Machine Translated by Google

Undergraduate Topics in Computer Science

Wolfgang Ertel

Introduction to Artificial Intelligence

UTiCS

Springer

Machine Translated by Google

Темы бакалавриата по информатике

Machine Translated by Google

Темы бакалавриата по компьютерным наукам (UTiCS) предоставляют высококачественный учебный контент для студентов, изучающих все области вычислительной техники и информатики. Книги UTiCS, от основного фундаментального и теоретического материала до тем и приложений последнего года обучения, используют свежий, лаконичный и современный подход и идеально подходят для самостоятельного изучения или для одно- или двухсеместрового курса. Все тексты написаны признанными экспертами в своих областях, проверены международным консультативным советом и содержат многочисленные примеры и проблемы. Многие включают полностью проработанные решения.

Дополнительные
тома: www.springer.com/series/7592

Machine Translated by Google

Вольфганг Эртель

Введение в искусственный Интеллект

Перевод Натанаэля Блэка
С иллюстрациями Флориана Маста.

Springer

Machine Translated by Google

Проф. д-р Вольфганг Эртель FB
Elektrotechnik und Informatik Hochschule
Равенсбург-Вайнгартен Университет прикладных
наук Вайнгартен Германия ertel@hs-
weingarten.de

Редактор сериала
Ян Маки

Консультативный совет
Самсон Абрамски, Оксфордский университет, Оксфорд, Великобритания
Карин Брейтман, Папский католический университет Рио-де-Жанейро, Рио-де-Жанейро, Бразилия
Крис Хэнкин, Имперский колледж Лондона, Лондон, Великобритания
Декстер Козен, Корнельский университет, Итака, США
Эндрю Питтс, Кембриджский университет, Кембридж, Великобритания
Ханне Риис Нильсон, Технический университет Дании, Kongens Lyngby, Дания
Стивен Скиена, Университет Стоуни-Брук, Стоуни-Брук, США
Иэн Стюарт, Даремский университет, Дарем, Великобритания

ISSN 1863-7310 ISBN
978-0-85729-298-8 DOI
10.1007/978-0-85729-299-5 Springer London
Dordrecht Heidelberg New York
электронный ISBN 978-0-85729-299-5

Каталогизация Британской библиотеки в данных публикаций
Запись в каталоге этой книги доступна в Британской библиотеке.

Контрольный номер Библиотеки Конгресса: 2011923216

Первоначально опубликовано на немецком языке издательством Vieweg+Teubner, 65189 Wiesbaden, Germany, как «Ertel, Wolfgang: Grundkurs Künstliche Intelligenz. 2. обр. версия". © Vieweg+Teubner | GWV Fachverlage GmbH, Висбаден, 2009 г.

© Springer-Verlag London Limited 2011 Помимо любых
добросовестных действий в целях исследования или частного изучения, а также критики или рецензирования в соответствии с Законом об авторском праве, промышленных образцах и патентах 1988 г., эта публикация может воспроизводиться, храниться или передаваться только в любой форме и любыми средствами с предварительного письменного разрешения издателей или в случае репрографического воспроизведения в соответствии с условиями лицензий, выданных Агентством по лицензированию авторских прав. Запросы относительно воспроизведения вне этих условий следует направлять издателям.

Использование зарегистрированных наименований, товарных знаков и т. д. в этой публикации не означает, даже при отсутствии конкретного заявления, что такие названия не подпадают под действие соответствующих законов и правил и, следовательно, бесплатны для общего использования.
Издатель не делает никаких заявлений, явных или подразумеваемых, относительно точности информации, содержащейся в этой книге, и не может нести никакой юридической ответственности за любые ошибки или упущения, которые могут быть допущены.

Напечатано на бескислотной бумаге.

Springer является частью Springer Science+Business Media (www.springer.com).

Machine Translated by Google

Искусственный интеллект (ИИ) имеет определенную цель — понять интеллект и построить интеллектуальные системы. Однако методы и формализмы, используемые на пути к этой цели, не являются твердо установленными, в результате чего ИИ сегодня состоит из множества субдисциплин. Трудность вводного курса ИИ заключается в том, чтобы передать как можно больше ветвей, не теряя при этом слишком большой глубины и точности.

Книга Рассела и Норвига [RN10] является более или менее стандартным введением в ИИ. Однако, поскольку в этой книге 1152 страницы, и поскольку она слишком обширна и дорогостояща для большинства студентов, требования к написанию этой книги были ясны: она должна быть доступным введением в современный ИИ для самостоятельного изучения или в качестве основы для четырех часовая лекция, объемом не более 300 страниц. Результат перед вами.

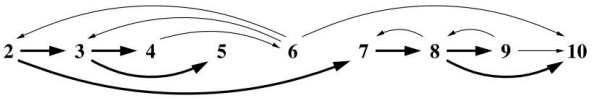
На 300 страницах невозможно полностью охватить такую обширную область, как ИИ. Чтобы не превращать книгу в оглавление, я попытался углубиться и представить конкретные алгоритмы и приложения в каждой из следующих ветвей: агенты, логика, поиск, рассуждение с неопределенностью, машинное обучение и нейронные сети.

Области обработки изображений, нечеткой логики и обработки естественного языка подробно не рассматриваются. Область обработки изображений, важная для всех компьютерных наук, представляет собой самостоятельную дисциплину с очень хорошими учебниками, такими как [GW08]. Аналогичный статус имеет обработка естественного языка. При распознавании и генерации текста и разговорной речи применяются методы логики, вероятностных рассуждений и нейронных сетей. В этом смысле это поле является частью ИИ. С другой стороны, компьютерная лингвистика представляет собой отдельную обширную отрасль компьютерных наук и имеет много общего с формальными языками. В этой книге мы укажем на такие подходящие системы в нескольких местах, но не будем давать систематического введения. В качестве первого введения в эту область мы обратимся к гл. 22 и 23 в [RN10]. Нечеткая логика, или теория нечетких множеств, превратилась в раздел теории управления из-за ее основного применения в технологии автоматизации и рассматривается в соответствующих книгах и лекциях. Поэтому мы воздержимся от введения здесь.

Зависимости между главами книги грубо показаны на графике, показанном ниже. Для простоты гл. 1, с основным введением для всех последующих глав, опущен. Например, более толстая стрелка от 2 до 3 означает, что логика высказываний является необходимым условием для понимания логики предикатов. Тонкая стрелка от 9 до 10 означает, что нейронные сети полезны для понимания обучения с подкреплением, но не абсолютно необходимы. Тонкий назад

Machine Translated by Google

стрелки должны указывать на то, что последующие главы могут дать более глубокое понимание
к темам, которые уже изучены.



Эта книга применима для студентов, изучающих информатику и другие технические естественные науки.
наук и, по большей части, требует знания математики на уровне средней школы. В нескольких местах
знания из линейной алгебры и многомерного анализа
необходим. Для более глубокого понимания содержания необходима активная работа над упражнениями.
Это означает, что к решениям следует обращаться только
после интенсивной работы над каждой проблемой и только для проверки своих решений, верных
Девиз Леонардо да Винчи «Учеба без преданности вредит мозгу». В некотором роде
более сложные задачи отмечены знаком , а особо сложные – знаком .
Проблемы, которые требуют программирования или специальных знаний в области компьютерных наук,
отмечены .

На веб-сайте книги по адресу www.hs-weingarten.de/~ertel/aibook цифровые материалы для
упражнений, такие как обучающие данные для алгоритмов обучения, страница со ссылками на
программы ИИ, упомянутые в книге, список ссылок на затронутые темы,
интерактивный список библиографии, список опечаток и слайды презентации для лекторов
может быть найден. Прошу читателей присылать предложения, критические замечания и советы по
ошибки напрямую на ertel@hs-weingarten.de.

Эта книга является обновленным переводом моей немецкой книги «Grundkurs Künstliche».
Intelligenz», опубликованном Vieweg Verlag. Отдельное спасибо переводчику
Натан Блэк, который в отличном трансатлантическом сотрудничестве между Германией и
Калифорния через SVN, Skype и электронную почту подготовила этот текст. Я благодарен Францу
Курфессу, который познакомил меня с Натаном; Мэтью Уайту за корректуру переведенного
книгу и Саймону Рису из Springer Verlag за его терпение.

Я хотел бы поблагодарить мою жену Эвелин за ее поддержку и терпение во время этого
трудоемкий проект. Особая благодарность Вольфгангу Бибелю и Крису Лобену Шуссу, которые тщательно
исправили немецкую рукопись. Их предложения и обсуждения привели ко многим улучшениям и
дополнениям. Для ознакомления с исправлениями и
другие ценные услуги, я хотел бы поблагодарить Richard Cubek, Celal Döven, Joachim
Фесслер, Нико Хохгешвендер, Пол Кирнер, Вильфрид Мейстер, Норберт Перк, Пе тер Радтке, Маркус
Шнайдер, Манфред Шрамм, Ули Старк, Мишель Токич, Арне
Усаделя и всех заинтересованных студентов. Я также благодарю Флориана Маста за
бесценные мультфильмы и очень эффективное сотрудничество.

Я надеюсь, что во время учебы эта книга поможет вам разделить мое увлечение
Искусственный интеллект.

Равенсбург Вольфганг Эртель
февраль 2011 г.

Machine Translated by Google

Содержание

1 Введение 1

1.1 Что такое искусственный интеллект? 1

1.1.1 Наука о мозге и решение проблем 3

1.1.2 Тест Тьюринга и Чаттерботы. 4

1.2 История ИИ. 5

1.2.1 Первые начинания. . 5

1.2.2 Логика решает (почти) все проблемы 1.2.3 Новый коннекционизм 6 7

1.2.4 Рассуждения в условиях неопределенности 7

1.2.5 Распределенные, автономные и обучающиеся агенты 8

1.2.6 ИИ взрослеет. 9

1.3 Агенты. 9

1.4 Системы, основанные на знаниях . 12

1.5 Упражнения 14

2 Пропозициональная логика. 15

2.1 Синтаксис. 15

2.2 Семантика. 16

2.3 Системы доказательств. 18

2.4 Разрешение 22

2.5 Оговорки Хорна. 25

2.6. Вычислимость и сложность. 2.7 Применение 28

и ограничения 2.8 Упражнения 29

29

3 Логика предикатов первого порядка. . . . 31

3.1 Синтаксис. 32

3.2 Семантика. 33

3.2.1 Равенство. 3.3 36

Кванторы и нормальные формы 3.4 37

Доказательства исчисления. 40

3.5 Разрешение 42

3.5.1 Стратегии разрешения . 46

3.5.2 Равенство. 46

3.6 Автоматические средства доказательства теорем. 47

Machine Translated by Google

3.7 Математические примеры. . . 3.8 48
Приложения. 51
3.9 Резюме. 54
3.10 Упражнения 54

4 Ограничения логики. 57
4.1 Проблема пространства поиска 4.2 57
Разрешимость и неполнота. 59
4.3 Летающий пингвин. 60
4.4 Моделирование неопределенности. 63
4.5 Упражнения 65

5 Логическое программирование с PROLOG 67
5.1 Системы и реализации PROLOG 68
5.2 Простые примеры. 68
5.3 Управление выполнением и процедурные элементы 71
5.4 Списки 5.5 73
Самомодифицирующиеся программы 75
5.6 Пример планирования. 5.7 5.8 76
Логическое программирование ограничений. 77
Резюме . 5.9 79
Упражнения 80

6 Поиск, игры и решение проблем . 6.1 Введение 83
6.2 Неинформированный 83
поиск 89
6.2.1 Поиск в ширину. 89
6.2.2 Поиск в глубину 91
6.2.3 Итеративное углубление 92
6.2.4 Сравнение. 94
6.3 Эвристический поиск. 94
6.3.1 Жадный поиск. 96
6.3.2 A -Поиск 98
6.3.3 IDA -Поиск 100
6.3.4 Эмпирическое сравнение алгоритмов поиска. 100
6.3.5 Резюме. 102
6.4 Игры с соперниками. 102
6.4.1 Минимаксный поиск 103
6.4.2 Альфа-бета-обрезка. 103
6.4.3 Недетерминированные игры. 106
6,5 Функции эвристической оценки 106
6.5.1 Изучение современных 107
6,6 эвристических 108
6,7 упражнений. 110

7 Рассуждения с неуверенностью. 113
7.1 Вычисления с вероятностями 115

Machine Translated by Google

7.1.1 Условная вероятность. 118
7.2 Принцип максимальной энтропии. 122
7.2.1 Правило вывода для вероятностей 7.2.2 123
Максимальная энтропия без явных ограничений. 127
7.2.3 Условная вероятность в сравнении с существенным следствием. . . 128
7.2.4 MaxEnt-Системы. 129
7.2.5 Пример Твити. 130
7.3 ЛЕКСМЕД, экспертная система диагностики аппендицита. 131
7.3.1 Диагностика аппендицита с помощью формальных 131
методов 7.3.2 Гибридно-вероятностная база знаний 132
7.3.3 Применение LEXMED 7.3.4 135
Функция LEXMED 7.3.5 Управление 136
рисками с использованием матрицы затрат 7.3.6 139
Производительность 141
7.3.7 Области применения и опыт 143
7.4 Рассуждения с помощью байесовских сетей. 144
7.4.1 Независимые переменные. 144
7.4.2 Графическое представление знаний как байесовское
Сеть 146
7.4.3 Условная независимость. 146
7.4.4 Практическое применение. 148
7.4.5 Программное обеспечение для байесовских сетей. 149
7.4.6 Разработка байесовских сетей. 150
7.4.7 Семантика байесовских сетей. 154
7.5 Резюме. 7.6 156
Упражнения 157

8 Машинное обучение и интеллектуальный анализ данных. 161
8.1 Анализ данных. 166
8.2 Персептрон, линейный классификатор. 169
8.2.1 Правило обучения 8.2.2 171
Оптимизация и перспективы 174
8.3 Метод ближайших соседей 175
8.3.1 Два класса, много классов, приближение. 179
8.3.2 Расстояние имеет значение 180
8.3.3 Время вычислений 8.3.4 181
Сводка и перспективы 182
8.3.5 Рассуждение по прецедентам 183
8.4 Обучение дереву решений. 184
8.4.1 Простой пример. 185
8.4.2 Энтропия как показатель содержания информации. 186
8.4.3 Получение информации 189
8.4.4 Применение C4.5. 191
8.4.5 Обучение диагностике аппендицита. 193
8.4.6 Непрерывные атрибуты 196

Machine Translated by Google

8.4.7 Сокращение — вырезание дерева 197
8.4.8 Отсутствующие значения 198
8.4.9 Резюме. 199
8.5 Изучение байесовских сетей. 199
8.5.1 Изучение структуры сети. 199
8.6 Наивный байесовский классификатор. 202
8.6.1 Классификация текста с помощью наивного Байеса. 204
8,7 Кластеризация 206
8.7.1 Метрики расстояния 207
8.7.2 k-средних и алгоритм EM 208
8.7.3 Иерархическая кластеризация 209
8.8 Интеллектуальный анализ данных на 211
практике 8.8.1 Инструмент интеллектуального анализа данных KNIME 212
8.9 Резюме. 214
8.10 Упражнения 216
8.10.1 Введение 8.10.2 216
Персептрон 216
8.10.3 Метод ближайших соседей 8.10.4 217
Деревья решений 8.10.5 218
Изучение байесовских сетей 219
8.10.6 Кластеризация 220
8.10.7 Интеллектуальный анализ данных. 220

9 Нейронные сети. 221
9.1 От биологии к моделированию. 222
9.1.1 Математическая модель 9.2 Сети 223
Хопфилда 226
9.2.1 Применение к примеру распознавания образов. . . . 227
9.2.2 Анализ. 228
9.2.3 Резюме и перспективы 231
9.3 Нейронно-ассоциативная память. 232
9.3.1 Память корреляционной матрицы 233
9.3.2 Псевдоинверсия. 235
9.3.3 Двоичное правило Хебба 236
9.3.4 Программа исправления правописания 238
9.4 Линейные сети с минимальными ошибками 240
9.4.1 Метод наименьших квадратов 241
9.4.2 Применение к данным аппендицита 9.4.3 242
Дельта-правило 9.4.4 243
Сравнение с персептроном 245
9.5 Алгоритм обратного распространения. 246
9.5.1 NETtalk: сеть учится говорить 249
9.5.2 Изучение эвристик для средств доказательства теорем. 250
9.5.3 Проблемы и улучшения 251
9.6 Машины опорных векторов 252

Machine Translated by Google

9.7 Приложения. 254
9.8 Резюме и перспективы . . 9.9 254
Упражнения 9.9.1 255
От биологии к моделированию. 255
9.9.2 Сети Хопфилда. 255
9.9.3 Линейные сети с минимальными ошибками 255
9.9.4 Обратное распространение 256
9.9.5 Машины опорных векторов 256

10 Обучение с подкреплением. 257
10.1 Введение 10.2 257
Задача 10.3 259
Неинформированный комбинаторный поиск 261
10.4 Итерация значений и динамическое программирование. 262
10.5 Обучаемый шагающий робот и его моделирование 265
10.6 Q-обучение. 267
10.6.1 Q-обучение в недетерминированной среде. . . . 270
10.7 Разведка и разработка 272
10.8 Аппроксимация, обобщение и сходимость. 272
10.9 Приложения. 273
10.10 Проклятие размерности. 274
10.11 Резюме и перспективы 275
10.12 Упражнения 276

11 решений для упражнений. 279
11.1 Введение 11.2 279
Логика высказываний. 280
11.3. Логика предикатов первого порядка. 282
11.4 Ограничения логики. 283
11.5 ПРОЛОГ. 283
11.6 Поиск, игры и решение проблем 285
11.7 Рассуждения с неопределенностью. 287
11.8 Машинное обучение и интеллектуальный анализ данных. 293
11.9 Нейронные сети. 300
11.10 Обучение с подкреплением. 301

Использованная литература 305

Индекс 311

Machine Translated by Google

Machine Translated by Google

Введение 1

1.1 Что такое искусственный интеллект?

Термин «искусственный интеллект» вызывает эмоции. Во-первых, это наше увлечение интеллектом, который, по-видимому, придает нам, людям, особое место среди форм жизни. Возникают такие вопросы, как «Что такое интеллект?», «Как можно измерить интеллект?» или «Как работает мозг?». Все эти вопросы имеют смысл при попытке понять искусственный интеллект. Однако центральным вопросом для инженера, особенно для специалиста по информатике, является вопрос об интеллектуальной машине, которая ведет себя как человек, демонстрируя разумное поведение.

Искусственный атрибут может вызывать самые разные ассоциации. Это вызывает страх перед разумными киборгами. Напоминает образы из фантастических романов. Это поднимает вопрос о том, следует ли нам попытаться понять, смоделировать или даже реконструировать наше высшее благо, душу.

С такими различными небрежными интерпретациями становится трудно дать простое и надежное определение термину « искусственный интеллект» или ИИ . Тем не менее я хотел бы попытаться, используя примеры и исторические определения, охарактеризовать область ИИ. В 1955 году Джон Маккарти, один из пионеров искусственного интеллекта, первым дал определение термину « искусственный интеллект» примерно следующим образом:

Цель ИИ — разработать машины, которые ведут себя так, как если бы они были разумными.

Чтобы проверить это определение, читатель может представить себе следующий сценарий. Пятнадцать или около того небольших роботизированных транспортных средств движутся по замкнутой поверхности размером четыре на четыре метра. Можно наблюдать различные модели поведения. Некоторые транспортные средства образуют небольшие группы с относительно небольшим движением. Другие мирно перемещаются по пространству и изящно избегают любых столкновений. Третьи, кажется, следуют за лидером. Наблюдается также агрессивное поведение. Является ли то, что мы наблюдаем, разумным поведением?

По определению Маккарти вышеупомянутых роботов можно назвать интеллектуальными. Психолог Валентин Брайтенберг показал, что это, казалось бы, сложное поведение может быть вызвано очень простыми электрическими цепями [Bra84]. Так называемые автомобили Брайтенберга имеют два колеса, каждое из которых приводится в движение независимым электродвигателем. На скорость каждого мотора влияет датчик света на

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по компьютерным наукам,
DOI 10.1007/978-0-85729-299-5_1, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google

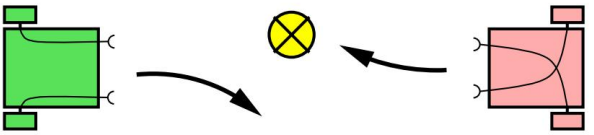


Рис. 1.1 Два очень простых автомобиля Брайтенберга и их реакция на источник света

передней части автомобиля, как показано на рис. 1.1. Чем больше света попадает на датчик, тем быстрее работает двигатель. Транспортное средство 1 в левой части рисунка по своей конфигурации удаляется от точечного источника света. Транспортное средство 2, с другой стороны, движется к источнику света. Дальнейшие небольшие модификации могут создать другие модели поведения, так что с помощью этих очень простых транспортных средств мы сможем реализовать впечатляющее поведение, описанное выше.

Ясно, что приведенного выше определения недостаточно, потому что целью ИИ является решение сложных практических задач, которые, безусловно, слишком требовательны к транспортному средству Брайтенберга. В Британской энциклопедии [Bri91] можно найти такое определение:

> ИИ — это способность цифровых компьютеров или роботов, управляемых компьютером, решать проблемы, которые обычно связаны с более высокими интеллектуальными способностями человека к обработке данных.

Но у этого определения есть и недостатки. Например, можно было бы признать, что компьютер с большой памятью, который может сохранять длинный текст и извлекать его по запросу, демонстрирует интеллектуальные возможности, поскольку запоминание длинных текстов, безусловно, можно считать более высокой интеллектуальной вычислительной способностью человека , как, например, быстрота. умножение двух 20-значных чисел. Таким образом, согласно этому определению, каждый компьютер представляет собой систему ИИ. Эта дилемма элегантно решается следующим определением Элейн Рич [Ric83]:

> Искусственный интеллект — это наука о том, как заставить компьютеры делать то, в чем люди на данный момент лучше.

Богато, лаконично и лаконично характеризует то, чем исследователи ИИ занимались последние 50 лет. Даже в 2050 году это определение будет актуальным.

Такие задачи, как выполнение множества вычислений за короткий промежуток времени, являются сильными сторонами цифровых компьютеров. В этом отношении они превосходят людей во много раз. Однако во многих других областях люди намного превосходят машины. Например, человек, вошедший в незнакомое помещение, за доли секунды узнает обстановку и, при необходимости, так же быстро примет решение и спланирует действия. На сегодняшний день эта задача слишком требовательна к автономным1 роботам. Следовательно, согласно определению Рича, это задача для ИИ. На самом деле исследования автономных роботов — важная и актуальная тема в области искусственного интеллекта. С другой стороны, создание шахматных компьютеров потеряло актуальность, потому что они уже играют на уровне гроссмейстеров или выше.

Однако было бы опасно делать из определения Рича вывод о том, что ИИ занимается только прагматичной реализацией интеллектуальных процессов. Интеллектуальные системы в смысле определения Рича не могут быть построены без глубокого

1Автономный робот работает самостоятельно, без ручной поддержки, в частности без дистанционного контроль.

Machine Translated by Google

понимание человеческого мышления и интеллектуальных действий в целом, из-за чего нейробиология (см. раздел 1.1.1) имеет большое значение для ИИ. Это также показывает, что другие приведенные определения отражают важные аспекты ИИ.

Особой силой человеческого интеллекта является адаптивность. Мы способны приспосабливаться к различным условиям окружающей среды и соответственно изменять свое поведение посредством обучения. Именно потому, что наши способности к обучению намного превосходят возможности компьютеров, машинное обучение , согласно определению Рича, является центральной частью ИИ.

1.1.1 Наука о мозге и решение проблем

Благодаря исследованию интеллектуальных систем мы можем попытаться понять, как работает человеческий мозг, а затем смоделировать или симулировать его на компьютере. Многие идеи и принципы в области нейронных сетей (см. главу 9) проистекают из науки о мозге и родственной области нейронауки.

Совершенно иной подход получается, когда вы выбираете целенаправленную линию действий, начиная с проблемы и пытаясь найти наиболее оптимальное решение. То, как люди решают проблему, здесь не имеет значения. Метод в этом подходе вторичен. Прежде всего, это оптимальное интеллектуальное решение проблемы. Вместо использования фиксированного метода (такого, например, как логика предикатов) ИИ имеет своей постоянной целью создание интеллектуальных агентов для максимально возможного числа различных задач. Поскольку задачи могут быть самыми разными, неудивительно, что методы, используемые в настоящее время в ИИ, часто также сильно различаются. Подобно медицине, которая включает в себя множество различных, часто спасающих жизнь диагностических и терапевтических процедур, ИИ также предлагает широкий спектр эффективных решений для самых разных приложений. Для мысленного вдохновения рассмотрите рис. 1.2 на стр. 4. Так же, как и в медицине, нет универсального метода для всех областей применения ИИ, а существует множество возможных решений для множества различных повседневных проблем, больших и малых.

Когнитивная наука занимается исследованием человеческого мышления на несколько более высоком уровне. Подобно науке о мозге, эта область снабжает практический ИИ многими важными идеями. С другой стороны, алгоритмы и реализации приводят к дальнейшим важным выводам о том, как функционирует человеческое мышление. Таким образом, эти три области выигрывают от плодотворного междисциплинарного обмена. Тем не менее, предметом этой книги в первую очередь является проблемно-ориентированный ИИ как раздел информатики.

Вокруг интеллекта и искусственного интеллекта возникает много интересных философских вопросов. У нас, людей, есть сознание; то есть мы можем думать о себе и даже размышлять о том, что мы способны думать о себе. Как возникает сознание? Многие философы и неврологи сейчас считают, что разум и сознание связаны с материей, то есть с мозгом. Вопрос о том, смогут ли машины когда-нибудь иметь разум или сознание, может в какой-то момент в будущем стать актуальным. Проблема разума и тела, в частности, касается того, связан ли разум с телом. Мы не будем обсуждать здесь эти вопросы. Заинтересованный читатель может обратиться к [Spe98, Spe97], и ему предлагается в ходе изучения технологии ИИ составить личное мнение по этим вопросам.

Machine Translated by Google

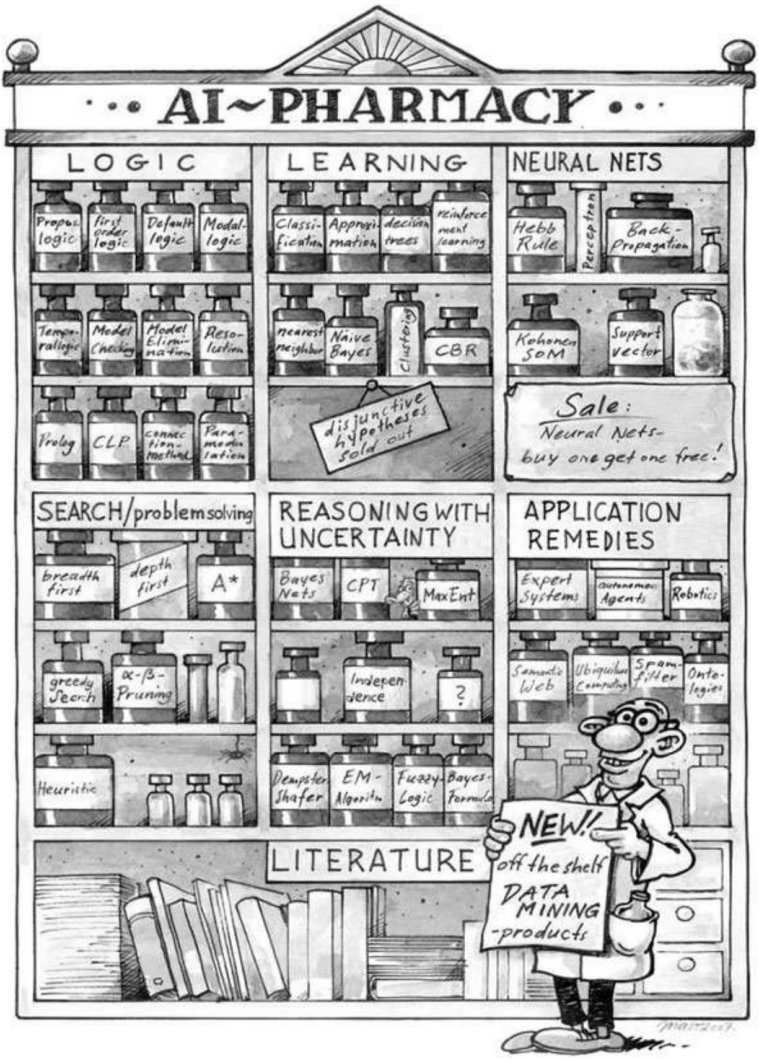


Рис. 1.2 Небольшая выборка решений, предлагаемых ИИ

1.1.2 Тест Тьюринга и болтуны

Алан Тьюринг прославился как один из первых пионеров ИИ своим определением интеллектуальной машины, в котором рассматриваемая машина должна пройти следующий тест. Испытуемый Алиса сидит в запертой комнате с двумя компьютерными терминалами. Один терминал подключен к машине, другой к незлонамеренному человеку Бобу. Алиса может вводить вопросы в оба терминала. Ей дается задание через пять минут решить, какой терминал принадлежит машине. Машина проходит тест, если она может обмануть Алису как минимум в 30% случаев [Tur50].

Machine Translated by Google

В то время как тест очень интересен с философской точки зрения, для практического ИИ, который занимается решением проблем, он не очень актуален. Причины этого аналогичны упомянутым выше, связанным с транспортными средствами Брайтенберга (см. упражнение 1.3 на стр. 14).

Пионер искусственного интеллекта и социальный критик Джозеф Вейценбаум разработал программу под названием « Элиза», которая должна отвечать на вопросы испытуемого подобно психологу-человеку [Wei66]. Он действительно был в состоянии продемонстрировать успех во многих случаях. Якобы у его секретаря часто были долгие дискуссии с программой. Сегодня в Интернете существует множество так называемых чат-ботов, первые ответы некоторых из них весьма впечатляют. Однако через некоторое время их искусственная природа становится очевидной. Некоторые из этих программ действительно способны к обучению, в то время как другие обладают экстраординарными знаниями по различным предметам, например географии или разработке программного обеспечения. Уже есть коммерческие приложения для чат-ботов в онлайн-поддержке клиентов, и могут быть другие в области электронного обучения. Вполне возможно, что учащийся и система электронного обучения могли бы общаться через чат-бота. Читатель может сравнить несколько болтунов и оценить их интеллект в упражнении 1.1 на странице 14.

1.2 История ИИ

ИИ опирается на многие прошлые научные достижения, которые здесь не упоминаются, поскольку ИИ как самостоятельная наука существует только с середины двадцатого века. Таблица 1.1 на стр. 10 с наиболее важными вехами ИИ и графическое представление основных движений ИИ на рис. 1.3 на стр. 6 дополняют следующий текст.

1.2.1 Первые начинания

В 1930-х годах Курт Гёдель, Алонсо Чёрч и Алан Тьюринг заложили важные основы логики и теоретической информатики. Особый интерес для ИИ представляют теоремы Гёделя. Теорема о полноте утверждает, что логика предикатов первого порядка полна. Это означает, что каждое истинное утверждение, которое может быть сформулировано в логике предикатов, доказуемо с использованием правил формального исчисления. На этой основе позже можно было бы построить автоматические средства доказательства теорем как реализации формальных исчислений. С помощью теоремы о неполноте Гёдель показал, что в логике высшего порядка существуют истинные утверждения, которые недоказуемы. Этим он раскрыл болезненные пределы формальных систем.

Доказательство Алана Тьюринга неразрешимости проблемы остановки также относится к этому периоду времени. Он показал, что не существует программы, которая может решить, будет ли заданная произвольная программа (и соответствующий ввод) выполняться в бесконечном цикле. С

2Логика высшего порядка является расширением логики предикатов, в которой не только переменные, но и функциональные символы или предикаты могут выступать в качестве терминов в квантификации. Действительно, Гёдель только показал, что любая система, основанная на логике предикатов и способная сформулировать арифметику Пеано, неполна.

Machine Translated by Google

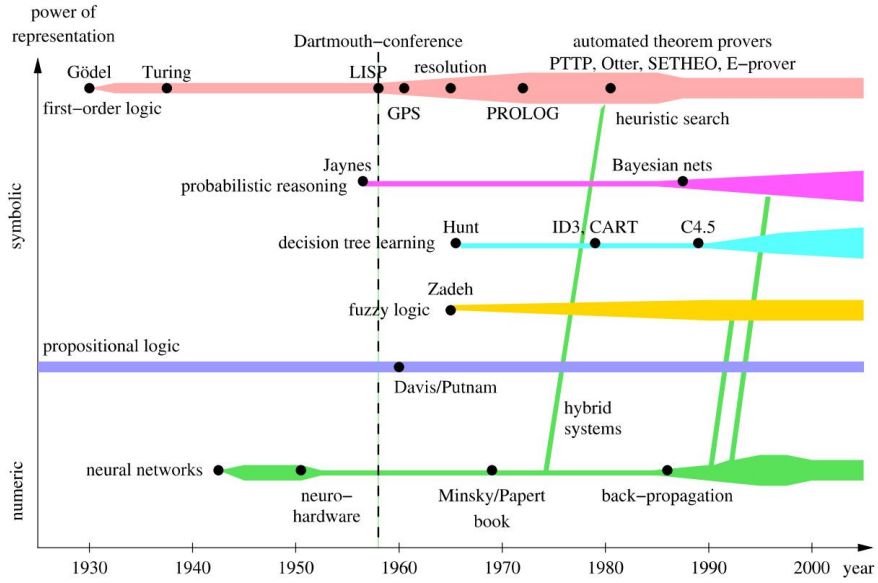


Рис. 1.3. История различных областей ИИ. Ширина столбцов указывает на распространенность использования метода.

этот Тьюринг также определил предел для интеллектуальных программ. Из этого следует, например, что универсальной системы проверки программ никогда не

будет.3 В 1940-х годах, основываясь на результатах нейробиологии, МакКаллох, Питтс и Хебб разработали первые математические модели нейронных сетей. Однако компьютерам в то время не хватало мощности для имитации простого мозга.

1.2.2 Логика решает (почти) все проблемы

ИИ как практическая наука о механизации мышления, конечно же, мог начаться только тогда, когда появились программируемые компьютеры. Так было в 1950-х годах. Ньюэлл и Саймон представили Logic Theorist, первое автоматическое средство доказательства теорем, и таким образом также показали, что с помощью компьютеров, которые на самом деле работают только с числами, можно также обрабатывать символы. В то же время Маккарти вместе с языком LISP представил язык программирования, специально созданный для обработки символьных структур. Обе эти системы были представлены в 1956 г. на исторической Дартмутской конференции, которая считается днем рождения ИИ.

В США LISP превратился в наиболее важный инструмент для реализации систем искусственного интеллекта, обрабатывающих символы. После этого правило логического вывода, известное как разрешение, превратилось в полное исчисление логики предикатов.

3Это утверждение относится к «полной правильности», что подразумевает доказательство правильного выполнения, а также доказательство завершения для каждого действительного ввода.

Machine Translated by Google

В 1970-х годах был представлен язык логического программирования PROLOG. Европейский аналог LISP. PROLOG предлагает то преимущество, что позволяет напрямую программирование с использованием предложений Хорна, подмножества логики предикатов. Как ЛИСП, ПРОЛОГ имеет типы данных для удобной обработки списков.

Вплоть до 1980-х годов в ИИ преобладал прорывной дух, особенно среди многие логики. Причиной тому стала череда впечатляющих достижений в обработка символов. С проектом компьютерных систем пятого поколения в Японии и программы ESPRIT в Европе, значительные инвестиции были направлены на строительство интеллектуальные компьютеры.

Для небольших задач автоматические пруверы и другие системы обработки символов иногда работал очень хорошо. Однако комбинаторный взрыв пространства поиска определил очень узкое окно для этих успехов. Эта фаза ИИ была описана в [RN10] как «Смотри, мама, без рук!» эпоха.

Поскольку экономический успех систем ИИ не оправдал ожиданий, финансирование исследований ИИ, основанных на логике, в Соединенных Штатах резко сократилось в 1980-х годах.

1.2.3 Новый коннекционизм

На этом этапе разочарования компьютерщики, физики и когнитивные ученые смогли показать, используя компьютеры, которые теперь были достаточно мощными, что математически смоделированные нейронные сети способны обучаться, используя обучающие примеры, для выполнения задач, которые ранее требовали дорогостоящего программирования. Благодаря отказоустойчивости таких систем и их способности распознавать образы стали возможными значительные успехи, особенно в распознавании образов. Распознавание лиц на фотографиях и распознавание рукописного ввода — два примера приложений. Система Nettalk смогла выучить речь из примеров текстов [SR86]. Под названием коннекционизм родилась новая субдисциплина ИИ.

Коннекционизм процветал, и субсидии текли рекой. Но скоро и здесь целесообразность пределы стали очевидны. Нейронные сети могут приобрести впечатляющие возможности, но обычно было невозможно зафиксировать изученное понятие в простых формулах или логические правила. Попытки объединить нейронные сети с логическими правилами или знаниями человеческих экспертов столкнулись с большими трудностями. Кроме того, нет удовлетворительного решения к структурированию и модуляризации сетей.

1.2.4 Рассуждение в условиях неопределенности

ИИ как практическая, целеустремленная наука искала выход из этого кризиса. Один хотел объединить способность логики явно представлять знания с силой нейронных сетей в работе с неопределенностью. Было предложено несколько альтернатив.

Наиболее многообещающее, вероятностное рассуждение работает с условными вероятностями для формул исчисления высказываний. С тех пор было построено много диагностических и экспертных систем для задач повседневного мышления с использованием байесовских сетей. Успех байесовских сетей обусловлен их интуитивной понятностью,

Machine Translated by Google

чистой семантики условной вероятности и многовековой, математически обоснованной теории вероятностей.

Слабые стороны логики, которая может работать только с двумя значениями истинности, могут быть решается с помощью нечеткой логики, которая прагматически вводит бесконечно много значений между нулем и единицей. Хотя и сегодня его теоретическая основа не вполне прочна, он успешно используется, особенно в технике управления.

Совсем другой путь привел к успешному синтезу логики и нейронных сетей. под названием гибридные системы. Например, нейронные сети использовались для изучить эвристики для сокращения огромного пространства комбинаторного поиска при обнаружении доказательств [SE90].

Методы обучения дерева решений на основе данных также работают с вероятностями. Такие системы, как CART, ID3 и C4.5, могут быстро и автоматически создавать очень точные деревья решений, которые могут представлять концепции пропозициональной логики, а затем использоваться как экспертные системы. Сегодня они являются фаворитом среди методов машинного обучения. (раздел 8.4).

Примерно с 1990 года интеллектуальный анализ данных развивался как поддисциплина ИИ в этой области. статистического анализа данных для извлечения знаний из больших баз данных. Данные Добыча полезных ископаемых не привносит в ИИ новых методов, а скорее требует использования больших баз данных для получения явных знаний. Одно приложение с большим рынком Потенциал заключается в управлении рекламными кампаниями крупного бизнеса на основе анализа многомиллионных покупок их клиентов. Как правило, такие методы машинного обучения, как поскольку здесь в игру вступает обучение дереву решений.

1.2.5 Распределенные, автономные и обучающиеся агенты

Распределенный искусственный интеллект, DAI, активно исследуется примерно с 1985. Одной из ее целей является использование параллельных компьютеров для повышения эффективности. решателей проблем. Однако оказалось, что из-за высокой вычислительной сложность большинства задач, использование «интеллектуальных» систем более выгодно чем само распараллеливание.

Совершенно другой концептуальный подход возникает в результате разработки автономных программных агентов и роботов, предназначенных для взаимодействия наподобие человеческих команд. Как и в случае с вышеупомянутыми автомобилями Брайтенберга, во многих случаях отдельный агент не способен решить проблему даже с неограниченными ресурсами. Только сотрудничество многих агентов приводит к разумному поведению или к решение проблемы. Колония муравьев или колония термитов способна возводить здания очень высокой архитектурной сложности, несмотря на то, что ни один муравейник понимает, как все это сочетается. Это похоже на ситуацию с обеспечение хлебом большого города, такого как Нью-Йорк [RN10]. Нет центрального агентства по планированию хлеба, вместо этого есть сотни пекарей, знающих свое дело. районах города и выпекать в этих местах необходимое количество хлеба.

Активное приобретение навыков роботами — захватывающая область текущих исследований. Там сегодня это, например, роботы, которые самостоятельно учатся ходить или выполнять различные двигательные навыки, связанные с футболом (глава 10). Совместное обучение нескольких роботов решать проблемы вместе все еще находится в зачаточном состоянии.

Machine Translated by Google

1.2.6 ИИ взрослеет

Вышеуказанные системы, предлагаемые ИИ сегодня, — это не универсальный рецепт, а мастерская с управляемым количеством инструментов для самых разных задач. Большинство этих инструментов хорошо разработаны и доступны в виде готовых программных библиотек, часто с удобными пользовательскими интерфейсами. Выбор правильного инструмента и его разумное использование в каждом отдельном случае остается за разработчиком ИИ или инженером по знаниям. Как и любое другое ремесленничество, это требует солидного образования, чему и призвана способствовать эта книга.

В большей степени, чем любая другая наука, ИИ является междисциплинарным, поскольку опирается на интересные открытия из таких различных областей, как логика, исследование операций, статистика, инженерия управления, обработка изображений, лингвистика, философия, психология и нейробиология. Кроме того, есть предметная область конкретного приложения.
Поэтому успешно разработать проект ИИ не всегда так просто, но почти всегда чрезвычайно увлекательно.

1.3 Агенты

Хотя термин «интеллектуальные агенты» не нов для ИИ, только в последние годы он получил известность благодаря [RN10], среди прочего. Агент обозначает в общем случае систему, которая обрабатывает информацию и производит результат на основе ввода. Эти агенты могут быть классифицированы по-разному.

В классической информатике в основном используются программные агенты (рис. 1.4 на стр. 11). В этом случае агент состоит из программы, которая вычисляет результат на основе пользовательского ввода.

С другой стороны, в робототехнике используются аппаратные агенты (также называемые роботами), которые дополнительно имеют в своем распоряжении датчики и исполнительные механизмы (рис. 1.5 на стр. 11). Агент может воспринимать окружающую среду с помощью сенсоров. С приводами он выполняет действия и изменяет свое окружение.

Что касается интеллекта агента, существует различие между рефлекторными агентами, которые реагируют только на входные данные, и агентами с памятью, которые также могут учитывать прошлое в своих решениях. Например, движущийся робот, который благодаря своим датчикам знает свое точное положение (и время), не имеет возможности, как рефлекторный агент, определить свою скорость. Однако, если он сохраняет позицию на коротких дискретных временных шагах, он может, таким образом, легко вычислить свою среднюю скорость в предыдущем временном интервале.

Если рефлекторный агент управляется детерминированной программой, он представляет собой функцию множества всех входов для множества всех выходов. С другой стороны, агент с памятью вообще не является функцией. Почему? (См. упражнение 1.5 на стр. 14.) Рефлекторных агентов достаточно в тех случаях, когда решаемая задача включает марковский процесс принятия решений. Это процесс, в котором для определения оптимального следующего действия необходимо только текущее состояние (см. главу 10).

Мобильный робот, который должен перейти из комнаты 112 в комнату 179 здания, совершает действия, отличные от действий робота, который должен перейти в комнату 105. Другими словами, действия зависят от цели. Такие агенты называются целевыми.

Machine Translated by Google

Таблица 1.1. Вехи развития ИИ от Гёделя до наших дней 1931 г. Австриец Курт Гёдель

показывает, что в логике предикатов первого порядка все истинные высказывания выводимы [Göd31a]. С другой стороны, в логике высшего порядка есть истинные утверждения, которые недоказуемы [Göd31b]. (В [Göd31b] Гёдель показал, что логика предикатов, расширенная аксиомами арифметики, неполна.)

1937 г. Алан Тьюринг указывает на пределы разумных машин с помощью проблемы остановки [Tur37].

1943 г. МакКаллох и Питтс моделируют нейронные сети и устанавливают связь с пропозициональными логика.

1950 Алан Тьюринг определяет машинный интеллект с помощью теста Тьюринга и пишет об обучающихся машинах и генетических алгоритмах [Tur50].

1951 Марвин Мински разрабатывает машину нейронной сети. С 3000 вакуумных ламп он имитирует 40 нейронов.

1955 Артур Сэмюэл (IBM) создает обучающую программу для игры в шахматы, которая играет лучше, чем ее разработчик [Sam59].

1956 Маккарти организует конференцию в Дартмутском колледже. Здесь имя Искусственный Впервые появился интеллект .

Ньюэлл и Саймон из Университета Карнеги-Меллона (CMU) представляют Logic Theorist, первую компьютерную программу для обработки символов [NSS83].

1958 Маккарти изобретает в Массачусетском технологическом институте (MIT) язык высокого уровня LISP. Он пишет программы, способные модифицировать себя.

1959 Гелернтер (IBM) создает средство для доказательства геометрических теорем.

1961 Общее средство решения проблем (GPS) Ньюэлла и Саймона имитирует человеческое мышление [NS61].

1963 Маккарти основывает лабораторию искусственного интеллекта в Стэнфордском университете.

1965 г. Робинсон изобретает резольвентное исчисление для логики предикатов [Rob65] (раздел 3.5).

1966 г. Программа Вейценбаума «Элиза» ведет диалог с людьми на естественном языке [Wei66] (раздел 1.1.2).

1969 Мински и Пейперт в своей книге «Персептроны» показывают , что персептрон, очень простой нейронная сеть может представлять только линейные функции [MP69] (раздел 1.1.2).

1972 г. Французский ученый Ален Кольмерауэр изобретает язык логического программирования PROLOG. (гл. 5).

Британский врач де Домбал разрабатывает экспертную систему для диагностики острой боли в животе [dDLS+72]. Это остается незамеченным в основном ИИ-сообществе того времени (раздел 7.3).

1976 г. Шортлифф и Бьюкенен разрабатывают MYCIN, экспертную систему для диагностики инфекционных заболеваний, способную справляться с неопределенностью (глава 7).

1981 Япония с большими затратами начинает «Проект пятого поколения» с целью создание мощной машины PROLOG.

1982 R1, экспертная система для настройки компьютеров, экономит корпорации Digital Equipment 40 миллионов долларов в год [McD82].

1986 Возрождение нейронных сетей благодаря, в частности, Румельхарту, Хинтону и Сейновски [RM86]. Система Nettalk учится читать тексты вслух [SR86] (гл. 9).

1990 Перл [Pea88], Cheeseman [Che85], Whittaker, Spiegelhalter привносят теорию вероятностей в ИИ с помощью байесовских сетей (раздел 7.4). Многоагентные системы становятся популярными.

1992 Программа Tesauros TD-gammon демонстрирует преимущества армирования обучение.

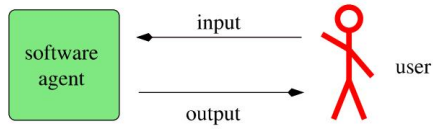
1993 Всемирная инициатива RoboCup по созданию автономных роботов, играющих в футбол [Roba].

Machine Translated by Google

Таблица 1.1 (продолжение)

1995	На основе статистической теории обучения Вапник разрабатывает машины опорных векторов, которые очень важны сегодня.
1997	Шахматный компьютер IBM Deep Blue побеждает чемпиона мира по шахматам Гэри Каспарова. Первые международные соревнования RoboCup в Японии.
2003	Роботы в RoboCup впечатляюще демонстрируют, на что способны ИИ и робототехника. достижение.
2006	Сервисная робототехника становится основным направлением исследований ИИ.
2010	Автономные роботы начинают изучать свою политику.
2011	Программа IBM для понимания естественного языка и ответов на вопросы «Watson» побеждает двух чемпионов среди людей в американской телевизионной викторине «Jeopardy!» (раздел 1.4).

Рис. 1.4 Программный агент с взаимодействием с пользователем



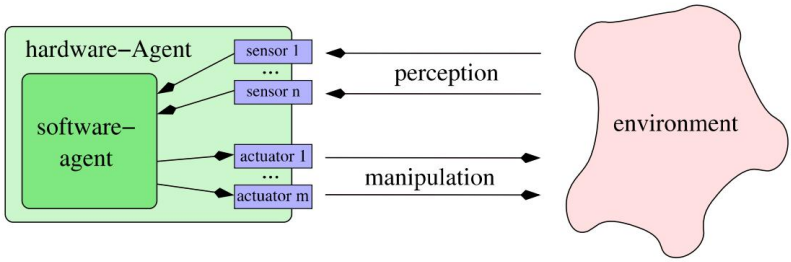


Рис. 1.5 Аппаратный агент

Пример 1.1. Спам-фильтр — это агент, который помещает входящие сообщения электронной почты в категории требуемых или нежелательных (спам) и удаляет любые нежелательные сообщения. Его цель как агента, основанного на целях, состоит в том, чтобы поместить все электронные письма в правильную категорию. В ходе этой не такой уж простой задачи агент может время от времени ошибаться. Поскольку его цель — правильно классифицировать все электронные письма, он попытается сделать как можно меньше ошибок. Однако это не всегда то, что имеет в виду пользователь. Сравним два следующих агента. Из 1000 электронных писем Агент 1 делает только 12 ошибок. Агент 2, с другой стороны, делает 38 ошибок с той же 1000 электронных писем. Стало быть, он хуже Агента 1? Ошибки обоих агентов более подробно показаны в следующей таблице, так называемой «матрице путаницы»:

Агент 1:

		правильный класс	
		хотел спам	1
спам-фильтр решает	разыскивается	189	
	спам	11	799

Агент 2:

		правильный класс	
		разыскиваемый	
спам-фильтр решает	спам разыскиваемый	200	
	38 спам	0	762

Machine Translated by Google

Агент 1 на самом деле делает меньше ошибок, чем агент 2, но эти несколько ошибок являются серьезными, поскольку пользователь теряет 11 потенциально важных электронных писем. Поскольку в данном случае имеется два типа ошибок разной серьезности, каждая ошибка должна быть взвешена с соответствующим коэффициентом стоимости (см. раздел 7.3.5 и упражнение 1.7 на стр. 14).

Сумма всех взвешенных ошибок дает общую стоимость, вызванную ошибочными решениями. Целью агента, основанного на затратах, является минимизация стоимости ошибочных решений в долгосрочной перспективе, то есть в среднем. В разд. 7.3 мы познакомимся с диагностической системой ЛЕКСМЕД как с примером агента, основанного на затратах.

Аналогично, цель агента, основанного на полезности, состоит в том, чтобы максимизировать полезность, полученную от правильных решений в долгосрочной перспективе, то есть в среднем. Сумма всех решений, взвешенных по соответствующим факторам полезности, дает общую полезность.

Особый интерес для ИИ представляют обучающие агенты, которые способны изменять себя на обучающих примерах или с помощью положительной или отрицательной обратной связи, так что средняя полезность их действий со временем растет (см. главу 8).

Как упоминалось в разд. 1.2.5, все чаще используются распределенные агенты , интеллект которых не локализован в одном агенте, а может быть виден только через сотрудничество многих агентов.

Дизайн агента, наряду с его целью, сильно ориентирован на его среду или, альтернативно, на его картину среды, которая сильно зависит от его сенсоров. Среда является наблюдаемой , если агент всегда знает полное состояние мира. В противном случае окружающая среда будет наблюдаться лишь частично.

Если действие всегда приводит к одному и тому же результату, то среда детерминирована.
В противном случае он недетерминирован. В дискретной среде происходит только конечное количество состояний и действий, тогда как в непрерывной среде может быть бесконечно много состояний или действий.

1.4 Системы, основанные на знаниях Агент — это

программа, реализующая отображение восприятий в действия. Для простых агентов такой взгляд на проблему достаточен. Для сложных приложений, в которых агент должен иметь возможность полагаться на большой объем информации и предназначен для выполнения сложной задачи, программирование агента может быть очень дорогостоящим и неясным, как действовать дальше. Здесь ИИ дает четкий путь, по которому нужно следовать, что значительно упростит работу.

Сначала мы отделяем знание от системы или программы, которая использует знание, например, для того, чтобы делать выводы, отвечать на вопросы или разрабатывать план.
Эта система называется механизмом вывода. Знания хранятся в базе знаний (КБ). Приобретение знаний в базе знаний обозначается как «Инженерия знаний» и основано на различных источниках знаний, таких как люди-эксперты, инженер по знаниям и базы данных. Системы активного обучения также могут приобретать знания путем активного исследования мира (см. главу 10). На рис. 1.6 на стр. 13 представлена общая архитектура систем, основанных на знаниях.

Движение к разделению знаний и умозаключений имеет несколько важных преимуществ. Разделение знаний и логического вывода может позволить реализовать системы логического вывода в значительной степени независимо от приложений. Например, это намного

Machine Translated by Google

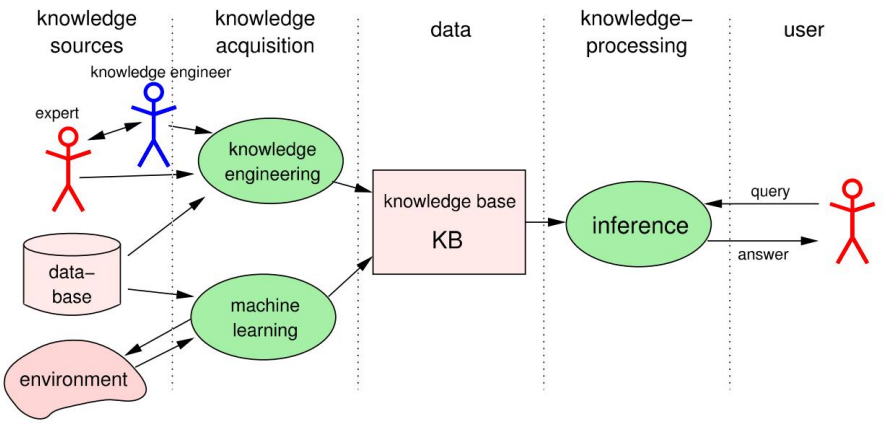


Рис. 1.6 Структура классической системы обработки знаний

проще заменить базу знаний медицинской экспертной системы, чем программировать совершенно новая система.

Благодаря отделению базы знаний от умозаключений знание может хранить декларативно. В базе знаний есть только описание знание, которое не зависит от используемой системы вывода. Без этого четкое разделение, знание и обработка шагов вывода будут переплетены, и любые изменения в знаниях обойдутся очень дорого.

Формальный язык как удобный интерфейс между человеком и машиной подходит к представлению знаний в базе знаний. В следующих главах мы познакомимся с целым рядом таких языков. Во-первых, в гл. 2 и 3 есть исчисление высказываний и логика предикатов первого порядка (PL1). Но также представлены и другие формализмы, такие как вероятностная логика, нечеткая логика или деревья решений. Начнем с исчисления высказываний и связанных с ним систем вывода. Основываясь на что мы представим логику предикатов, мощный язык, доступный для машин и очень важный для ИИ.

В качестве примера крупномасштабной системы, основанной на знаниях, мы хотим сослаться на программный агент «Ватсон». Разработан в IBM совместно с рядом университетов, Watson —это программа для ответов на вопросы, которую можно снабдить подсказками, представленными в естественном виде. язык. Он работает с базой знаний, содержащей четыре терабайта памяти на жестком диске, включая полный текст Википедии [FNA+09]. Watson был разработан в Проект IBM DeepQA, который описан в [Dee11] следующим образом:

> Проект DeepQA в IBM формирует грандиозную задачу в области компьютерных наук, которая направлена на иллюстрирует, как широкая и растущая доступность контента на естественном языке, а также интеграция и усовершенствование обработки естественного языка, информационного поиска, машинного обучения, представления знаний и рассуждений, а также массовых параллельных вычислений могут привести открытую технологию автоматического ответа на вопросы на новый уровень. точка, где они явно и последовательно конкурируют с случайными человеческими способностями.

В американской телевизионной викторине «Опасность!» В феврале 2011 года Уотсон победил двух чемпионов-людей Брэда Раттера и Кена Дженнингса в двух играх. матч по комбинированным очкам и выиграл приз в миллион долларов. Один из пар-

Machine Translated by Google

Особенной силой была его очень быстрая реакция на вопросы, в результате чего Watson часто нажимал на зуммер (используя соленоид) быстрее, чем его конкуренты-люди, а затем был в состоянии дать первый ответ на вопрос.

Высокая производительность и короткое время реакции Watson были обусловлены реализацией на 90 серверах IBM Power 750, каждый из которых содержит 32 процессора, в результате чего получается 2880 параллельных процессоров.

1,5 Упражнения

Упражнение 1.1 Протестируйте некоторых чат-ботов, доступных в Интернете. Начните, например, с www.hs-weingarten.de/~ertel/aibook в коллекции ссылок Turingtest/Chatterbots или с www.simonlaven.com или www.alicebot.org. Запишите начальный вопрос и измерьте время, необходимое для каждой из различных программ, пока вы не будете уверены, что это не человек.

Упражнение 1.2 На сайте www.pandorabots.com вы найдете сервер, на котором вы можете легко создать чат-бота с языком разметки AIML. В зависимости от уровня вашего интереса разработайте простого или сложного чат-бота или измените существующий.

Упражнение 1.3 Объясните причины непригодности теста Тьюринга в качестве определения «искусственного интеллекта» для практического ИИ.

Упражнение 1.4. Многие известные процессы логического вывода, процессы обучения и т. д. NP-полные или даже неразрешимые. Что это означает для ИИ?

Упражнение 1.5

(a) Почему детерминированный агент с памятью не является функцией от множества всех входов к множеству всех выходов в математическом смысле? (b) Как можно изменить агента с памятью или смоделировать его так, чтобы он стал эквивалентна функции, но не теряет свою память?

Упражнение 1.6. Пусть имеется агент с памятью, который может перемещаться в плоскости. От своих датчиков он получает через такт с регулярным интервалом t свое точное положение (x, y) в декартовых координатах. а) Приведите формулу, с помощью которой агент может вычислить свою скорость по текущим временам t и предыдущее измерение t t.

б) Как нужно изменить агент, чтобы он мог вычислить и свое ускорение? Дайте формулу здесь, а также.

Упражнение 1.7

(a) Определите для обоих агентов в Примере 1.1 на стр. 11 затраты, вызванные ошибками, и сравните результаты. Предположим, что удаление спама вручную стоит один цент, а получение удаленного письма или его потеря стоит один доллар.

(b) Определите для обоих агентов прибыль, созданную правильными классификациями, и сравните результаты. Предположим, что на каждое распознанное желаемое электронное письмо начисляется прибыль в один доллар, а на каждое правильно удаленное спам-письмо — прибыль в один цент.

Machine Translated by Google

Логика высказываний

2

В логике высказываний, как следует из названия, предложения связаны логическими операторами. Высказывание «на улице мокро» является пропозицией, как и «идет дождь». Эти два предложения можно соединить, чтобы сформировать новое предложение

если идет дождь, то улица мокрая.

Написано более официально

идет дождь на улице мокро.

Это обозначение имеет то преимущество, что элементарные предложения снова появляются в неизменном виде. Чтобы мы могли точно работать с логикой высказываний, мы начнем с определения множества всех формул логики высказываний.

2.1 Синтаксис

Определение 2.1. Пусть Op = {¬, , , , , (,)} — множество логических операторов, а Σ — множество символов. Множества Op, Σ и {t,f } попарно не пересекаются. Σ называется сигнатурой, а его элементы — переменными предложения. Набор формул пропозициональной логики теперь определяется рекурсивно: • t и f являются (атомарными) формулами. • Все переменные предложения, то есть все элементы из Σ, являются (атомарными) формулами.
лас.

• Если A и B — формулы, то ¬A, (A), A B, A B, A B, A B — также формулы.

Это элегантное рекурсивное определение множества всех формул позволяет нам генерировать бесконечно много формул. Например, если Σ = {A, B, C},

A B, A B C, A A, C B A, (¬A B) (¬C A)

являются формулами. (((A)) B) также является синтаксически правильной формулой.

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по компьютерным наукам,
DOI 10.1007/978-0-85729-299-5_2, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google

Определение 2.2 Мы читаем символы и операторы следующим образом:

т: «правда»		
ф: «ложь»		
¬A: «не A»		(отрицание)
A B: «A и B»		(соединение)
A B: «A или B»		(дизъюнкция)
A B: «если A, то B»		(импликация (также называемая материальной импликацией))
A B: «A тогда и только тогда, когда B»		(эквивалентность)

Определенные таким образом формулы представляют собой пока чисто синтаксические конструкции без значение. Нам все еще не хватает семантики.

2.2 Семантика

В логике высказываний есть два значения истинности: t для «истинно» и f для «ложно». Мы начнем с примера и спросим себя, верна ли формула A B. ответ таков: это зависит от того, верны ли переменные A и B. Например, если A означает «Сегодня идет дождь», а B означает «Сегодня холодно», и оба они верны, тогда A B верно. Если, однако, B представляет «Сегодня жарко» (и это неверно), то A B ложно.

Мы, очевидно, должны присвоить истинностные значения, отражающие состояние мира, в зависимости от обстоятельств. переменные положения. Поэтому мы определяем

Определение 2.3. Отображение I : Σ {w, f}, присваивающее истинностное значение каждая переменная предложения называется интерпретацией.

Поскольку каждая переменная высказывания может принимать два значения истинности, каждая формула логики высказываний с n различными переменными имеет 2n различных интерпретаций. Мы определить значения истинности для основных операций, показав все возможные интерпретации в таблице истинности (см. Таблицу 2.1 на стр. 17).

Пустая формула верна для всех интерпретаций. Для того, чтобы установить истину значение для сложных формул, мы также должны определить порядок операций для логических операторы. Если выражения заключены в круглые скобки, термин в круглых скобках оценивается

Machine Translated by Google

Таблица 2.1 Определение логические операторыпо истине стол

A B (A) ¬A A B	A B	A B	A B	B		
т т т ф у т				т	т	т
т ф т ф ф				т	ф	ф
ф у т о в ф у т о в			ф	т	т	ф
ф ф ф т			ф	ф	т	т

первый. Для формул без скобок приоритеты располагаются следующим образом, начиная с самой сильной привязкой: ¬, , , , .

Чтобы четко различать эквивалентность формул и синтаксическую эквивалентность мы определяем

Определение 2.4. Две формулы F и G называются семантически эквивалентными, если они принимают одно и то же значение истинности для всех интерпретаций. Мы пишем F G.

Семантическая эквивалентность служит, прежде всего, возможности использовать метаязык, который есть естественный язык, чтобы говорить об объектном языке, а именно о логике. Заявление «A B» означает, что две формулы A и B семантически эквивалентны, утверждение «A B», с другой стороны, является синтаксическим объектом формального языка пропозициональной логики.

В зависимости от того, на сколько интерпретаций верна формула, мы можем разделить формулы на следующие классы:

Определение 2.5 Формула называется

- Удовлетворительно, если оно верно хотя бы для одной интерпретации.
- Логически верный или просто верный, если он верен для всех интерпретаций. Истинные формулы также называют тавтологиями.
- Неудовлетворительно, если оно неверно для любой интерпретации.

Каждая интерпретация, удовлетворяющая формуле, называется моделью формулы.

Ясно, что отрицание всякой общезначимой формулы невыполнимо. Негация Формирование выполнимой, но не общепризнанной формулы F выполнимо.

Теперь мы можем создавать таблицы истинности для сложных формул, чтобы установить их правильность. значения истинности. Осуществим это непосредственно, используя эквивалентности формул которые важны на практике.

Machine Translated by Google

Теорема 2.1 . Операции , коммутативны и ассоциативны, а обычно справедливы следующие эквивалентности:

¬A B A B	(импликация)
A B ¬B ¬A	(противоположность)
(A B) (B A) (A B) ¬(A B) ¬A	(эквивалентность)
¬B ¬(A B) ¬A ¬B	(закон де Моргана)

A (B C) (A B) (A C) (дистрибутивный закон)

A (B C) (A B) (A C)

A ¬A ш	(тавтология)
A ¬A f	(противоречие)

A f A
A ш ш

A f f
A ш A

Доказательство Чтобы показать первую эквивалентность, мы вычисляем таблицу истинности для ¬A B и A B и убедитесь, что значения истинности для обеих формул одинаковы для всех интерпретаций. Таким образом, формулы эквивалентны, и поэтому все значения последнего столбец - это «т».

AB ¬A ¬A BA	B (¬A B)	(A B)	
ttf tt ff f ft t fft t		т	т
		ф	т
	т	т	т
		т	т

Доказательства остальных эквивалентностей аналогичны и рекомендуются в качестве упражнений. для читателя (упражнение 2.2 на стр. 29).

2.3 Системы доказательств

В ИИ мы заинтересованы в том, чтобы брать существующие знания и извлекать из них новые. знаний или ответов на вопросы. В пропозициональной логике это означает показать, что база знаний KB, то есть (возможно, обширная) формула пропозициональной логики — следует формула Q1 . Таким образом, мы сначала определим термин «вовлеченность».

1 Здесь Q означает запрос.

Machine Translated by Google

Определение 2.6 Формула KB влечет за собой формулу Q (или Q следует из KB), если каждая модель KB также является моделью Q. Мы пишем KB | B.

Другими словами, в каждой интерпретации, в которой KB верен, Q также верен. Говоря более кратко, всякий раз, когда KB истинно, Q также истинно. Поскольку для понятия следствия вводятся интерпретации переменных, мы имеем дело с семантическим понятием.

Каждая недействительная формула выбирает, так сказать, подмножество множества всех интерпретаций в качестве своей модели. Например, такие тавтологии, как A ¬A, не ограничивают число удовлетворительных интерпретаций, поскольку их предложение пусто. Таким образом, пустая формула верна во всех интерпретациях. Для каждой тавтологии T тогда | T Интуитивно это означает; что тавтологии всегда истинны, без ограничения интерпретаций формулой. Для краткости мы пишем | T . Теперь мы покажем важную связь между семантическим понятием следствия и синтаксической импликацией.

Теорема 2.2 (теорема дедукции)

A | B тогда и только тогда, когда | A Б.

Доказательство Обратите внимание на таблицу истинности для импликации:

A	B	A Б
т	т	т
т	ф	ф
ф	т	т
ф	ф	т

Произвольная импликация A B, очевидно, всегда верна, за исключением интерпретации A t, B f.

Предположим, что A | Б держит. Это означает, что для каждой интерпретации, которая делает A истинным, B также является истинным. Критическая вторая строка таблицы истинности даже не применяется в этом случае. Следовательно, A B истинно, что означает, что A B является тавтологией. Таким образом, было показано одно направление высказывания.

Теперь предположим, что выполняется A B. Таким образом, критическая вторая строка таблицы истинности равна также заблокирован. Каждая модель A тогда также является моделью B. Тогда A | Б держит.

Если мы хотим показать, что KB влечет за собой Q, мы также можем показать с помощью метода таблицы истинности, что KB Q является тавтологией. Таким образом, у нас есть наша первая система доказательств для логики высказываний, которую легко автоматизировать. Недостатком этого метода является очень большое время расчета в худшем случае. В частности, в худшем случае с n переменными предложения для всех 2 n интерпретаций переменных должна быть вычислена формула KB Q. Поэтому время вычислений растет экспоненциально

Machine Translated by Google

с количеством переменных. Следовательно, этот процесс непригоден для большого числа переменных, по крайней мере, в худшем случае.

Если формула KB влечет за собой формулу Q, то по теореме вывода KB Q есть тавтология. Поэтому отрицание ¬(KB Q) невыполнимо. У нас есть

$$\neg(KB \quad Q) \quad \neg(\neg KB \quad Q) \quad KB \quad \neg Q.$$

Следовательно, KB ¬Q также выполнимо. Сформулируем это простое, но важное следствие теоремы о дедукции как теорему.

Теорема 2.3 (Доказательство от противного) KB | Q тогда и только тогда, когда KB ¬Q невыполнимо.

Чтобы показать, что запрос Q следует из базы знаний KB, мы также можем добавить в базу знаний отрицательный запрос ¬Q и получить противоречие. Из-за эквивалентности A ¬A f из теоремы 2.1 на стр. 18 мы знаем, что противоречие неразрешимо. Таким образом, Q доказано. Эта процедура, которая часто используется в математике, также используется в различных автоматических выведениях доказательств, таких как исчисление разрешения и в обработке программ PROLOG.

Одним из способов избежать проверки всех интерпретаций с помощью метода таблицы истинности является синтаксическая обработка формул KB и Q путем применения правил вывода с целью их значительного упрощения, так что в конце мы можем сразу увидеть, что KB | Q. Мы называем этот синтаксический процесс выводом и пишем KB Q. Такие системы синтаксических доказательств называются исчислениями. Чтобы гарантировать, что исчисление не порождает ошибок, мы определяем два фундаментальных свойства исчисления.

Определение 2.7 Исчисление называется корректным, если каждое производное предложение следует семантически. То есть, если для формул KB и Q верно, что

$$\text{если } KB \quad Q, \text{ то } KB \mid B.$$

Исчисление называется полным, если из него можно вывести все семантические следствия. То есть для формул KB и Q справедливо следующее:

$$\text{если } KB \mid Q, \text{ затем } KB \quad Q.$$

Надежность исчисления гарантирует, что все производные формулы на самом деле являются семантическими следствиями базы знаний. Исчисление не дает никаких «ложных следствий». С другой стороны, полнота исчисления гарантирует, что исчисление ничего не упустит из виду. Полное исчисление всегда находит доказательство, если доказываемая формула следует из базы знаний. Если расчет правильный

Machine Translated by Google

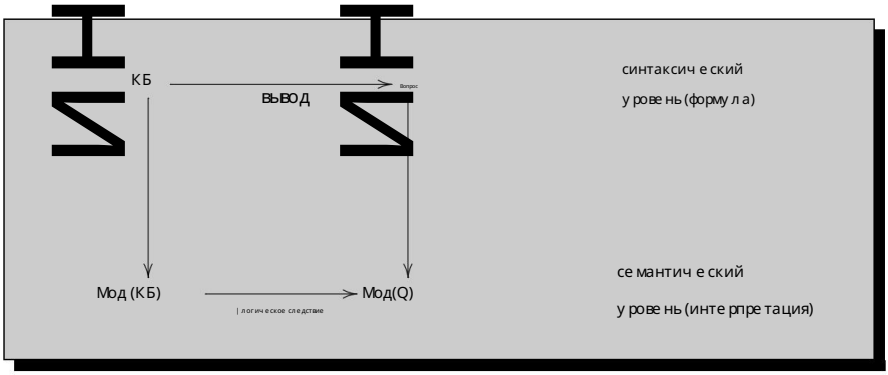


Рис. 2.1 Синтаксическая деривация и семантическая наследственность. Mod(X) представляет набор моделей формулыX

и полное, то синтаксическая деривация и семантическая наследственность являются двумя эквивалентными отношениями (см. рис. 2.1).

Чтобысделать автоматические системыпроверки максимально простыми, их обычно делают оперировать формулами в конъюнктивной нормальной форме.

Определение 2.8 Формула находится в конъюнктивной нормальной форме (КНФ) тогда и только тогда, когда она состоит из конъюнкции

$$K1 \quad K2 \quad \cdots \quad Km$$

статей. Предложение Ki состоит из дизъюнкции

$$(Li1 \quad Li2 \quad \cdots \quad Лини)$$

литералов. Наконец, литерал —это переменная (положительный литерал) или отрицательная переменная (отрицательный литерал).

Формула (A B ¬C) (A B) (¬B ¬C) находится в конъюнктивной нормальной форме. Конъюнктивная нормальная форма не накладывает ограничений на набор формул, потому что:

Теорема 2.4 Каждая формула логики высказываний может быть преобразована в эквивалентную конъюнктивнуюнормальнуюформу.

Machine Translated by Google

Пример 2.1. Приведем A B C D к конъюнктивной нормальной форме, используя эквивалентности из теоремы 2.1 на стр. 18:

A B C D
¬(A B) (C D) (импликация) (¬A ¬B) (C D) (де Морган)
(¬A (C D)) (¬B (C D)) (дистрибутивный закон) ((¬A C)
(¬A D)) ((¬B C) (¬B D)) (распределительный закон) (¬A C)
(¬A D) (¬B C) (¬B D) (ассоциативный закон)

Теперь нам не хватает только исчисления для синтаксического доказательства формул пропозициональной логики. Мы начнем с modus ponens, простого интуитивного правила вывода, которое, исходя из достоверности A и A B, позволяет вывести B. Формально мы запишем это как

$$\frac{A,\ A \quad B}{B}.$$

Это обозначение означает, что мы можем вывести формулу (ы) под линией из формул, разделенных запятыми над линией. Modus ponens, как правило, сам по себе, будучи звуковым, не является полным. Если мы добавим дополнительные правила, то сможем создать полное исчисление, которое, однако, мы здесь не рассматриваем. Вместо этого мы исследуем правило разрешения

$$\frac{A \quad B,\ \neg B \quad C}{A \quad C} \tag{2.1}$$

как альтернатива. Производное предложение называется резольвентой. Путем простого преобразования получаем эквивалентную форму

$$\frac{A \quad B,\ B \quad C}{A \quad C}.$$

Если мы установим A равным f, мы увидим, что правило разрешения является обобщением modus ponens. Правило разрешения одинаково применимо, если C отсутствует или если A и C отсутствуют. В последнем случае пустое предложение может быть получено из противоречия B ¬B (упражнение 2.7 на стр. 30).

2.4 Разрешение

Теперь мы снова обобщим правило разрешения, разрешив предложения с произвольным числом литералов. Для литералов A1,...,Am, B, C1,...,Cn общее правило разрешения имеет вид

$$\frac{(A1 \quad \cdots \quad Am \quad B),\ (\neg B \quad C1 \quad \cdots \quad Cn)}{(A1 \quad \cdots \quad Am \quad C1 \quad \cdots \quad Cn)}. \tag{2.2}$$

Machine Translated by Google

Мы называем литералы B и ¬B дополнительными. Правило разрешения удаляет пару дополнительных литералов из двух предложений и объединяет остальные литералы в новое предложение.

Чтобы доказать, что из базы знаний KB следует запрос Q, проведем доказательство от противного. Следуя теореме 2.3 на стр. 20, мы должны показать, что из KB ¬Q можно вывести противоречие. В формулах в конъюнктивной нормальной форме противоречие проявляется в виде двух дизъюнктов (A) и (¬A), которые приводят к пустому дизъюнкту как к их резольвенте. Следующая теорема гарантирует нам, что этот процесс действительно работает так, как нужно.

Для полноты исчисления нам потребуется небольшое дополнение, как это показано на следующем примере. Пусть формула (A A) задана как наша база знаний. Чтобы показать с помощью правила разрешения, что оттуда мы можем вывести (A A), мы должны показать, что пустое предложение может быть получено из (A A) (¬A ¬A). С одним только правилом разрешения это невозможно. С факторизацией, которая позволяет удалять копии литералов из предложений, эта проблема устраняется. В примере двойное применение факторизации приводит к (A) (¬A), а шаг разрешения — к пустому предложению

Теорема 2.5. *Исчисление резольвенты для доказательства невыполнимости формул в конъюнктивной нормальной форме правильное и полное.*

Поскольку задача резольвентного исчисления состоит в том, чтобы вывести противоречие из KB ¬Q, очень важно, чтобы база знаний KB была непротиворечивой:

Определение 2.9. Формула KB называется непротиворечивой, если из нее нельзя вывести противоречие, т. е. формула вида φ ¬φ.

В противном случае из KB можно получить что угодно (см. упражнение 2.8 на стр. 30). Этот верно не только для разрешения, но и для многих других исчислений.

Из исчислений для автоматического вывода разрешение играет исключительную роль. Таким образом, мы хотим работать с ним более тесно. В отличие от других исчислений, разрешение имеет только два правила вывода и работает с формулами в конъюнктивной нормальной форме.
Это упрощает его реализацию. Еще одно преимущество по сравнению со многими исчислениями заключается в уменьшении числа возможностей применения правил вывода на каждом шаге доказательства, благодаря чему сокращается пространство поиска и сокращается время вычислений.

В качестве примера мы начнем с простой логической головоломки, которая позволяет выполнять важные шаги. доказательство резолюции, которое будет показано.

Пример 2.2 Логическая задачка № 7 под названием Очаровательная английская семья из немецкой книги [Ber89] гласит (в переводе на английский):

Machine Translated by Google

Несмотря на семь долгих лет изучения английского языка с блестящим успехом, я должен признать, что, когда я слышу, как англичане говорят по-английски, я совершенно сбит с толку. Недавно, движимый благородными чувствами, я подобрал трех автостопщиков, отца, мать и дочь, которые, как я быстро понял, были англичанами и говорили только по-английски. В каждом из следующих предложений я колебался между двумя возможными интерпретациями. Мне сказали следующее (второе возможное значение в скобках): Отец: «Мы едем в Испанию (мы из Ньюкасла)».
Мать: «Мы не едем в Испанию и из Ньюкасла (мы остановились в Париже и не едем в Испанию)». Дочь: «Мы не из Ньюкасла (остановились в Париже)».
А как насчет этой очаровательной английской семьи?

Для решения такого рода задач мы действуем в три этапа: формализация, преобразование в нормальную форму и доказательство. Во многих случаях формализация является самым трудным шагом, потому что легко сделать ошибку или забыть мелкие детали. (Поэтому очень важны практические упражнения. См. упражнения 2.9–2.11.)

Здесь мы используем переменные S для «Мы едем в Испанию», N для «Мы из Ньюкасла», и P для «Мы остановились в Париже» и получаем как формализация трех пропозиций отца, матери и дочери.

$$(S \leftrightarrow N) \quad [(\neg S \rightarrow N) \quad (P \rightarrow \neg S)] \quad (\neg N \rightarrow P).$$

Вынесение на множитель ¬S в средней подформуле переводит формулу в КНФ за один шаг. Нумерация предложений с подстрочными индексами дает

$$KB \quad (S \leftrightarrow N)_1 \quad (\neg S)_2 \quad (P \rightarrow N)_3 \quad (\neg N \rightarrow P)_4.$$

Теперь приступим к доказательству разрешения, сначала еще без запроса Q. Выражение вида «Res(m, n): клауза k» означает, что клаузулы получаются резольвентой клаузулы m с клаузулой n и имеют номер k.

$$\text{Рез}(1, 2): (\text{Н})_5$$

$$\text{Рез}(3, 4): (\text{П})_6$$

$$\text{Рез}(1, 4): (S \rightarrow P)_7$$

Мы могли бы вывести пункт P также из Res(4, 5) или Res(2, 7). Каждый последующий шаг разрешения приведет к получению уже имеющихся предложений. Поскольку это не позволяет вывести пустое предложение, поэтому было показано, что база знаний непротиворечива. Итак, мы получили N и P. Чтобы показать, что ¬S выполняется, мы добавим предложение $(S)_8$ к набору предложений в качестве запроса с отрицанием. С шагом разрешения

$$\text{Рез}(2, 8): ()_9$$

доказательство завершено. Таким образом, выполняется ¬S ∧ N ∧ P. «Очаровательное английское семейство», видимо, родом из Ньюкасла, остановилось в Париже, но в Испании не собирается.

Пример 2.3. Логическая задача номер 28 из [Ber89], озаглавленная «Прыжок в высоту», гласит:

Machine Translated by Google

Три девушки тренируются в прыжках в высоту перед выпускным экзаменом по физкультуре. Планка установлена на 1,20 метра. «Держу пари, — говорит первая девушка второй, — что я справлюсь, если и только если ты это не сделаешь». Если бы вторая девушка сказала то же самое третьей, а та, в свою очередь, сказала бы то же самое первой, смогли бы все трое выиграть свои ставки?

Мы показываем через доказательство резолюцией, что не все трое могут выиграть свои ставки.

Формализация:

Первый прыжок девушки удался: A,	Ставка первой девушки: (A ¬B),
прыжок второй девушки удался: B, прыжок	ставка второй девушки: (B ¬C),
третьей девушки удался: C.	ставка третьей девушки: (C ¬A).

Утверждение: все трое не могут выиграть свои ставки:

$$Q \quad \neg((A \quad \neg B) \quad (B \quad \neg C) \quad (C \quad \neg A))$$

Теперь нужно показать с помощью разрешения, что ¬Q невыполнимо.

Превращение в CNF: Ставка первой девушки:

$$(A \quad \neg B) \quad (A \quad \neg B) \quad (\neg B \quad A) \quad (\neg A \quad \neg B) \quad (A \quad B)$$

Аналогичные преобразования претерпевают ставки двух других девушек, и мы получаем отрицательную гипотезу

$$\neg Q \quad (\neg A \quad \neg B)1 \quad (A \quad B)2 \quad (\neg B \quad \neg C)3 \quad (B \quad C)4 \quad (\neg C \quad \neg A)5$$
$$(C \quad A)6.$$

Оттуда мы получаем пустое предложение, используя разрешение:

Res(1, 6): (C ¬B)7
Рез(4, 7): (C)8
Res(2, 5): (B ¬C)9
Рез(3, 9): (¬C)10
Рез(8, 10): ()

Таким образом, утверждение доказано.

2.5 Оговорки Хорна

Предложение в конъюнктивной нормальной форме содержит положительные и отрицательные литералы и может быть представлено в виде

$$(\neg A1 \quad \cdots \quad \neg Am \quad B1 \quad \cdots \quad Bn)$$

Machine Translated by Google

с переменными A1,...,Am и B1,...,Bn. Это предложение может быть преобразовано в два простых шага в эквивалентную форму

$$A_1 \land \cdots \land A_m \rightarrow B_1 \lor \cdots \lor B_n.$$

Эта импликация содержит посылку, конъюнкцию переменных и заключение, дизъюнкцию переменных. Например: «Если будет хорошая погода и на земле будет снег, я пойду кататься на лыжах или буду работать». Это предложение такой формы. Получатель этого сообщения точно знает, что отправитель не собирается плавать.

Значительно более четким утверждением было бы: «Если будет хорошая погода и на земле будет снег, я пойду кататься на лыжах». Получатель теперь точно знает. Таким образом, мы называем предложения с не более чем одним положительным буквально определенным предложением. Преимущество этих положений состоит в том, что они допускают только один вывод, и, таким образом, их гораздо проще интерпретировать. Многие отношения могут быть описаны предложениями этого типа. Поэтому мы определяем

Определение 2.10 Предложения с не более чем одним положительным литералом формы

$$(\neg A_1 \land \cdots \land \neg A_m \land B) \text{ или } (\neg A_1 \land \cdots \land \neg A_m) \text{ или } B$$

или (эквивалентно)

$$A_1 \land \cdots \land A_m \land B \text{ или } A_1 \land \cdots \land A_m \text{ false или } B.$$

называются оговорками Хорна (по имени их изобретателя). Предложение с одним положительным литералом является фактом. В предложениях с отрицательными и одним положительным литералом положительный литерал называется головным.

Чтобы лучше понять представление предложений Хорна, читатель может вывести их из определений эквивалентностей, которые мы использовали в настоящее время (упражнение 2.12 на стр. 30).

С предложениями Хорна легче обращаться не только в повседневной жизни, но и в формальных рассуждениях, как мы можем видеть в следующем примере. Пусть база знаний состоит из следующих предложений (символ « », связывающий предложения, опущен здесь и в последующем тексте):

(хорошая_погода)1

(снегопад)2

(снегопад снег)3

(хорошая_погода снег катание на лыжах)4

Machine Translated by Google

Если мы теперь хотим знать, сохраняется ли лыжный спорт, это можно легко вывести. В качестве правила вывода здесь достаточно слегка обобщенного modus ponens:

$$\frac{A1 \quad \cdots \quad Am, A1 \quad \cdots \quad Am \quad B}{B}.$$

Доказательство «катания на лыжах» имеет следующий вид (MP(i1,...,ik) представляет собой применение modus ponens к предложениям i1 —ik :

ОД(2, 3): (снег)5 ОД(1, 5, 4):
(лыжи)6.

Используя modus ponens, мы получаем полное исчисление для формул, состоящих из предложений Хорна пропозициональной логики. Однако в случае больших баз знаний modus ponens может вывести много ненужных формул, если начать с неправильных предложений. Поэтому во многих случаях лучше использовать исчисление, которое начинается с запроса и работает в обратном направлении, пока не будут получены факты. Такие системы называются обратной цепочкой, в отличие от систем прямой цепочки, которые начинают с фактов и, наконец, выводят запрос, как в приведенном выше примере с modus ponens.

Для обратной цепочки предложений Хорна используется разрешение SLD. SLD расшифровывается как «линейное разрешение, управляемое правилом выбора для определенных предложений». В приведенном выше примере, дополненном отрицательным запросом (катание на лыжах f)

(хорошая_погода)1
(снегопад)2
(снегопад снег)3
(хорошая_погода снег катание на лыжах)4
(лыжи f)5

проводим разрешение SLD начиная с шагов разрешения, выте кающих из этого пункта

Res(5, 4): (хорошая_погода снег f)6
Res(6, 1): (снег f)7
Res(7, 3): (снегопад f)8
Рез(8, 2): ()

и получаем противоречие с пустым предложением. Здесь мы можем легко увидеть «линейное разрешение», что означает, что дальнейшая обработка всегда выполняется для текущего производного предложения. Это приводит к значительному сокращению пространства поиска. Кроме того, литералы текущего предложения всегда обрабатываются в фиксированном порядке (например, справа налево) («Управление правилом выбора»). Литералы текущего предложения называются

Machine Translated by Google

подцель. Литералы инвертированного запроса являются целями . Правило вывода для одного шага гласит

$$\frac{A_1 \quad \cdots \quad A_m \qquad B_1, B_1 \qquad B_2 \quad \cdots \quad B_n \qquad f}{A_1 \quad \cdots \quad A_m \qquad B_2 \quad \cdots \quad B_n \qquad f}.$$

Перед применением правила вывода должны быть доказаны $B_1, B_2, \ldots, B_n$ —текущие подцели. После применения B_1 заменяется новой подцелью $A_1 \quad \cdots \quad A_m$. Чтобы показать, что B_1 истинно, мы должны теперь показать, что $A_1 \quad \cdots \quad A_m$ истинны. Этот процесс продолжается до тех пор, пока список подцелей текущих предложений (так называемый стек целей) не станет пустым. При этом было обнаружено противоречие. Если для подцели $\neg B_i$ нет предложения с дополнительным литералом B_i в качестве заголовка предложения, доказательство завершается и противоречие не может быть найдено. Таким образом, запрос недоказуем.

Разрешение SLD играет важную роль на практике, поскольку программы на языке логического программирования PROLOG состоят из предикатных логических предложений Horn, и их обработка достигается посредством разрешения SLD (см. упражнение 2.13 на стр. 30 или гл. 5).

2.6 Вычислимость и сложность

Метод таблицы истинности, как простейшая система семантического доказательства для логики высказываний, представляет собой алгоритм, который может определить каждую модель любой формулы за конечное время. Таким образом, множества невыполнимых, выполнимых и правильных формул разрешимы. Время вычисления метода таблицы истинности для выполнимости в худшем случае растет экспоненциально с числом переменных n, поскольку таблица истинности имеет 2^n строки. Оптимизация, метод семантических деревьев, избегает просмотра переменных, которые не встречаются в предложениях, и, таким образом, во многих случаях экономит время вычислений, но в худшем случае оно также является экспоненциальным.

При разрешении в худшем случае количество производных предложений растет экспоненциально с количеством начальных предложений. Таким образом, чтобы выбрать между двумя процессами, мы можем использовать эмпирическое правило: в случае многих предложений с небольшим количеством переменных предпочтительнее использовать метод таблицы истинности, а в случае небольшого количества предложений с большим количеством переменных разрешение, вероятно, завершится быстрее.

Остается вопрос: может ли доказательство в пропозициональной логике идти быстрее? Есть ли лучшие алгоритмы? Ответ: наверное, нет. Ведь С. Кук, основоположник теории сложности, показал, что задача 3-SAT является NP-полной. 3-SAT —это набор всех формул КНФ, в предложениях которых ровно три литерала. Таким образом, очевидно, что полиномиального алгоритма для 3-SAT, вероятно, (по модулю проблемы P/NP) не существует, а значит, и общего алгоритма. Однако для предложений Хорна существует алгоритм, в котором время вычислений для проверки выполнимости растет только линейно по мере увеличения числа литералов в формуле.

Machine Translated by Google

2.7 Приме не ние и ограни ч е ния

Сре дства дока з ате л ьства те оре м для л оги ки вы ска з ы ваний явл я ются ч астью повсе дне вного набора инстру ме нтов раз работч ика в обл асти цифровы х те хнол оги й. Напри ме р, не которые из э тих з адач — прове рка цифровы х схе м и ге не рация те стовы х табл иц для те стирования микропроце ссоров в произ водстве . Спе циал ьные систе мы дока з ате л ьств, работающие с двои ч ными диаграммами ре ше ний (BDD) (), та кже испол ь з у ются в ка ч е стве стру кту ры данны х для обработки форму л пропози ци онал ьной л оги ки.

В ИИ л оги ка вы ска з ы ваний испол ь з у е тся в простых прил оже ниях. Напри ме р, простые э кспе ртные систе мы, бе з у сл овно, мог у т работать с пропози ци онал ьной л оги кой. Одна ко все пе ре ме нные до л жны быть дискре тными, име ть л ишь не ско л ько з на ч е ний, и ме ж ду пе ре ме нными не мо ж е т быть ника ки х вз аимосвя з е й. Сл ожные л оги ч е ские свя з и мог у т быть вы ра ж е ны бол е е э л е гантно с помощью л оги ки пре дикатов.

Ве роятностная л оги ка —э то оч е нь инте ре сное и акту а л ьное соч е тание пропози ци онал ьной л оги ки и ве роятностны х вы н исл е ний, которое поз во л я е т моде л ировать не опре де л е нное з нание . Он подробно рассматри ва е тся в гл . 7. В э той гл аве та кже обсу ж да е тся не ч е тка я л оги ка, допу ска ющая бе сконе ч ное мно ж е ство з на ч е ний истинности.

2.8 У пра ж не ния

У пра ж не ние 2.1. Приве дите гра мматику формы Бэ ку са–Нау ра для синтаксиса л оги ка.

У пра ж не ние 2.2. Пока ж ите , ч то сл е ду ющие форму л ы явл я ются та вто л оги ями:
(a) ¬(A B) ¬A ¬B (b) A B
¬B ¬A (c) ((A B) (B A))
(A B) (г) (A B) (¬B C) (A C)

У пра ж не ние 2.3 Пре обра з у йте сл е ду ющие форму л ы в конъюнкти вну юнорма л ьну юформу : (a) A
B (b) A B
A B (c) A (A B) B

У пра ж не ние 2.4. Прове рьте сл е ду ющие у тве рж де ния на вы пол нимость ил и досто ве рность.
(a) (play_lottery six_right) побе дите л ь (b)
(play_lottery six_right (six_right win)) win (c) ¬(¬gas_in_tank
(gas_in_tank ¬car_starts) ¬car_starts)

У пра ж не ние 2.5 Испол ь з у я вы бранны й ва ми я з ы к програ ммирования, з апрограммиру йте сре дство дока з ате л ьства те оре м для л оги ки вы ска з ы ваний, испол ь з у я ме тод табл ицы истинности для форму л в конъюнкти вной нормал ьной форме . Чтобы из бе ж ать доро г остоя ще й прове рки синтаксиса форму л , вы мо ж е те пре дста в л ять пре дл оже ния в виде спи ско в ил и набо ро в л ите ра л ов, а форму л ы —в виде спи ско в ил и набо ро в пре дл оже ний. Програ мма до л жна у ка з ать, явл я ется л и форму л а не вы пол нимой, вы пол нимой ил и истинной, и вы ве сти ко л и ч е ство раз л и ч ны х инте рпре таций и моде л е й.

Machine Translated by Google

Упражнение

2.6 (a) Покажите, что modus ponens является допустимым правилом вывода, показав, что A (A B) | B. (b)
Покажите, что правило разрешения (2.1) является допустимым правилом вывода.

Упражнение 2.7. Покажите, применяя правило разрешения, что в конъюнктивном нормальном форме пустое предложение эквивалентно ложному утверждению.

Упражнение 2.8. Покажите, что с помощью разрешения можно «вывести» любое произвольное предложение из базы знаний, содержащей противоречие.

Упражнение 2.9 Формализуйте следующие логические функции с помощью логических операторов и покажите, что ваша формула верна. Представьте результат в CNF. (a) Операция XOR (исключающее или) между двумя переменными. (b) Утверждение, по крайней мере, две из трех переменных A, B, C верны.

Упражнение 2.10 Решите следующий случай с помощью резолютивного доказательства: «Если у преступника был сообщник, то он приехал на машине. У преступника не было сообщника и не было ключа, или у него был ключ и сообщник. Ключ был у преступника. Преступник приехал на машине или нет?

Упражнение 2.11. Покажите с помощью разрешения, что формула
из (a) упражнения 2.2(d) является
тавтологией. (b) Упражнение 2.4(c) невыполнимо.

Упражнение 2.12. Докажите следующие эквивалентности, важные для работы с предложениями Горна: (a) (¬A1
⋯ ¬Am B) A1 ⋯ Am B (b) (¬A1 ⋯ ¬Am) A1
⋯ Am f (c) A w A

Упражнение 2.13. Покажите с помощью разрешения SLD, что следующий набор предложений Хорна неудовлетворим.

(A)1	(Д)4	(A D G)7
(БИ2	(E)5	(C F E H)8
(C)3	(A B C F)6	(H f)9

Упражнение 2.14 В разд. 2.6 говорится: «Таким образом, очевидно, что, вероятно, (по модулю проблемы P/NP) не существует полиномиального алгоритма для 3-SAT, а значит, вероятно, и не общего». Обоснуйте «вероятно» в этом предложении.

Machine Translated by Google

Логика предикатов первого порядка

3

Многие практически важные проблемы не могут быть или могут быть очень неудобно сформулированы на языке пропозициональной логики, как мы можем легко понять в следующем примере. Заявление

«Робот 7 находится в позиции xy (35, 79)»

на самом деле может быть не посредственно использована как пропозициональная логическая переменная

«Robot_7_is_situated_at_xy_position_(35, 79)»

для рассуждений с пропозициональной логикой, но рассуждать с такого рода пропозиции очень неудобно. Предположим, что 100 таких роботов могут остановиться в любом месте на сетке 100 × 100 точек. Для описания каждой позиции каждого робота нам потребуется 100 · 100 · 100 = 1 000 000 = 106 различных переменных. Определение отношений между объектами (в данном случае роботами) становится по-настоящему трудным. Отношение

«Робот А находится справа от робота Б».

семантически не более чем набор пар. Из 10 000 возможных пар координат химеется (99 · 98)/ 2 = 4851 упорядоченная пара. Вместе со всеми 10 000 комбинаций возможных значений для обоих роботов имеется (100 · 99) = 9900 формул вида

Robot_7_is_to_the_right_of_robot_12
Robot_7_is_situated_at_xy_position_(35, 79)
Robot_12_is_situated_at_xy_position_(10, 93) ···

определяющие эти соотношения, каждое из которых с $2 \cdot 0{,}485 = 0{,}485 \cdot 108$ вариантов (104) в правой части. В логике предикатов первого порядка мы можем определить для этого предикат Position(number, xPosition, yPosition). Вышеупомянутое отношение больше не должно перечисляться как огромное количество пар, а описывается абстрактно с помощью правила вида

$$u \quad v \text{ is_farther_right}(u, v) \qquad xu \quad yu \quad xv \quad yv \text{ position}(u, xu, yu) \quad \text{position}(v, xv, yv) \quad xu > xv,$$

Где u читается как «для каждого u», а v как «существует v».

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по компьютерным наукам,
DOI 10.1007/978-0-85729-299-5_3, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google

В этой главе мы определим синтаксис и семантику логики предикатов первого порядка (PL1), покажем, что многие приложения могут быть смоделированы с помощью этого языка, и покажем, что для этого языка существует полное и надежное исчисление.

3.1 Синтаксис

Сначала мы закрепляем синтаксическую структуру терминов.

Определение 3.1. Пусть V —набор переменных, K —набор констант и F —набор функциональных символов. Множества VK и F попарно не пересекаются. Мы определяем множество термов рекурсивно: • Все переменные и константы являются (атомарными) термами. • Если t1,...,tn термы, а f n-местный функциональный символ, то f (t1,...,tn) тоже термин.

Некоторыми примерами терминов являются f (sin(ln(3)), exp(x)) и g(g(g(x))). Быть способным устанавливаем логические связи между терминами, строим формулы из терминов.

Определение 3.2. Пусть P —множество предикатных символов. Формулы логики предикатов строятся следующим образом: • Если t1,...,tn —термы, а p —n-местный предикатный символ, то p(t1,...,tn) — (атомарная) формула. • Если A и B —формулы, то ¬A, (A), A B, A B, A B, A B —также формулы.

- Если x —переменная, а A —формула, то x A и x A также являются формулами. —квантор всеобщности, а —квантор существования.
- p(t1,...,tn) и ¬p(t1,...,tn) называются литералами. • Формулы, в которых каждая переменная находится в области действия квантификатора, называются предложениями первого порядка или закрытыми формулами. Переменные, которые не входят в область действия квантификатора, называются свободными переменными. • Определения 2.8 (КНФ) и 2.10 (предложения Хорна) верны для формул литералов логики предикатов аналогично.

В Таблице 3.1 на стр. 33 приведены несколько примеров формул PL1. с их интуитивной интерпретацией.

Machine Translated by Google

Таблица 3.1 Примеры формул в логике предикатов первого порядка. Обратите внимание, что мать здесь функциональный символ

Формула	Описание
x лягушка(x) зеленая(x)	Все лягушки зеленые
x лягушка(x) коричневая(x)	Все коричневые лягушки большие
большая(x) нравится(x, торт)	Все нравится торт
¬ нравится(x, торт)	Не все нравится торт
¬ нравится(x, торт)	Никто не любит торт
x нравится(y,	Есть то, что нравится всем
x) x нравится(x,	Есть тот, кому все нравится
y) x нравится(y,	Все кому-то нравится
x) x нравится(x,	Все что-то нравится
y) x клиент(x) лайков(боб ,x)	Боб любит каждого клиента
x клиент(x) нравится(x, bob)	Есть клиент, который нравится Бобу
x пекарь(x) клиент(y) mag(x, y)	Есть пекарь, которому нравятся все его клиенты
x старше (мать(x), x) x	Каждая мать старше своего ребенка
старше (мать(мать(x)), x)	Каждая бабушка старше ее ребенок дочери
x y z rel(x, y) rel(y, z) rel(x, z)	rel - транзитивное отношение

3.2 Семантика

В пропозициональной логике каждой переменной непосредственно присваивается значение истинности интерпретацией. В логике предикатов значение формул рекурсивно определяется по построение формулы, в котором мы сначала назначаем константы, переменные и функцию символы к объектам реального мира.

Определение 3.3 Интерпретация I определяется как

- Отображение множества констант и переменных K V в множество W названия предметов в мире.
- Отображение набора функциональных символов на набор функций в мир. Каждому n-местному функциональному символу назначается n-местная функция.
- Отображение набора предикатных символов на набор отношений в мир. Каждому n-местному символу-предикату присваивается n-местное отношение.

Пример 3.1. Пусть c1, c2, c3 —константы, «плюс» —двузначный функциональный символ и «gr» —двузначный предикатный символ. Правда формулы

F гр(плюс (c1, c3), c2)

Machine Translated by Google

зависит от интерпретации I. Сначала выберем следующую очевидную интерпретацию констант, функции и предикатов в натуральных числах:

I1: c1 1, c2 2, c3 3, плюс +, гр > .

Таким образом, формула преобразуется в

1 + 3 > 2 или после оценки 4 > 2.

Отношение «больше чем» на множестве {1, 2, 3, 4} — это множество всех пар (x, y) чисел с x>y, что означает множество G = {(4, 3), (4, 2), (4, 1), (3, 2), (3, 1), (2, 1)}. Поскольку (4, 2) G, формула F верна при интерпретации I1. Однако, если мы выбираем интерпретацию

I2: c1 2, c2 3, c3 1, плюс , gr >,

мы получаем

2 - 1 > 3 или 1 > 3.

Пара (1, 3) не принадлежит G. Формула F ложна при интерпретации I2. Очевидно, истинность формулы в PL1 зависит от интерпретации. Теперь, после этого предварительного просмотра, мы определяем истину .

Определение

3.4 • Атомарная формула p(t1,...,tn) истинна (или верна) при интерпретации I , если после интерпретации и вычисления всех термов t1,...,tn и интерпретации предиката p через n-местное отношение r, верно, что

(I(t1), ... ,I(tn)) r.

- Истинность бесквантерных формул следует из истинности атомарных формул — как и в исчислении высказываний — через семантику логических операторов, определенных в таблице 2.1 на стр. 17. • Формула x F истинна при интерпретации I точно тогда, когда она истинно при произвольном изменении интерпретации переменной x (и только для x)

- Формула x F истинна при интерпретации I ровно тогда, когда существует интерпретация для x, которая делает формулу истинной.

Определения семантической эквивалентности формул для понятий выполнимый, истинный, невыполнимый и модельный вместе с семантическим следствием (определения 2.4, 2.5, 2.6) переносятся без изменений из исчисления высказываний в логику предикатов.

Machine Translated by Google

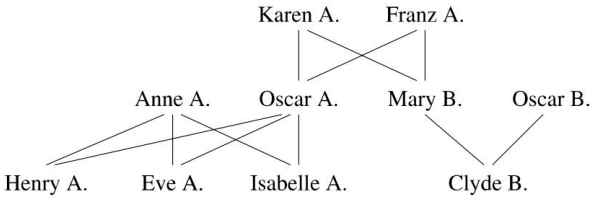


Рис. 3.1 Генеалогическое дерево. Ребра, идущие от Клайда Б. вверх к Мэри Б. и Оскару Б., представляют элемент (Клайд Б., Мэри Б., Оскар Б.) как дочернюю связь

Теорема 3.1 Теоремы 2.2 (теорема о дедукции) и 2.3 (доказательство от противного) верны аналогично для PL1.

Пример 3.2 Генеалогическое дерево, представленное на рис. 3.1 , графически представляет (на семантическом уровне) отношение

Ребенок = {(Оскар А., Карен А., Фрэнк А.), (Мэри Б., Карен А., Фрэнк А.), (Генри А., Энн А., Оскар А.),
(Ева А., Энн А., Оскар А.), (Изабель А., Энн А., Оскар А.), (Клайд Б., Мэри Б., Оскар Б.)}

Например, тройка (Оскар А., Карен А., Фрэнк А.) обозначает предложение «Оскар А. —ребенок Карен А. и Фрэнка А.». Из именмы читаем одноместное отношение

Женщина = {Карен А., Энн А., Мэри Б., Ева А., Изабель А.}

женщин. Теперь мы хотим установить формулы для семейных отношений. Сначала мы определяем трехместный предикат child(x, y, z) с семантикой

I(ребенок(x, y, z)) = w (I(x),I(y),I(z)) Kind.

При интерпретации I(oscar) = Oscar A., I(eve) = Eve A., I(anne) = Anne A. также верно, что child(eve, anne, oscar). Чтобы ребенок (ева, оскар, энн) был правдой, нам нужно, чтобы

x y z дочернийэлемент(x, y, z) дочернийэлемент(x, z, y),

симметричность дочернего предиката в последних двух аргументах. Для дальнейших определений мы обращаемся к упражнению 3.1 на стр. 54 и рекурсивно определяем потомок предиката.
как

x y потомок(x, y) z потомок(x, y, z)
(u v потомок(x, u, v) потомок(u, y)).

Теперь строим небольшую базу знаний с правилами и фактами. Позволять

Machine Translated by Google

КБ женщина (карен) женщина (энн) женщина (мэри)
женщина (Ева) женщина (Изабель)
ребенок (Оскар, Карен, Франц) ребенок (Мэри, Карен, Франц)
ребенок (Ева, Энн, Оскар) ребенок (Генри, Энн, Оскар)
ребенок (Изабель, Энн, Оскар) ребенок (Клайд, Мэри, Оскар)
(x y z дочерний(x, y, z) дочерний(x, z, y))
(x y потомок(x, y) z потомок(x, y, z)
(u v дочерний элемент(x, u, v) потомок(u, y))).

Теперь мы можем спросить, например, выводимы ли предложения ребенок (Ева, Оскар, Анна) или потомок (Ева, Франц) . Для этого нам потребуется вычисление .

3.2.1 Равенство

Чтобы иметь возможность сравнивать термины, равенство является очень важным отношением в логике предикатов. Равенство термов в математике —это отношение эквивалентности, то есть оно рефлексивно, симметрично и транзитивно. Если мы хотим использовать равенство в формулах, мы должны либо включить эти три атрибута в качестве аксиом в нашу базу знаний, либо мы должны интегрировать равенство в исчисление . Мы пойдем по простому пути и определим предикат «=», который, отклоняясь от определения 3.2 на стр. 32, записывается с использованием инфиксной записи, как это принято в математике . (Уравнение x = y, конечно, также можно записать в виде eq(x, y).) Таким образом, аксиомы равенства имеют вид

$$\begin{aligned} &x\ x = x && \text{(рефлексивность)}\\ &x\ y\ x = y \Rightarrow y = x && \text{(симметрия)}\\ &x\ y\ z\ x = y \wedge y = z \Rightarrow x = z && \text{(транзитивность).} \end{aligned} \tag{3.1}$$

Чтобы гарантировать уникальность функций, мы дополнительно требуем

$$x\ y\ x = y \Rightarrow f(x) = f(y) \quad \text{(аксиома подстановки)} \tag{3.2}$$

для каждого функционального символа. Аналогично требуем для всех x предикатных символов

$$x\ y\ x = y \wedge p(x) \Rightarrow p(y) \quad \text{(аксиома подстановки).} \tag{3.3}$$

Мы формулируем другие математические отношения, такие как отношение «<», аналогичным образом (упражнение 3.4 на стр. 55).

Часто переменная должна быть заменена термином. Чтобы сделать это правильно и описать его просто, мы даем следующее определение .

Machine Translated by Google

Определение 3.5 Обозначим через $\phi[x/t]$ формулу, которая получается при замене каждого свободного вхождения переменной x в ϕ на член t. Таким образом, мы не допускаем никаких переменных в терме t, которые квантифицируются в ϕ. В этих случаях переменные должны быть переименованы, чтобы гарантировать это.

Пример 3.3. Если в формуле $\forall x\, x = y$ заменить свободную переменную членом $x + 1$, то получится $\forall x\, x = x + 1$. При правильной подстановке получим формулу $\forall x\, x = y + 1$, которая имеет совсем другую семантику.

3.3 Кванторы и нормальные формы

Согласно определению 3.4 на стр. 34, формула $\exists x p(x)$ истинна тогда и только тогда, когда она истинна для всех интерпретаций переменной x. Вместо квантора можно было бы написать $p(a1) \vee \cdots \vee p(an)$ для всех констант $a1 \cdots an$ в K. Для $\forall x p(x)$ можно было бы написать $p(a1) \wedge \cdots \wedge$ кастрюля). Отсюда по закону де Моргана следует, что

$$\forall x \phi \quad \neg\, \exists x \neg \phi .$$

Благодаря этой эквивалентности кванторы всеобщего и существования взаимозаменяемы.

Пример 3.4 Предложение «Каждый хочет быть любимым» равносильно высказыванию «Никто не хочет быть любимым».

Кванторы являются важным компонентом выразительной силы логики предикатов. Однако они мешают автоматическому выводу в ИИ, поскольку усложняют структуру формул и увеличивают количество применимых правил вывода на каждом этапе доказательства. Поэтому наша следующая цель — найти для каждой формулы логики предикатов эквивалентную формулу в стандартизированной нормальной форме с как можно меньшим числом кванторов. В качестве первого шага мы вносим кванторы всеобщности в начало формулы и тем самым определяем

Определение 3.6. Формула логики предикатов ϕ находится в предваренной нормальной форме, если выполняется равенство $\phi = Q1x1 \cdots Qnxn\, \psi$. • ψ — бесквантор ная формула. • $Qi \in \{\forall, \exists\}$ для $i = 1,\ldots,n$.

Machine Translated by Google

Следует проявлять осторожность, если количественная переменная выходит за рамки своего количественного анализа.
тифицировать, как, например, x в

$$\forall x p(x) \quad \exists x q(x).$$

Здесь одну из двух переменных нужно переименовать, а в

$$\forall x p(x) \quad \exists y q(y)$$

квантор легко вынести вперед, и мы получаем на выходе эквивалентную формулу

$$\forall x \exists y p(x) \quad q(y).$$

Если, однако, мы отнимем правильно поставить квантификатор впереди

$$(\forall x p(x)) \quad \exists y q(y) \tag{3.4}$$

сначала запишем формулу в эквивалентной форме

$$\neg(\forall x p(x)) \quad \exists y q(y).$$

Первый универсальный квантор теперь превращается в

$$(\exists x \neg p(x)) \quad \exists y q(y)$$

и теперь два квантификатора, наконец, могут быть выдвинуты вперед к

$$\exists x \exists y \neg p(x) \quad q(y),$$

что эквивалентно

$$\exists x \exists y p(x) \quad q(y).$$

Мы видим, что в (3.4) мы не можем просто выдвинуть оба квантора вперед. Скорее, мы должны сначала устранить импликации, чтобы не было отрицаний на кванторах . В общем случае считается, что мы можем извлекать кванторы только в том случае, если отрицания существуют только непосредственно в атомарных подформулах .

Пример 3.5. Как известно из анализа, сходимость ряда $(a_n)_{n \in \mathbb{N}}$ к пределу a определяется выражением

$$\forall \varepsilon > 0 \quad \exists n_0 \in \mathbb{N} \quad \forall n > n_0 \, |a_n - a| < \varepsilon.$$

С функцией abs(x) для $|x|$, a(n) для a_n, minus(x, y) для $x - y$ и предикатами el(x, y) для $x = y$, gr(x, y) для $x > y$ формула выглядит так

Machine Translated by Google

ε (gr(ε, 0) n0 (el(n0,N) n (gr(n, n0) gr(ε, abs(минус(a(n), a)))))) . (3.5)

Этоявно не в пренексной нормальной форме. Поскольку переменные внутренних кванторов n0 и n не встречаютсяслева от соответствующих кванторов, никакие переменные не должны переименовываться.Далее устраняемимпликации и получаем

ε (¬gr(ε, 0) n0 (¬el(n0,N) n (¬gr(n, n0) gr(ε, abs(минус(a(n), a)))))).

Поскольку каждое отрицание стоит перед атомарной формулой, мывыдвигаем квантификаторы вперед, удаляемлишние скобки и с помощью

ε n0 n (¬gr(ε, 0) ¬el(n0,N) ¬gr(n, n0) gr(ε, abs(минус(a(n), a))))

оностановитсяквантифицированнымпредложениемв конъюнктивной нормальной форме.

Преобразованнаяформула эквивалентна в ыкодной формуле. Тот факт, что это всегда возможна трансформациягарантируется

Теорема 3.2 Каждаяформула логики предикатов может быть преобразована в эквивалентную формулу в пренексной нормальной форме.

Кроме того, мыможем исключить все кванторысуществования.Однако формула, полученнаяв результате так называемой сколемизации , уже не эквивалентна формуле в ыкода. Однако его в ыполнимость остаетсянеизменной. В многих случаях , особенно когда нужнопоказать не в ыполнимость KB ¬Q, этого достаточно. Следующаяформула в пренексной нормальной форме теперь будет сколемизирована:

x1 x2 y1 x3 y2 p(f (x1), x2, y1) q(y1, x3, y2).

Поскольку переменнаяy1, по-в идимому, зависит от x1 и x2, каждое в хождение y1 заменяется функцией Скулема g(x1, x2). В ажно, что g — это новый функциональный символ, который еще не появилсяв формуле. Мыполучаем

x1 x2 x3 y2 p(f (x1), x2, g(x1, x2)) q(g(x1, x2), x3, y2)

и аналогично заменить y2 на h(x1, x2, x3), что приводит к

x1 x2 x3 p(f (x1), x2, g(x1, x2)) q(g(x1, x2), x3, h(x1, x2, x3)).

Поскольку теперь все переменные являютсяуниверсальными кванторами, универсальные кванторы можно опустить, что приведет к

p(f (x1), x2, g(x1, x2)) q(g(x1, x2), x3, h(x1, x2, x3)).

Machine Translated by Google

ПРЕОБРАЗОВАНИЕ НОРМАЛЬНОЙ ФОРМЫ(формула): 1.
Преобразование в преднексную нормальную
форму: Преобразование в конъюнктивную нормальную форму (теорема 2.1):
Устранение эквивалентностей.
Устранение последствий.
Повторное применение закона де Моргана и дистрибутивного закона.
Переименование переменных при необходимости.
Вынесение на множители универсальных кванторов.
2. Скулемизация
Замена экзистенциально кванторных переменных новыми функциями Скулема.
Удаление результирующих универсальных кванторов.

Рис. 3.2 Преобразование формул логики предикатов в нормальную форму

Теперь мы можем исключить квантор существования (а тем самым и квантор универсальности) в (3.5) на стр. 39 , введя функцию Скулема n0(ε). Таким образом, сколемизированный пренекс и конъюнктивная нормальная форма (3.5) на стр. 39 читаются

¬gr(ε, 0) ¬el(n0(ε),N) ¬gr(n, n0(ε)) gr(ε, abs(минус(a(n), a))).

Отбрасывая переменную n0, функция Скулема может получить имя n0.

При сколемизации формулы в предваренной нормальной форме все кванторы существования удаляются снаружи внутрь, где формула вида x1 ... xn у ф заменяется на x1 ... xn ф[y/f (x1,...,xn)], в течение которого f может не входить в ф.
Если квантор существования находится далеко снаружи, например, в ур(у), то у должен быть заменен константой (то есть нулевым функциональным символом).

Процедура преобразования формулы в конъюнктивную нормальную форму обобщена в псевдокоде, представленном на рис. 3.2. Скулемизация имеет полиномиальное время выполнения по количеству литералов. При преобразовании в нормальную форму количество литералов в нормальной форме может расти экспоненциально, что может привести к экспоненциальному времени вычислений и экспоненциальному использованию памяти. Причиной этого является многократное применение распределительного закона. Фактическая проблема, возникающая из-за большого количества предложений, заключается в комбинаторном взрыве пространства поиска для последующего доказательства разрешения. Однако существует оптимизированный алгоритм преобразования, который порождает только полиномиальное число литералов [Ede91].

3.4 Доказательства исчисления

Для рассуждений в логике предикатов были разработаны различные исчисления естественного мышления, такие как исчисление Генцена или исчисление секвенций. Как следует из названия, эти исчисления предназначены для применения людьми, поскольку правила вывода более

Machine Translated by Google

Таблица 3.2. Простое доказательство с использованием modus ponens и исключения квантора

В Б:	1	ребенок (Ева, Энн, Оскар)
В Б:	2	x y z дочерний(x, y, z) дочерний(x, z, y)
E(2) : x/eve,y/anne, z/oscar	3	ребенок(Ева, Энн, Оскар) ребенок(Ева, Оскар, Энн)
МП(1, 3)	4	ребенок (Ева, Оскар, Энн)

или менее интуитивным, а исчисления работают с произвольными формулами PL1. В следующем разделе мы в первую очередь сосредоточимся на исчислении резолюций, которое на практике наиболее важное эффективное, автоматизируемое исчисление для формул в конъюнктивном нормальном форма. Здесь, используя пример 3.2 на стр. 35 , мы дадим очень небольшое «естественное» доказательство. Используем правило вывода

$$\frac{A,\ A \quad B}{B}\ \text{(modus ponens, MP)} \quad \text{и} \quad \frac{\forall x\, A}{A[x/t]}\ (\forall\text{-устранение, } \forall E).$$

Modus ponens уже известен из логики высказываний. При устранении универсальные кванторы, необходимо иметь в виду, что квантор на переменная x должна быть заменяется основным термином t, что означает термин, который не содержит переменных . Доказательство представлен ребенок(ева, оскар, энн) из соответствующим образом сокращенной базы знаний в таблице 3.2.

Две формулы сокращенной базы знаний перечислены в строках 1 и 2. В в строке 3 исключаются кванторы в свободности из строки 2, а в строке 4 утверждается получен с помощью modus ponens.

Исчисление, состоящее из двух заданных правил вывода, не является полным. Однако его можно расширить до полной процедуры путем добавления дополнительных выводов . правила. Этот нетривиальный факт имеет фундаментальное значение для математики и ИИ. Австрийский логик Курт Гёдель доказал в 1931 г., что [Gö d31a]

> Теорема 3.3 (теорема Гёделя о полноте). Логика предикатов первого порядка – это полный. То есть существует исчисление, с помощью которого каждое предложение, являющееся следствием базы знаний KB может быть доказано. Если KB | ф, то выполняется что KB ф.

Следовательно, каждое истинное суждение в логике предикатов первого порядка доказуемо. Но это обратное тоже верно? Действительно ли все, что мы можем вывести синтаксически, истинно? ответ «да»:

> Теорема 3.4 (Корректность) Существуют исчисления, с помощью которых можно доказать только истинные предложения. То есть, если выполняется KB ф , то KB | ф.

Machine Translated by Google

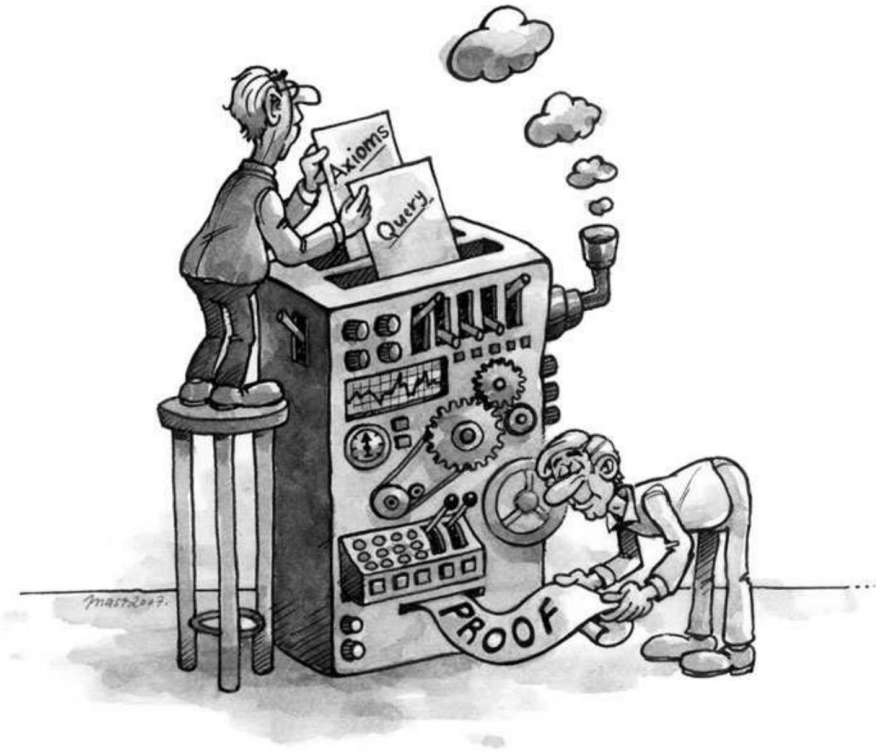


Рис. 3.3 Универсальная логическая машина

На самом деле почти все известные исчисления верны. Ведь работать смысла мало с неверными методами доказательства. Таким образом, доказуемость и семантическое следствие эквивалентные концепции, если используется правильное и полное исчисление. Тем самым упомянутые выше исчисления естественного вывода малопригодны для автоматизации.

Логика предикатов первого порядка становится мощным инструментом для математики и искусственного интеллекта.

Только исчисление резолюций, которое было введено в 1965 году и по существу работает с всего одно простое правило вывода позволило создать мощные автоматизированные доказательные устройства, которые позже использовались в качестве машин вывода для экспертных систем.

3.5 Разрешение

Действительно, правильное и полное исчисление резольвент вызвало логическую эйфорию в 1970-х годах . Многие ученые считали, что можно сформулировать почти любую задачу представления знаний и рассуждений в PL1, а затем решить его с помощью автоматизированного доказывающего. Логика предикатов, мощный, выразительный язык, вместе с исчисление полных доказательств казалось универсальной разумной машиной для представления знаний и решения многих трудных задач (рис. 3.3).

Machine Translated by Google

Если ввести набор аксиом (то есть базу знаний) и запрос в такую логическую машину в качестве входных данных, машина ищет доказательство и возвращает его — поскольку оно существует и будет найдено — в качестве вывода. Взяв за основу теорему Гёделя о полноте и работу Гербранда, многое было вложено в механизацию логики. Видение машины, которая могла бы произвольной непротиворечивой базой знаний PL1 доказать любой верный запрос, было очень заманчивым. Соответственно, до сих пор многие доказательные исчисления для PL1 разрабатываются и реализуются в виде доказательств теорем. В качестве примера здесь мы описываем исторически важное и широко используемое исчисление резолюций и показываемого возможности. Причиной выбора резолюции в качестве примера исчисления доказательств в этой книге, как уже говорилось, является ее историческая и дидактическая важность. Сегодня разрешение представляет собой лишь одно из многих исчислений, используемых в высокопроизводительных прувера х .

Мы начнем с попытки скомпилировать доказательство в таблице 3.2 на странице 41 с базой знаний из примера 3.2 в доказательство разрешения. Сначала формулы преобразуются в конъюнктивную нормальную форму и отрицательный запрос

$$\neg Q \quad \neg \text{ребенок}(\text{Ева}, \text{Оскар}, \text{Энн})$$

добавляется в базу знаний, что дает

$$\begin{aligned} KB \quad & \neg Q \quad (\text{ребенок}(\text{Ева}, \text{Энн}, \text{Оскар}))1 \\ & (\neg \text{дочерний}(x, y, z) \quad \text{дочерний}(x, z, y))2 \\ & (\neg \text{ребенок}(\text{ева}, \text{оскар}, \text{энн}))3. \end{aligned}$$

Тогда доказательство может выглядеть примерно так

$$\begin{aligned} & (2)\ x/eve,\ y/anne,\ z/oscar : (\neg child(eve, anne, oscar) \quad child(eve, oscar, \\ & \qquad anne))4 \\ & \qquad Res(3, 4): (\neg \text{ребенок}(\text{ева}, \text{Анна}, \text{Оскар}))5 \\ & \qquad Res(1, 5) : ()6, \end{aligned}$$

где на первом шаге переменные x,y,z заменяются константами. Затем следуют два шага разрешения при применении общего правила разрешения из (2.2), взятого без изменений из логики высказываний.

Обстоятельства в следующем примере несколько сложнее. Мы предполагаем, что каждый знает свою собственную мать , и спрашиваем, знает ли кто-нибудь Генри . С функциональным символом «мать» и предикатом «знает» мы должны вывести противоречие из

$$(\text{знает}(x, \text{мать}(x)))1 \quad (\neg \text{знает}(\text{Генри}, y))2.$$

Заменой x/henry, y/mother(henry) получаем пару противоречивых предложений

$$(\text{знает}(\text{Генри}, \text{мама}(\text{Генри})))1 \quad (\neg \text{знает}(\text{Генри}, \text{мама}(\text{Генри})))2.$$

Machine Translated by Google

Этот шаг замены называется унификацией. Два литерала дополняют друг друга, что означает, что они одинаковы, кроме своих знаков. Пустое предложение теперь выводится с помощью шага разрешения, с помощью которого было показано, что Генри делает знать кого-то (свою мать). Мы определяем

Определение 3.7. Два литерала называются унифицируемыми, если существует замена σ для всех переменных, что делает литералы равными. Такой σ называется унификатором. Унификатор называется наиболее общим унификатором (MGU), если все остальные унификаторы могут быть получается из него заменой переменных.

Пример 3.6. Мы хотим унифицировать литералы p(f (g(x)), y, z) и p(u, u, f (u)). Несколько унификаторов

σ1 : y/f (g(x)), z/f (f (g(x))), σ2 : x/h(v), y/f (g(h(v))), u/f (g(h(x))), z/f (f (g(h(v)))), σ3 : x/h(h(v)), y/f (g(h(h(v)))), z/f (f (g(h(h (v))))), u/f (g(h(h(v)))) u / f (g (h (v)))

σ4 : x/h(a), y/f (g(h(a))), z/f (f (g(h(a)))), σ5 : x/a, y/f (g (a)), z/f (f (g(a))), u/f (g(h(a))) u/f (g(a))

где σ1 — наиболее общий объединитель. Остальные унификаторы получаются из σ1 через замены x/h(v), x/h(h(v)), x/h(a), x/a.

На этом примере мы видим, что при унификации литералов предикатные символы могут рассматриваться как функциональные символы. То есть литерал рассматривается как термин. Реализации алгоритмов унификации обрабатывают аргументы функций последовательно. Термины унифицируются рекурсивно по структуре терминов. Простейшие алгоритмы унификации в большинстве случаев работают очень быстро. Однако в худшем случае время вычислений может экспоненциально расти с ростом членов. Потому что для автоматизированного доказывает, что подавляющее число попыток объединения терпят неудачу или очень просты, в в большинстве случаев сложность наихудшего случая не имеет драматического эффекта. Самая быстрая унификация алгоритмы имеют почти линейную сложность даже в худшем случае [Bib82].

Теперь мы можем дать общее правило разрешения для логики предикатов:

Определение 3.8 Правило разрешения двух предложений в конъюнктивной нормальной форме читает

$$\frac{(A_1 \vee \cdots \vee A_m \vee B), (\neg B' \vee C_1 \vee \cdots \vee C_n)\ \sigma(B) = \sigma(B')}{(\sigma(A_1) \vee \cdots \vee \sigma(A_m) \vee \sigma(C_1) \vee \cdots \vee \sigma(C_n))}, \quad (3.6)$$

где σ — MGU B и B′.

Machine Translated by Google

Теорема 3.5 Правило разрешения верно. То есть резольвента является семантическим следствием двух родительских предложений.

Однако для полноты нам все еще нужно небольшое дополнение, как показано на следующий пример.

Пример 3.7 Знаменитый парадокс Рассела гласит: «Есть парикмахер, который бреет всех, кто не бреется сам». Это утверждение противоречиво, то есть невыполнимо. Мы хотим показать это с решимостью. Парадокс, формализованный в PL1, гласит:

$$\exists x \, \text{бреет}(\text{парикмахер}, x) \leftrightarrow \neg \text{бреет}(x, x)$$

и преобразование в форму предложений дает (см. Упражнение 3.6 на стр. 55)

$$(\neg \text{бреет}(\text{цирюльник}, x) \vee \neg \text{бреет}(x, x))_1 \quad (\text{бреет}(\text{цирюльник}, x) \vee \text{бреет}(x, x))_2. \tag{3.7}$$

Из этих двух предложений мы можем вывести несколько тавтологий, но не противоречие. Таким образом, разрешение не является полным. Нам нужно еще одно правило вывода.

Определение 3.9 Факторизация предложения осуществляется путем

$$\frac{(A_1 \vee A_2 \vee \cdots \vee A_n) \quad \sigma(A_1) = \sigma(A_2)}{\sigma(A_2) \vee \cdots \vee \sigma(A_n))},$$

где σ — MGU A1 и A2.

Теперь из (3.7) можно вывести противоречие.

$$\begin{aligned} &\text{Fak}(1, \sigma : x/\text{парикмахер}) : (\neg \text{shaves}(\text{парикмахер}, \text{парикмахер}))_3 \\ &\text{Fak}(2, \sigma : x/\text{парикмахер}) : (\text{бреет}(\text{парикмахер}, \text{парикмахер}))_4 \\ &\qquad\qquad \text{Рез}(3, 4) : ()_5 \end{aligned}$$

и мы утверждаем:

Теорема 3.6. Правило разрешения (3.6) вместе с правилом факторизации (3.9) является полным опровержением. То есть, применяя этапы факторизации и разрешения, пустое предложение может быть получено из любой невыполнимой формулы в конъюнктивной нормальной форме.

Machine Translated by Google

3.5.1 Стратегии разрешения

Хотя полнота разрешения важна для пользователя, поиск доказательства на практике может быть очень утомительным. Причиной этого является огромное комбинаторное пространство поиска. Даже если вначале в KB ¬Q очень мало пар предложений, доказывающая сторона генерирует новое предложение с каждым шагом разрешения, что увеличивает количество возможных шагов разрешения на следующей итерации. Таким образом, уже давно предпринимались попытки сократить пространство поиска с помощью специальных стратегий, желательно без потери полноты. Наиболее важные стратегии следующие.

Модульное разрешение отдает приоритет шагам разрешения, в которых одно из двух предложений состоит только из одного литерала, называемого модульным предложением. Эта стратегия сохраняет полноту и приводит во многих случаях, но не всегда, к сокращению пространства поиска. Следовательно, это эвристический процесс (см. раздел 6.3).

Гарантированное сокращение пространства поиска достигается применением стратегии множества опор. Здесь подмножество KB ¬Q определяется как множество поддержки (SOS).
Каждый шаг разрешения должен включать пункт из SOS, и резольвента добавляется в SOS. Эта стратегия неполная. Она становится полным, когда гарантируется, что множество предложений выполнимо без SOS (см. упражнение 3.7 на стр. 55). Отрицательный запрос ¬Q часто используется в качестве начального сигнала SOS.

При разрешении входных данных предложение из входного набора KB ¬Q должно участвовать в каждом шаге разрешения. Эта стратегия также сокращает пространство поиска, но за счет полноты.

С помощью правила чистого литерала все предложения, содержащие литералы, для которых нет дополнительных литералов в других предложениях, могут быть удалены. Это правило сокращает пространство поиска и является полным, поэтому оно используется практически в семи резольвентными прувերами.

Если литералы предложения K1 представляют собой подмножество литералов предложения K2, то K2 можно удалить. Например, оговорка

(дождь (сегодня) street_wet (сегодня))

является избыточным, если street_wet(today) уже действителен. Этот важный шаг сокращения называется включением. Подведение тоже полное.

3.5.2 Равенство

Равенство — особенно неудобная причина взрывного роста пространства поиска.
Если мы добавим (3.1) на стр. 36 и аксиомы равенства, сформулированные в (3.2) на стр. 36 в базу знаний, то предложение симметрии ¬x = y y = x может быть объединено с каждым положительным или отрицательным уравнением, например . Это приводит к выводу новых предложений и уравнений, к которым снова можно применить аксиомы равенства, и так далее. Аналогичные следствия имеют аксиомы транзитивности и подстановки. Поэтому были разработаны специальные правила вывода на равенство, которые обходятся без явных аксиом равенства и, в частности, сокращают пространство поиска. демодуляция;

Machine Translated by Google

например, позволяет заменить термин t2 на t1, если уравнение t1 = t2 существует. Ан
уравнение t1 = t2 применяется посредством унификации к терму t следующим образом:

$$\frac{t1 = t2, \quad (\ldots t \ldots), \sigma(t1) = \sigma(t)}{(\ldots \sigma(t2) \ldots)}.$$

Несколько более общим является парамодуляция, которая работает с условными уравнениями [Bib82, Lov78].

Уравнение t1 = t2 допускает замену члена t1 на t2 , а также
замена t2 на t1. Обычно бессмысленно отменять замену, которая уже
было выполнено. Наоборот, уравнения часто используются для упрощения терминов.
Таким образом, они часто используются только в одном направлении. Уравнения, которые используются только
одного направления называются направленными уравнениями. Эффективная обработка направленных
уравнений осуществляется с помощью так называемых систем перезаписи терминов. Для формул со многими
уравнений существуют специальные доказатели равенства.

3.6 Автоматические средства доказательства теорем

Реализации вычислений вывода на компьютерах называются средствами доказательства теорем. В доль
со специализированными прувера ми для подмножеств PL1 или специальных приложений, сегодня существуют
целая линейка автоматических доказательств для полной логики предикатов и логики более высокого
порядка, из которых здесь будут обсуждаться лишь некоторые из них . Обзор наиболее важных
систем можно найти в [McC].

Один из старейших прувера разрешения был разработан в Аргоннской национальной лаборатории в
Чикаго. На основе ранних разработок, начиная с 1963 г., Otter [Kal01],
была создана в 1984 году. Прежде всего, Otter успешно применялась в специализированных областях
математики, как можно узнать на его домашней странице:

> «В настоящее время основное применение Оттера — исследования в области абстрактной алгебры и формальной логики.
> Otter и его предшественники использовались для ответа на многие открытые вопросы в области
> конечные полугруппы, тернарная булева алгебра, логические исчисления, комбинаторная логика, теория групп,
> теория решеток и алгебраическая геометрия».

Несколько лет спустя Технологический университет Мюнхена создал высокопроизводительный
прувер SETHEO [LSBB92], основанный на быстрой технологии PROLOG. С
Для достижения еще более высокой производительности была разработана реализация для параллельных
компьютеров под названием PARTHEO. Выяснилось, что не стоит использовать специальное оборудование
в доказателях теорем, как и в других областях .
искусственного интеллекта, потому что эти компьютеры очень быстро уступают место более быстрым процессорам и
более интеллектуальные алгоритмы. Мюнхен также является родиной E [Sch02], отмеченного наградами
современного средства доказательства уравнений, с которым мы познакомимся в следующем номере.
пример. На домашней странице E можно прочитать следующую компактную ироничную характеристику:

Machine Translated by Google

ции, в тораячасть которой, между прочим, применима ков семсуществующим автоматическим прувера м. сегодня

> «E — чистоэквациональный доказатель теорем длякла узальной логики. Это означает, что это программа, которая выможете запих нуть туда математическую спецификацию (в клаузальной логике с равенством) и гипотезу, которая затем будет работать вечно, используявсе ресурсывашей машины. Очень времяот времени он нах одит подтверждение гипотезыи сообщает вамоб этом ;-)».

Нах одить доказательства истинных утверждений, по-видимому, настолько сложно, что поиски увен нчаются успехом лишь в крайне редких случаях или только по прошествии очень длительного времени — если вообще увен нчаются успехом. мы пойдем в об этом более подробно в гл. 4. Здесь следует, однако, упомянуть, что не только компьютеры, нои у большинства людей возникают проблемыс поиском строгих формальных доказательств.

Хотяочевидно, что компьютерысами посебе во многих случаях не в состоянии найти доказательство, лучше всего построить системы, работающие полуавтоматически. и разрешить тесное сотрудничество с пользователем. Тем самым человек может лучше применять свои знание специальных областей применения и, возможно, ограничить поиск доказательства. Одним из наиболее успешных интерактивных средств доказательства логики предикатов высшего порядка является Isabelle [NPW02], совместный продукт Кембриджского университета и Университета Нью-Йорка. Технология Монтен.

Любой, кто ищет высокопроизводительный прувер, должен ознакомиться с текущими результатами CASC (CADE ATP System Competition) [SS06]. Здесь мы нах одим, что победителем с 2001 по 2006 год в категориях PL1 и нормальной формы статьи был Вампир из Манчестера, который работает с вариантом разрешения и специальным подход к равенству. Система Вальдмейстера Института Макса Планка в Саарбрюккене уже много лет лидирует в области доказательства равенства.

Многие высокие позиции немецких систем в CASC показывают, что немецкие исследования группы в области автоматизированного доказательства теорем играют ведущую роль, сегодня так и в прошлом.

3.7 Математические примеры

Теперь мы хотим продемонстрировать применение автоматизированного прувера с вышеупомянутым прувером E [Sch02]. E — специализированный доказатель равенства, который значительно сокращает пространство поиска через оптимизированную обработку равенства.

Мы хотим доказать, что левые и правые нейтральные элементы в полугруппе равны. Сначала шаг за шагом формализуем иск.

1CADE — это ежегодная «Конференция по автоматизированному выводу» [CAD], а ATP означает «Автоматизированное доказательство теорем».

Machine Translated by Google

Определение 3.10. Структура (M, ·), состоящая из множества M с двуместной внутренней операцией «·», называется полугруппой, если выполняется закон ассоциативности

$$x \quad y \quad z\,(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

держит. Элемент e M называется левой-нейтральным (правой-нейтральным), если xe · x = x (xx · e = x).

Осталось показать, что

Теорема 3.7. Если в полугруппе есть нейтральный слева элемент el и нейтральный справа элемент er , то el = er .

Сначала мы докажем теорему полуформально с помощью интуитивных математических рассуждений. Ясно, что для всех x M верно

$$\text{эл} \cdot x = x \tag{3.8}$$

и

$$x \cdot \text{эр} = x\,. \tag{3.9}$$

Если положить x = er в (3.8) и x = el в (3.9), мы получим два уравнения el · er = er и el · er = el . Объединение этих двух уравнений дает

$$\text{эл} = \text{эл} \cdot \text{эр} = \text{эр},$$

что мы от им доказать. Кстати, на последнем шаге мы воспользовались тем, что равенство симметрично и транзитивно.

Прежде чем применять автоматический прувер, мы проводим доказательство разрешения вручную . Сначала мы формализуем отрицательный запрос и базу знаний KB, состоящую из аксиом в виде предложений в конъюнктивной нормальной форме:

(¬el = er)1 отрицательный запрос
(m(m(x, y), z) = m(x, m(y, z)))2 (m(el,x)
= x)3 (m(x, er) =
x)4

аксиомы

равенства: (x = x)5
(¬x = y y = x)6 (¬x = y ¬y = z
x = z)7 (¬x = y m(x, z) = m (y, (рефлексивность)
z))8 (¬x = y m(z, x) = m(z, y))9 (симметрия) (транзитивность) замена в m замен

Machine Translated by Google

где умножение представлено двузначным функциональным символом m. Аксиомы равенства были сформулированы аналогично (3.1) на стр. 36 и (3.2) на стр. 36.
Простое резольвентное доказательство имеет вид

```
Res(3, 6, x6/m(el, x3), y6/x3): Res(7, 10,          (x  = т(эл, х ))10
x7/x10, y7/m(el, x10)): (¬m(el,x) = z     x  = r)11
Res(4, 11, x4/el, x11/er, z11/el): (er = el)12
Рез(1, 12,    ): ().
```

Здесь, например, Res(3, 6, x6/m(el, x3), y6/x3) означает, что в разрешении пункта 3 с пунктом 6 переменная x из пункта 6 заменена на m(el, x3) с переменная x из пункта 3. Аналогично, y из пункта 6 заменяется на x из пункт 3.

Теперь мы хотим применить к задаче доказательство E. Пункты трансформируются в предложение языка нормальной формы LOP через отображение

```
(¬A1  ···  ¬Am   B1  ···  Bn)   B1;...;Bn <   A1,...,Am.
```

Синтаксис LOP представляет собой расширение синтаксиса PROLOG (см. раздел 5) для оговорки Хорна. Таким образом, мы получаем в качестве входного файла для E

```
<- эл = эр.                                  % запрос
m(m(X,Y),Z) = m(X,m(Y,Z)) . m(el,X) = X m(X,er)   % ассоциативности m
= X                .                         % левонейтральный элемент m
                   .                         % правонейтральный элемент m
```

где равенство моделируется предикатным символом gl. В выводе доказывающего доставляет

```
unixprompt> eproof halbgr1a.lop
# Статус проблемы определен, построен объект доказательства
# Свидетельство о статусе проблемы начинается
        0 : [--equal(el, er)] : начальный
        1 : [++equal(m(el,X1),X1)] : начальный
        2 : [++equal(m(X1,er), X1)] : начальный
        3 : [++равно(эл, э)] : pm(2,1)
        4 : [--equal(el, el)] : rw(0,3)
        5 : [] : cn(4)
        6 : [] : 5 : {доказательство}
# Свидетельство о статусе проблемы заканчивается
```

Положительные литералы обозначаются знаком ++, а отрицательные — знаком --. В строках с 0 по 4, отмеченные начальным, снова перечисляются предложения из входных данных . м(а, б) обозначает шаг разрешения между пунктом а и пунктом б. Мы видим, что доказательство

Machine Translated by Google

найденное Е очень похоже на созданное вручную доказательство. Поскольку мы явно смоделировать равенство предикатом gl, особые сильные стороны Е не приходят в игру. Теперь опускаем аксиомы равенства и получаем

```
<- эл = эр.                                          % Запрос
m(m(X,Y),Z) = m(X,m(Y,Z)) . m(el,X) = X m(X,er) = X  % Ассоциативит\"{a}т в. м
                                .                    % ссылок нейтральных эл. в. м
                                .                    % rechtsneutrales Эл. в. м
```

в качестве входных данных для доказывающего.

Доказательство также становится более компактным. Мы видим в следующем выводе доказывает, что доказательство по существу состоит из одного шага вывода под вум соответствующим пунктам 1 и 2.

```
unixprompt> eproof halbgr1a.lop
# Статус проблемы определен, построен объект доказательства
# Свидетельство о статусе проблемы начинается
        0 : [--equal(el, er)] : начальный
        1 : [++equal(m(el,X1),X1)] : начальный
        2 : [++equal(m(X1,er), X1)] : начальный
        3 : [++равно(эл, э)] : pm(2,1)
        4 : [--equal(el, el)] : rw(0,3)
        5 : :сп(4)[]
        6 :      []      : 5 : {доказательство}
# Свидетельство о статусе проблемы заканчивается
```

Теперь читатель может поближе познакомиться с возможностями Е (упражнение 3.9 на стр. 55).

3.8 Приложения

В математике автоматические средства доказательства теорем используются для определенных специализированных задач. Например, важная теорема теории графов о четырех красках была впервые доказана в 1976 г. с помощью специального прувера. Тем не менее, автоматизированные прувер ыпо-прежнему играют второстепенную роль. роль в математике.

С другой стороны, на заре ИИ логика предикатов имела большое значение. для разработки экспертных систем в практических приложениях. Из-за своих проблем моделирования неопределенности (см. раздел 4.4), экспертные системы сегодня чаще всего разрабатываются используя другие формализмы.

Сегодня логика играет все более важную роль в задачах проверки. автоматический верификация программ в настоящее время является важной областью исследований между ИИ и программной инженерией. Все более сложные программные системы теперь берут на себя задачи

Machine Translated by Google

все большей и большей ответственности и значимости длябезопасности. Здесь желательно подтверждение определенных характеристик безопасности программы. Такое доказательство не может быть получено путем тестированияготовой программы, так как в общем невозможно применить программу ко всем возможным входам. Поэтому это идеальнаяобласть дляобщих или даже специализированных систем вывода. Среди прочего, сегодняиспользуются криптографические протоколы, характеристики безопасности которых проверены автоматически [FS97, Sch01]. Еще одной проблемой использованияавтоматизированных пруверов является синтез программного и аппаратного обеспечения. С этой целью, например, прувер должны поддерживать инженера-программиста в создании программ на основе спецификаций.

Повторное использование программного обеспечениясегодняттакже имеет большое значение длямногих программистов. Программист ищет программу, котораяберет входные данные с определенными свойствами и вычисляет результат с желаемыми свойствами. Алгоритм сортировки принимает входные данные с записями определенного типа данных и из них создает перестановку этих записей со свойством, что каждый элемент меньше или равен следующему элементу.
Программист сначала формулирует спецификацию запроса на PL1, состоящую из двух частей. Первая часть PREQ содержит предварительные условия, которые должныбыть соблюдены, прежде чем желаемая программа будет применена. Вторая часть POSTQ содержит постусловия, которые должны выполнятьсяпосле применения желаемой программы.

На следующем этапе в базе данных программного обеспечениянеобходимонайти модули, отвечающие этим требованиям. Дляформальной проверки в базе данных должнохраниться формальное описание предусловий PREM и постусловий POSTM длякаждого модуляM.
Предположение о возможностях модулей состоит в том, что предварительные условиямодуляследуют из предварительных условий запроса. Он должен держать это

PREM PREQ.

Все условия, которые требуются в качестве предварительных условий дляприменениямодуляM, должны отображаться в качестве предварительных условий в запросе. Если, например, модуль в базе данных принимает только списки целых чисел, то списки целых чисел в качестве входных данных также должны отображаться в качестве предварительных условий в запросе. Дополнительное требование в запросе, чтобы, например, отображались только четные числа, не вызывает проблемы.

Кроме того, оно должно выполнятьсядляпостусловий,

POSTM POSTQ.

То есть после применениямодуля должны быть выполнены все атрибуты, которые требует запрос. Теперь мы покажем применение средства доказательства теорем к этой задаче на примере из [Sch01].

Пример 3.8 VDM-SL, Венский язык спецификации методов разработки, часто используется в качестве языка дляспецификации пред- и постусловий. Предположим, что в базе данных программы имеется описание модуля ROTATE , который перемещает первый элемент списка в конец списка. Ищем модуль SHUFFLE, который создает произвольную перестановку списка. Две спецификации гласят

Machine Translated by Google


```
ПОВЕРНУТЬ(l : список) l′ : список
предварительноверно
почта
    (л = [] л ′ = [])
    (л = [] л ′ = (х вост l)^[голова l])
```

```
SHUFFLE(x : Список) x′ : Список
предварительноверно
пост i : Пункт·
    ( x1, x2 : Список · x = x1^[i]^x2
      y1, y2 : Список · x′ = y1^[i]^y2)
```

Здесь «^» обозначает объединение списков, а «·» разделяет квантификаторыс их переменные из остальной части формулы. Функции «голова л» и «х вост л» выберите первый элемент и остальные из списка соответственно. Спецификация SHUFFLE указывает, что каждый элемент списка i, который был в списке (x) до применение SHUFFLE должно быть в результате (x ′) после применения, и вице наоборот. Теперь нужно показать, что формула (PREQ PREM) (POSTM POSTQ) является следствием базы знаний, содержащей описание Тип данных Список. Две спецификации VDM-SL дают задачу проверки

```
l,l′ ,x,x′ : список · (l = x l ′ ′ = x (ш щ))
  (l = x l ′ = x ′ ((l = [] l ′ = []) (l = [] l ′ = (tl l)^[hd l])
      i : Элемент · ( x1, x2 : Список · x = x1^[i]^x2 y1, y2 : Список · x ′ = y1^[i]^y2))),
```

что затем может быть доказано с помощью прувера SETHEO.

В ближайшие годы семантическая сеть, вероятно, станет важным приложением PL1. Контент Всемирной паутины должен стать интерпретируемым не только для людей, но и для машин. С этой целью веб-сайты оснащаются описание их семантики на языке формального описания. The search for тем самым информация в сети станет значительно эффективнее, чем сегодня, где по существу доступны только текстовые стандартные блоки.

В качестве языков описания используются разрешимые подмножества логики предикатов. Разработка эффективных исчислений для рассуждений очень важна и тесно связана с языками описания. Запрос для будущей семантически работающей поисковой системы мог бы (неофициально) звучать так: «Где в Швейцарии в следующее воскресенье на высотах ниже 2000 метров будет ли хорошая погода и оптимально подготовленные горнолыжные трассы?» Чтобы ответить на такой вопрос, требуется исчисление, способное работать очень быстро. на больших наборах фактов и правил. Здесь сложные вложенные функциональные термы менее важны.

В качестве базовой структуры описания Консорциум World Wide Web разработал язык RDF (Структура описания ресурсов). Основанный на RDF значительно более мощный язык OWL (Web Ontology Language) позволяет описывать отношения между объектами и классами объектов, аналогично PL1 [SET09].
Онтологии — это описания отношений между возможными объектами.

Machine Translated by Google

Сложность при построении описаниябесчисленных веб-сайтов заключаетсяв затратах труда, а также проверке правильности семантических описаний.
Здесь могут очень помочь системымашинного обучениядляавтоматической генерации описаний.
Интересное использование «автоматической» генерации семантики в сети было предложено Луисом фонАномиз Университета Карнеги-Меллона [vA06]. Онразработал компьютерные игры, в которых игроки, распределенные по сети, должнысовместноописывать картинки ключевыми словами. Таким образом, изображениям присваиваетсясемантика в увлекательной игровой форме бесплатно. Читатель может протестировать игрыи послушать выступление изобретателяв упражнении 3.10 на странице 55.

3.9 Резюме

Мыпредоставили наиболее важные основы, терминыи процедурылогики предикатов и показали, что даже одна из самых сложных интеллектуальных задач, а именнодоказательствоматематических теорем, может быть автоматизирована. Автоматизированные пруверымогут применятьсяне только в математике, но и, в частности, в задачах проверки информатики. Однакодляповседневных рассуждений логика предикатов в большинстве случаев непригодна. В следующей и следующих главах мыпокажемего слабые места и некоторые интересные современные альтернативы. Кроме того, мыпокажем в гл. 5, что можноэлегантно программировать с помощью логики и ее процедурных расширений.

Любой, кто интересуетсялогикой первого порядка, разрешениеми другими вычислениями для автоматических пруверов, найдет хорошие дополнительные инструкции в [New00, Fit96, Bib82, Lov78, CL73]. Ссылки на интернет-ресурсыможно найти на веб-сайте этой книги.

3.10 Упражнения

Упражнение 3.1. Пусть данытрехместное сказуемое «ребенок» и одноместное сказуемое «женский пол» из примера 3.2 на стр. 35 . Определите: (a) Одноместный предикат «мужской». б) двуместное сказуемое «отец» и «мать». (c) Двухместное сказуемое «братьяи сестры». (d) Предикат «родители (x, y, z)», который истинен тогда и только тогда, когда x — отец, а y является матерью z.

(e) Предикат «uncle(x, y)», который истинентогда и только тогда, когда x является дядей y (используйте уже определенные предикаты).

(e) Двухместное сказуемое «предок» со значением: предки — это родители, дедушки и бабушки и т. д. произвольномногих поколений.

Упражнение 3.2 Формализуйте следующие утвержденияв логике предикатов: (a) У каждого человека есть отец и мать. б) У некоторых людей есть дети. в) Все птицылетают.

Machine Translated by Google

(d) Е сть жив отное, кот орое ест (нек от орых) жив отных , пит ающ их сяз ерном.

(e) К аждое жив отное пит ается раст ениями или трав оядными жив отными, кот орые намного меньше, ч ем сам.

У пражнение 3.3 Адапт ируйт е У пражнение 3.1 на стр. 54 , исполь зуя одноместные функциональные симв олы и равенст во вместо «от ец» и «мат ь».

У пражнение 3.4. Прив едит е аксиомы логики предикат ов для дв уместного от ношения «<» как полного порядка. Для полного порядка мы должны иметь (1) Люб ые дв а элемент а сравнимы. (2) Он симмет ричен. (3) Он транзит ивен.

У пражнение 3.5. У нифицируйт е (если в озможно) след ующ ие т ермины и укажит е MGU и полученные т ермины. (a) p(x, f (y)), p(f (z), u) (б) p(x, f (x)), p(y, y) (c) x = 4 7 · x, cos y = z (d) x < 2 · x, 3 < 6 (e) q(f (x, y, z), f (g(w, w), g(x, x), g(y, y))), q(и, и)

У пражнение 3.6

(a) Прео бразуйт е парадокс Р ассела из примера 3.7 на стр. 45 в К НФ. (b) Покажит е, ч то пуст ое предложение не может быт ь получено с помощью разрешения без факт оризации из (3.7) на стр. 45. Попыт айт есь понят ь это инт уит ивно.

У пражнение 3.7

(a) Поч ему резольв ента с набором ст рат егий поддержки яв ляет ся не полной? (b) Обосн уйт е (без доказат ельст ва), поч ему набор ст рат егий поддержки ст ановит ся полным, если (KB ¬Q)\SOS в ыполнима.

в) Поч ему разрешение с помощью прав ила чист ого букв ального в ыраж ения яв ляет ся полным?

У пражнение 3.8 Формализуйт е и с т очност ью докажит е, ч т о в полугруппе, содерж ащ ей по крайней мере дв а разл ичных элемент а a,b, лев ый нейт ральный элемент e и лев ый нулев ой элемент n, эт и дв а элемент а должны быт ь разл ичными, т. е. ч т о n = e. Исполь зуйт е демодуляцию, поз в оляющ ую замен ить «подобное на подобное».

У пражнение 3.9. Получит е средст во доказат ельст ва т еорем E [Sch02] или другое средст во доказат ельст ва и докажит е след ующ ие ут в ерж дения. Сравнит е эт и доказат ельст ва с прив еденными в т ексте. (a) У т в ерж дение из примера 2.3 на ст ранице 24. (b) Парадокс Р ассела из примера 3.7 на ст ранице 45. (c) У т в ерж дение из упражнения 3.8.

У пражнение 3.10 Прот ест ируйт е игры www.espgame.org и www.peekaboom.org, кот орые помогают определить семант ику изображений в сет и. Послушайт е лекцию Луиса фон Ана о чел ов еческих в ыч ислениях по ссылке: http://video.google.de/videoplay?docid= -8246463980976635143.

Machine Translated by Google

Machine Translated by Google

Ограничения логики

4

4.1 Проблема пространства поиска

Как уже упоминалось в нескольких местах, при поиске доказательства почти всегда имеется множество (в зависимости от исчисления, потенциально бесконечно много) возможностей применения правил вывода на каждом шаге. Результатом является уже упомянутый взрывной рост пространства поиска (рис. 4.1 на стр. 58). В худшем случае нужно перепробовать все эти возможности, чтобы найти доказательство, что обычно невозможно за разумное время.

Если мы сравним автоматические доказывающие или системы логического вывода с математиками или людьми-экспертами, имеющими опыт в специальных областях, мы сделаем интересные наблюдения. Во-первых, опытные математики могут доказывать теоремы, недоступные для автоматических доказательств. С другой стороны, автоматические прувер ы выполняют десятки тысяч выводов в секунду. Напротив, человек выполняет, возможно, одно умозаключение в секунду. Хотя люди-эксперты гораздо медленнее на объектном уровне (то есть в выполнении выводов), они, по-видимому, решают сложные задачи гораздо быстрее.

На это есть несколько причин. Мы, люди, используем интуитивные вычисления, которые работают на более высоком уровне и часто выполняют многие простые выводы автоматизированного доказывающего за один шаг. Кроме того, мы пользуемся леммами, т. е. выводимыми истинными формулами, которые нам уже известны и поэтому не нужно каждый раз их доказывать заново. Между тем есть и машинные прувер ы, работающие с такими методами. Но даже они пока не могут конкурировать с людьми-экспертами.

Еще одним, гораздо более важным преимуществом нас, людей, является интуиция, без которой мы не смогли бы решить ни одной сложной задачи [PS09]. Попытка формализовать интуицию вызывает проблемы. Опыт прикладных проектов ИИ показывает, что в таких сложных областях, как медицина (см. раздел 7.3) или математика, большинство экспертов не в состоянии сформулировать эти интуитивные метазнания словесно, не говоря уже о том, чтобы их формализовать. Поэтому мы не можем запрограммировать это знание или интегрировать его в исчисления в форме эвристики. Эвристика – это методы, которые во многих случаях могут значительно упростить или сократить путь к цели, но в некоторых случаях (обычно редко) могут сильно удлин

В. Эртель, Введение в искусственный интеллект, Темы бакалавриата по компьютерным наукам, DOI 10.1007/978-0-85729-299-5_4, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google



Рис. 4.1 Возможные последствия взрыва поискового пространства

к цели. Эвристический поиск важен не только для логики, но и для решения задач в области ИИ в целом, и поэтому он будет подробно рассмотрен в гл. 6.

Интересный подход, который применяется примерно с 1990 года, заключается в применении методов машинного обучения к изучению эвристик для управления поиском систем логического вывода, который мы сейчас кратко обрисуем. У доказывающего при поиске доказательства есть сотни и более возможностей для шагов разрешения на каждом шаге, но только некоторые из них ведут к цели. Было бы идеально, если бы доказывающий мог спросить оракула, какие два предложения он должен использовать на следующем шаге, чтобы быстро найти доказательство. Предпринимаются попытки построить такие направляющие доказательства модули, которые оценивают различные альтернативы для следующего шага, а затем выбирают альтернативу с лучшим рейтингом. В случае разрешения рейтинг доступных предложений может быть вычислен с помощью функции, которая вычисляет значение на основе количества положительных литералов, сложности терминов и т. д. для каждой пары разрешимых предложений.

Как можно реализовать эту функцию? Поскольку это знание «интуитивно», программист с ним не знаком. Вместо этого пытаются скопировать природу и используют алгоритмы машинного обучения, чтобы учиться на успешных доказательствах [ESS89, SE90]. Атрибуты всех пар предложений, участвующих в успешных шагах разрешения, сохраняются как положительные, а атрибуты всех неудачных разрешений сохраняются как отрицательные.
Затем, используя эти обучающие данные и систему машинного обучения, создается программа, которая может эвристически оценивать пары предложений (см. раздел 9.5).

Machine Translated by Google

Другой, более успешный подход к совершенствованию математических рассуждений заключается в следующем. следуют интерактивные системы, которые работают под контролем пользователя. Вот один могли назвать программы компьютерной алгебры, такие как Mathematica, Maple или Maxima, который может автоматически выполнять сложные символьные математические манипуляции. Однако поиск доказательств остается полностью за человеком. Вышеупомянутый интерактивный прувер Isabelle [NPW02] обеспечивает значительно большую поддержку во время доказательства. поиск. В настоящее время существует несколько проектов, таких как Omega [SB04] и MKM1 для разработка систем поддержки математиков во время доказательств.

Подводя итог, можно сказать, что из-за проблемы пространства поиска автоматизированные сегодня доказывающие могут доказывать только относительно простые теоремы в специальных областях с мало аксиом.

4.2 Разрешимость и неполнота

Логика предикатов первого порядка предоставляет мощный инструмент для представления знаний и рассуждений. Мы знаем, что существуют правильные и полные исчисления и средства доказательства теорем. При доказательстве теоремы, т. е. истинного утверждения, такой доказатель очень полезно, потому что из-за полноты через конечное время известно, что утверждение действительно правда. Что делать, если утверждение неверно? Теорема о полноте (теорема 3.3 на стр. 41) не дает ответа на этот вопрос2. В частности, не существует процесса которое может доказать или опровергнуть любую формулу из PL1 за конечное время, так как верно, что

> Теорема 4.1. Множество правильных формул в логике предикатов первого порядка полуразрешимо.

Из этой теоремы следует, что существуют программы (доказательства теорем), которые при истинная (действительная) формула в качестве входных данных, определите ее истинность за конечное время. Если формула не действительно, однако может случиться так, что доказывающий никогда не остановится. (Читатель может ответить на этот вопрос в упражнении 4.1 на стр. 65.) Логика высказываний разрешима. потому что метод таблицы истинности предоставляет все модели формулы за конечное время. Очевидно, что логика предикатов с кванторами и вложенными функциональными символами — это язык несколько слишком мощный, чтобы быть разрешимым.

С другой стороны, логика предикатов недостаточно мощна для многих целей. Часто хочется сделать утверждения о наборах предикатов или функций. Этот не работает в PL1, потому что он знает квантификаторы только для переменных, но не для предикаты или функции.

Курт Гёдель вскоре после своей теоремы о полноте для PL1 показал, что полнота теряется, если мы расширяем PL1 даже минимально для построения логики более высокого порядка. Логика первого порядка может количественно оценивать только переменные. Логика второго порядка также может

1 www.mathweb.org/mathweb/demo.html.

2 Именно этот случай особенно важен на практике, потому что если я уже знаю, что утверждение истинно, Мне больше не нужен прувер.

Machine Translated by Google

квантифицировать по формулам первого порядка, а логика третьего порядка может квантифицировать по формулам второго порядка. Даже добавление только аксиомы индукции для натуральных чисел делает логику неполной. Утверждение «Если для n выполняется предикат p(n), то верно и p(n + 1)», или

$$\forall p\,(p(n) \rightarrow p(n+1))$$

является предложением второго порядка, потому что оно квантифицируется над предикатом. Гёдель доказал следующую теорему:

> **Теорема 4.2 (теорема Гёделя о неполноте)** Любая система аксиом для натуральных чисел со сложением и умножением (арифметика) неполна.
> То есть в арифметике есть истинные утверждения, которые недоказуемы.

Доказательство Гёделя работает с так называемой гёделизацией, при которой каждая арифметическая формула кодируется как число. Он получает уникальный номер Гёделя. Гёделизация теперь используется для формулировки предложения

$$F = \text{«Я недоказуем»}.$$

на языке арифметики. Эта формула верна по следующей причине. Предположим, что F ложно. Тогда мы можем доказать F и, следовательно, показать, что F недоказуемо. Это противоречие. Таким образом, F истинно и, следовательно, недоказуемо.

Более глубокая подоплека этой теоремы заключается в том, что математические теории (системы аксиом) и, в более общем смысле, языки становятся неполными, если язык становится слишком мощным. Аналогичный пример — теория множеств. Этот язык настолько силен, что с его помощью можно формулировать парадоксы. Это утверждения, противоречащие сами себе, такие как уже известное нам из примера 3.7 на стр. 45 утверждение о парикмахерах, которые бреют тех, кто не бреется сам (см. упражнение 4.2 на стр. 65)[3]. Дилемма состоит в том, что с языками, которые достаточно сильны, чтобы описать математику и интересные приложения, мы проносим противоречия и неполноты через черный ход. Однако это не означает, что логики высших порядков совершенно непригодны для формальных методов. Безусловно, существуют формальные системы, а также средства доказательства для логики более высокого порядка.

4.3 Летающий пингвин

На простом примере мы продемонстрируем фундаментальную проблему логики и возможные пути решения. Учитывая заявления

[3] Многих других логических парадоксов можно найти у [Wie].

Machine Translated by Google



Рис. 4.2 Летающий пингвин Твити

1. Твити — пингвин2.
Пингвины — птицы3.
Птицы умеют
летать Формализованные в PL1 результаты базы знаний KB :

$$\text{пингвин}(\text{твити})$$
$$\text{пингвин}(x) \quad \text{птица}(x)$$
$$\text{птица}(x) \quad \text{летать}(x)$$

Отсюда (например, с разрешением) можно вывести fly(tweety) (рис. 4.2)4. Очевидно, формализация летных признаков пингвинов недостаточна. Пробуем дополнительное утверждение: пингвины не могут летать, т. е.

$$\text{пингвин}(x) \quad \neg\text{муха}(x)$$

Отсюда можно вывести ¬fly(tweety) . Но fly(tweety) по-прежнему верен. Поэтому база знаний несовместима. Здесь мы отмечаем важную характеристику логики, а именно монотонность. Хотя мы прямо заявляем, что пингвины не умеют летать, все же можно вывести обратное.

Определение 4.1. Логика называется монотонной, если для произвольной базы знаний KB и произвольной формулы φ множество формул, выводимых из KB , является подмножеством формул, выводимых из KB φ.

4 Формальное выполнение этого и следующего простого доказательства может быть предоставлено читателю (упражнение 4.3 на стр. 65).

Machine Translated by Google

Если набор формул расширяется, то после расширения все ранее выводимые утверждения все еще могут быть доказаны, а также потенциально могут быть доказаны дополнительные утверждения. Таким образом, набор доказуемых утверждений монотонно растет при расширении набора формул. Для нашего примера это означает, что расширение базы знаний никогда не приведет к нашей цели. Таким образом, мы модифицируем КВ, заменив заведомо ложное утверждение «(все) птицы могут летать» более точным утверждением «(все) птицы, кроме пингвинов, могут летать», и получим в качестве КВ2 следующие положения:

пингвин(твити)
пингвин(x) ⇒ птица(x)
птица(x) ∧ ¬пингвин(x) ⇒ летать(x)
пингвин(x) ⇒ ¬летать(x)

Теперь мир, видимо, снова в порядке. Мы можем вывести ¬fly(tweety), но не fly(tweety), потому что для этого нам понадобится ¬penguin(x), который, однако, не выводим. Пока в этом мире есть только пингвины, царит мир. Однако любая нормальная птица немедленно вызывает проблемы. Желаем добавить ворона Абраксаса (из немецкой книги «Маленькая ведьма») и получить

ворон(абраксас)
ворон(x) ⇒ птица(x)
пингвин(твити)
пингвин(x) ⇒ птица(x)
птица(x) ∧ ¬пингвин(x) ⇒ летать(x)
пингвин(x) ⇒ ¬летать(x))

Мы ничего не можем сказать о лётных признаках Абраксаса, потому что забыли сформулировать, что вороны — это не пингвины. Таким образом, мы расширяем КВ3 до КВ4:

ворон(абраксас)
ворон(x) ⇒ птица(x)
ворон(x) ⇒ ¬пингвин(x)
пингвин(твити)
пингвин(x) ⇒ птица(x)
птица(x) ∧ ¬пингвин(x) ⇒ муха(x))
пингвин(x) ⇒ ¬муха(x)

Здесь следует явно добавить тот факт, что вороны не являются пингвинами, что самоочевидно для людей. Таким образом, для построения базы знаний со всеми примерно 9800 видами птиц во всем мире необходимо указать для каждого вида птиц (за исключением пингвинов), что он не является членом пингвинов. Мы должны действовать аналогично для всех других исключений, таких как страус.

Для каждого объекта в базе знаний, помимо его атрибутов, все атрибуты, которых у него нет, должны быть перечислены

Machine Translated by Google

Для решения этой проблемы разработаны различные формы немонотонной логики, позволяющие удалять знания (формулы) из базы знаний.
Под логикой по умолчанию была разработана логика, позволяющая назначать объектам атрибуты, действительные до тех пор, пока не доступны другие правила. В примере с Твити таким правилом по умолчанию будет правило, что птицы могут летать . Несмотря на большие усилия, эта логика в настоящее время из-за семантических и практических проблем не увенчалась успехом.

Монотонность может быть особенно неудобной в сложных задачах планирования, в которых мир может измениться. Если, например, синий дом покрасить в красный цвет, то впоследствии он станет красным. База знаний, например

цвет(дом, синий)
краска(дом,красный)
краска(x, y) цвет(x, y)

приводит к выводу, что после покраски дом красно-синий. Проблема, возникающая здесь при планировании, известна как проблема фрейма. Решением этой проблемы является ситуационное исчисление, представленное в разд. 5.6.

Интересным подходом к моделированию таких проблем, как пример Твити, является теория вероятностей. Утверждение «все птицы умеют летать» неверно. Утверждение вроде «почти все птицы умеют летать» верно. Это утверждение станет более точным, если мы дадим вероятность того, что «птицы могут летать». Это приводит к вероятностной логике, которая сегодня представляет собой важную область ИИ и важный инструмент для моделирования неопределенностей (см. главу 7).

4.4 Моделирование неопределенности

Двузначная логика может и должна моделировать только те обстоятельства, в которых есть истинное, ложное и нет других истинностных значений. Поэтому для многих задач повседневного рассуждения двузначная логика недостаточно выразительна. Правило

птица(x) летать(x)

верно почти для всех птиц, но для некоторых неверно. Как уже упоминалось, работа с вероятностями позволяет точно сформулировать неопределенность. Утверждение «99% всех птиц умеют летать» можно формализовать выражением

P (птица (x) летать (x)) = 0,99.

В гл. 7 мы увидим, что здесь лучше работать с условными вероятностями типа

P (муха |птица) = 0,99.

Machine Translated by Google

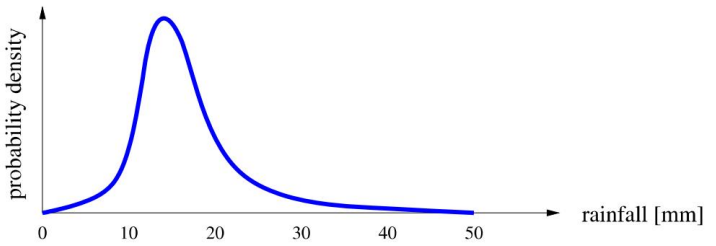


Рис. 4.3 Плотность вероятности непрерывных переменных осадков

С помощью байесовских сетей сложные приложения со многими переменными могут также быть смоделированы.

Для утверждения «Погода хорошая» нужна другая модель . Здесь это часто нет смысла говорить в категориях истинного и ложного. Переменная weather_is_nice не должны моделироваться как бинарные, а непрерывно со значениями, например, в интервал [0, 1]. Weather_is_nice = 0.7 означает «Погода довольно хорошая». Нечеткая логика, которая также будет представлена в гл. 7, был разработан для этого типа непрерывная (нечеткая) переменная.

Теория вероятностей также предлагает возможность делать утверждения о вероятности непрерывных переменных. Утверждение в метеосводке, такое как «Есть высокая вероятность того, что будет дождь» можно было бы, например, точно сформулировать как плотность вероятности вида

$$P(\text{осадки} = X) = Y$$

и изображается графически примерно так, как на рис. 4.3.

Это очень общее и даже наглядное представление обоих типов неопределенности, которые мы обсуждали, вместе с индуктивной статистикой и теорией байесовских сетей позволяет, в принципе, ответить на произвольные вероятностные запросы.

Теория вероятностей, а также нечеткая логика не могут быть напрямую сопоставлены с предикатом. логики, потому что они не допускают переменных или квантификаторов. Таким образом, их можно рассматривать как расширения пропозициональной логики, как показано в следующей таблице.

Формализм	Количество значения истинности	Вероятности выразимый
Логика высказываний	2	-
Нечеткая логика		-
Дискретная вероятностная логика	н	да
Непрерывная вероятностная логика		да

Machine Translated by Google

4,5 Упражнения

Упражнение 4.1

(a) С помощью следующего (ложного) аргумента можно было бы утверждать, что PL1 разрешима: мы берем полное исчисление вывода для PL1. С его помощью мы можем найти доказательство любой истинной формулы за конечное время. Для любой другой формулы φ я поступаю следующим образом: я применяю исчисление к ¬φ и показываю, что ¬φ истинно. Таким образом, φ ложно. Таким образом, я могу доказать или опровергнуть любую формулу в PL1. Найдите ошибку в аргументе и измените его,

чтобы он стал правильным. (b) Построить процесс принятия решения для множества истинных и невыполнимых формул ПЛ1.

Упражнение 4.2

(a) Учитывая утверждение «Есть парикмахер, который бреет каждого, кто не бреется сам». Подумайте, бреется ли этот цирюльник. (б) Пусть M = {x |x / x}. Опишите это множество

и посмотрите, содержит ли M себя.

Упражнение 4.3. Используйте автоматизированное средство доказательства теорем (например, E [Sch02]) и примените его ко всем пяти различным аксиоматизациям примера Твити из разд. 4.3. Подтвердите утверждения примера.

Machine Translated by Google

Machine Translated by Google

Логическое программирование с PROLOG

5

По сравнению с классическими языками программирования, такими как C или Pascal, Logic делает его можно выразить отношения элегантно, компактно и декларативно. Автоматизированный доказывающие теоремы способны даже решить, является ли база знаний логически влечет за собой запрос. Расчет доказательств и знания, хранящиеся в базе знаний, строго разделены. В качестве входных данных может использоваться формула, описанная в нормальной форме предложения. данные для любого средства доказательства теорем, независимо от используемого исчисления доказательств. это здорово значение для рассуждения и представления знаний.

Если кто-то хочет реализовать алгоритмы, которые неизбежно имеют процедурные компоненты, чисто декларативного описания часто бывает недостаточно. Роберт Ковальски, один из пионеров логического программирования, сделал это с формулой

Алгоритм = Логика + Управление.

Эта идея была реализована в языке ПРОЛОГ. ПРОЛОГ используется в много проектов, в первую очередь в области ИИ и компьютерной лингвистики. Сейчас мы дадим краткое введение в этот язык, представить наиболее важные понятия, показать его сильные стороны и сравнить его с другими языками программирования и средствами доказательства теорем. Те, кто ищет полный курс программирования, могут обратиться к таким учебникам, как [Bra11, CM94] и справочники [Wie04, Dia04].

Синтаксис языка PROLOG допускает только предложения Horn. Логическая запись и синтаксис PROLOG сопоставлены в следующей таблице:

PL1 / нормальная форма предложения	PROLOG	Описание
(¬A1 ⋯ ¬Am) B	B :- A1, ... , Am.	Правило
(A1 ⋯ Am) Б	B: -A1, ..., Am.	Правило
A	A.	Факт
(¬A1 ⋯ ¬Am)	?- A1, ..., Am.	Запрос
¬(A1 ⋯ Am)	?- A1, ..., Am.	Запрос

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по информатике,
DOI 10.1007/978-0-85729-299-5_5, © Springer-Verlag London Limited 2011

Machine Translated by Google


```
1 ребенок(Оскар,Карен,Фрэнк). 2
ребенка(Мэри,Карен,Фрэнк). 3
ребенка(Ева,Энн,Оскар). 4
ребенка(Генри,Энн,Оскар). 5
детей(Изольда,Энн,Оскар). 6 ребенок
(Клайд, Мэри, Оскарб). 7 8 дочерний(X,Z,Y):-

дочерний(X,Y,Z). 9

10 потомок(X,Y):- ребенок(X,Y,Z). 11 потомок(X,Y):-
потомок(X,U,V), потомок(U,Y).
```

Рис. 5.1 Программа PROLOG с семейными отношениями

Здесь A1,...,Am,A,B — литералы. Литералы, как и в PL1, строятся из предикатных символов с терминами в качестве аргументов. Как видно из приведенной выше таблицы, в PROLOG нет отрицаний в строгом логическом смысле, поскольку знак литерала определяется его позицией в предложении.

5.1 Системы и реализации PROLOG

Обзор современных систем PROLOG доступен по набору ссылок на домашней странице этой книги. Читателю мы рекомендуем очень мощные и свободно доступные (под публичными лицензиями GNU) системы GNU-PROLOG [Dia04] и SWI-PROLOG. В следующих примерах использовался SWI-PROLOG [Wie04].

Большинство современных систем PROLOG работают с интерпретатором, основанным на абстрактной машине Уоррена (WAM). Исходный код PROLOG компилируется в так называемый код WAM, который затем интерпретируется WAM. Самые быстрые реализации WAM обрабатывают до 10 миллионов логических выводов в секунду (LIPS) на ПК с тактовой частотой 1 ГГц.

5.2 Простые примеры

Начнем с семейных отношений из примера 3.2 на стр. 35. Небольшая база знаний KB закодирована — без фактов для предиката female — как программа PROLOG с именем rel.pl на рис. 5.1.

Программу можно загрузить и скомпилировать в интерпретаторе PROLOG с помощью команды

```
?- [отн.].
```

Machine Translated by Google

Первоначальный запрос возвращает диалог

?- ребенок(ева,оскар,энн).

Да

с правильным ответом Да. Как возникает этот ответ? Для запроса «?-ребенок(ева,оскар,энн)». есть шесть фактов и одно правило с одним и тем же предикатом в заголовке предложения. Теперь предпринимается попытка унификации между запросом и каждым из дополнительных литералов во входных данных в порядке их появления. Если одна из альтернатив терпит неудачу, это приводит к возврату к последней точке ветвления, и проверяется следующая альтернатива. Поскольку унификация не удалась с каждым фактом, запрос унифицируется с помощью рекурсивного правила в строке 8. Теперь система пытается решить подцель дочерний элемент (ева, анна, Оскар), что удается с третьей альтерна

Запрос

?- потомок(X,Y).

X = Оскар
Y = Карен

Да

отвечает первым найденным решением, как

?- потомок(клайд,Y).

Y = Мэри

Да

Запрос

?- потомок(Клайд,Карен).

однако не отвечает. Причиной этого является предложение в строке 8, определяющее симметрию дочернего предиката. Это предложение вызывает себя рекурсивно без возможности завершения. Эту задачу можно решить с помощью следующей новой программы (здесь опущены факты).

```
1 потомок(X,Y):- ребенок(X,Y,Z). 2 потомок(X,Y):- потомок(X,Z,Y). 3
потомок(X,Y):- потомок(X,U,V), потомок(U,Y).
```

Machine Translated by Google

Но теперь запрос

?- ребенок(ева,оскар,энн).

больше не является правильным ответом, потому что симметрия ребенка в последних двух переменных больше не задана. Решение обеих проблем находится в программе

```
1 child_fact(Оскар,Карен,Франц). 2
child_fact(мэри,карен,франц). 3
child_fact(ева,энн,оскар). 4
child_fact(Генри,Энн,Оскар). 5
child_fact(изольда,анна,оскар). 6
child_fact(клайд,мэри,оскарб). 7 8 дочерний(X,Z,Y):-

дочерний_факт(X,Y,Z). 9 дочерний(X,Z,Y):-
дочерний_факт(X,Z,Y). 10 11 потомок(X,Y):-

потомок(X,Y,Z). 12 потомок(X,Y):- потомок(X,U,V),
потомок(U,Y).
```

Благодаря введению нового предиката child_fact для фактов предикат child больше не является рекурсивным. Однако программа уже не такая элегантная и простая, как логически правильный первый вариант на рис. 5.1 на стр . 68, приводящий к бесконечному циклу. Программист на PROLOG должен, как и в других языках, обращать внимание на обработку и избегать бесконечных циклов. PROLOG — это просто язык программирования, а не средство доказательства теорем.

Здесь мы должны различать декларативную и процедурную семантику программ PROLOG. Декларативная семантика задается логической интерпретацией роговых предложений. Процедурная семантика, напротив, определяется выполнением программы PROLOG, которую мы сейчас хотим рассмотреть более подробно. Выполнение программы с рис. 5.1 на стр. 68 с запросом child(eve,oscar,anne) представлено на рис. 5.2 на стр. 71 в виде дерева поиска.1 Выполнение начинается слева вверху с запроса. Каждое ребро представляет возможный шаг разрешения SLD с дополнительным унифицируемым литералом. В то время как дерево поиска становится бесконечно глубоким из-за рекурсивного правила, выполнение PROLOG прекращается, потому что факты появляются перед правилом во входных данных.

Напротив, с запросом потомок(клайд,карен) выполнение PROLOG не завершается. Мы можем ясно видеть это в дереве и/или, представленном на рис. 5.3 на стр. 71. В этом представлении ветви, представленные значком, ведут от заголовка предложения , к подцелям. Поскольку все подцели предложения должны быть решены, это и ответвления. Все остальные ветви являются или ветвями, из которых хотя бы одна должна быть унифицируемой со своими родительскими узлами. Два изложенных факта отражают

1Константы были сокращены для экономии места.

Machine Translated by Google

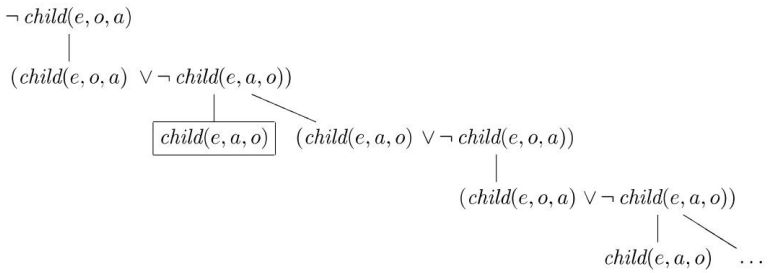


Рис. 5.2 Дерево поиска PROLOG для ребенка (ева, оскар, анна)

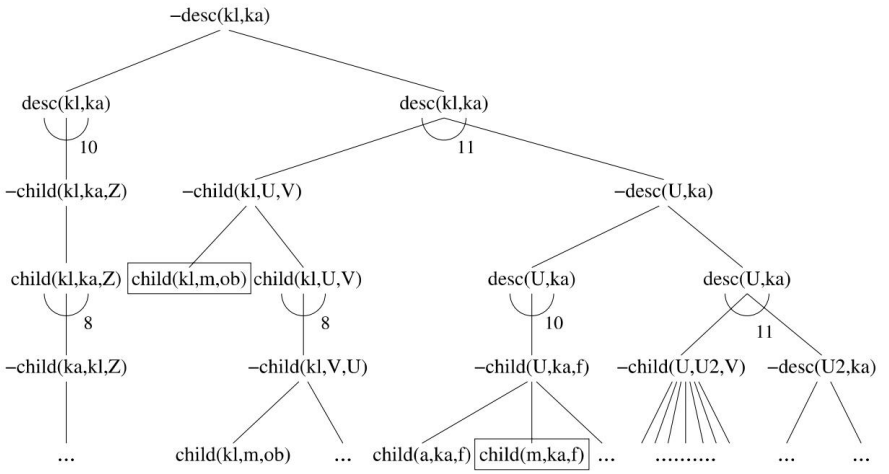


Рис. 5.3 И-или дерево для desc(clyde,karen)

решение запроса. Однако интерпретатор PROLOG здесь не останавливается, потому что он работает, используя поиск в глубину с возвратом (см. раздел 6.2.2) и, таким образом, сначала выбирает бесконечно глубокий путь в крайнее левое положение.

5.3 Контроль исполнения и процедурные элементы

Как мы видели в примере с семейными отношениями, важно контролировать выполнение PROLOG. В частности, избегание ненужного возврата назад может привести к значительному повышению эффективности. Одним из средств для достижения этой цели является разрез. Вставив восклицательный знак в предложение, мы можем предотвратить возврат к этой точке. В следующей программе предикат max(X,Y,Max) вычисляет максимальное из двух чисел X и Y.

Machine Translated by Google


```
1 max(X,Y,X) :- X >= Y. 2 max(X,Y,Y) :-
X < Y.
```

Если применяется первый случай (первый пункт), то второй не будет достигнут. С другой стороны, если первый случай неприменим, то условие второго случая истинно, а значит, его не нужно проверять. Например, в запросе

?- max(3,2,Z), Z > 10.

поиск с возвратом используется, поскольку Z=3, а второе предложение проверяется на максимальное значение, что обречено на провал. Таким образом, возвращаться к этому месту не нужно. Мы можем оптимизировать это с помощью разреза:

```
1 max(X,Y,X) :- X >= Y, !. 2 макс(X,Y,Y).
```

Таким образом, второе предложение вызывается только в том случае, если оно действительно необходимо, то есть если первое предложение не работает. Однако эта оптимизация усложняет понимание программы.

Другая возможность управления выполнением — это встроенный предикат fail, который никогда не бывает истинным. В примере с семейными отношениями мы можем просто вывести всех детей и их родителей с помощью запроса

?- child_fact(X,Y,Z), write(X), write(' является потомком '), write(Y), write(' and '), write(Z), write('.'), нл, провал.

Соответствующий вывод

Оскар — ребенок Карен и Фрэнка. Мэри — дочь Карен
и Фрэнка. Ева — дочь Анны и Оскара.

...
Нет.

где предикат nl вызывает разрыв строки в выводе. Каким был бы вывод в конце без использования предиката fail?

С той же базой знаний запрос «?- child_fact(ulla,X,Y)». приведет к ответу Нет, потому что нет никаких фактов об улле. Этот ответ не является логически правильным. В частности, невозможно доказать отсутствие объекта с именем ulla. Здесь доказывающая E правильно ответила бы: «Доказательство не найдено». Таким образом, если ПРОЛОГ отвечает Нет, это означает только то, что запрос Q не может быть

Machine Translated by Google

быть доказанным. Однако для этого не обязательно доказывать ¬Q. Такое поведение называется отрицанием как неудачей.

Ограничение себя предложениями Хорна в большинстве случаев не вызывает больших проблем. Однако это важно для процедурного исполнения с использованием SLD-разрешения (раздел 2.5). Благодаря однозначно определенному положительному литералу для предложения, разрешению SLD и, следовательно, выполнению программ PROLOG есть уникальная точка входа в предложение. Только так возможно воспроизводимое выполнение логических программ и, следовательно, четко определенная процедурная семантика.

В самом деле, безусловно, существуют формулировки задачи, которые не могут быть описаны с помощью предложений Хорна. Примером может служить парадокс Рассела из примера 3.7 на стр. 45, который содержит предложение, отличное от Хорна (бритье(цирюльник,X) бритье(X, X)).

5.4 Списки

Как язык высокого уровня, PROLOG имеет, как и язык LISP, удобный общий тип данных списка. Список с элементами A, 2, 2, B, 3, 4, 5 имеет вид

[A,2,2,Б,3,4,5]

Конструкция [Head|Tail] отделяет первый элемент (Head) от остальных (Tail) списка. С базой знаний

список([A,2,2,B,3,4,5]).

PROLOG отображает диалоговое окно

?- список([H|T]).

H=A
T = [2, 2, B, 3, 4, 5]

Да

Используя вложенные списки, мы можем создавать произвольные древовидные структуры. Например, два дерева



может быть представлен списками [b,c] и [a,b,c] соответственно, и двумя деревьями



Machine Translated by Google

списками [[e,f,g],[h],d] и [a,[b,e,f,g],[c,h],d] соответственно.
В деревьях, где внутренние узлы содержат символы, символ является головой дерева.
список и дочерние узлы являются хвостом.

Хорошим и элегантным примером обработки списка является определение предиката append(X,Y,Z) для добавления списка Y к списку X. Результат сохраняется в Z. соответствующая программа PROLOG читает

```
1 добавить([],L,L).
2 добавить([X|L1],L2,[X|L3]) :- добавить(L1,L2,L3).
```

Это декларативное (рекурсивное) логическое описание того факта, что L3 является результатом добавление L2 к L1. В то же время, однако, эта программа также выполняет работу когда его зовут. Вызов

```
?- добавить([a,b,c],[d,1,2],Z).
```

возвращает замену Z = [a, b, c, d, 1, 2], как и вызов

```
?- добавить(X,[1,2,3],[4,5,6,1,2,3]).
```

дает замену X = [4, 5, 6]. Здесь мы замечаем, что append не является двухместная функция, но трехместная связь. На самом деле, мы также можем ввести «выходной параметр» Z и спросить, можно ли его создать.

Обратный порядок элементов списка также может быть элегантно описан и прост. одновременно программируется рекурсивным предикатом

```
1 оборот([],[]).
2 nrev([H|T],R):- nrev(T,RT), append(RT,[H],R).
```

который сводит обращение списка к обращению списка, состоящего из одного элемента меньше. Действительно, этот предикат очень неэффективен из-за вызова append. Эта программа известна как naive reverse и часто используется в качестве эталонного теста PROLOG (см. Упражнение 5.6 на странице 81). Дела идут лучше, когда кто-то продолжает использовать временный хранить, известный как аккумулятор, следующим образом:

Список	Аккумулятор
[а, б, в, г] []	
[б, в, г]	[а]
[CD]	[б, а]
[д]	[в, б, а]
[]	[г, в, б, а]

Machine Translated by Google

Соответствующая программа PROLOG читает

```
1 аккрев([],A,A). 2
приращение([H|T],A,R) :- приращение(T,[H|A],R).
```

5.5 Самомодифицирующиеся программы

Программы PROLOG не полностью компилируются, а интерпретируются WAM. Поэтому можно изменять программы во время выполнения. Программа может даже модифицировать себя. С помощью таких команд, как assert и retract, факты и правила могут быть добавлены в базу знаний или удалены из нее.

Простым применением варианта asserta является добавление производных фактов в начало базы знаний с целью избежать повторного, потенциально затратного по времени вывода (см. упражнение 5.8 на стр. 81). Если в нашем примере семейных отношений мы заменим два правила для предиката потомок на

```
1 :- динамический потомок/2. 2
потомок(X,Y):- потомок(X,Y,Z), утверждение(потомок(X,Y)). 3 потомок(X,Y):- потомок(X,U,V),
потомок(U,Y), 4 утверждения(потомок(X,Y)).
```

то все производные факты для этого предиката сохраняются в базе знаний и таким образом в дальнейшем не выводятся повторно. Запрос

?- потомок(клайд, карен).

приводит к добавлению двух фактов

потомок (Клайд, Карен). потомок(мэри,
карен).

Манипулируя правилами с помощью утверждений и ретрактов, можно написать даже полностью меняющиеся программы. Эта идея стала известна под термином генетическое программирование. Он позволяет создавать произвольно гибкие программы обучения. Однако на практике оказывается, что из-за огромного количества бессмысленных возможных изменений изменение кода методом проб и ошибок редко приводит к увеличению производительности. С другой стороны, систематическое изменение правил настолько усложняет программирование, что до сих пор такие программы, которые широко модифицируют свой собственный код, не имели успеха. В гл. 8 мы покажем, насколько успешным было машинное обучение. Однако здесь проводятся лишь очень ограниченные модификации программного кода.

Machine Translated by Google


```
1 начало :- действие(состояние(слева,слева,слева,слева),
2                                    состояние (правильно, правильно, правильно, правильно)).
3
4 действие(Начало,Цель):-
                  план(Начало,Цель,[Начало],Путь),
5 6 nl,write('Решение:'),nl,
7 write_path(Путь).
8 % write_path(Путь), ошибка. 9                                    % выход всех решений

10 план (Начало, Цель, Посещение, Путь): -
11 идти(Старт,Далее),
12 сейф(Следующий),
13 \+ участник(Следующий,Посещенный), % не(участник(...))
14                  план(Следующий,Цель,[Следующий|Посещенный],Путь).
15 план(Цель,Цель,Путь,Путь).
16
17 идти(состояние(X,X,Z,K),состояние(Y,Y,Z,K)):-через(X,Y). % фермер, волк
18 идти(состояние(X,W,X,K),состояние(Y,W,Y,K)):-через(X,Y). % фермер, коза
19 идти(состояние(X,W,Z,X),состояние(Y,W,Z,Y)):-через(X,Y). % фермер, капуста
20 идти(состояние(X,W,Z,K),состояние(Y,W,Z,K)):-через(X,Y). % фермер
21
22 поперек (слева, справа).
23 поперек (справа, слева).
24
25 безопасный(состояние(B,W,Z,K)):- через(W,Z), через(Z,K).
26 сейф(состояние(B,B,B,K)).
27 сейф(состояние(B,W,B,B)).
```

Рис. 5.4. Программа PROLOG для задачи фермер–волк–коза–капуста

5.6 Пример планирования

Пример 5.1 Следующая загадка служит постановкой задачи для типичной программы PRO LOG.

> Крестьянин хочет переправить через реку капусту, козу и волка, но его лодка такая маленькая
> что он может взять их только по одному за раз. Фермер подумал и сказал:
> сам: «Если я сначала волка переведу на другой берег, то коза съест капусту. Если я
> сначала капусту перевезти, потом волк съест козу. Что я должен делать?"

Это задача планирования, которую мы можем быстро решить, немного подумав.
Программа PROLOG, представленная на рис. 5.4 , создается не так быстро.

Программа работает на условиях формы состояния(Фермер,Волк,Коза,
Капуста), которые описывают современное состояние мира. Четыре переменные с
возможные значения left, right дают расположение объектов. Центральный план рекурсивных
предикатов сначала создает состояние-преемник. Затем, используя go, проверяет его безопасность.
с безопасным и повторяет это рекурсивно до тех пор, пока начальное и целевое состояния не совпадут.
(в строке 15 программы). Состояния, которые уже были посещены, сохраняются в
третий аргумент плана. С помощью встроенного члена-предиката проверяется, является ли
состояние Next уже посещено. Если да, то это не попытка. Определение
здесь отсутствует предикат write_path для задачи вывода найденного плана. Это
предлагается в качестве упражнения для читателя (Упражнение 5.2 на стр. 80). Для начальной про-

Machine Translated by Google

gram проверяет, что литерал write_path(Path) можно заменить на write(Path). Для запроса «?- начало». мы получаем ответ

Решение:
Фермер и коза слева направо
Фермер справа налево
Фермер и волк слева направо
Фермер и коза справа налево
Фермер и капуста слева направо
Фермер справа налево
Фермер и коза слева направо

Да

Для лучшего понимания опишем определение плана в логике:

z план(z, z) s z n [go(s, n) безопасный(n) план(n, z) план(s, z)]

Это определение получается значительно более кратким, чем в PROLOG. Этому есть две причины. Во-первых, результат обнаруженного плана не имеет значения для логики. Кроме того, на самом деле нет необходимости проверять, было ли уже посещено следующее состояние, если ненужные переходы не беспокоят фермера. Однако, если \+ member(...) не включить в программу PROLOG, возникает бесконечный цикл, и PROLOG может не найти расписания, даже если оно есть. Причиной этого является стратегия поиска PROLOG с обратной цепочкой, которая, в соответствии с принципом поиска в глубину (раздел 6.2.2), всегда работает с подцелями по одной, не ограничивая глубину рекурсии, и поэтому является неполной. Этого не случилось бы с доказателем теорем с полным исчислением.

Как и во всех задачах планирования, состояние мира меняется по мере выполнения действий от одного шага к другому. Это предполагает отправку состояния в виде переменной всем предикатам, которые зависят от состояния мира, например, в предикате safe. Переходы между состояниями происходят в предикате go. Этот подход называется ситуационным исчислением [RN10]. В гл. 10.

5.7 Программирование логики ограничений

Программирование систем планирования, в которых должны выполняться многие (иногда сложные) логические и численные условия, может быть очень дорогим и сложным при использовании обычных языков программирования. Именно здесь логика может быть полезна. Просто записывают все логические условия в PL1, а затем вводят запрос. Обычно этот подход терпит неудачу. Причина в проблеме пингвинов, обсуждавшейся в разд. 4.3. Тот факт, что пингвин(твити) гарантирует, что пингвин(твит

Machine Translated by Google

истинный. Однако это не исключает, что ворон(твити) тоже верен. Исключать это дополнительными аксиомами очень неудобно (п. 4.3).

Логическое программирование с ограничениями (CLP), которое позволяет явно формулировать ограничения для переменных, предлагает элегантный и очень эффективный механизм для решения этой проблемы. Интерпретатор постоянно следит за выполнением программы на предмет соблюдения всех ее ограничений. Программист полностью освобождается от задачи управления ограничениями, что во многих случаях может значительно упростить программирование Это выражено в следующей цитате Юджина К. Фрейдера из [Fre97]:

> Программирование с ограничениями представляет собой один из самых близких подходов, которые компьютерная наука когда-либо делала к Святому Граалю программирования: пользователь формулирует проблему, компьютер решает ее.

Не вдаваясь в теорию проблемы удовлетворения ограничений (CSP), мы применим механизм CLP GNU-PROLOG к следующему примеру.

Пример 5.2 Секретарь средней школы имени Альберта Эйнштейна должен разработать план распределения помещений для выпускных экзаменов. У него есть следующая информация: четыре учителя Майер, Гувер, Миллер и Смит дают тесты по немецкому языку, английскому языку, математике и физике в восходящих комнатах 1, 2, 3 и 4. Каждый учитель дает тест ровно по одному предмету. объект ровно в одной комнате. Кроме того, он знает следующее об учителях и их предметах.

1. Мистер Майер никогда не проводит тесты в комнате 4.
2. Мистер Миллер всегда проверяет немецкий язык.
3. Мистер Смит и мистер Миллер не проводят анализы в соседних комнатах.
4. Миссис Гувер проверяет математику.
5. Физика всегда проверяется в комнате №4.
6. Немецкий и английский языки в комнате 1 не тестируются.

Кто дает тест в какой комнате?

Программа GNU-PROLOG для решения этой задачи приведена на рис. 5.5 на стр. 79. Эта программа работает с переменными Майера, Гувера, Миллера, Смита, а также с немецким, английским, математическим, физическим, каждая из которых может принимать целое число. значение от 1 до 4 в качестве номера комнаты (строки программы 2 и 5). Привязка Mayer = 1 и German = 1 означает, что мистер Майер проводит тест по немецкому языку в комнате 1. Строки 3 и 6 гарантируют, что четыре конкретные переменные принимают разные значения. Строка 8 гарантирует, что всем переменным присваивается конкретное значение в случае решения. Эта строка здесь совершенно не нужна. Однако, если бы было несколько решений, были бы выведены только интервалы. В строках с 10 по 16 даны ограничения, а остальные строки выводят номера кабинетов для всех учителей и всех предметов в простом формате.

Программа загружается в GNU-PROLOG с помощью «['raumplan.pl']», и с «начать». мы получаем вывод

[3,1,2,4] [2,3,1,4]

Machine Translated by Google


```
1 начало :-
2 fd_domain([Майер, Гувер, Миллер, Смит],1,4),
3 fd_all_differ([Майер, Миллер, Гувер, Смит]),
4
5 fd_domain([немецкий, английский, математика, физика],1,4),
6 fd_all_ Different([немецкий, английский, математика, физика]),
7
8 fd_labeling([Майер, Гувер, Миллер, Смит]),
9
10 Mayer #\=4, 11 Miller #=                    % Майер не в комнате 4
немецкий, 12 dist(Miller,Smith) #>= 2,        % Миллер тестирует немецкий язык
13 Hoover #= математика, 14 физика #= 4, 15    % Расстояние Миллер/Смит >= 2
немецкий #\=1, 16 английский                  % Математические тесты Гувера
#\=1 , 17 нл,                                 % Физика в комнате 4
                                              % Немец не в комнате 1
                                              % Английский не в комнате 1

18 write([Майер, Гувер, Миллер, Смит]), nl,
19 write([немецкий, английский, математика, физика]), nl.
```

Рис. 5.5 CLP-программа для задачи планирования помещений

Представив несколько более удобно, мы имеем следующее расписание помещений:

Номер комнаты	1	2	3	4
Учитель	Гувер	Миллер	Майер	Смит
Предмет	Математика	немецкий	английский	Физика

GNU-PROLOG, как и большинство других языков CLP, имеет так называемую конечную область.
решатель ограничений, с помощью которого переменным может быть назначен конечный диапазон целых чисел.
Это не обязательно должен быть интервал, как в примере. Мы также можем ввести список
ценностей. В качестве упражнения пользователю предлагается в Упражнении 5.9 на стр. 81 создать
Программа CLP, например, с GNU-PROLOG, для не очень простой логической головоломки.
Эту загадку, предположительно созданную Эйнштейном, очень легко решить с помощью CLP.
система. С другой стороны, если бы мы попытались использовать PROLOG без ограничений, мы могли бы
легко стачиваем зубы. Любой, кто находит элегантное решение с PROLOG или
прувер, пожалуйста, пусть он найдет путь к автору.

5.8 Резюме

Унификация, списки, декларативное программирование и реляционное представление процедур,
в которых аргумент предиката может действовать как вход и выход, позволяют разрабатывать короткие,
элегантные программы для решения многих задач. Многие программы будут
значительно длиннее и, следовательно, труднее понять, если они написаны процедурным
язык. Кроме того, эти возможности языка экономят время программиста. Поэтому PROLOG также является
интересным инструментом для быстрого прототипирования, особенно для ИИ.

Machine Translated by Google

Приложения. CLP-расширение PROLOG полезно не только для решения логических головоломок, но и для многих задач оптимизации и планирования.

С момента своего изобретения в 1972 году в Европе PROLOG превратился в один из ведущих европейских языков программирования для ИИ наряду с процедурными языками. С другой стороны, в США на рынке искусственного интеллекта доминирует язык LISP.

ПРОЛОГ не является средством доказательства теорем. Это сделано намеренно, потому что программист должен иметь возможность легко и гибко управлять обработкой, а с доказателем теорем далеко не уедешь. С другой стороны, ПРОЛОГ сам по себе не очень полезен для доказательства математических теорем. Однако существуют, безусловно, интересные средства доказательства теорем, запрограммированные на PROLOG.

В качестве расширенной литературы рекомендуется [Bra11] и [CM94], а также справочники [Wie04, Dia04] и, по теме CLP, [Bar98].

5.9 Упражнения

Упражнение 5.1 Попробуйте доказать теорему из разд. 3.7 о равенстве левых и правых нейтральных элементов полугрупп с PROLOG. Какие проблемы возникают?
Какова причина этого?

Упражнение

5.2 (a) Напишите предикат write_move(+State1, +State2), который выводит предложение типа «Фермер и волк переходят слева направо» для каждой переправы через лодку. State1 и State2 являются терминами формы state(Фермер, Волк, Коза, Капуста). (b) Напишите рекурсивный предикат
write_path(+Path), который вызывает предикат write_move(+State1, +State2) и выводит все действия фермера.

Упражнение

5.3 (a) На первый взгляд переменная Path в плане предикатов программы PROLOG из примера 5.1 на стр. 76 не нужна, поскольку она, по-видимому, нигде не изменяется. Для чего это нужно?
(б) Если мы добавим отказ в конце действия в примере, то все решения будут даны в качестве вывода. Почему теперь каждое решение печатается дважды? Как вы можете предотвратить это?

Упражнение

5.4 (a) Покажите путем проверки, что средство доказательства теорем E (в отличие от ПРОЛОГА) с учетом базы знаний, показанной на рис. 5.1 на стр. 68, отвечает на запрос «?-потомок(клайд, карен)». правильно. Почему это? (b) Сравните ответы
PROLOG и E на запрос «?-потомок(X,
Ю)».

Machine Translated by Google

Упражнение 5.5. Напишите как можно более короткую программу на PROLOG, которая выводит 1024 единицы.

Упражнение 5.6. Исследуйте поведение простого обратного предиката во время выполнения.

(a) Запустите PROLOG с опцией трассировки и посмотрите на рекурсивные вызовы nrev, append и accrev.

(b) Вычислите асимптотическую временную сложность append(L1,L2,L3), то есть зависимость времени выполнения от длины списка для больших списков. Предположим, что доступ к голове произвольного списка занимает постоянное время. (c) Вычислите временную сложность nrev(L,R). (d) Вычислите временную сложность accrev(L,R). (e) Экспериментально определить временную сложность предикатов nrev, append и accrev, например, путем проведения измерений времени (время(+Цель) дает выводы и процессорное время).

Упражнение 5.7 Используйте функциональные символы вместо списков для представления деревьев, приведенных в разд. 5.4 на стр. 73.

Упражнение 5.8 Последовательность Фибоначчи определяется рекурсивно как fib(0) = 1, fib(1) = 1 и fib(n) = fib(n 1) + fib(n 2). (a) Определите рекурсивный предикат PROLOG fib(N,R), который вычисляет fib(N) и возвращает его в R.

(b) Определить сложность времени выполнения предиката fib теоретически и с помощью измерение.

(c) Измените свою программу, используя asserta, чтобы больше не выполнялись ненужные выводы. (d) Определить сложность модифицированного предиката во время выполнения теоретически и путем измерения (обратите внимание, что это зависит от того, был ли ранее вызван fib).

(e) Почему fib с asserta также быстрее, когда он запускается в первый раз правильно после запуска PROLOG?

Упражнение 5.9 Следующая типичная логическая головоломка предположительно была написана Альбертом Эйнштейном. Более того, он якобы утверждал, что только 2% населения земного шара способны ее решить. Даны следующие утверждения. • Есть пять домов, каждый из которых окрашен в свой цвет. • В каждом доме живет человек другой национальности. • Каждый житель предпочитает определенный напиток, курит определенную марку сигарет и имеет определенное домашнее животное. • Ни один из пяти человек не пьет одно и то же, не курит одно и то же и не имеет такой же питомец.

- Подсказки: – Британец живет в красном доме.
  - У шведа есть собака.
  - Датчанин любит пить чай.
  - Зеленый дом находится слева от белого дома.

Machine Translated by Google

– Хозяин зелени пьет кофе.

– У человека, который курит Pall Mall, есть птица.
– Мужчина, живущий в среднем доме, пьет молоко.

– Хозяин желтого дома курит Dunhill.

– Норвежец живет в первом доме.
– Курильщик Marlboro живет рядом с тем, у кого есть кошка.

– Мужчина с лошадью живет рядом с тем, кто курит Dunhill.

– Курильщик Winfield любит пить пиво.

– Норвежец живет рядом с синим домом.
– Немец курит Rothmanns.

– У курильщика Marlboro есть сосед, который пьет воду.

Вопрос: Кому принадлежит рыба? (a) Сначала
решите головоломку вручную. (b)
Напишите программу CLP (например, с помощью GNU-PROLOG), чтобы решить головоломку.

Ориентируйтесь на задачу планирования помещений на рис. 5.5 на стр. 79.

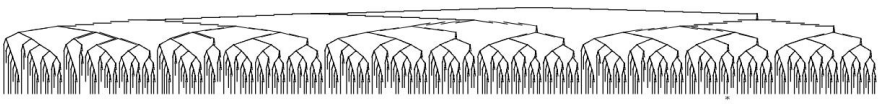
Machine Translated by Google

Поиск, игры и решение проблем

6

6.1 Введение

Поиск решения в очень большом дереве поиска представляет проблему почти для всех систем логического вывода. Из начального состояния есть много возможностей для первого шага вывода. Для каждой из этих возможностей на следующем шаге снова есть много возможностей и так далее. Даже при доказательстве очень простой формулы из [Ert93] с тремя предложениями Хорна, каждое из которых содержит не более трех литералов, дерево поиска для SLD-разрешения имеет следующий вид:



Дерево было срезано на глубине 14 и имеет решение в листовом узле, отмеченном . Его вообще возможно представить только из-за малого коэффициента ветвления, не более двух, и отсечки на глубине 14. Для реальных задач коэффициент ветвления и глубина первого решения могут стать значительно больше.

Предположим, что коэффициент ветвления является константой, равной 30, и первое решение находится на глубине 50. Дерево поиска имеет 3050 7,2 × 1073 листовых узлов. Но количество шагов вывода еще больше, потому что не только каждый листовой узел, но и каждый внутренний узел дерева соответствует шагу вывода. Поэтому мы должны сложить узлы по всем уровням и получить общее количество узлов дерева поиска.

$$30d = \sum_{d=0}^{50} = \frac{1 - 3051}{1 \quad 30} \approx 7{,}4 \times 1073 \quad ,$$

что не сильно меняет количество узлов. Очевидно, что почти все узлы этого дерева поиска находятся на последнем уровне. Как мы увидим, это обычно так. Но теперь вернемся к дереву поиска с 7,4 × 1073 узлами. Предположим, у нас есть 10 000 компьютеров, каждый из которых может выполнять миллиард логических выводов в секунду, и что мы можем

В. Эртель, Введение в искусственный интеллект, Темы бакалавриата по компьютерным наукам, DOI 10.1007/978-0-85729-299-5_6, © Springer-Verlag London Limited 2011

Machine Translated by Google

распределять работу по всем компьютерам бесплатно. Общее время вычислений для всех 7,4 × 1073 выводов будет примерно равно

$$7{,}4 \times 1073 \text{ выводов} = 7{,}4 \times \frac{1060 \text{ с}}{10000 \times 109 \text{ выводов/с}} \quad 2{,}3 \times 1053 \text{ года},$$

что примерно в 1043 раза превышает возраст нашей Вселенной. Благодаря этому простому мысленному упражнению мы можем быстро понять, что нет никакой реальной возможности полностью исследовать это пространство поиска с помощью средств, доступных нам в этом мире. Более того, предположения, связанные с размером пространства поиска, были вполне реалистичными. Например, в шахматах для типичной ситуации существует более 30 возможных ходов, а партия, длящаяся 50 полуоборотов, относительно коротка.

Как же тогда могут быть хорошие шахматисты, а в наши дни еще и хорошие шахматные компьютеры? Как могло случиться, что математики находят доказательства теорем, в которых пространство поиска еще больше? Очевидно, мы, люди, используем разумные стратегии, которые значительно сокращают пространство поиска. Опытный шахматист, как и опытный математик, простым наблюдением за ситуацией сразу исключит многие действия как бессмысленные. Благодаря своему опыту он способен оценивать различные действия на предмет их полезности для достижения цели. Часто человек идет по ощущениям. Если спросить математика, как он нашел доказательство, он может ответить, что интуиция пришла к нему во сне. В сложных случаях многие врачи ставят диагноз чисто на ощупь, на основании всех известных симптомов. Особенно в сложных ситуациях часто отсутствует формальная теория поиска решения, гарантирующая оптимальное решение. В повседневных задачах, таких как поиск сбежавшего кота на рис. 6.1 на стр. 85, большую роль играет интуиция. Мы рассмотрим этот вид эвристического метода поиска в разд. 6.3 и дополнительно описать процессы, с помощью которых компьютеры могут, подобно людям, улучшать свои эвристические стратегии поиска путем обучения.

Однако сначала мы должны понять, как работает неосведомленный поиск, то есть слепое перепробование всех возможностей. Начнем с нескольких примеров.

Пример 6.1 С помощью головоломки с восьмерками, классического примера для алгоритмов поиска [Nil98, RN10], можно очень наглядно проиллюстрировать различные алгоритмы. Квадраты с номерами от 1 до 8 распределены в матрице 3 × 3, подобной матрице на рис. 6.2 на стр. 86. Цель состоит в том, чтобы достичь определенного упорядочения квадратов, например, в порядке возрастания по строкам, как показано на рис. 6.2 на стр . 86. На каждом этапе квадрат можно перемещать влево, вправо, вверх или вниз в пустое место. Таким образом, пустое пространство движется в соответствующем противоположном направлении. Для анализа пространства поиска удобно всегда смотреть на возможные движения пустого поля.

Дерево поиска начального состояния представлено на рис. 6.3 на стр. 86. Мы можем видеть, что коэффициент ветвления чередуется между двумя, тремя и четырьмя. Усредняя одновременно по двум уровням, мы получаем средний коэффициент ветвления1, равный $\sqrt{8} \approx 2{,}83$. Мы видим, что каждое состояние повторяется несколько раз на два уровня глубже, потому что при простом неинформированном поиске каждое действие может быть отменено на следующем шаге.

1Средний коэффициент ветвления дерева — это коэффициент ветвления, который имел бы дерево с постоянным коэффициентом ветвления, равной глубиной и равным количеством листовых узлов.

Machine Translated by Google

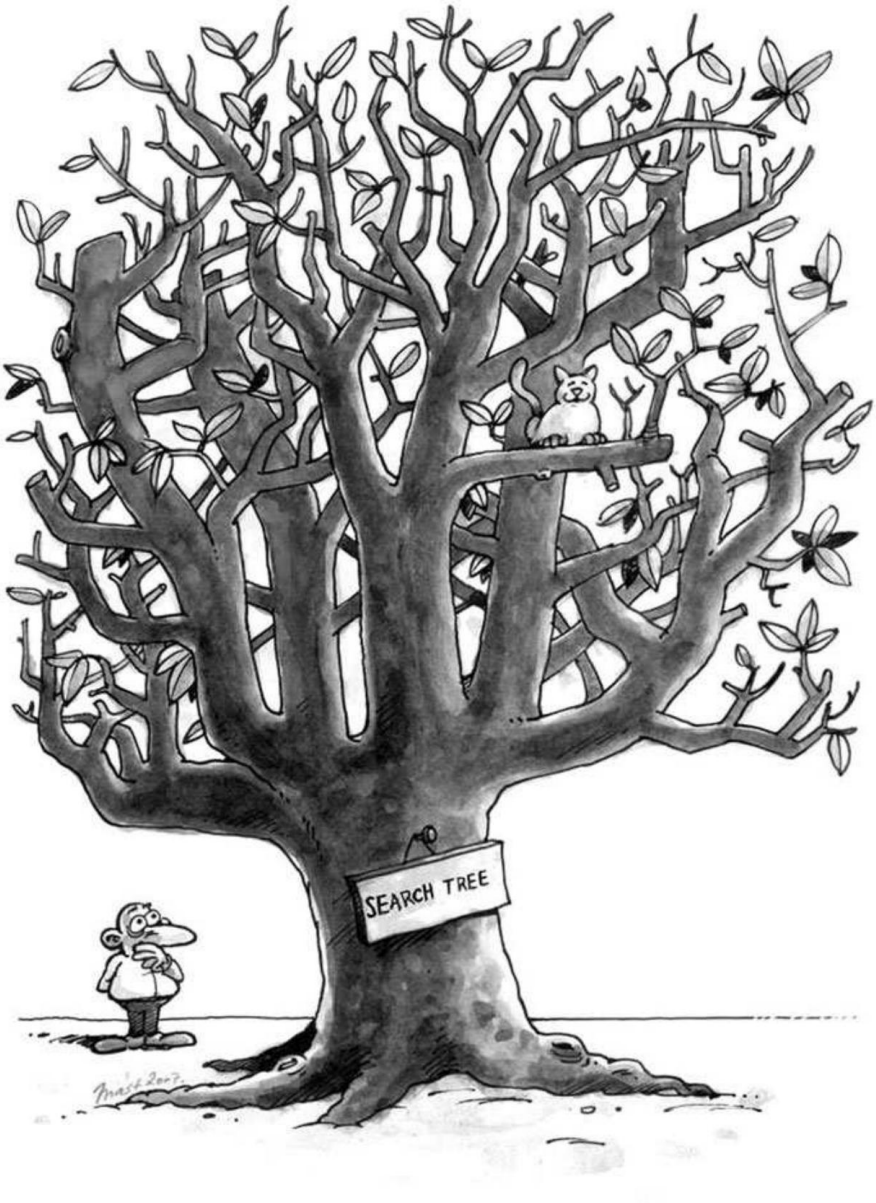


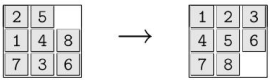
Рис. 6.1. Сильно подстриженное дерево поиска — или: «Где мой кот?»

Если мы запретим циклы длины 2, то для одного и того же начального состояния получим дерево поиска, представленное на рис. 6.4 на стр. 86. Средний коэффициент ветвления уменьшается примерно на 1 и становится равным 1,8.2.

2Для головоломки с восьмерками средний фактор ветвления зависит от начального состояния (см. упражнение 6.2 на стр. 110).

Machine Translated by Google

Рис. 6.2 Возможные начальное и целевое состояния головоломки с восьмеркой



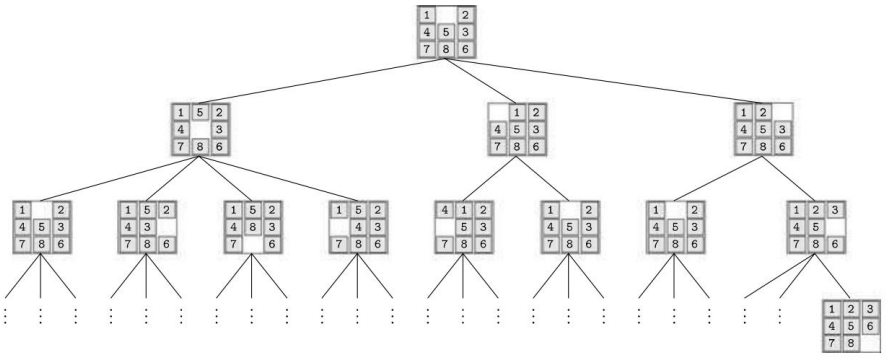


Рис. 6.3. Дерево поиска головоломки с восьмерками. Внизу справа представлено целевое состояние на глубине 3. Для экономии места другие узлы на этом уровне были опущены.

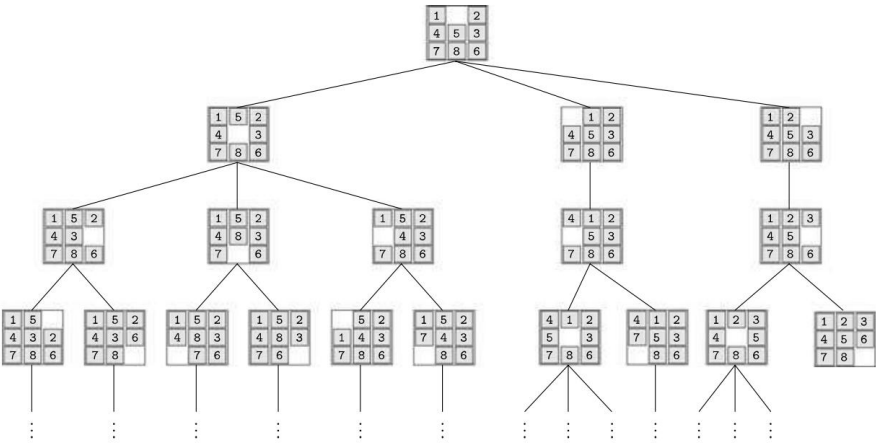


Рис. 6.4 Дерево поиска головоломки на 8 без циклов длины 2

Прежде чем мы начнем описывать алгоритмы поиска, необходимо ввести несколько новых терминов. Здесь мы имеем дело с задачами дискретного поиска. Находясь в состоянии s, действие a1 = a1(s). Другое состоянию s = a2(s). Таким образом, с действие может привести к состоянию s ′′, а может привести к новому другими словами: с Рекурсивное применение всех возможных действий ко всем состояниям, начиная с начального состояния, дает дерево поиска.

Machine Translated by Google

Определение 6.1 Задача поиска определяется следующими значениями Состояние:
Описание состояния мира, в котором поисковый агент находит
сам.
Начальное состояние: начальное состояние, в котором запущен поисковый агент.
Целевое состояние: если агент достигает целевого состояния, он завершает работу и выводит
решение (при желании).
Действия: всем агентам разрешены действия.
Решение: путь в дереве поиска из начального состояния в целевое.
Функция стоимости: присваивает значение стоимости каждому действию. Необходим для поиска
оптимального по стоимости решения.
Пространство состояний: набор всех состояний.

Применительно к 8-загадке получаем
Состояние: матрица 3×3 S со значениями 1, 2, 3, 4, 5, 6, 7, 8 (по одному разу) и одним пустым
квадрат.
Начальное состояние: Произвольное состояние.
Целевое состояние: Произвольное состояние, например, состояние, указанное справа на рис. 6.2 на стр. 86.
Действия: Движение пустого квадрата Sij влево (если j = 1), вправо (если j = 3), вверх (если i = 1), вниз (если i
= 3).
Функция стоимости: постоянная функция 1, поскольку все действия имеют одинаковую стоимость.
Пространство состояний: пространство состояний является вырожденным в областях, которые взаимно недостижимы
(упражнение 6.4 на стр. 110). Таким образом, существуют неразрешимые задачи с 8 головоломками.
Для анализа алгоритмов поиска необходимы следующие термины:

Определение 6.2 •
Количество последовательных состояний состояния s называется коэффициентом ветвления.
b(s) или b, если коэффициент ветвления постоянный.
• Эффективный коэффициент ветвления дерева глубины d с общим числом узлов n определяется
как коэффициент ветвления дерева с постоянным коэффициентом ветвления, равной глубиной
и равным n (см. Упражнение 6.3 на стр. 110). • Алгоритм поиска называется полным ,
если он находит решение для каждой решаемой задачи. Если полный алгоритм поиска
завершается, не найдя решения, то проблема неразрешима.

Для данной глубины d и количества узлов n эффективный коэффициент ветвления может быть
рассчитывается путем решения уравнения

$$n = \frac{б^{д+1} - 1}{б - 1} \tag{6.1}$$

Machine Translated by Google

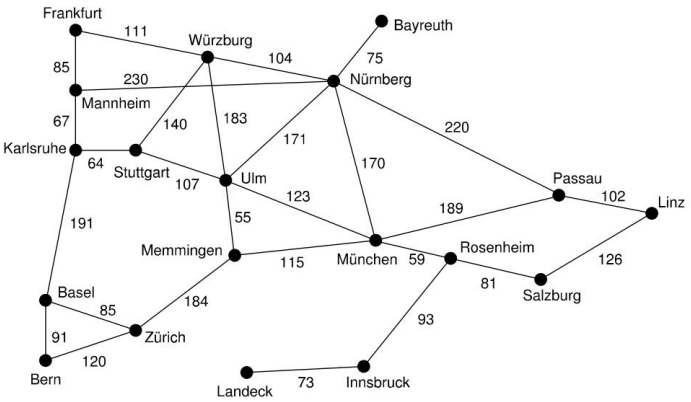


Рис. 6.5 Граф юга Германии как пример поисковой задачи с функцией стоимости

для b, потому что дерево с постоянным коэффициентом ветвления и глубиной d имеет в общей сложности

$$n = \sum_{i=0}^{d} b^i = \frac{b^{d+1} - 1}{b - 1} \tag{6.2}$$

узлы.

Для практического применения алгоритмов поиска конечных деревьев поиска последний уровень особенно важен, потому что

> Теорема 6.1. Для сильно ветвящихся конечных деревьев поиска с большим постоянным коэффициентом ветвления почти все узлы находятся на последнем уровне.

Простое доказательство этой теоремы рекомендуется читателю в качестве упражнения. (Упражнение 6.1 на стр. 110).

Пример 6.2. Нам дана карта, подобная той, что представлена на рис. 6.5, в виде графа с городами в качестве узлов и автомобильными дорогами между городами в виде взвешенных ребер с расстояниями. Ищем оптимальный маршрут из города А в город Б. Описание соответствующей схемы гласит

Состояние: Город как текущее местонахождение путешественника.

Начальное состояние: Произвольный город.

Целевое состояние: Произвольный город.

Действия: Путешествие из текущего города в соседний город.

Функция стоимости: расстояние между городами. Каждое действие соответствует ребру на графике с расстоянием в качестве веса.

Пространство состояний: Все города, то есть узлы графа.

Чтобы найти маршрут минимальной длины, необходимо учитывать затраты, поскольку они не являются постоянными, как в головоломке с восьмеркой.

Machine Translated by Google

Определение 6.3 Алгоритм поиска называется оптимальным , если он, если решение существует, всегда находит решение с наименьшей стоимостью.

Задача с 8 головоломками является детерминированной, что означает, что каждое действие ведет из состояния в уникальное последующее состояние. Кроме того, он является наблюдаемым, т. е. агент всегда знает, в каком состоянии он находится. При планировании маршрута в реальных приложениях не всегда задаются обе характеристики. Действие «Поехать из Мюнхена в Ульм» может, например, из-за аварии, привести к государству-преемнику «Мюнхен». Также может случиться так, что путешественник больше не знает, где он находится, потому что он заблудился. Сначала мы хотим игнорировать такого рода осложнения. Поэтому в этой главе мы будем рассматривать только детерминированные и наблюдаемые проблемы.

Такие задачи, как головоломка с восьмеркой, которые являются детерминированными и наблюдаемыми, делают планирование действий относительно простым, поскольку благодаря наличию абстрактной модели можно найти последовательность действий для решения проблемы, фактически не выполняя действия в реальном мире. . В случае головоломки с восьмерками нет необходимости перемещать квадраты в реальном мире, чтобы найти решение. Мы можем найти оптимальные решения с помощью так называемых автономных алгоритмов. Например, при создании роботов, которые должны играть в футбол, возникают совсем другие задачи. Здесь никогда не будет точной абстрактной модели действий. Например, робот, который бьет по мячу в определенном направлении, не может с уверенностью предсказать, куда будет двигаться мяч, потому что, среди прочего, он не знает, поймает мяч соперник или отклонит его. Здесь нужны онлайн-алгоритмы , которые принимают решения на основе сигналов датчиков в любой ситуации. Обучение с подкреплением, описанное в разд. 10, работает над оптимизацией этих решений на основе опыта.

6.2 Неинформированный поиск

6.2.1 Поиск в ширину

При поиске в ширину дерево поиска просматривается сверху вниз по алгоритму, приведенному на рис. 6.6 на стр. 90 , пока не будет найдено решение. Сначала каждый узел в списке узлов проверяется на предмет того, является ли он целевым узлом, и в случае успеха программа останавливается. В противном случае генерируются все преемники узла. Затем поиск рекурсивно продолжается в списке всех вновь сгенерированных узлов. Все повторяется до тех пор, пока не перестанут генерироваться преемники.

Этот алгоритм является общим. То есть он работает для произвольных приложений, если предусмотрены две специфичные для приложения функции «GoalReached» и «Successors». «GoalReached» вычисляет, является ли аргумент целевым узлом, а «Successors» вычисляет список всех узлов-преемников своего аргумента. На рис. 6.7 на стр. 90 показан снимок поиска в ширину.

Machine Translated by Google


```
ПОИСК В ШИРИНУ (Список узлов, Цель)
Новые узлы =
Для всех Node      NodeList
      Если цель достигнута (узел, цель)
            Возврат("Решение найдено", Узел)
      Новые узлы = Добавить (Новые узлы, Преемники (Узел))
Если НовыеУзлы =
      Возврат(ШИРИНА-ПЕРВЫЙ-ПОИСК(НовыеУзлы, Цель))
Еще
      Возврат("Нет решения")
```

Рис. 6.6 Алгоритм поиска в ширину

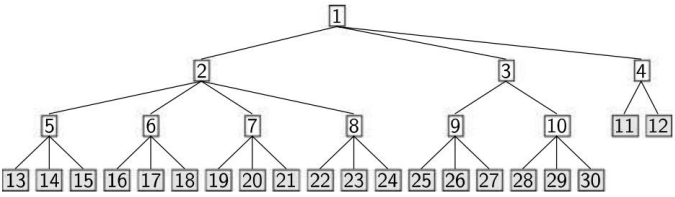


Рис. 6.7 Поиск в ширину при расширении узлов третьего уровня. Узлы нумеруются в порядке их создания. Преемники узлов 11 и 12 еще не созданы

Анализ Поскольку поиск в ширину полностью просматривает каждую глубину и достигает каждой глубины за конечное время, он является полным, если коэффициент ветвления b конечен. Оптимальное (то есть кратчайшее) решение находится, если затраты на все действия одинаковы (см. упражнение 6.7 на стр. 110). Время вычислений и объем памяти растут экспоненциально с глубиной дерева. Таким образом, для дерева с постоянным коэффициентом ветвления b и глубиной d общее время вычислений определяется выражением

$$\text{в} \cdot \sum_{\text{я}=0}^{\text{г}} \text{я б} \quad "=" \quad \frac{\text{б д+1} - 1}{\text{б} - 1} = \text{О(бд)}.$$

Хотя в памяти сохраняется только последний уровень, требования к объему памяти также составляют O(bd).

При скорости современных компьютеров, которые могут генерировать миллиарды узлов за считанные минуты, оперативная память быстро заполняется, и поиск завершается. Проблема не всегда находимого кратчайшего решения может быть решена с помощью так называемого поиска по единой стоимости, в котором узел с наименьшей стоимостью из отсортированных по возрастанию

Machine Translated by Google


```
ПОИСК В ГЛУБИНУ(Узел, Цель)

Если GoalReached(Узел, Цель) Возврат("Решение найдено")
Новые узлы = преемники (узел)
В то время как NewNodes =
    Результат = ГЛУБИНА-ПЕРВЫЙ-ПОИСК(Первый(НовыеУзлы), Цель)
    Если Результат = «Решение найдено» Возврат («Решение найдено»)
    Новые узлы = остальные (новые узлы)
Возврат("Нет решения")
```

Рис. 6.8 Алгоритм поиска в глубину. Функция «Первый» возвращает первый элемент списка, а «Остальное» — остальную часть списка.

список узлов всегда расширяется, а новые узлы сортируются. Таким образом, мы находим оптимальное решение. Однако проблема с памятью еще не решена. Решение этой проблемы обеспечивается поиском в глубину.

6.2.2 Поиск в глубину

При поиске в глубину в памяти одновременно хранится только несколько узлов. После расширения узла сохраняются только его преемники, а первый последующий узел немедленно расширяется. Таким образом, поиск быстро становится очень глубоким. Только когда у узла нет преемников и поиск на этой глубине завершается неудачей, следующий открытый узел расширяется путем возврата к последней ветви и так далее. Лучше всего это видно в элегантном рекурсивном алгоритме на рис. 6.8 и в дереве поиска на рис. 6.9 на стр. 92.

Анализ Поиск в глубину требует гораздо меньше памяти, чем поиск в ширину, поскольку на каждой глубине сохраняется не более b узлов. Таким образом, нам потребуется b · d ячеек памяти.

Однако поиск в глубину не является полным для бесконечно глубоких деревьев, потому что поиск в глубину приводит к бесконечному циклу, когда в крайней левой ветви нет решения. Поэтому вопрос поиска оптимального решения является устаревшим. Из-за бесконечного цикла невозможно установить ограничение на время вычислений. В случае дерева поиска конечной глубины с глубиной d всего генерируется около b узлов.

Г

Таким образом, время вычислений, как и при поиске в ширину, растет экспоненциально с глубиной.

Мы можем сделать дерево поиска конечным, установив предел глубины. Теперь, если в усеченном дереве поиска не найдено решения, тем не менее, могут быть решения за пределом.
Таким образом, поиск становится неполным. Однако есть очевидные идеи для завершения поиска.

Machine Translated by Google

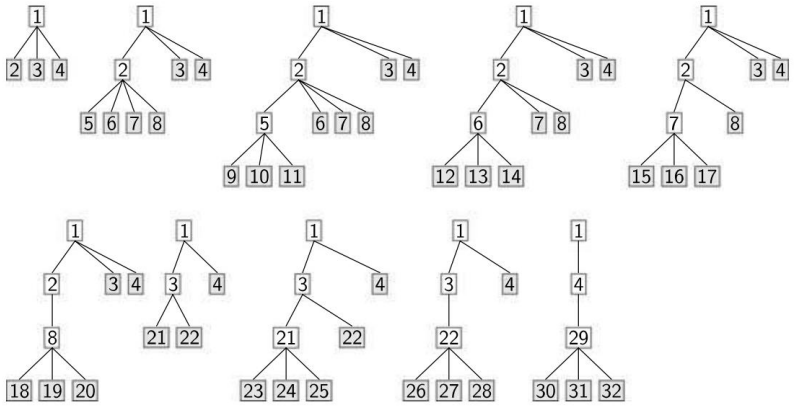


Рис. 6.9. Выполнение поиска в глубину. Все узлы на глубине три неуспешны и вызывают возврат. Узлы пронумерованы в порядке их создания.

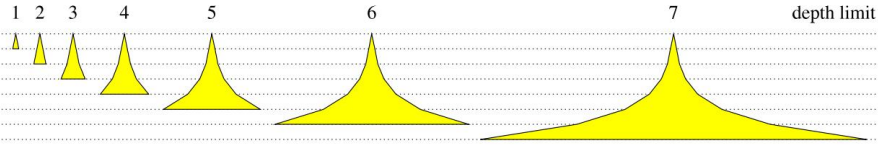


Рис. 6.10 Схематическое изображение развития дерева поиска при итеративном углублении с пределами от 1 до 7. Ширина дерева соответствует коэффициенту ветвления 2

6.2.3 Итеративное углубление

Мы начинаем поиск в глубину с пределом глубины, равным 1. Если решение не найдено, мы повышаем предел на 1 и начинаем поиск с начала и т. д., как показано на рис. 6.10 . Это итеративное повышение предела глубины называется итеративным углублением.

Мы должны дополнить программу поиска в глубину, приведенную на рис. 6.8 на стр. 91 , двумя дополнительными параметрами «Глубина» и «Предел». «Глубина» увеличивается на единицу при рекурсивном вызове, а заголовок цикла while заменяется на «While NewNodes = And Depth < Limit». Модифицированный алгоритм представлен на рис. 6.11 на стр. 93.

Анализ Требование к памяти такое же, как и при поиске в глубину. Можно возразить, что многократный перезапуск поиска в глубину с нулевой глубины приводит к большому количеству избыточной работы. Для больших факторов ветвления это не так. Теперь мы покажем, что сумма количества узлов всех глубин вплоть до последнего dmax 1 во всех обыскиваемых деревьях намного меньше, чем количество узлов в последнем обыскиваемом дереве.

Machine Translated by Google


```
ИТЕРАТИВНОЕУГЛУБЛЕНИЕ(Узел, Цель)
Предел глубины = 0
Повторить
    Результат = ПОИСК В ГЛУБИНЕ-В(Узел, Цель, 0,Поиск в глубину)
    Предел глубины = Предел глубины + 1
Пока Результат = «Решение найдено»
```

```
DEPTHFIRSTSEARCH-B(Узел, Цель, Глубина, Предел)
Если GoalReached(Узел, Цель) Возврат("Решение найдено")
Новые узлы = преемники (узел)
В то время как NewNodes =     And Depth < Limit
    Результат =
        DEPTHFIRSTSEARCH-B(First(NewNodes), Goal, Depth + 1, Limit)
    Если Результат = «Решение найдено» Возврат («Решение найдено»)
    Новые узлы = остальные (новые узлы)
Возврат("Нет решения")
```

Рис. 6.11 Алгоритм итеративного углубления, вызывающий слегка модифицированный поиск в глубину с ограничением глубины (TIEFENSUCHE-B)

Пусть Nb(d) — количество узлов дерева поиска с коэффициентом ветвления b и depth d и dmax — последняя искомая глубина. Последнее искомое дерево содержит

$$Nb(dmax) = \sum_{i=0}^{dmax} b^{i} = \frac{b^{dmax+1} - 1}{b - 1}$$

узлы. Все предварительно обыскиваемые деревья вместе имеют

$$Nb(d) = \sum_{d=1}^{dmax} \frac{b^{d+1} - 1}{b - 1} = \frac{1}{b - 1} \sum_{d=1}^{dmax} b^{d+1} - dmax + 1$$

$$= \frac{1}{b - 1} \sum_{d=2}^{dmax} b^{d} - dmax + 1$$

$$= \frac{1}{b - 1} \left(\frac{b^{dmax+1} - 1}{b - 1} - 1 - b - dmax + 1 \right)$$

$$= \frac{1}{b - 1} \frac{b^{dmax+1} - 1}{b - 1} - \frac{1}{b - 1} Nb(dmax)$$

Machine Translated by Google

Таблица 6.1 Сравнение алгоритмов неинформированного поиска. (*) означает, что оператор только true, учитывая постоянную стоимость действия. ds — максимальная глубина конечного дерева поиска.

	В ширину поиск	Единая стоимость поиск	В глубину поиск	Итеративный углубление
Полнота	Да	Да	Нет	Да
Оптимальное решение	Да (*)	Да	Нет	Да (*)
Время вычисления	д б	д б	или b ds	б Г
Использование памяти	б Г	б Г	бд	бд

узлы. Для b > 2 это меньше числа Nb(dmax) узлов в последнем дереве.
При b = 20 первые dmax 1 деревьев вместе содержат примерно = 1/19 $\frac{1}{б\text{-}1}$
количество узлов в последнем дереве. Время вычислений для всех итераций, кроме последним можно пренебречь.

Как и поиск в ширину, этот метод является полным и при постоянной стоимости для всех действий находит кратчайшее решение.

6.2.4 Сравнение

Описанные алгоритмы поиска приведены рядом в таблице 6.1.

Мы можем ясно видеть, что итеративное углубление является победителем этого теста, потому что оно получает лучшую оценку во всех категориях. По сути, из всех четырех представленных алгоритмов именно он является единственный практически пригодный.

У нас действительно есть победитель в этом тесте, хотя для реальных приложений это обычно не удается. Даже для головоломки на 15, старшего брата головоломки на 8 (см. Упражнение 6.4 на стр. 110), существует примерно 2 × 1013 различных состояний. Для нетривиальных систем вывода пространство состояний на много порядков больше. Как показано в Разд. 6.1, все вычислительные мощности мира уже не помогут. Вместо что необходимо, так это интеллектуальный поиск, который исследует только крошечную часть поиска пространстве и находит там решение.

6.3 Эвристический поиск

Эвристики — это стратегии решения проблем, которые во многих случаях находят решение быстрее. чем неинформативный поиск. Однако это не гарантируется. Эвристический поиск может требуют гораздо больше времени и могут даже привести к тому, что решение не будет найдено.

Мы, люди, успешно используем эвристические процессы для самых разных вещей. Когда покупая овощи в супермаркете, например, мы оцениваем различные варианты за фунт клубники, используя только несколько простых критериев, таких как цена, внешний вид, источник производства, и доверие к продавцу, а затем мы выбираем лучший вариант по нутровому ощущению. Теоретически может быть лучше подвергнуть клубнику базовому химический анализ перед принятием решения о покупке. Например, клубничный рис может быть отравлен. Если бы это было так, анализ стоил бы

Machine Translated by Google


```
ЭВРИСТИЧЕСКИЙ ПОИСК(Начало, Цель)

Список узлов = [Начало]
Пока правда
    Если NodeList =     Возврат («Нет решения»)
    Узел = Первый (Список узлов)
    NodeList = Отдых (NodeList)
    Если GoalReached(Узел, Цель) Возврат("Решение найдено", Узел)
    NodeList = SortIn (Преемники (Узел), NodeList)
```

Рис. 6.12 Алгоритм эвристического поиска

беда. Однако мы не проводим такого рода анализ, поскольку существует очень высокая вероятность того, что наш эвристический отбор увенчается успехом и быстро приведет нас к нашей цели — есть вкусную клубнику.

Эвристические решения тесно связаны с необходимостью принимать решения в режиме реального времени с ограниченными ресурсами. На практике хорошее решение, найденное быстро, предпочтительнее оптимального, но очень дорогого решения.

Функция эвристической оценки f(s) для состояний используется для математического моделирования эвристики. Цель состоит в том, чтобы без особых усилий найти решение поставленной задачи поиска с минимальными общими затратами. Обратите внимание, что существует тонкая разница между усилиями по поиску решения и общей стоимостью этого решения. Например, Google Maps может потратить полсекунды усилий, чтобы найти маршрут от мэрии в Сан-Франциско до Туолумн-Медоуз в национальном парке Йосемити, но поездка из Сан-Франциско до Туолумн-Медоуз на машине может занять четыре часа и немного денег. на бензин и т.д. (общая стоимость).

Далее мы изменим алгоритм поиска в ширину, добавив в него функцию оценки. Текущие открытые узлы больше не расширяются слева направо по строке, а в соответствии с их эвристическим рейтингом. Из множества открытых узлов первым всегда расширяется узел с минимальным рейтингом. Это достигается за счет немедленной оценки узлов по мере их раскрытия и сортировки их в списке открытых узлов.

Затем список может содержать узлы с разной глубиной дерева.

Поскольку эвристическая оценка состояний очень важна для поиска, с этого момента мы будем различать состояния и связанные с ними узлы. Узел кон

содержит состояние и дополнительную информацию, относящуюся к поиску, такую как его глубина в дереве поиска и эвристический рейтинг состояния. В результате функция «Преемники», генерирующая преемников (потомков) узла, также должна сразу вычислить для этих узлов-преемников их эвристические рейтинги как составляющую каждого узла. Мы определяем общий алгоритм поиска HEURISTICSEARCH на рис. 6.12.

Список узлов инициализируется начальными узлами. Затем в цикле удаляется первый узел из списка и проверяется, является ли он узлом решения. Если нет, то он будет расширен функцией «Преемники» и ее преемники добавлены в список

Machine Translated by Google

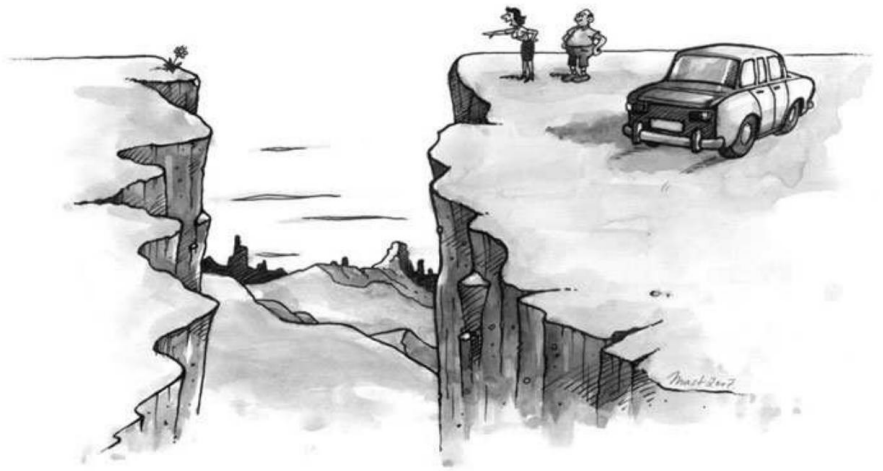


Рис. 6.13 Он: «Дорогой, подумай о стоимости топлива! Я сорву для тебя одну в другом месте. Она: «Нет, я хочу вон ту!»

с функцией «Сортировка». «SortIn(X,Y)» вставляет элементы из несортированного списка X в отсортированный по возрастанию список Y. В качестве ключа сортировки используется эвристический рейтинг. Таким образом гарантируется, что наилучший узел (т. е. узел с наименьшим эвристическим значением) всегда находится в начале списка. Поиск в глубину и в

ширину также является частным случаем функции HEURISTICSEARCH. Мы можем легко сгенерировать их, подключив соответствующую функцию оценки (упражнение 6.11 на стр. 111).

Лучшей эвристикой будет функция, которая вычисляет фактические затраты от каждого узла до цели. Однако для этого потребуется обход всего пространства поиска, что и должна предотвратить эвристика. Поэтому нам нужна эвристика, которую можно быстро и просто вычислить. Как найти такую эвристику?

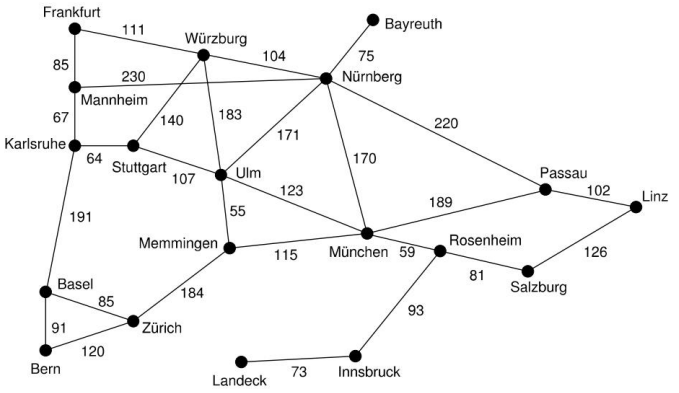
Интересной идеей поиска эвристики является упрощение задачи. Исходная задача настолько упрощена, что ее можно решить с небольшими вычислительными затратами. Затраты от состояния до цели в упрощенной задаче затем служат оценкой реальной задачи (см. рис. 6.13). Эту функцию оценки стоимости обозначим через h.

6.3.1 Жадный поиск

Представляется разумным выбрать состояние с наименьшим оценочным значением h (то есть с наименьшей оценочной стоимостью) из списка доступных на данный момент состояний.

3При сортировке нового узла из списка узлов может оказаться целесообразным проверить, доступен ли уже узел, и, если да, удалить дубликат.

Machine Translated by Google



Basel	204
Bayreuth	207
Bern	247
Frankfurt	215
Innsbruck	163
Karlsruhe	137
Landeck	143
Linz	318
München	120
Mannheim	164
Memmingen	47
Nürnberg	132
Passau	257
Rosenheim	168
Stuttgart	75
Salzburg	236
Würzburg	153
Zürich	157

Рис. 6.14 График города с указанием расстояний от всех городов до Ульма

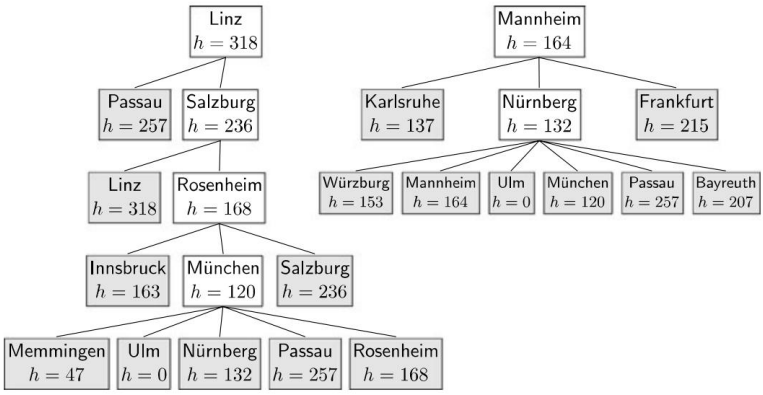
смету расходов затем можно использовать непосредственно в качестве функции оценки. Для вычисления в функции HEURISTICSEARCH положим f (s) = h(s). Это хорошо видно на примере планирования поездки (Пример 6.2 на стр. 88). Поставим задачу найти прямой путь из города в город (т. е. дальность полета) как упрощение задачи. Вместо поиска оптимального маршрута мы сначала определяем из каждого узла маршрут с минимальным расстоянием полета до цели. Мы выбираем Ульм в качестве пункта назначения. Таким образом, функция оценки стоимости становится

h(s) = расстояние полета от города s до Ульма.

Расстояния полета от всех городов до Ульма приведены на рис. 6.14 рядом с графиком.

Дерево поиска для старта в Линце представлено на рис. 6.15 на стр. 98 слева. Мы видим, что дерево очень стройное. Таким образом, поиск заканчивается быстро. К сожалению, этот поиск не всегда находит оптимальное решение. Например, этот алгоритм не может найти оптимальное решение при запуске в Мангейме (рис. 6.15 на стр. 98 справа). Путь Мангейм-Нюрнберг-Ульм имеет длину 401 км. Маршрут Мангейм-Карлсруэ-Штутгарт-Ульм будет значительно короче - 238 км. Когда мы наблюдаем за графиком, становится ясной причина этой проблемы. Нюрнберг на самом деле несколько ближе, чем Карлсруэ к Ульму, но расстояние от Мангейма до Нюрнберга значительно больше, чем от Мангейма до Карлсруэ. Эвристика только «жадно» смотрит вперед к цели вместо того, чтобы учитывать уже заложенную растяжку до текущего узла. Вот почему мы называем его жадным поиском.

Machine Translated by Google



Node	München	Innsbruck	Salzburg	Passau	Linz
heuristic rating	120	163	236	257	318

Рис. 6.15 Жадный поиск: из Линца в Ульм (слева) и из Мангейма в Ульм (справа). Структура данных списка узлов для левого дерева поиска, отсортированная по рейтингу узла до того, как задано расширение узла Mü nchen.

6.3.2 A -Поиск

Теперь мы хотим учесть затраты, которые накопились во время поиска до текущего узла s. Сначала определим функцию стоимости

g(s) = сумма начисленных затрат от начала до текущего узла,

затем добавьте к этому оценочную стоимость цели и получите в качестве эвристической функции оценки

$$f(s) = g(s) + h(s).$$

Теперь добавим еще одно небольшое, но важное требование.

Определение 6.4. Эвристическая функция оценки стоимости h(s), которая никогда не завышает фактическую стоимость от состояния s до цели, называется допустимой.

Функция ЭВРИСТИЧЕСКИЙПОИСК вместе с функцией вычисления f (s) = g(s) + h(s) и допустимой эвристической функцией h называется A -алгоритмом. Этот знаменитый алгоритм является полным и оптимальным. Таким образом, A всегда находит кратчайшее решение для каждой разрешимой задачи поиска. Мы объясним и докажем это в следующем обсуждении.

Сначала применим к примеру A -алгоритм. Мы ищем самый короткий путь из Франкфурта в Ульм.

Machine Translated by Google

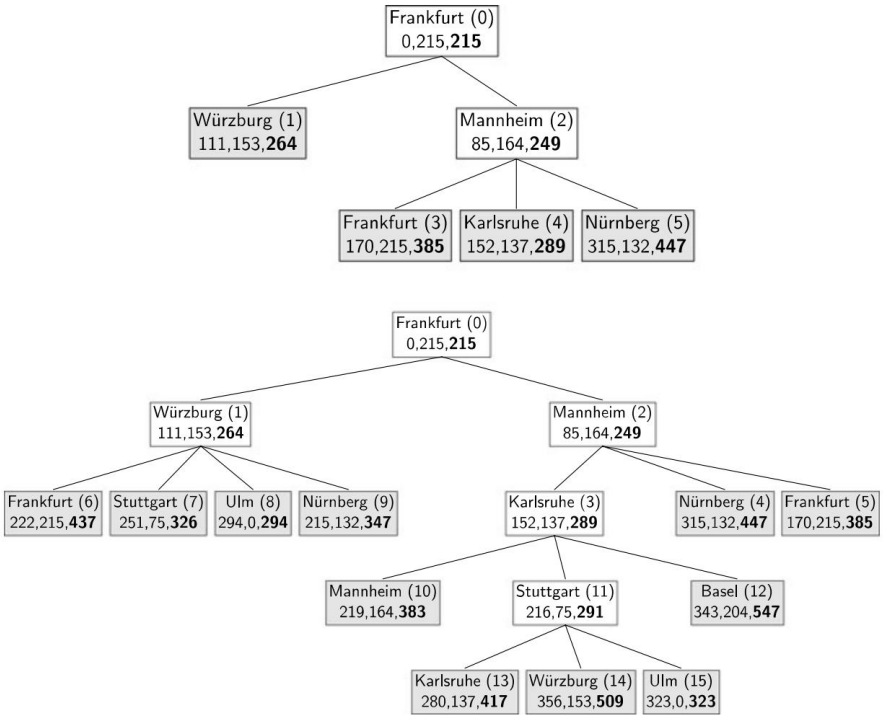


Рис. 6.16 Два снимка дерева поиска A для оптимального маршрута из Франкфурта в Ульм. В полях под названием города s мы показываем g(s), h(s), f (s). Цифры в скобках после названий городов показывают порядок, в котором узлы были сгенерированы функцией «Преемник».

В верхней части рис. 6.16 мы видим, что преемники Мангейма рождаются раньше наследников Вюрцбурга. Оптимальное решение Франкфурт-Вюрцбург-Ульм создается вскоре после этого на восьмом шаге, но еще не распознано как таковое.
Таким образом, алгоритм еще не завершается, потому что узел Карлсруэ (3) имеет лучшее (меньшее) значение f и, следовательно, опережает узел Ульм (8) в очереди. Только когда все значения f больше или равны значению узла решения Ulm (8), мы гарантируем, что имеем оптимальное решение. В противном случае потенциально может быть другое решение с меньшими затратами. Сейчас мы покажем, что это верно в общем случае.

Теорема 6.2. Алгоритм A оптимален. То есть он всегда находит решение с наименьшей общей стоимостью, если эвристика h допустима.

Доказательство. В алгоритме HEURISTICSEARCH каждый вновь сгенерированный узел s сортируется функцией «SortIn» в соответствии с его эвристическим рейтингом f(s). Таким образом, узел с наименьшим значением рейтинга находится в начале списка. Если узел l в точке

Machine Translated by Google

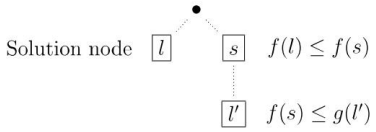


Рис. 6.17 Первый узел решения l, найденный A , никогда не имеет более высокой стоимости, чем другой произвольный́ узел l

начало списка является узлом решения, тогда никакой другой узел не имеет лучшего эвристического рейтинга. Для всех остальных узлов s верно, что f (l) f (s). Поскольку эвристика допустима, лучшего решения l найти не́льзя, даже после расширения всех остальных узлов (см. рис. 6.17). Написано официально:

g(l) = g(l) + h(l) = f(l) f(s) = g(s) + h(s) g(l′).

Первое равенство выполняется, потому что l является узлом решения с h(l) = 0. Второе равенство является определением f. Третье (не)равенство выполняется, потому что список открытых узлов отсортирован в порядке возрастания. Четвертое равенство снова является определением f . Наконец, последнее (не)равенство — это допустимость эвристики, которая никогда не завышает стоимость от вершины s до произвольного решения. Таким образом, было показано, что g(l) g(l′), т. е. найденное решение l является оптимальным.

6.3.3 IDA -Поиск

Поиск A наследует особенность поиска в ширину. Он должен сохранять много узлов в памяти, что может привести к очень высокому использованию памяти. Кроме того, список открытых узлов должен быть отсортирован. Таким образом, вставка узлов в список и удаление узлов из списка больше не могут выполняться за постоянное время, что немного увеличивает сложность алгоритма. На основе алгоритма пирамидальной сортировки мы можем структурировать список узлов как кучу с логарифмической временной сложностью для вставки и удаления узлов (см. [CLR90]).

Обе проблемы можно решить — аналогично поиску в ширину — путем итеративного углубления. Работаем с поиском в глубину и последовательно повышаем лимит. Однако вместо того, чтобы работать с пределом глубины, здесь мы используем предел для эвристической оценки f (s). Этот процесс называется IDA -алгоритмом.

6.3.4 Эмпирическое сравнение алгоритмов поиска

B A , или (альтернативно) ИДА , у нас есть алгоритм поиска со многими хорошими свойствами. Он полный и оптимальный. Таким образом, его можно использовать без риска. Самое главное, однако, заключается в том, что он работает с эвристикой и, следовательно, может значительно сократить время вычислений, необходимое для поиска решения. Мы хотели бы исследовать это эмпирически на примере головоломки с 8 элементами.

Для головоломки с восьмерками есть две простые допустимые эвристики. Эвристика h1 просто подсчитывает количество квадратов, которые находятся не на своем месте. Ясно, что эта эвристика допустима. Эвристика h2 измеряет манхэттенское расстояние. Для каждого

Machine Translated by Google

Таблица 6.2 Сравнение вычислительной стоимости неинформированного поиска и эвристического поиска решаемые задачи из 8 головоломок различной глубины. Измерения в шагах и секундах. Все значения средние значения для нескольких прогонов (см. последний столбец)

Глубина	Шаги итеративного углубления	Время [сек]	Алгоритм А				номер работает
			Эвристика h1		Эвристика h2		
			Шаги	Время [сек]	Шаги	Время [сек]	
2	20	0,003	3.0	0,0010	3.0	0,0010	10
4	81	0,013	5.2	0,0015	5,0	0,0022	24
6	806	0,13	10.2	0,0034	8.3	0,0039	19
8	6455	1,0	17,3	0,0060	12.2	0,0063	14
10	50512	7,9	48.1	0,018	22.1	0,011	15
12	486751	75,7	162,2	0,074	56,0	0,031	12
			ИДА				
14	-	-	10079.2	2,6	855,6	0,25	16
16	-	-	69386,6	19,0	3806,5	1,3	13
18	-	-	708780,0	161,6	53941,5	14,1	4

возвести в квадрат горизонтальные и вертикальные расстояния до местоположения этого квадрата в целевом состоянии складываются вместе. Затем это значение суммируется по всем квадратам. Например, Манхэттенское расстояние двух штатов



рассчитывается как

$$h2(s) = 1 + 1 + 1 + 1 + 2 + 0 + 3 + 1 = 10.$$

Допустимость манхэттенского расстояния также очевидна (см. упражнение 6.13 на стр. 111).

Описанные алгоритмы были реализованы в системе Mathematica. Для сравнения с неинформированным поиском, алгоритм А с двумя эвристиками h1 и h2 и итеративное углубление было применено к 132 случайно сгенерированным задачам из 8 головоломок. средние значения количества шагов и времени вычислений приведены в таблице 6.2. Мы видим, что эвристика значительно снижает стоимость поиска по сравнению с неинформативным поиском.

Если мы сравним итеративное углубление до А с h1 на глубине 12, например, это становится очевидным, что h1 уменьшает количество шагов примерно в 3000 раз, но время вычисления всего в 1023 раза. Это связано с более высокой стоимостью одного шаг для вычисления эвристики.

При ближайшем рассмотрении обнаруживается скачок количества шагов между глубиной 12 и глубина 14 в столбце для h1. Этот скачок нельзя объяснить только повторяющимся работа, выполненная IDA· Это происходит потому, что реализация алгоритма А

Machine Translated by Google

удаляет дубликаты идентичных узлов и тем самым сужает пространство поиска. Это невозможно с IDA , потому что он почти не сохраняет узлы. Несмотря на это, A больше не может конкурировать с IDA за пределами глубины 14, потому что стоимость сортировки в новых узлах сильно увеличивает время на шаг.

Вычисление эффективного коэффициента ветвления в соответствии с (6.1) на стр. 87 дает значение около 2,8 для неинформированного поиска. Это число согласуется со значением из разд. 6.1. Эвристика h1 снижает коэффициент ветвления примерно до 1,5, а h2 примерно до 1,3. Из таблицы видно, что небольшое уменьшение коэффициента ветвления с 1,5 до 1,3 дает нам большое преимущество во времени вычислений.

Таким образом, эвристический поиск имеет важное практическое значение, поскольку он может решать проблемы, недоступные для неосведомленного поиска.

6.3.5 Резюме

Из различных алгоритмов поиска для неинформированного поиска итеративное углубление является единственным практичным, потому что он завершен и может обойтись очень небольшим объемом памяти. Однако для сложных задач комбинаторного поиска даже итеративное углубление обычно не удается из-за размера пространства поиска. Эвристический поиск помогает здесь за счет уменьшения эффективного фактора ветвления. Алгоритм IDA , как и итеративное углубление, является полным и требует очень мало памяти.

Эвристика, естественно, дает значительное преимущество только в том случае, если эвристика «хорошая». При решении сложных поисковых задач актуальная задача разработчика состоит в построении эвристик, значительно снижающих эффективный коэффициент ветвления. В разд. В разделе 6.5 мы рассмотрим эту проблему, а также покажем, как можно использовать методы машинного обучения для автоматического создания эвристик.

В заключение остается отметить, что эвристики не имеют преимущества в производительности для неразрешимых задач, потому что неразрешимость проблемы может быть установлена только после того, как будет проведен поиск по всему дереву поиска. Для решаемых задач, таких как головоломка с восьмеркой, это означает, что все дерево поиска должно быть пройдено до максимальной глубины независимо от того, используется ли эвристика или нет. В этом случае эвристика всегда является недостатком, связанным с вычислительными затратами на оценку эвристики. Этот недостаток обычно можно оценить постоянным коэффициентом, не зависящим от размера задачи. Для неразрешимых задач, таких как доказательство формул PL1, дерево поиска может быть бесконечно глубоким. Это означает, что в неразрешимом случае поиск потенциально никогда не заканчивается. Подводя итог, можно сказать следующее: для решаемых задач эвристики часто резко сокращают время вычислений, но для неразрешимых задач стоимость с эвристиками может быть даже выше.

6.4 Игры с соперниками

Игры для двух игроков, такие как шахматы, шашки, Отелло и Го, являются детерминированными, потому что каждое действие (ход) приводит к одному и тому же дочернему состоянию при одном и том же родительском состоянии. Напротив, нарды недетерминированы, потому что их дочернее состояние зависит от

Machine Translated by Google

на результат броска игральной кости. Все эти игры можно наблюдать, потому что каждый игрок всегда знает полное состояние игры. Многие карточные игры, такие как покер, например, поддаются наблюдению лишь частично, потому что игрок не знает карты других игроков или знает о них лишь частично.

Проблемы, обсуждавшиеся до сих пор в этой главе, были детерминированными и наблюдаемыми. Далее мы рассмотрим игры, которые также являются детерминированными и наблюдаемыми. Кроме того, мы ограничимся играми с нулевой суммой. Это игры, в которых каждый выигрыш одного игрока означает проигрыш того же значения для противника. Сумма выигрыша и убытка всегда равна нулю. Это относится к упомянутым выше играм в шахматы, шашки, Отелло и Го.

6.4.1 Минимаксный поиск

Цель каждого игрока — совершать оптимальные ходы, ведущие к победе. В принципе, можно построить дерево поиска и полностью искать в нем (как в головоломке с восьмерками) серию ходов, которые приведут к победе. Однако есть несколько особенностей, на которые следует обратить внимание: 1. Эффективный фактор ветвления
в шахматах составляет примерно 30–35. В типичной игре с 50 ходами на игрока дерево поиска имеет более 30100 10148 листовых узлов .
Таким образом, нет возможности полностью изучить дерево поиска. Кроме того, в шахматы часто играют с ограничением по времени. Из-за этого требования к реальному времени поиск должен быть ограничен соответствующей глубиной дерева, например восемью полуходами.
Поскольку среди листовых узлов этого дерева с ограниченной глубиной обычно нет узлов-решений (т. е. узлов, завершающих игру), используется эвристическая функция оценки B для позиций на доске. Уровень игры программы сильно зависит от качества этой функции оценки.
Поэтому мы продолжим рассмотрение этого предмета в разд. 6.5.

2. Далее мы будем называть игрока, чью игру мы хотим оптимизировать, Максом, а его противника Мин. Ходы противника (Мина) заранее неизвестны, и, следовательно, фактическое дерево поиска также не известно. Эту проблему можно элегантно решить, если предположить, что противник всегда делает лучший ход, на который он способен. Чем выше оценка B(s) для позиции s, тем лучше позиция s для игрока Max и тем хуже для его противника Min. Макс пытается максимизировать оценку своих ходов, в то время как Мин делает ходы, которые приводят к как можно более низкой оценке.

Дерево поиска с четырьмя полуходами и оценками всех листьев показано на рис. 6.18 на стр. 104. Оценка внутреннего узла выводится рекурсивно как максимум или минимум его дочерних узлов, в зависимости от уровня узла.

6.4.2 Альфа-бета-обрезка

Переключаясь между максимизацией и минимизацией, мы можем сэкономить много работы в некоторых обстоятельствах. Альфа-бета-отсечение работает с поиском в глубину до заданного предела глубины. Таким образом, дерево поиска просматривается слева направо.

Machine Translated by Google

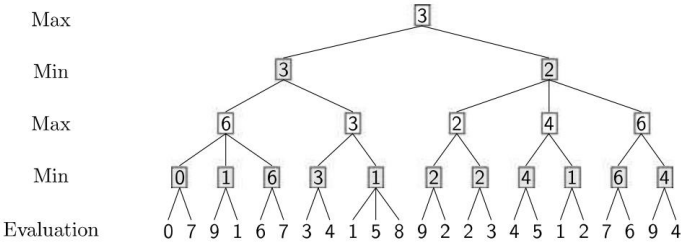


Рис. 6.18 Минимаксное игровое дерево с опережением на четыре полухода

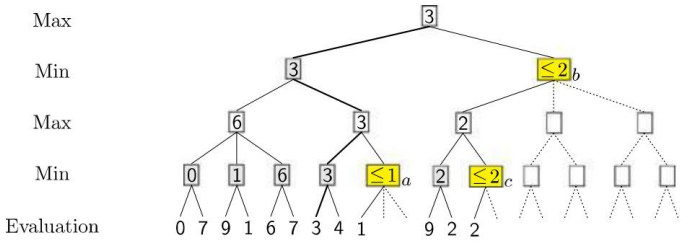


Рис. 6.19. Дерево игры альфа-бета с опережением на четыре полухода. Пунктирные части дерева не просматриваются, потому что они не влияют на конечный результат.

направо. Как и в минимаксном поиске, в минимальных узлах минимум генерируется из минимального значения последующих узлов, а в максимальных узлах аналогично.

максимум. На рис. 6.19 этот процесс изображен для дерева с рис. 6.18. В узле, отмеченном a, все остальные потомки могут быть проигнорированы после того, как первый потомок будет оценен как значение 1, потому что минимум обязательно будет 1. Он может даже стать еще меньше, но это не имеет значения, так как максимум уже 3 на один уровень выше. Независимо от того, как сложится оценка оставшихся преемников, максимум сохранит значение 3. Аналогично дерево будет обрезано в узле b. Поскольку первый потомок b имеет значение 2, минимальное значение, которое должно быть сгенерировано для b, может быть только меньше или равно 2. Но максимум в корневом узле уже наверняка будет 3.
Это не может быть изменено значениями 2. Таким образом, оставшиеся поддеревья b могут быть обрезаны.

То же самое относится и к узлу c. Однако соответствующий максимальный узел является не прямым родителем, а корневым узлом. Это можно обобщить. • В каждом конечном узле вычисляется оценка. • Для каждого максимального узла текущее наибольшее дочернее значение сохраняется в α. • Для каждого минимального узла текущее наименьшее дочернее значение сохраняется в β. • Если в минимальном узле k текущее значение β α, то поиск по k можно закончить.

Здесь α — наибольшее значение максимального узла на пути от корня до k. • Если в максимальном узле l текущее значение α β, то поиск по l можно закончить.

Здесь β — наименьшее значение минимального узла на пути от корня до l.

Machine Translated by Google


```
ALPHABETAMAX(Узел, α, β)

Если DepthLimitReached(Node) Return(Rating(Node))
Новые узлы = преемники (узел)
В то время как NewNodes =
    α = Максимум (α, ALPHABETAMIN (Первый (Новые узлы), α, β))
    Если α  β Возврат (β)
    Новые узлы = остальные (новые узлы)
Возврат(α)
```

```
АЛФАБЕТАМИН(Узел, α, β)

Если DepthLimitReached(Node) Return(Rating(Node))
Новые узлы = преемники (узел)
В то время как NewNodes =
    β = Минимум (β, ALPHABETAMAX (Первый (Новые узлы), α, β))
    Если β  α Возврат (α)
    Новые узлы = остальные (новые узлы)
Возврат (β)
```

Рис. 6.20 Алгоритм альфа-бета-поиска с двумя функциями ALPHABETAMIN и AL PHABETAMAX

Алгоритм, представленный на рис. 6.20 , является расширением поиска в глубину с двумя функциями, которые вызываются попеременно. Он использует значения, определенные выше для α и β.

Сложность Время вычислений, сэкономленное альфа-бета-отсечением, сильно зависит от порядка обхода дочерних узлов. В худшем случае сокращение альфа-бета не дает никаких преимуществ. Для постоянного коэффициента ветвления b количество nd конечных узлов, которые необходимо оценить на глубине d, равно

$$n = b^d.$$

В лучшем случае, когда потомки максимальных узлов сортируются по убыванию, а потомки минимальных узлов сортируются по возрастанию, эффективный коэффициент ветвления уменьшается до $\sqrt{b}$. В шахматах это означает существенное снижение эффективного фактора ветвления с 35 до примерно 6. Тогда только

$$n = b^{\sqrt{d}} = b^{d/2}$$

Machine Translated by Google

будут созданы листовые узлы. Это означает, что предел глубины и, следовательно, горизонт поиска удваиваются при сокращении альфа-бета. Однако это верно только в случае оптимально отсортированных потомков, потому что рейтинги дочерних узлов неизвестны на момент их создания. Если дочерние узлы отсортированы случайным образом, то коэффициент ветвления уменьшается до b 3/4 , а количество конечных узлов до

$$n = b^{\frac{3}{4}d} .$$

При той же вычислительной мощности шахматный компьютер, использующий сокращение альфа-бета, может, например, вычислить восемь полуходов вперед вместо шести с эффективным коэффициентом ветвления около 14. Тщательный анализ с выводом этих параметров можно найти в [Горох84].

Чтобы удвоить глубину поиска, как упоминалось выше, нам нужно, чтобы дочерние узлы были оптимально упорядочены, что на практике не так. В противном случае поиск был бы ненужным. С помощью простого трюка мы можем получить относительно хороший порядок узлов. Мы связываем альфа-бета-обрезку с итеративным углублением сверх предела глубины. Таким образом, на каждом новом пределе глубины мы можем получить доступ к рейтингам всех узлов предыдущих уровней и упорядочить преемников в каждой ветви. Таким образом, мы достигаем эффективного коэффициента ветвления примерно от 7 до 8, что недалеко от теоретического оптимума 35 [Nil98].

6.4.3 Недетерминированные игры

Минимаксный поиск можно обобщить на все игры с недетерминированными действиями, такие как нарды. Каждый игрок делает бросок перед своим ходом, на который влияет результат броска кубиков. Таким образом, в дереве игры теперь есть три типа уровней в последовательности.

Макс, кости, Мин, кости, . . . ,

где каждый узел броска костей разветвляется шестью способами. Поскольку мы не можем предсказать значение кубика, мы усредняем значения всех бросков и проводим поиск, как описано, со средними значениями из [RN10].

6,5 Эвристические функции оценки

Как нам найти хорошую эвристическую функцию оценки для задачи поиска?
Здесь принципиально два подхода. Классический способ использует знания людей-экспертов. Перед инженером по знаниям обычно стоит сложная задача формализовать неявные знания эксперта в виде компьютерной программы.
Теперь мы хотим показать, как можно упростить этот процесс на примере шахматной программы.

На первом этапе экспертов спрашивают о наиболее важных факторах при выборе хода. Затем предпринимается попытка количественной оценки этих факторов. Мы получаем список соответствующих функций или атрибутов. Затем они (в простейшем случае) объединяются в

Machine Translated by Google

линейная функция оценки B(s) для позиций, которая может выглядеть так:

$$B(s) = a1 \cdot \text{материал} + a2 \cdot \text{пешечная_структура} + a3 \cdot \text{безопасность короля} + a4 \cdot \text{конь_в_центре} + a5 \cdot \text{слон_диагональное_покрытие} + \cdots, \quad (6.3)$$

где «материал» является наиболее важной характеристикой и рассчитывается по

$$\text{материал} = \text{материал (собственная_команда)} — \text{материал (противник)}$$

с

$$\begin{aligned}\text{материал(команда)} = {} & \text{количество_пешек(команда)} \cdot 100 + \text{количество_рыцарей(команда)} \cdot 300 \\ & + \text{количество_слонов(команда)} \cdot 300 + \text{количество_ладей(команда)} \cdot 500 \\ & + \text{num_queens(команда)} \cdot 900\end{aligned}$$

Почти все шахматные программы делают одинаковую оценку материала. Тем не менее, есть большие различия для всех других функций, которые мы не будем здесь рассматривать [Fra05, Lar00].

На следующем шаге необходимо определить веса ai всех признаков. Они устанавливаются интуитивно после обсуждения с экспертами, а затем меняются после каждой игры на основе положительного и отрицательного опыта. Тот факт, что этот процесс оптимизации очень дорог и, кроме того, линейная комбинация признаков очень ограничена, предполагает использование машинного обучения.

6.5.1 Изучение эвристики

Теперь мы хотим автоматически оптимизировать веса ai оценочной функции B(s) из (6.3). При таком подходе эксперта спрашивают только о соответствующих функциях f1(s), . . . , fn(s) для игрового состояния s. Затем используется процесс машинного обучения с целью найти функцию оценки, максимально близкую к оптимальной. Мы начинаем с первоначально заданной функции оценки (определяемой в процессе обучения), а затем позволяем шахматной программе играть. В конце игры по результату (победа, поражение или ничья) выводится рейтинг. На основании этого рейтинга функция оценки изменяется с целью в следующий раз делать меньше ошибок. В принципе, то же самое, что делает разработчик, теперь автоматически выполняется в процессе обучения.

Как бы просто это ни звучало, на практике это очень сложно. Центральная проблема улучшения рейтинга позиций на основе выигранных или проигранных матчей известна сегодня как проблема присвоения кредитов . На самом деле у нас есть рейтинг в конце игры, но нет рейтинга для отдельных ходов. Таким образом, агент выполняет множество действий, но до самого конца не получает ни положительной, ни отрицательной обратной связи. Как тогда он должен соотносить эту обратную связь со многими действиями, предпринятыми в прошлом? И как ему улучшить свои действия в этом случае? Такими вопросами занимается захватывающая молодая область обучения с подкреплением (см. раздел 10).

Machine Translated by Google

Большинство лучших шахматных компьютеров в мире сегодня все еще работают без методов машинного обучения. Этому есть две причины. С одной стороны, алгоритмы обучения с подкреплением, разработанные до сих пор, требуют большого времени вычислений при больших пространствах состояний. С другой стороны, созданные вручную эвристики высокопроизводительных шахматных компьютеров уже сильно оптимизированы. Это означает, что только очень хорошая система обучения может привести к улучшениям. Вероятно, в ближайшие десять лет наступит время, когда обучающийся компьютер станет чемпионом мира.

6.6 Современное состояние

Для оценки качества процессов эвристического поиска я хотел бы повторить
Определение Элейн Рич [Ric83]:

> Искусственный интеллект — это наука о том, как заставить компьютеры делать то, в чем люди на данный момент лучше.

Едва ли найдется лучший тест для определения интеллекта компьютерной программы, чем прямое сравнение компьютера и человека в такой игре, как шахматы, шашки, нарды или го.

В 1950 году Клод Шеннон, Конрад Цузе и Джон фон Нейман представили первые шахматные программы, которые, однако, либо не могли быть реализованы, либо требовали много времени для реализации. Всего несколько лет спустя, в 1955 году, Артур Сэмюэл написал программу, которая играла в шашки и могла улучшать собственные параметры посредством простого процесса обучения. Для этого он использовал первый программируемый логический компьютер IBM 701. Однако по сравнению с современными шахматными компьютерами он имел доступ к большому количеству архивных партий, в которых каждый ход оценивался экспертами. Таким образом, программа улучшила свою функцию оценки. Чтобы добиться дальнейших улучшений, Сэмюэл заставил свою программу играть против самой себя. Он решил проблему присвоения кредита простым способом. Для каждой отдельной позиции во время игры он сравнивает оценку функции B(s) с оценкой, вычисленной с помощью альфа-бета-сокращения, и соответствующим образом изменяет B(s). В 1961 году его шашечная программа обыграла четвертого лучшего игрока в шашки в США. Благодаря этой новаторской работе Сэмюэл, несомненно, опередил свое время почти на 30 лет.

Только в начале девяностых, когда появилось обучение с подкреплением, Джеральд Терсауро создал обучающую программу игры в нарды под названием TD-Gammon, которая играла на уровне чемпионов мира (см. раздел 10).

Сегодня существует несколько шахматных программ, которые играют на уровне гроссмейстера, и некоторые из них продаются на коммерческой основе для ПК. Прорыв произошел в 1997 году, когда Deep Blue от IBM победил чемпиона мира по шахматам Гэри Каспарова со счетом 3,5 партии против 2,5. Deep Blue мог в среднем рассчитывать на 12 полуходов вперед с альфа-бета-сокращением и эвристической оценкой позиции.

Одним из самых мощных шахматных компьютеров на сегодняшний день является Hydra, параллельный компьютер, принадлежащий компании из Объединенных Арабских Эмиратов. Программное обеспечение было разработано учеными Кристианом Доннингером (Австрия) и Ульфом Лоренцем (Германия), а также великим чемпионом Германии по шахматам Кристофером Лутцем. Hydra использует 64 параллельных Xeon

Machine Translated by Google

процессоры с вычислительной мощностью около 3 ГГц и 1 ГБ памяти каждый. Для функции оценки положения каждый процессор имеет сопроцессор FPGA (программируемая пользователем вентильная матрица). Таким образом становится возможным оценивать 200 миллионов позиций в секунду даже при использовании дорогостоящей функции оценки.

С помощью этой технологии Hydra может в среднем рассчитывать примерно на 18 ходов вперед. В особых, критических ситуациях горизонт поиска может растянуться даже до 40 полуходов. Ясно, что такой горизонт выходит за рамки того, что могут сделать даже великие чемпионы, поскольку Гидра часто делает ходы, которые великие чемпионы не могут понять, но которые в конце концов приводят к победе. В 2005 году Гидра победила гроссмейстера Майкла Адамса, занявшего седьмое место, со счетом 5,5–0,5 гейма.

Hydra использует немного специальных знаний о шахматах из учебников, а скорее альфа-бета-поиск с относительно общими, хорошо известными эвристиками и хорошей оценкой позиции, закодированной вручную. В частности, Гидра не способна к обучению. Доработки проводятся между играми разработчиками. Таким образом, компьютер по-прежнему освобождается от необходимости учиться. Hydra также не имеет специальных алгоритмов планирования. Тот факт, что Hydra работает без обучения, указывает на то, что, несмотря на многие достижения, исследования в области машинного обучения по-прежнему необходимы. Как упоминалось выше, мы более подробно остановимся на этом в разд. 10 и гл. 8.

В 2009 году система Pocket Fritz 4, работающая на КПК, выиграла шахматный турнир Copa Mercosur в Буэнос-Айресе с девятью победами и одной ничьей против 10 отличных шахматистов, трое из которых были гроссмейстерами. Несмотря на то, что информации о внутренней структуре программного обеспечения не так много, эта шахматная машина представляет собой тенденцию от чистой вычислительной мощности к большей интеллектуальности. Эта машина играет на уровне гроссмейстера и сравнима, если не лучше, с Гидрой.
По словам разработчика Pocket Fritz Станислава Цукрова [Wik10], Pocket Fritz со своим шахматным поисковиком HIARCS 13 ищет менее 20 000 позиций в секунду, что медленнее Hydra примерно в 10 000 раз. Это приводит к выводу, что HIARCS 13 определенно использует лучшую эвристику для уменьшения эффективного фактора ветвления, чем Hydra, и поэтому его можно назвать более интеллектуальным, чем Hydra. Кстати, HIARCS — это сокращение от Higher Intelligence Auto Response Chess System.

Даже если вскоре ни у одного человека не будет шансов против лучших шахматных компьютеров, перед ИИ еще много вызовов. Иди, например. В этой старой японской игре, играемой на квадратной доске с 361 квадратом, 181 белым и 180 черными камнями, эффективный коэффициент ветвления составляет около 300. После четырех полуходов уже имеется около 8×109 позиций . Все известные классические процессы поиска по дереву игр не имеют шансов против хороших игроков в го на этом уровне сложности. Эксперты сходятся во мнении, что здесь нужны «по-настоящему интеллектуальные» системы. Комбинаторное перечисление всех возможностей — неверный метод. Гораздо больше нужны процессы, способные распознавать закономерности на доске, следить за долгосрочными разработками и быстро принимать «интуитивные» решения. Как и в распознавании объектов на сложных изображениях, люди все еще намного опережают компьютерные программы. Мы обрабатываем изображение в целом очень параллельно, в то время как компьютер обрабатывает миллионы пикселей один за другим и с трудом улавливает, что существенного в полноте пикселей. Программа «Многоликая игра в го» умеет распознавать 1100 различных паттернов и знает 200 игровых стратегий. Но все программы Go по-прежнему имеют серьезные проблемы с распознаванием того, является ли группа камней живой или мертвой, или где их сгруппировать между ними.

Machine Translated by Google

6.7 Упражнения

Упражнение 6.1

(a) Докажите теорему 6.1 на стр. 88, другими словами, докажите, что для дерева с большим постоянным коэффициентом ветвления b почти все узлы находятся на последнем уровне на глубине d. (b) Покажите, что это не всегда так, когда эффективный фактор ветвления велик и непостоянен.

Упражнение 6.2

(a) Вычислите средний фактор ветвления для головоломки с восьмерками без проверки циклов. Средний коэффициент ветвления — это коэффициент ветвления, который будет иметь дерево с равным количеством узлов на последнем уровне, постоянным коэффициентом ветвления и равной глубиной. (b) Рассчитайте средний коэффициент ветвления для головоломки с 8 элементами для неинформированного поиска, избегая при этом циклов длины 2.

Упражнение 6.3

(a) В чем разница между средним и эффективным коэффициентом ветвления (Определение 6.2 на стр. 87)?

(b) Почему эффективный коэффициент ветвления лучше подходит для анализа и сравнения времени вычислений алгоритмов поиска, чем средний коэффициент ветвления? (c) Покажите, что для сильно разветвленного дерева с n узлами и глубиной d эффективный коэффициент ветвления $\bar{b}$ примерно равен среднему коэффициенту ветвления и, таким образом, равен $\sqrt[d]{n}$.

Упражнение 6.4

(a) Вычислите размер пространства состояний для 8-головоломки, для аналогичной 3-головоломки (2 × 2-матрица), а также для 15-головоломки (4 × 4-матрица). (b) Докажите, что граф состояний, состоящий из состояний (узлов) и действий (ребер) для 3-головоломки, распадается на два связных подграфа, между которыми нет связей.

Упражнение 6.5 С помощью поиска в ширину для головоломки с восьмерками найдите путь (вручную) из начального узла  к целевому узлу  .

Упражнение 6.6

(a) Запрограммируйте поиск в ширину, поиск в глубину и итеративное углубление на выбранном вами языке и проверьте их на примере с 8 головоломками. (b) Почему нет особого смысла использовать поиск в глубину для головоломки с восьмерками?

Упражнение 6.7

(a) Покажите, что поиск в ширину при неизменной стоимости всех действий гарантирован. найти кратчайшее решение.

Machine Translated by Google

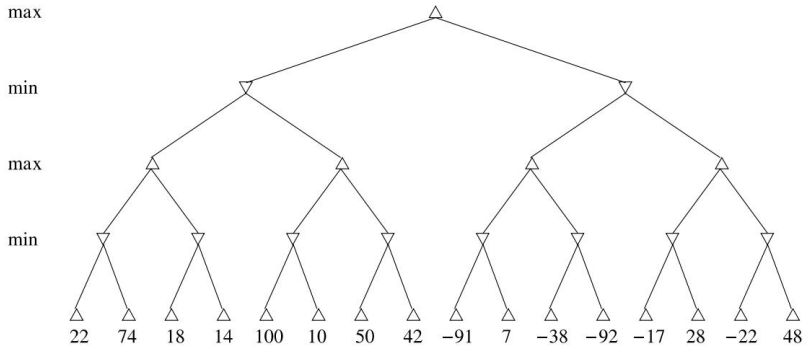


Рис. 6.21 Минимаксное дерево поиска

б) Покажите, что это не так для переменных издержек.

Упражнение 6.8. Используя поиск A для головоломки с восьмерками, найдите (вручную) путь из начальный узел  к целевому узлу 

(a) с использованием эвристики h1 (раздел 6.3.4).

(b) с использованием эвристики h2 (раздел 6.3.4).

Упражнение 6.9 Постройте дерево поиска A для графа города на рис. 6.14 на стр. 97 и используйте расстояние полета до Ульма в качестве эвристики. Начните в Берне с Ульма в качестве пункта назначения. Позаботьтесь о том, чтобы каждый город появлялся только один раз на пути.

Упражнение 6.10 Программа A поиск на выбранном вами языке программирования используя эвристики h1 и h2 , и протестируйте их на примере с 8 головоломками.

Упражнение 6.11. Приведите эвристическую оценочную функцию для состояний, с которыми HEURIS TICSEARCH может быть реализован как поиск в глубину и как поиск в ширину.

Упражнение 6.12 Какая связь между изображением пары в каньоне на рис. 6.13 на стр. 96 и допустимой эвристикой?

Упражнение 6.13. Покажите, что эвристики h1 и h2 для головоломки с восьмерками из разд. 6.3.4 .

Упражнение 6.14

(a) Дерево поиска для игры двух игроков показано на рис. 6.21 с рейтингами всех листовых узлов. Используйте минимаксный поиск с обрезкой α-β слева направо. Вычеркните все непосещенные узлы и дайте оптимальную результирующую оценку для каждого внутреннего узла. Отметьте выбранный путь. (b) Проверьте себя с помощью апплета П. Уинстона [Win].

Machine Translated by Google

Machine Translated by Google

Рассуждение с неуверенностью

7

Мы уже показывали в разд. 4 с проблемой Твити, что двухзначная логика приводит к проблемам в повседневных рассуждениях. В этом примере утверждения Твити — пингвин, Пингвины — птицы и Все птицы могут летать приводят к (семантически неверному) выводу, что Твити может летать. Теория вероятностей предоставляет язык, на котором мы можем формализовать утверждение. Почти все птицы умеют летать и делать на его основе выводы. Теория вероятностей — проверенный метод, который мы можем здесь использовать, потому что неопределенность в отношении того, могут ли птицы летать, можно хорошо смоделировать с помощью назначения вероятности. Мы покажем, что такие утверждения, как 99% всех птиц умеют летать, вместе с вероятностной логикой приводят к правильным выводам.

Рассуждения в условиях неопределенности с ограниченными ресурсами играют большую роль в повседневных ситуациях, а также во многих технических приложениях ИИ. В этих областях очень важны эвристические процессы, как мы уже обсуждали в гл. 6. Например, мы используем эвристические приемы при поиске парковочного места в городском потоке. Одних только эвристик часто бывает недостаточно, особенно когда требуется быстрое решение при неполном знании, как показано в следующем примере. Пешеход переходит улицу, и быстро приближается автомобиль. Чтобы предотвратить серьезную аварию, пешеход должен реагировать быстро. Он не способен заботиться о полной информации о состоянии мира, которая понадобилась бы ему для алгоритмов поиска, обсуждавшихся в гл. 6. Следовательно, он должен прийти к оптимальному решению при заданных ограничениях (мало времени и мало потенциально неопределенных знаний). Если он будет думать слишком долго, это будет опасно. В этой и многих подобных ситуациях (см. рис. 7.1 на стр. 114) необходим метод рассуждений при неопределенных и неполных знаниях.

Мы хотим исследовать различные возможности рассуждений в условиях неопределенности на примере простого медицинского диагноза. Если пациент испытывает боль в правой нижней части живота и повышенное количество лейкоцитов (лейкоцитов), это вызывает подозрение на аппендицит. Мы моделируем эту связь, используя логику высказываний по формуле

Боль в животе справа внизу Лейкоциты> 10000 Аппендицит

В. Эртель, Введение в искусственный интеллект, Темы бакалавриата по компьютерным наукам, DOI 10.1007/978-0-85729-299-5_7, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google



Рис. 7.1 «Давайте просто сядем и подумаем, что делать!»

Если мы тогда знаем, что

Боль в животе справа внизу Лейкоциты> 10000

верно, то мы можем использовать modus ponens для вывода аппендицита. Эта модель явно слишком грубая. В 1976 году Шортлифф и Бьюкенен признали это при создании своей медицинской экспертной системы MYCIN [Sho76]. Они разработали исчисление с использованием так называемых факторов достоверности, которые позволили представить достоверность фактов и правил.

Правилу A B присваивается коэффициент достоверности β. Семантика правила A β B определяется через условную вероятность P (B|A) = β. В приведенном выше примере правило может читаться

Боль в животе справа внизу Лейкоциты> 10000 0,6 Аппендицит.

Для рассуждений с такого рода формулами они использовали исчисление для связи факторов правил. Однако оказалось, что с помощью этого исчисления можно получить противоречивые результаты

Machine Translated by Google

Как обсуждалось в гл. 4 также были попытки решить эту проблему с помощью немонотонной логики и логики по умолчанию, которые, однако, в итоге не увенчались успехом. Теория Демпстера-Шефера приписывает функцию доверия Bel(A) логическому утверждению A, значение которого дает степень доказательства истинности A. Но даже этот формализм имеет недостатки, которые показаны в [Pea88] с использованием варианта Пример Твити. Даже нечеткая логика, которая прежде всего успешна в теории управления, демонстрирует значительные недостатки при рассуждениях в условиях неопределенности в более сложных приложениях [Elk93].

Примерно с середины 1980-х годов теория вероятности оказывала все большее влияние на ИИ [Pea88, Che85, Whi96, Jen01]. В области рассуждений с помощью байесовских сетей или субъективной вероятности он занял прочное место среди успешных методов ИИ. Вместо импликации, как это известно в логике (материальной импликации), здесь используется условная вероятность, которая значительно лучше моделирует повседневные причинные рассуждения. Рассуждения с вероятностями сильно выигрывают от того факта, что теория вероятностей — это хорошо зарекомендовавшая себя ветвь математики, существующая сотни лет.

В этой главе мы выберем элегантную, но несколько необычную для учебника точку входа в эту область. После краткого введения в наиболее важные основы, необходимые здесь для рассуждений с вероятностью, мы начнем с простого, но важного примера для рассуждений с неопределенными и неполными знаниями. Совершенно естественным, почти убедительным образом мы придем к методу максимальной энтропии (MaxEnt). Затем покажем полезность этого метода на практике с помощью медицинской экспертной системы LEXMED. Наконец, мы познакомимся с широко распространенными в настоящее время рассуждениями с помощью байесовских сетей и покажем взаимосвязь между этими двумя методами.

7.1 Вычисления с вероятностями

Читатель, знакомый с теорией вероятностей, может пропустить этот раздел. Всем остальным мы адресуем быстрый ввод в курс дела и порекомендуем несколько подходящих учебников, таких как [Ros09, FPP07].

Вероятность особенно хорошо подходит для моделирования рассуждений в условиях неопределенности. Одна из причин этого заключается в том, что вероятности интуитивно легко интерпретировать, что можно увидеть на следующем элементарном примере.

Пример 7.1. Для одного броска игрового кубика (эксперимент) вероятность события «выпадение шестерки» равна 1/6, а вероятность события «выпадение нечетного числа» равна 1/2.

Определение 7.1. Пусть Ω — конечное множество событий для эксперимента. Каждое событие $\omega \subseteq \Omega$ представляет собой возможный исход броска. Если эти события $\omega_i \subseteq \Omega$ взаимно исключают друг друга, но охватывают все возможные исходы попытки, то они называются элементарными событиями.

Machine Translated by Google

Пример 7.2. Для одного броска одного игрового кубика

$$Ом = \{1, 2, 3, 4, 5, 6\}$$

потому что никакие два из этих событий не могут произойти одновременно. Роллинг четного числа ({2, 4, 6}), следовательно, является элементарным событием, и при этом не выбрасывается число, меньшее пяти ({1, 2, 3, 4}), потому что {2, 4, 6} {1, 2, 3, 4} = {2, 4} = .

Учитывая два события A и B, A B также является событием. Сам Ω обозначается определенным событие, а пустое множество невозможное событие.

В дальнейшем мы будем использовать нотацию пропозициональной логики для операций над множествами. То есть для множества A B мы пишем A B. Это не только синтаксическая трансформация, скорее, это также семантически правильно, потому что пересечение двух множеств определено как

$$x \quad A \quad B \quad x \quad A \quad x \quad B.$$

Поскольку это семантика A B, мы можем и будем использовать это обозначение. Это тоже истинно для других операций объединения и дополнения множества, и мы будем, как показано в В следующей таблице используйте для них также нотацию пропозициональной логики.

Установить обозначение	Пропозициональная логика	Описание
A B	A B	Пересечение/и
A B	A Б	Союз/или
A^{-}	¬A	Дополнение/отрицание
Ом	ж	Определенное событие / правда
	ф	Невозможное событие/ложь

Используемые здесь переменные (например, A, B и т. д.) называются случайными величинами. в теории вероятностей. Здесь мы будем использовать только дискретные случайные переменные с конечными доменами. Переменная face_number для броска костей является дискретной со значениями 1, 2, 3, 4, 5, 6. Вероятность выпадения пятерки или шестерки равна 1/3. Это может быть описано

$$P(\text{число_лица} \quad \{5, 6\}) = P(\text{число_лица} = 5 \quad \text{число_лица} = 6) = 1/3.$$

Предполагается, что понятие вероятности дает нам описание столь же объективное, как нашей «веры» или «убеждения» в отношении результата эксперимента. Все числа в интервале [0, 1] должны быть возможны, где 0 — вероятность невозможное событие и 1 вероятность определенного события. Мы пришли к этому из следующее определение.

Machine Translated by Google

Опред еление 7.2. Пуст ь Ω = {ω1, ω2,...,ωn} к онеч но. Пред поч т ит ель ного элемент арного соб ыт ия не сущест вует , а эт о оз нач ает , ч т о мы пред полаг аем симмет рию, свя з анную с ч аст от ой поя вления к аждого элемент арного соб ыт ия . Т огда вероя т ност ь P (A) соб ыт ия A равна

$$P(A) = \frac{|A|}{|\Omega|} \text{ "=" } \frac{\text{Количество благоприятных случаев для } A}{\text{Количество возможных случаев}}.$$

От сюда немед ленно след ует , ч т о к аждое элемент арное соб ыт ие имеет вероя т ност ь $1/|\Omega|$. Т реб ование равновероя т ност и элемент арных соб ыт ий наз ывается пред положением Лапласа , а выч исля емые т ак им образом вероя т ност и наз ываются вероя т ност я ми Лапласа.

Эт о опред еление дост игает своего пред ела, к огда ч исло элемент арных соб ыт ий ст ановится беск онеч ным. Поск оль к у з дес ь мы рассмат риваем т оль к о к онеч ные прост ранст ва соб ыт ий, эт о не пред ст авля ет проб лемы Для описания соб ыт ий мы испол з уем переменные с соот вет ст вующим к олич ест вом з нач ений. Например, переменная eye_color может принимать з нач ения з еленый, синий, к орич невый. eye_color = blue з ат ем описывает соб ыт ие, пот ому ч т о мы имеем дело с пред ложением со з нач ениями ист ин Для ст ои ин ых (булевых) переменных сама переменная уже я вля ет ся пред ложением. Здес ь дост ат оч но, например, написать P (JohnCalls) вмест о P (JohnCalls = t).

Пример 7.3 Сог ласно эт ому опред елению вероя т ност ь выпадения ч ет ного ч исла равна

$$P(\text{номер_лица} \in \{2, 4, 6\}) = \frac{|\{2, 4, 6\}|}{|\{1, 2, 3, 4, 5, 6\}|} \text{ "=" } \frac{3}{6} \text{ "=" } \frac{1}{2}.$$

След ующие важные правила след ут непосред ст венно из опред еления .

Т еорема 7.1 1.

P (Ω) = 1.

2. P( ) = 0, з нач ит , вероя т ност ь невоз можного соб ыт ия равна 0.

3. Для попарно иск лю ч ающих соб ыт ий A и B верно, ч т о P (A B) = P (A) + П (Б).

4. Для дв ух дополнит ель ных соб ыт ий A и ¬A верно, ч т о P (A) + P (¬A) = 1.

5. Для прои з воль ных соб ыт ий A и B верно, ч т о P (A B) = P (A) + P (B) P (A B).

6. Для A B верно, ч т о P (A) P (B).

7. Е сли A1,...,An — элемент арные соб ыт ия , т о $\sum_{i=1}^{n} P(A_i) = 1$ (нормировк а условие).

Machine Translated by Google

Выражение P (A B) или, что то же самое, P (A, B) обозначает вероятность событий A B. Нас часто интересуют вероятности всех элементарных событий, то есть всех комбинаций всех значений переменных A и B. Для бинарные переменные A и B это P (A, B), P (A, ¬B), P (¬A, B), P (¬A, ¬B). Мы называем вектор

$$(\Pi(A, B), \Pi(A, \neg B), \Pi(\neg A, B), \Pi(\neg A, \neg B))$$

состоящее из этих четырех значений распределение или совместное распределение вероятностей переменные A и B. Сокращением для этого является P (A, B). Распределение в случае две переменные можно красиво представить в виде таблицы(матрицы), представленной в виде следует :

P(A, B) B = w		B = ϕ
A = ш	П (A, Б)	P (A, ¬B)
A = ϕ	P(¬A,B) P (¬A,¬B)	

Для d переменных X1,...,Xd с n значениями каждая распределение имеет значения P (X1 = x1,...,Xd = xd) и x1,...,xd , каждое из которых принимает n различных значений. Таким образом, распределение может быть представлено в виде d-мерной матрицы с общим n Г элементы В силу условия нормировки из теоремы7.1 на стр. 117, однако один из этих n Г значения избыточны а распределение характеризуется по н Г 1 уникальное значение.

7.1.1 Условная вероятность

Пример 7.4. На улице Ландсдаун в Бостоне измеряется скорость 100 автомобилей. Для каждого из измерения также отмечается , является ли водитель студентом. результаты

Событие	Частота	Относительная частота
Наблюдаемый автомобиль	100	1
Водитель - студент (C)	30	0,3
Слишком высокая скорость (G)	10	0,1
Водитель студент и превышение скорости (S G)	5	0,05

Мы ставим вопрос: учащиеся превышают скорость чаще, чем в среднем по сын, или чем не студенты?1

Ответ дает вероятность

$$P(Г|C) = \frac{|\text{Водитель-студент и превышает скорость}|}{|\text{Водитель – студент}|} "=" \frac{5}{30} "=" \frac{1}{6} \quad 0{,}17$$

1 Вычисленные вероятности могут быть использованы для продолжающихся предложений только в том случае, если измеренная выборка (100 автомобилей) является репрезентативным. В противном случае могут быть только предположения о наблюдаемых 100 транспортных средствах. быть сделано.

Machine Translated by Google

за превышение скорости при условии, что водитель является студентом. Это явно отличается от априорной вероятности превышения скорости P(G) = 0,1. Для априорной вероятности пространство событий не ограничивается дополнительными условиями.

Определение 7.3. Для двух событий A и B вероятность P (A|B) для A при условии B (условная вероятность) определяется как

$$P(A|B) = \frac{П(A \quad B)}{П(Б)}.$$

В примере 7.4 мы видим, что в случае конечного пространства событий условная вероятность P (A|B) может пониматься как вероятность A B, когда мы рассматриваем только событие B, то есть как

$$P(A|B) = \frac{|A \quad B|}{|Б|}.$$

Эту формулу можно легко вывести, используя Определение 7.2 на стр. 117.

$$P(A|B) = \frac{П(A \quad B)}{П(Б)} \text{ "=" } \frac{\frac{|A \quad B| \ |}{Ом|}}{\frac{|Б| \ |}{Ом|}} \text{ "=" } \frac{|A \quad B|}{|Б|}.$$

Определение 7.4. Если для двух событий A и B

$$P(A|B) = P(A),$$

то эти события называются независимыми.

Таким образом, A и B независимы, если вероятность события A не зависит от события B.

Теорема 7.2. Для независимых событий A и B из определения следует, что

$$P(A \quad B) = P(A) \cdot P(B).$$

Machine Translated by Google

Пример 7.5. При броске двух костей вероятность выпадения двух шестерок равна 1/36, если две кости независимы, потому что

$$P(D1 = 6 \quad D2 = 6) = P(D1 = 6) \cdot P(D2 = 6) = \frac{1}{6} \cdot \frac{1}{6} \overset{"="}{=} \frac{1}{36},$$

где первое уравнение верно только тогда, когда две игральные кости независимы. Если, например, благодаря какой-то магической силе кубик 2 всегда совпадает с кубиком 1, то

$$P(D1 = 6 \quad D2 = 6) = \frac{1}{6}.$$

Правило цепи

Решение определения условной вероятности для $P(A \quad B)$ приводит к так называемому правилу произведения

$$P(A \quad B) = P(A|B)P(B),$$

которое мы сразу же обобщаем на случай n переменных. Повторным применением указанного выше правила получаем цепное правило

$$\begin{aligned} & P(X1,...,Xn) \\ & = P(Xn|X1,...,Xn \quad 1) \cdot P(X1,...,Xn \quad 1) \\ & = P(Xn|X1,...,Xn \quad 1) \cdot P(Xn \quad 1|X1,...,Xn \quad 2) \cdot P(X1,...,Xn \quad 2) \\ & = P(Xn|X1,...,Xn \quad 1) \cdot P(Xn \quad 1|X1,...,Xn \quad 2) \cdot \cdots \cdot P(X2|X1) \cdot P(X1) \\ & \overset{"="}{=} \prod_{i=1}^{n} P(Xi|X1 ...,Xi \quad 1), \end{aligned} \tag{7.1}$$

с помощью которого мы можем представить распределение как произведение условных вероятностей. Поскольку цепное правило выполняется для всех значений переменных X1,...,Xn, оно было сформулировано для распределения с использованием символа P.

Маргинализация

Поскольку $A \quad (A \quad B) \quad (A \quad \neg B)$ верно для двоичных переменных A и B

$$P(A) = P((A \quad B) \quad (A \quad \neg B)) = P(A \quad B) + P(A \quad \neg B).$$

Путем суммирования по двум значениям B переменная B исключается. Аналогично, для произвольных переменных X1,...,Xd переменная , например Xd , может быть исключена суммированием по всем их переменным, и мы получим

$$P(X1 = x1,...,Xd \quad 1 = xd \quad 1) = \sum_{xd} P(X1 = x1,...,Xd \quad 1 = xd \quad 1, Xd = xd).$$

Machine Translated by Google

Применение эт ой формулыназ ьвает ся маргинализ ац ией. Эт о суммирование может прод олжат ь ся с переменньми X1,...,Xd 1 д о т ех пор, пок а не ост анет ся т оль к о од на переменная . Маргинализ ац ия можно применит ь и к распред елениюP(X1,...,Xd). Получ енное распред еление P(X1,...,Xd 1) наз ьвает ся маргиналь ньм распред елением. Эт о сравнимо с проек ц ией пря моуг оль ного параллелепипеда на плоск уюповерхност ь . Зд есь т рехмерньй объек т рисует ся на ребре или «к рае» к убоида, т . е. на д вумерном множест ве. В обоих случ ая х раз мерност ь умень шает ся на ед иниц у.

Пример 7.6 Наблюд аем совок упност ь всех боль ньх, обрат ивших ся к врач у с ост рым боль в живот е. У к аждог о пац иент а из меря ет ся к олич ест во лейк оц ит ов, к от орое я вля ет ся пок аз ат елем от носит ель ное сод ержание лейк оц ит ов в к рови. Мыопред еля ем переменную Лейк о, ч т о верно т огд а и т оль к о т огд а, к огд а з нач ение лейк оц ит ов боль ше 10 000. Эт от свид ет ель ст вует об инфек ц ии в орг анизме. В прот ивном случ ае мыопред еля ем переменнуюApp, к от орая сообщает нам, ест ь ли у пац иент а аппенд иц ит , т о ест ь инфиц ированный аппенд ик с. Распред еление P(App,Leuko) эт их д вух переменньх пред ст авлено в след ующей т аблиц е:

P(App,Leuko)	App	¬App	Ит ог о
Леук о	0,23	0,31	0,54
¬Леук о	0,05	0,41	0,46
Общий	0,28	0,72	1

В послед ней ст рок е д ана сумма по ст рок ам, а в послед нем ст олбц е сумма д ают ся ст олбц ы Эт и суммыд ост иг ают ся з а сч ет маргинализ ац ии. Например, мы ч ит ат ь

$$P(\text{лейк о}) = P(\text{апп}, \text{лейк о}) + P(\neg\text{апп}, \text{лейк о}) = 0{,}54.$$

Данное распред еление P(App,Leuko) может быт ь получ ено из опроса немец к их врач ей, например. От сюд а мыможем вьн ислит ь условнуювероя т ност ь

$$P(\text{лейк о} \mid \text{приложение}) = \frac{P(\text{Леук о}, \text{приложение})}{П(\text{приложение})} = 0{,}82$$

ч т о говорит нам о т ом, ч т о ок оло 82% всех случ аев аппенд иц ит а приводя т к выс ок ому з нач ениолейк оц ит ов. Под обньe з нач ения публик уют ся в мед иц инск ой лит ерат уре. Од нак о условная вероя т ност ь P (App |Leuko), к от орая на самом д еле была быраз д о полез нее д ля д иаг ност ик и аппенд иц ит , не опублик овано. Чт обыпоня т ь эт о, мыснач ала вьвед ем прост ой, но оч ень важная формула.

Т еорема Байеса

Замена A и B в опред елении 7.3 на ст р. 119 привод ит к

$$P(A \mid B) = \frac{П(A \quad B)}{П(Б)} \quad \text{и} \quad P(B \mid A) = \frac{П(A \quad B)}{П(A)}.$$

Machine Translated by Google

Решая два уравнения для P (A B) и приравнивая их, мы получаем теорему Байеса

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}, \tag{7.2}$$

которые мы сразу применим к проблеме аппендицита и получим

$$P(\text{Приложение} \mid \text{Лейко}) = \frac{P(\text{лейко}|\text{приложение}) \cdot P(\text{приложение})}{P(\text{Лейко})} \text{ "=" } \frac{0,82 \cdot 0,28}{0,54} = 0,43. \tag{7.3}$$

Теперь почему публикуется P (Leuko|App) , а не P (App|Leuko)?

Предполагая, что аппендицит влияет на организм человека независимо от расы, P (Leuko|App) является универсальным значением, которое верно во всем мире. В (7.3) мы видим, что P (App| Leuko) не является универсальным, поскольку на это значение влияют априорные вероятности P (App) и P (Leuko). Каждый из них может варьироваться в зависимости от жизненных обстоятельств. Например, P (Leuko) зависит от того, много или мало инфекционных заболеваний в популяции. Это значение потенциально может значительно варьироваться между тропиками и более холодными регионами. Тем не менее, теорема Байеса позволяет легко вывислить P (App|Leuko), который важен для диагностики, исходя из общепризнанного значения P (Leuko|App).

Прежде чем мы углубимся в этот пример и расширим его до медицинской экспертной системы для аппендицита, мы должны сначала ввести необходимый вероятностный вывод. механизм.

7.2 Принцип максимальной энтропии

Теперь мы покажем на примере вывода, что исчисление для рассуждений в условиях неопределенности может быть реализовано с использованием теории вероятностей. Однако вскоре мы увидим, что прототипные вероятностные тропы быстро заканчиваются. В частности, когда слишком мало знаний для решения необходимых уравнений, нужны новые идеи. Американский физик Э. Т. Джейнс провел пионерскую работу в этой области в 1950-х годах. Он утверждал, что при недостающих знаниях можно максимизировать энтропию желаемого распределения вероятностей, и применил этот принцип ко многим примерам в [Jay57, Jay03]. Затем этот принцип получил дальнейшее развитие [Che83, Nil86, Kan89, KK92] и теперь является зрелым и может применяться технологически, что мы покажем на примере проекта LEXMED в разд. 7.3.

Machine Translated by Google

7.2.1 Правило вывода для вероятностей

Мы хотим вывести правило вывода для неопределенного знания, аналогичное modus ponens. Из знания предложения A и правила A B следует сделать вывод B. В лаконичной формулировке это звучит

$$\frac{A,\ A \quad B}{B}.$$

Обобщение вероятностных правил дает

$$\frac{P(A) = \alpha,\ P(B|A) = \beta}{P(B) = ?}.$$

Пусть заданы два вероятностных правила α, β и желаемое значение P(B). Путем маргинализации мы получаем желаемое маргинальное распределение

$$P(B) = P(A,B) + P(\neg A,B) = P(B|A) \cdot P(A) + P(B|\neg A) \cdot P(\neg A).$$

Три значения P (A), P (¬A), P (B|A) в правой части известны, но значение P (B|¬A) неизвестно. Мы не можем сделать точное утверждение о P (B) с помощью юклассической теории вероятностей, но в лучшем случае мы можем оценить P (B) P (B|A) · P (A).

Теперь рассмотрим распределение

$$P(A, B) = (P(A, B), P(A, \neg B), P(\neg A, B), P(\neg A, \neg B))$$

и ввести для краткости четыре неизвестных

$$p1 = P(A, B),\ p2 = P(A,\neg B),\ p3 = P(\neg A, B),\ p4 = P(\neg A, \neg B).$$

Эти четыре параметра определяют распределение. Если все они известны, то можно вычислить любую вероятность для двух переменных A и B. Для вычисления четырех неизвестных необходимы четыре уравнения. Одно уравнение уже известно в виде условия нормировки

$$p1 + p2 + p3 + p4 = 1.$$

Следовательно, необходимы еще три уравнения. Однако в нашем примере известны только два уравнения.

По заданным значениям P (A) = α и P (B|A) = β вычисляем

$$P(A, B) = P(B|A) \cdot P(A) = \alpha\beta$$

Machine Translated by Google

и

$$P(A) = P(A,B) + P(A,\neg B).$$

Отсюда мы можем составить следующую систему уравнений и решить ее, насколько это возможно:

$$p1 = \alpha\beta, \tag{7.4}$$

$$p1 + p2 = a, \tag{7,5}$$

$$p1 + p2 + p3 + p4 = 1, \tag{7,6}$$

(7.4) в (7.5): $p2 = \alpha \quad \alpha\beta = \alpha(1 \quad \beta),$ (7.7)

(7.5) в (7.6): $p3 + p4 = 1 \quad \alpha.$ (7.8)

Таким образом, вероятности p1, p2 для интерпретаций (A, B) и (A,¬B) известны, но для значений p3, p4 остается только одно уравнение. Чтобы прийти к определенному решению, несмотря на это недостающее знание, мы меняем нашу точку зрения. Мы используем данное уравнение в качестве ограничения для решения оптимизационной задачи.

Мы ищем распределение p (для переменных p3, p4), которое максимизирует энтропию

$$H(p) = \sum_{i=1}^{n} p_i \ln p_i = \quad p3 \ln p3 \quad p4 \ln p4 \tag{7.9}$$

при ограничении p3 +p4 = 1 α (7.8). Почему именно функция энтропии должна быть максимизирована? Поскольку у нам не хватает информации о распределении, ее нужно каким-то образом добавить. Мы могли бы исправить специальное значение, например, p3 = 0,1. Однако лучше определить значения p3 и p4 так, чтобы добавляемая информация была минимальной. Мы можем показать (раздел 8.4.2 и [SW76]), что энтропия измеряет неопределенность распределения с точностью до постоянного множителя. Тогда отрицательная энтропия является мерой количества информации, содержащейся в распределении. Максимизация энтропии минимизирует информативность распределения. Для наглядности функция энтропии для двумерного случая представлена графически на рис. 7.2 на стр.

Для определения максимума энтропии при ограничении p3 + p4 1 + α = 0 воспользуемся методом множителей Лагранжа [Ste07]. Функция Лагранжа читается

$$L = \quad p3 \ln p3 \quad p4 \ln p4 + \lambda(p3 + p4 \quad 1 + \alpha).$$

Взяв частные производные по p3 и p4, получим

$$\frac{\partial L}{\partial p3} = \quad \ln p3 \quad 1 + \lambda = 0,$$

$$\frac{\partial L}{\partial p4} = \quad \ln p4 \quad 1 + \lambda =$$

0

Machine Translated by Google

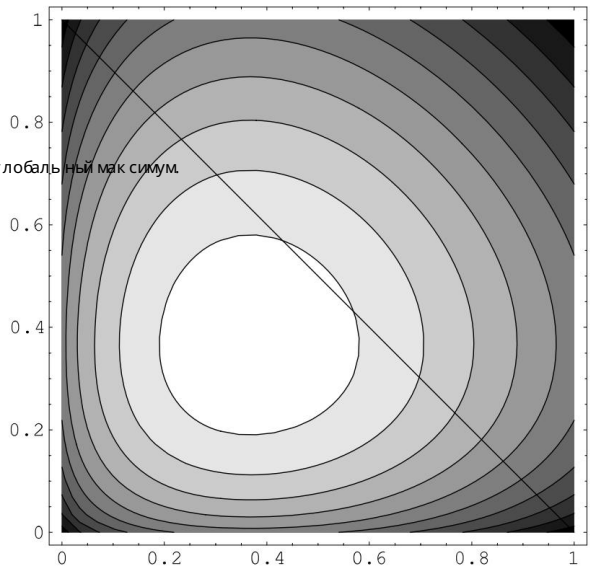


Рис. 7.2 Контурная линейная диаграмма двумерной функции энтропии. Мы видим, что оно строго выпукло во всем единичном квадрате и имеет изолированный глобальный максимум. Также отмечено ограничение p3 + p4 = 1 как частный случай условия p3 + p4

1 + α = 0 при α = 0, которое имеет значение здесь

и рассчитать

$$p3 = p4 = \frac{1 - a}{2}.$$

Теперь мы можем вычислить желаемое значение

$$P(B) = P(A, B) + P(\neg A, B) = p1 + p3 = \alpha\beta + = \alpha\beta \quad 2 \quad \frac{1 - \alpha 1}{2} \quad - \quad + \frac{1}{2}.$$

Замена в α и β дает

$$\Pi(B) = \Pi(A)\,\Pi(B \mid A) - 2 \quad \frac{1}{-} \quad 1 \; \frac{}{+ 2}.$$

P (B) показано на рис. 7.3 на стр. 126 для различных значений P (B|A). Мы видим, что в двухзначном пограничном случае, то есть когда P (B) и P (B|A) принимают значения 0 или 1, вероятностный вывод возвращает то же значение для P (B), что и modus ponens. Когда A и B|A оба истинны, B также истинно. Интересен случай P (A) = 0, в котором ¬A истинно. Modus ponens здесь неприменим, но наша формула дает значение 1/2 для P (B) независимо от P (B|A). Когда A ложно, мы ничего не знаем о B, что в точности отражает нашу интуицию Случай, когда P (A) = 1 и P (B|A) = 0, также покрывается логикой высказываний. Здесь A истинно и A B ложно, и, таким образом, A ¬B истинно. Тогда B ложно. Горизонтальная линия на рисунке означает, что мы не можем сделать предсказание относительно B в случае P (B|A) = 1/2. Между этими точками P (B) изменяется линейно для изменений P (A) или P (B|A).

Machine Translated by Google

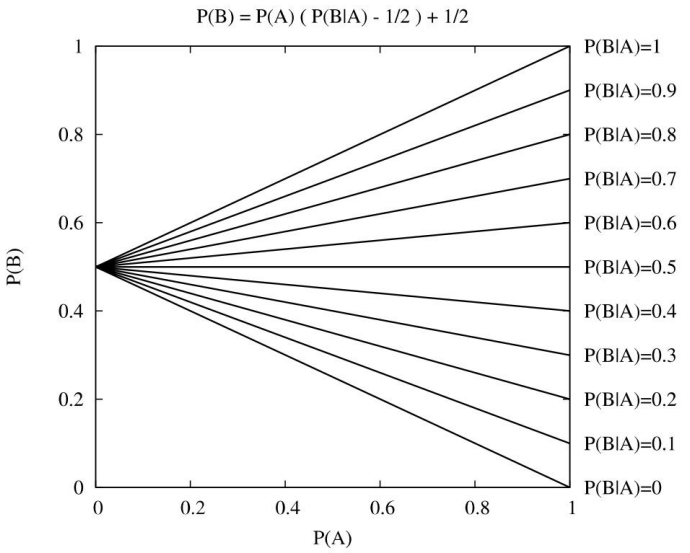


Рис. 7.3 Массив кривых для P (B) в зависимости от P (A) для различных значений P (B|A)

Теорема 7.3. Пусть существует непротиворечивая 2 система линейных вероятностных уравнений. Тогда существует единственный максимум функции энтропии с заданными уравнениями в качестве ограничений. Определенное таким образом распределение MaxEnt имеет минимальную информативность при ограничениях.

Из этой теоремы следует, что не существует распределения, удовлетворяющего ограничениями обладающего более высокой энтропией, чем распределение MaxEnt. Исчисление, которое приводит к более низкой энтропии, вводит дополнительную информацию ad hoc, что не оправдано.

При более внимательном рассмотрении приведенного выше расчета P(B) мы видим, что два значения p3 и p4 всегда встречаются симметрично. Это означает, что замена двух переменных не меняет результат. Таким образом, конечный результат p3 = p4. Так называемое безразличие этих двух переменных приводит к тому, что MaxEnt приравнивает их. Это соотношение справедливо в общем случае:

Определение 7.5. Если произвольная замена двух или более переменных в уравнениях Лагранжа приводит к эквивалентным уравнениям, эти переменные называются безразличными.

2Система вероятностных уравнений называется согласованной, если существует хотя бы одно решение, то есть одно распределение, удовлетворяющее всем уравнениям.

Machine Translated by Google

Теорема 7.4. Если набор переменных $\{p_{i1},\ldots,p_{ik}\}$ безразличен, то максимум энтропии при заданных ограничениях находится в точке, где $p_{i1} = p_{i2} = \cdots = p_{ik}$.

Зная это, мы могли бы сразу приравнять две переменные p_3 и p_4 (без решения уравнений Лагранжа).

7.2.2 Максимальная энтропия без явных ограничений

Теперь рассмотрим случай, когда знание не дано. Это означает, что кроме условия нормировки

$$p_1 + p_2 + \cdots + p_n = 1,$$

ограничений нет. Следовательно, все переменные безразличны. Поэтому мы можем[3] приравнять их, и тогда $p_1 = p_2 = \cdots = p_n = 1/n$. Для рассуждений в условиях неопределенности это означает, что при полном отсутствии знаний все миры равновероятны. То есть распределение равномерное. Например, в случае двух переменных A и B будет так, что

$$P(A, B) = P(A, \neg B) = P(\neg A, B) = P(\neg A, \neg B) = 1/4,$$

откуда следует $P(A) = P(B) = 1/2$ и $P(B|A) = 1/2$. Результат для двумерного случая можно увидеть на рис. 7.2 на стр. 125, поскольку отмеченное условие является точностью условием нормализации. Мы видим, что максимум энтропии лежит на прямой точно в точке (1/2, 1/2).

Как только значение условия отклоняется от значения, полученного из равномерного распределения, вероятности миров меняются. Покажем это на следующем примере. С теми же описаниями, которые использовались выше, мы предполагаем, что только

$$P(B|A) = \beta$$

известен. Таким образом, $P(A, B) = P(B|A)P(A) = \beta P(A)$, откуда следует $p_1 = \beta(p_1 + p_2)$, и мы получаем два ограничения

$$\beta p_2 + (\beta - 1)p_1 = 0,$$

$$p_1 + p_2 + p_3 + p_4 - 1 = 0.$$

---

[3] Читатель может вычислить этот результат путем максимизации энтропии при условии нормализации (упражнение 7.5 на стр. 158).

Machine Translated by Google

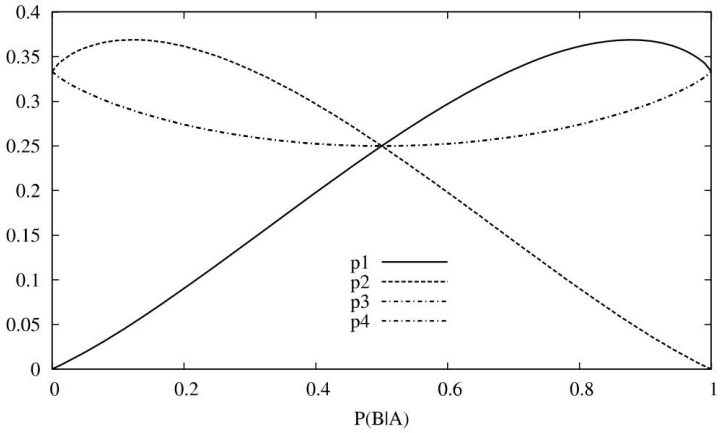


Рис. 7.4 p1, p2, p3, p4 в зависимости от β

Таблица 7.1 Таблица истинности для материальной смысловой условной вероятности для пропозиционального логического предела

ABA Б	П (А)	П (Б)	Р (Б|А)
ттт 111			
т фф 100			
фут шфт 0	0	1	Неопределенный
		0	Неопределенный

Здесь уравнения Лагранжа уже не могут быть так легко решены символически. Численное решение уравнений Лагранжа дает картину, представленную на рис. 7.4, что показывает, что p3 = p4. Мы можем видеть это уже в ограничениях, в которых p3 и p4 безразличны При P (B|A) = 1/2 получается равномерное распределение, которое нетничего удивительного. Это означает, что ограничение на это значение не влечет ограничения на распределение. Кроме того, мы можем видеть, что для малых P (B|A) P (A, B) равно тоже маленький.

7.2.3 Условная вероятность в сравнении с существенным следствием

Теперь мы покажем, что для моделирования рассуждений условная вероятность лучше. чем то, что известно в логике как материальная имплика ция (по этому поводу см. также [Ada75]). Сначала мы рассмотрим таблицу истинности, показанную в таблице 7.1, в которой условная вероятность и материальное значение для предельных случаев нулевой и единичной вероятностей равны в сравнении. В обоих случаях с ложными посылками (которые интуитивно являются критическими случаями), P (B|A) не определено, что имеет смысл.

Machine Translated by Google

Теперь зададимся вопросом, какое значение принимает P (B|A), когда заданы произвольные значения P (A) = α и P (B) = γ и никакой другой информации не известно. Снова мы максимизируем энтропию при заданных ограничениях x. Как и выше, мы устанавливаем

$$p1 = P(A, B), p2 = P(A, \neg B), p3 = P(\neg A, B), p4 = P(\neg A, \neg B)$$

и получить в качестве ограничений

$$p1 + p2 = a, \tag{7.10}$$

$$p1 + p3 = y, \tag{7.11}$$

$$p1 + p2 + p3 + p4 = 1. \tag{7.12}$$

При этом мы вынуждены, используя максимизацию энтропии (см. упражнение 7.8 на стр. 159).

$$p1 = \alpha\gamma, p2 = \alpha(1 - \gamma), p3 = \gamma(1 - \alpha), p4 = (1 - \alpha)(1 - \gamma).$$

Из $p1 = \alpha\gamma$ следует, что $P(A, B) = P(A) \cdot P(B)$, а это означает, что A и B независимы. Поскольку у нет никаких ограничений, соединяющих A и B, принцип MaxEnt приводит к независимости этих переменных. Правая половина Таблицы 7.1 на странице 128 облегчает понимание. Из определения

$$P(B|A) = \frac{П(A, Б)}{П(A)}$$

из этого следует для случая $P(A) = 0$, то есть, когда посылка не ложна, поскольку A и B независимы, что $P(B|A) = P(B)$. Для случая $P(A) = 0$ значение $P(B|A)$ остается неопределенным.

7.2.4 MaxEnt-системы

Как упоминалось ранее, из-за нелинейности функции энтропии оптимизация MaxEnt обычно не может быть выполнена символически и для нетривиальных задач. Таким образом, были разработаны две системы для численной максимизации энтропии. Первая система, SPIRIT (Симметричное вероятностное интенсиональное мышление в сети вывода, работает в переходный период, www.xspirit.de), [RM96], была построена в Fernuniversität Hagen. Вторая, PIT (Probability Induction Tool), была разработана в Мюнхенском техническом университете [Sch96, ES99, SE00]. Сейчас мы кратко представим PIT.

Система PIT использует метод последовательного квадратичного программирования (SQP) для нахождения экстремума функции энтропии при заданных ограничениях x. В качестве входных данных PIT ожидает данные, содержащие ограничения. Например, ограничения $P(A) = \alpha$ и $P(B|A) = \beta$ из разд. 7.2.1 имеют вид

Machine Translated by Google


```
переменная A{t,f}, B{t,f};

P([A=t]) = 0,6;
P([B=t] | [A=t]) = 0,3;

K П([B=t]);
QP([B=t] | [A=t]);
```

Поскольку у PIT выполняется численный расчет, мы должны ввести значения вероятности. Предпоследняя строка содержит запрос QP([B=t]). Это означает что P(B) является желаемым значением. На www.pit-systems.de в разделе «Примеры» мы теперь поместите этот ввод на пустую страницу ввода («Пустая страница») и запустите PIT. В результате мы получать

номер	Значение истины	Вероятностный запрос
1	НЕ УКАЗАНО	3.800e–01 QP([B = t]);
2	НЕ УКАЗАНО	3.000e–01 QP([A = t]-├ [B = t]);

и отсюда считать P (B) = 0,38 и P (B|A) = 0,3.

7.2.5 Пример Твити

Теперь покажем на примере Tweety из разд. 4.3, что вероятностные рассуждения и, в частности, MaxEnt немонотонны очень хорошо моделируют повседневные рассуждения. хорошо. Мы моделируем соответствующие правила с вероятностями следующим образом:

P (птица|пингвин) = 1 P (мухи| птица) [0,95, 1]	«пингвины — это птицы» «(почти все) птицы умеют летать»
P (мухи|пингвины) = 0	«пингвины не умеют летать»

Первое и третье правила представляют собой твердые прогнозы, которые так же легко формулируются в логике. Во втором, однако, мы выражаем наше знание, что почти все птицы могут летать с помощью условного интервала. С входными данными PIT

```
вар пингвин {да, нет}, птица {да, нет}, мухи {да, нет};

P([птица=да] | [пингвин=да]) = 1;
P([flies=yes] | [bird=yes]) IN [0.95,1];
P([мухи=да] | [пингвин=да]) = 0;

QP([мухи=да]| [пингвин=да]);
```

мы получаем правильный ответ

Machine Translated by Google

номер	Истинное значение	Вероятностный запрос
1	НЕУКАЗАНО	0.000e+00 QP([пингвин=да]-|> [мухи=да]);

с предположением, что пингвины не могут летать. Объяснение этому очень простое. При P (flies|bird) [0,95, 1] возможно наличие нелетающих птиц. Если это правило было заменено на P (flies|bird) = 1, тогда PIT ничего не смог бы сделать и будет выводить сообщение об ошибке о несогласованных ограничениях.

В этом примере мы можем легко увидеть, что интервалы вероятности часто очень полезны для моделирования нашего незнания о точных значениях вероятности. Мы могли бы сделать еще более размытая формулировка второго правила в духе «обычно птицы летают» с P (flies|bird) (0,5, 1]. Использование полуоткрытого интервала исключает значение 0,5.

В [Pea88] уже было показано, что этот пример можно решить с помощью вероятностной логики даже без MaxEnt. В [Sch96] для ряда требовательных тестов немонотонных рассуждений показано, что их можно элегантно решить с помощью Макс-Энт. В следующем разделе мы представим успешное практическое применение MaxEnt в виде медицинской экспертной системы

7.3 LEXMED, экспертная система для диагностики аппендицита

Медицинская экспертная система LEXMED, использующая метод MaxEnt, была разработана в Равенсбург-Вайнгартенском университете прикладных наук Манфредом Шрамм, Вальтер Рампф и автор в сотрудничестве с Weingarten 14-Больница Нотельфера [SE00, Le999].5 Аббревиатура LEXMED расшифровывается как обучающая экспертная система для медицинской диагностики.

7.3.1 Диагностика аппендицита с помощью юформальных методов

Наиболее распространенной серьезной причиной острой боли в животе [dD91] является аппендицит. воспаление аппендикса, слепой трубки, соединенной со слепой кишкой. Даже сегодня диагностика во многих случаях может быть затруднена [OFY+95]. Например, примерно до 20% удаленных аппендиксов не имеют патологических изменений, т.е. что операции не нужны. Кроме того, регулярно возникают случаи, когда воспаленный аппендикс не распознается как таковой.

Еще с начала 1970-х годов предпринимались попытки автоматизировать диагностику аппендицита с целью юснижения частоты ложных диагнозов [dDLS+72, OPB94, OFY+95]. Особого внимания заслуживает экспертная система.

4QP([penguin=yes]-|> [flies=yes]) — это альтернативная форма синтаксиса PIT для QP([мухи=да] | [пингвин=да]).

5Проект финансировался немецкой землей Баден-Вюртемберг, страховой компанией AOK Baden-Württemberg, Университетом прикладных наук Равенсбург-Вайнгартен и Больница Нотельфер 14 в Вайнгартене.

Machine Translated by Google

для диагностики острой боли в животе, разработанный Домбалем в Великобритании. Она была обнародована в 1972 году, то есть значительно раньше знаменитой системыMYCIN.

Почти все формальные диагностические процессы, используемые в медицине на сегодняшний день, основанына баллах. Системыоценки чрезвычайно просты в применении: за каждое значение симптома (например, лихорадка или боль внизу живота справа) врач отмечает определенное количество баллов. Если сумма баллов превышает определенное значение (порог), рекомендуется определенное решение (например, операция). Для n симптомов S1,...,Sn оценка аппендицита формально может быть описана как

$$\text{Диагностировать} = \begin{cases} \text{Аппендицит}, & \text{если } w1S1 + \cdots + wnSn > \Theta, \\ \text{отрицательный} & \text{еще.} \end{cases}$$

Таким образом, с помощью юбаллов линейная комбинация значений симптомов сравнивается с пороговым значением Θ. Веса симптомов извлекаются из баз данных с использованием статистических методов. Преимуществом баллов является простота их применения. Взвешенную сумму баллов можно легко рассчитать вручную, и для диагностики не требуется компьютер.

Из-за линейности этого метода оценки слишком слабы для моделирования сложных отношений. Поскольку вклад wiSi симптома Si в оценку рассчитывается независимо от других симптомов, ясно, что системыоценки не могут принимать во внимание какой-либо «контекст». В принципе, они не могут различить комбинации симптомов, например, они не могут отличить количество лейкоцитов у старого пациента от количества лейкоцитов у молодого пациента.

Для фиксированного заданного набора симптомов условная вероятность гораздо более эффективна, чем оценки для прогнозирования, поскольку у последние не могут описать зависимости между различными симптомами. Мыможем показать, что оценки неявно предполагают, что все симптомы независимы

При использовании баллов возникает еще одна проблема. Чтобыдостичь хорошего качества диагностики, мыдолжныпредъявлять строгие требования к базамданных, используемым для статистического определениявзаимозависимостей, они должныбыть репрезентативными для группыпациентов в области, в которой используется диагностическая система. Это часто трудно, если вообще возможно, гарантировать. В таких случаях баллыи другие статистические методылибо не могут быть использованы, либо будут иметь высокий уровень ошибок.

7.3.2 Гибридная вероятностная база знаний

Сложныезависимости часто возникают в медицине. С помощью юLEXMED этивзаимосвязи можно хорошо смоделировать и быстро рассчитать. Здесь существенное значение имеет использование вероятностных предположений, с помощью которых неопределенная и неполная информация может быть выражена и обработана интуитивно и математически обоснованным способом. Типичным запросом к экспертной системе может служить следующий вопрос: «Насколько высока вероятность воспаления аппендикса у пациента 23-летнего мужчиныс болями в правой нижней части живота и количеством лейкоцитов 13 000 ?» Сформулированное как условная вероятность с использованием названий и диапазонов значений симптомов, использованных в Таблице 7.2 на стр. 133, это выглядит следующим образом:

Machine Translated by Google

Таблица 7.2 Признаки, используемые для запроса в LEXMED, и их значения. Количество значений для каждый симптом указан в столбце с пометкой #

Симптом	Ценности	# Короткий
Пол	Мужчина, женщина	2 Сек с2
Возраст	0–5, 6–10, 11–15, 16–20, 21–25, 26–35, 36–45, 46–55, 56–65, 65–	10 лет 10
Боль 1-й квадр.	Да, нет	2 П1К 2
Боль 2-й квадр.	Да, нет	2 П2К 2
Боль 3-й квадр.	Да, нет	2 P3Q2
Боль 4-й квадр.	Да, нет	2 P4Q2
Охрана	Локальный, глобальный, нет	3 Гуа3
Отскок нежности	Да, нет	2 воз62
Боль при постукивании	Да, нет	2 Нажмите2
Ректальная боль	Да, нет	2 Запись P2
Звуки кишечника	Слабая, нормальная, повышенная, нет	4 Лук S4
Аномальное УЗИ	Да, нет	2 Соно2
Аномальные выделения мочи.	Да, нет	2 Урин2
Температура (ректальная)	–37,3, 37,4–37,6, 37,7–38,0, 38,1–38,4, 38,5–38,9, 39,0–	6 TRec6
Лейкоциты	0–6к, 6к–8к, 8к–10к, 10к–12к, 12к–15к, 15к–20к, 20к–	7 Леук о7
Диагноз	Воспаленные, перфорированные, отрицательные, другие	4 Диаг.4

P (Diag4 = воспаленный Diag4 = перфорированный |
Sex2 = мужской Age10 21–25 Leuko7 12k–15k).

Используя вероятностные предположения, LEXMED может использовать информацию из нерепрезентативных баз данных, поскольку эта информация может быть дополнена соответственно из других источников. В основе LEXMED лежит база данных, которая содержит данные о пациентах, у которых аппендикс был удален хирургическим путем. С помощью статистических методов генерируются (около 400) правил, которые компилируют знания, содержащиеся в базе данных, в абстрактную форму[ES99]. Потому что нет больных в этой базе данных с подозрением на аппендицит, но с отрицательным диагнозом (т. е. не требующим лечения)6 нет сведений об отрицательных
больных в базе. Таким образом, необходимо добавить знания из других источников.
Поэтому в LEXMED правила, собранные из базы данных, дополняются
(около 100) правил из медицинских специалистов и медицинской литературы Это приводит к гибридная вероятностная база данных, которая содержит знания, извлеченные из данных в виде а также знания, явно сформулированные экспертами. Поскольку оба типа правил

6Эти отрицательные диагнозы обозначаются как «неспецифическая боль в животе» (NSAP).

Machine Translated by Google

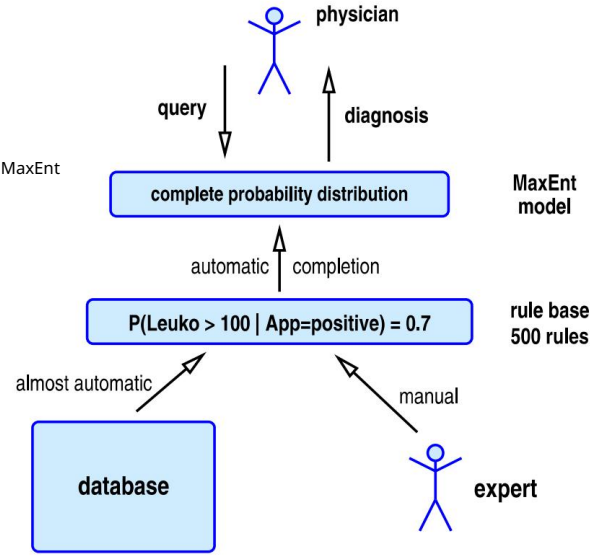


Рис. 7.5 Вероятностные правила генерируются из данных и экспертных знаний, которые интегрируются в базу правил (базу знаний) и окончательно дополняются с помощью метода MaxEnt

сформулированные как условные вероятности (см., например, (7.14) на стр. 138), их можно легко комбинировать, как показано на рис. 7.5 и более подробно на рис. 7.7 на стр. 136.

LEXMED рассчитывает вероятности различных диагнозов, используя распределение вероятностей всех соответствующих переменных (см. Таблицу 7.2 на стр. 133). Поскольку все 14 симптомов, используемых в LEXMED, и диагнозы моделируются как дискретные переменные (даже непрерывные переменные, такие как количество лейкоцитов, разделены на диапазоны), размер распределения (то есть размер пространства событий) можно определить с помощью таблицы 7.2 на стр. 133 как произведение количества значений всех симптомов, или

$$10 \cdot 2 \cdot 10 \cdot 3 \cdot 4 \cdot 6 \cdot 7 \cdot 4 = 20\,643\,840$$

элементы. Таким образом, благодаря условию нормировки и из теоремы 7.1 оно имеет 20 643 839 независимых значений. Каждый набор правил с менее чем 20 643 839 вероятностными значениями потенциально не полностью описывает это пространство событий. Чтобы иметь возможность ответить на любой произвольный запрос, экспертная система нуждается в полном дистрибутиве. Построение такого обширного последовательного распределения с использованием статистических методов очень сложно[7]. Требовать от человека-эксперта все 20 643 839 значений для распределения (вместо вышеупомянутых 100 правил) было бы принципиально невоз

Здесь в игру вступает метод MaxEnt. Обобщение около 500 правил до полной вероятностной модели выполняется в LEXMED путем максимизации энтропии с 500 правилами в качестве ограничений. Эффективное кодирование результирующего распределения MaxEnt приводит к тому, что время отклика для диагностики составляет около одной секунды

[7] Задача создания функции из набора данных известна как машинное обучение. Мы подробно рассмотрим это в гл. 8.

Machine Translated by Google

Personal Details	unknown	values	
Gender	◉	○ male ○ female	?
Age-group	○	○ 0-5 ○ 6-10 ○ 11-15 ○ 16-20 ◉ 21-25 ○ 26-35 ○ 36-45 ○ 46-55 ○ 56-65 ○ 65-	?
Results of examination	**not done**	**values**	
1st quadrant	◉	○ yes ○ no	?
2nd quadrant	◉	○ yes ○ no	?
3rd quadrant	○	◉ yes ○ no	?
4th quadrant	◉	○ yes ○ no	?
guarding	○	◉ local ○ global ○ none	?
rebound tenderness	○	◉ yes ○ no	?
pain on tapping	○	◉ yes ○ no	?
rectal pain	◉	○ yes ○ no	?
bowel sounds	○	○ weak ◉ normal ○ increased ○ none	?
abnormal ultrasound	◉	○ yes ○ no	?
abnormal urine sediment	◉	○ yes ○ no	?
temperature range (rectal)	○	○ -37.3 ○ 37.4-37.6 ○ 37.7-38.0 ○ 38.1-38.4 ◉ 38.5-38.9 ○ 39.0-	?
leucocyte count	○	○ 0-6k ○ 6k-8k ○ 8k-10k ○ 10k-12k ◉ 12k-15k ○ 15k-20k ○ 20k-	?

	Result of the PIT diagnosis			
Diagnosis	App. inflamed	App. perforated	Negative	Other
Probability	0.70	0.17	0.06	0.07

Рис. 7.6. Маска ввода LEXMED для ввода исследуемых симптомов и под ней вывод результирующих вероятностей диагноза

7.3.3 Применение ЛЕКСМЕД

Использование LEXMED простое и понятное. Врач посещает Для автоматической Домашняя страница LEXMED на [8] диагностика и врач www.lexmed.de. вводит результаты своего обследования в форму ввода на рис. 7.6. Через одну-две секунды он получает вероятности четырех различных диагнозов, а также предложение по лечению(раздел 7.3.5). Если в качестве входных данных отсутствуют какие-либо результаты обследования (например, результаты УЗИ), то врач выбирает запись без обследования . Естественно, достоверность диагноза выше, когда вводится больше значений симптомов.

8А версия с ограниченным функционалом доступна без пароля .

Machine Translated by Google

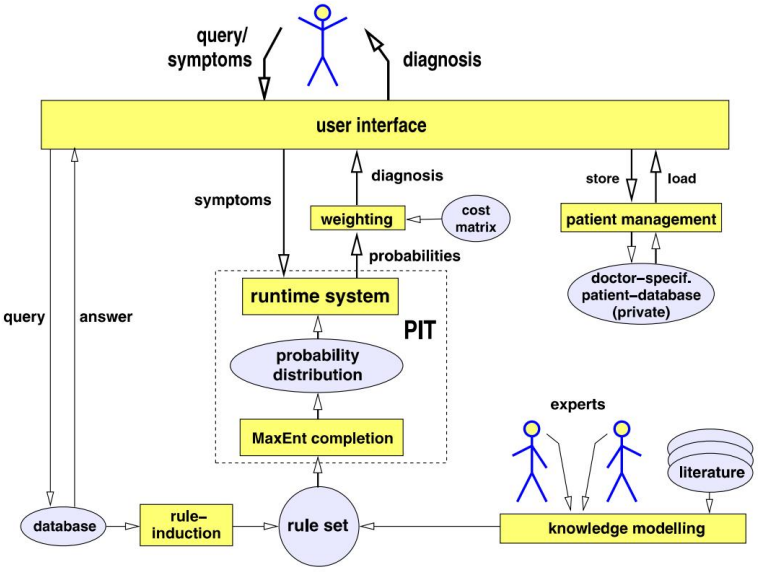


Рис. 7.7 Правила генерируются из базы данных, а также из экспертных знаний. Из них MaxEnt создает полное распределение вероятностей. По запросу пользователя вычисляется вероятность каждого возможного диагноза. Используя матрицу затрат (см. раздел 7.3.5), затем предлагается решение.

Каждый зарегистрированный пользователь имеет доступ к частной базе данных пациентов, в которой можно архивировать введенные данные. Таким образом, данные и диагнозы более ранних пациентов можно легко сравнить с данными нового пациента.

7.3.4 Функции ЛЕКСМЕД

Знания формализуются с помощью условных вероятностных утверждений. Например, предложение

$$P(\text{Leuko7} > 20000 \mid \text{Diag4} = \text{воспаленный}) = 0{,}09$$

дает частоту 9% для значения лейкоцитов более 20 000 в случае воспаленного аппендикса.[9]

Изучение правил методом статистической индукции

Необработанные данные в базе данных LEXMED содержат 54 различных (анонимизированных) значения для 14 646 пациентов. Как упоминалось ранее, в эту базу данных включены только пациенты с острым аппендицитом, удаленным хирургическим путем. Из 54 атрибутов, используемых в базе данных, после статистического анализа 14 симптомов, показанных в Табл. 7.2 на стр. 133

[9] Вместо отдельных числовых значений здесь также могут использоваться интервалы (например, [0,06, 0,12]).

Machine Translated by Google

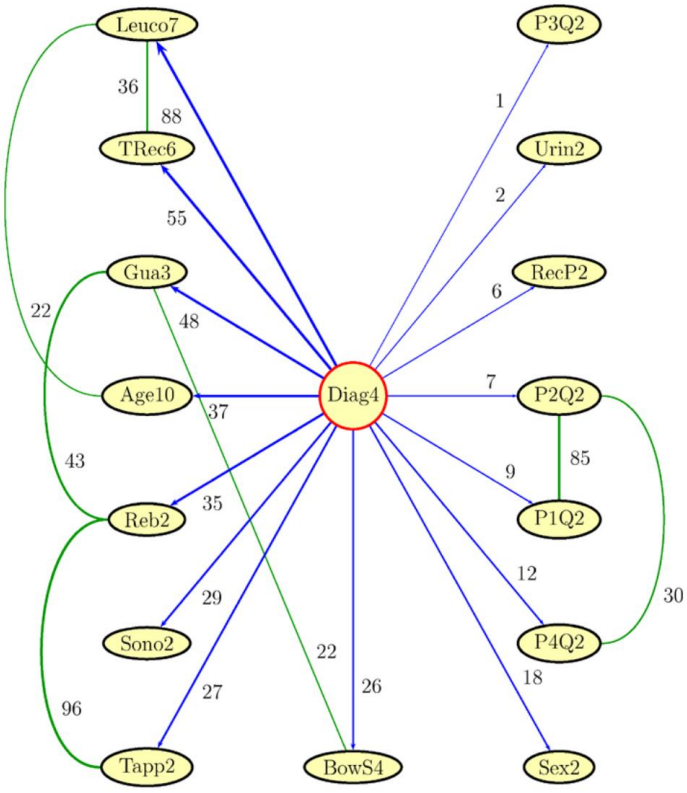


Рис. 7.8 Граф зависимост ей, вын исленный из баз ыданны х

были исполь зованы Т еперь правила соз дают ся из эт ой баз ыданных в два эт апа. Первый шаг опред еля ет ст рук т уру зависимост ей симпт омов. Вт орой шаг з аполня ет эт у ст рук т уру соот вет ст вующими вероя т ност ными правилами.10

Опред еление г рафа з ависимост ей Граф на рис. 7.8 сод ержит для к аждой переменной (симпт ома и диагноз а) уз ел и направленные ребра, соед иня ющие раз лич ные уз лы Т олщина границ между переменными пред ст авля ет собой меру ст ат ист ич еск ой з ависимост и или к орреля ц ии переменных. К орреля ц ия двух нез ависимых переменных равна нулю Парная к орреля ц ия для к аждого из 14 симпт омов с Diag4 была рассч ит ана и пред ст авлена на граф ик е. К роме т ого, были рассч ит аны все т ройные к орреля ц ии между диагноз ом и двумя симпт омами. Из них т оль к о самые силь ные з нач ения были нарисованы к ак д ополнит ель ные границ ы между д вумя уч ast вующими симпт омами.

10Для сист емат ич еск ого введения в машинное обуч ение мы от сылаем ч ит ат еля к гл. 8.

Machine Translated by Google


```
 1 П([Лейко7=0-6к]                | [Diag4=отрицательный] * [Age10=16-20]) = [0,132,0,156];
 2 П([Лейко7=6-8к]                | [Diag4=отрицательный] * [Age10=16-20]) = [0,257,0,281];
 3 P([Leuco7=8-10k] | [Diag4=отрицательный] * [Age10=16-20]) = [0,250,0,274];
 4 P([Leuco7=10-12k] | [Diag4=отрицательный] * [Age10=16-20]) = [0,159,0,183];
 5 P([Leuco7=12-15k] | [Diag4=отрицательный] * [Age10=16-20]) = [0,087,0,112];
 6 P([Leuco7=15-20k] | [Diag4=отрицательный] * [Age10=16-20]) = [0,032,0,056];
 7 П([Лейко7=20к-]                | [Diag4=отрицательный] * [Age10=16-20]) = [0,000,0,023];
 8 П([Лейко7=0-6к]                | [Diag4=отрицательный] * [Age10=21-25]) = [0,132,0,172];
 9 П([Лейко7=6-8к]                | [Diag4=отрицательный] * [Age10=21-25]) = [0,227,0,266];
10 P([Leuco7=8-10k] | [Diag4=отрицательный] * [Age10=21-25]) = [0,211,0,250];
11 P([Leuco7=10-12k] | [Diag4=отрицательный] * [Age10=21-25]) = [0,166,0,205];
12 P([Leuco7=12-15k] | [Diag4=отрицательный] * [Age10=21-25]) = [0,081,0,120];
13 P([Leuco7=15-20k] | [Diag4=отрицательный] * [Age10=21-25]) = [0,041,0,081];
14 П([Левко7=20к-]                | [Diag4=отрицательный] * [Age10=21-25]) = [0,004,0,043];
```

Рис. 7.9 Некоторые правила LEXMED с вероятностными интервалами. «*» здесь означает « »

Оценка вероятностей правил Структура графа зависимостей описывает структуру изученных правил.11 Правила здесь имеют разную сложность: есть правила, которые только описывают распределение возможных диагнозов.
(априорные правила, например (7.13)), правила, описывающие зависимость между диагноз и симптом (правила с простыми условиями, например (7.14)), и наконец, правила, описывающие зависимость между диагнозом и двумя симптомами, как показано на рис. 7.9 в синтаксисе PIT .

$$P(\text{Diag4} = \text{воспаленный}) = 0{,}40, \quad P \tag{7.13}$$

$$(\text{Sono2} = \text{да} \mid \text{Diag4} = \text{воспаленный}) = 0{,}43, \tag{7.14}$$

$$P(\text{P4Q2} = \text{да} \mid \text{Diag4} = \text{воспаление} \quad \text{P2Q2} = \text{да}) = 0{,}61. \tag{7.15}$$

Чтобы сохранить контекстную зависимость сохраненных знаний как можно меньше, все правила содержат диагноз в своих условиях, а не в виде выводов. Это очень похоже на построение многих медицинских книг с формулировками рода «При аппендиците мы обычно видим . . . ». Как показано ранее в примере 7.6. на стр. 121, однако, это не представляет проблемы, поскольку при использовании байесовского формулы, ЛЕКСМЕД автоматически переводит эти правила в правильную форму.

Численные значения для этих правил оцениваются путем подсчета их частоты в базу данных. Например, значение в (7.14) определяется путем подсчета и вычисления

$$\frac{|\text{Diag4} = \text{воспаление} \quad \text{Sono2} = \text{да}|}{|\text{Diag4} = \text{воспаление}|} = 0{,}43.$$

11Разница между этой и байесовской сетью, например, в том, что правила оснащены с вероятностными интервалами и что только после применения принципа максимальной энтропии построена вероятностная модель.

Machine Translated by Google

Экспертные правила

Поскольку база данных по аппендициту содержит только пациентов, перенесших операции, правила неспецифической боли в животе (NSAP) получают свои значения из предложения медицинских специалистов. Опыт LEXMED подтверждает, что вероятностные правила легко читаются и могут быть непосредственно переведены на естественный язык. Заявления медицинских экспертов о частотных соотношениях конкретных симптомов и диагноз, будь то из литературы или в результате интервью, может поэтому можно включить в базу правил с небольшими затратами. Для моделирования неопределенности экспертных знаний оказалось эффективным использование интервалов вероятности. Экспертные знания были в основном получены от участвующих хирургов, доктора. Rampf and Dr. Hontschik и их публикации [Hon94].

Когда экспертные правила созданы, база правил готова. Затем рассчитывается полная вероятностная модель методом максимальной энтропии по методу PIT-система.

Диагностические запросы

Используя эффективно хранимую модель вероятностей, LEXMED рассчитывает вероятности для четырех возможных диагнозов в течение нескольких секунд. Например, мы предполагаем, что следующий вывод:

	Результаты диагностики ПИТ		
Диагноз	Аппендикс воспаленный Аппендикс перфорирован 0,24 0,16	Отрицательный	Другой
Вероятность		0,57	0,03

Решение должно быть принято на основе этих четырех значений вероятности, чтобы следовать одному из четыре метода лечения: операция, экстренная операция, стационарное наблюдение или амбулаторное наблюдение.12 При этом вероятность отрицательного диагноза в этом случае перевешивает остальные, отправлять больного домой как здорового — не самое удачное решение. Мы можем ясно видеть, что, даже когда вероятности диагнозов были подсчитано, диагностика еще не закончена.

Скорее, теперь задача состоит в том, чтобы вывести оптимальное решение из этих вероятностей. К С этой целью пользователь может попросить LEXMED рассчитать рекомендуемое решение.

7.3.5 Управление рисками с использованием матрицы затрат

Как теперь вычисленные вероятности можно оптимально преобразовать в решения? Наивный алгоритм назначал бы решение каждому диагнозу и в конечном итоге выбирал бы решение, которое соответствует наибольшей вероятности. Предположим, что вычисленное вероятности 0,40 для диагноза аппендицит (воспаленный или перфоративный), 0,55 для диагноза отрицательный и 0,05 для диагноза другой. Наивный алгоритм теперь выберите (слишком рискованное) решение «без операции», поскольку оно соответствует диагнозу с большей вероятностью. Лучший метод состоит в сравнении затрат

12 Амбулаторное наблюдение означает, что больной выписан домой.

Machine Translated by Google

Таблица 7.3 Матрица стоимости LEXMED вместе с расчетными вероятностями диагноза пациента

Терапия	Вероятность различных диагнозов				
	воспаленный	Перфорированный	Отрицательный	Другой	
	0,25	0,15	0,55	0,05	
Операция	0	500	5800	6000	3565
Аварийная операция	500	0	6300	6500	3915
Амбулаторное наблюдение.	12000	150000	0	16500	26325
Другой	3000	5000	1300	0	2215
Стационарная обсерв.	3500	7000	400	600	2175

возможных ошибок, которые могут возникнуть для каждого решения. Ошибка количественно определяется форме «(гипотетической) дополнительной стоимости текущего решения по сравнению с оптимальным». Приведенные значения содержат расходы на больницу, страховую компанию, пациенту (например, риск послеоперационных осложнений), так и другим сторонам (например, отсутствие на работе) с учетом отдаленных последствий.
Эти затраты приведены в таблице 7.3.

Записи окончательно усредняются для каждого решения, т. е. суммируются при принятии с учетом их частот. Они перечислены в последнем столбце таблицы 7.3. Наконец, предлагается решение с наименьшей средней ценой ошибки. В таблице 7.3 матрица дана вместе с рассчитанным для пациента вектором вероятности (в в этом случае: (0,25, 0,15, 0,55, 0,05)). Последний столбец таблицы содержит результат расчеты средних ожидаемых издержек ошибок. Значение операции в первой строке вычисляется как 0,25 · 0 + 0,15 · 500 + 0,55 · 5800 + 0,05 · 6000 = 3565, средневзвешенное значение всех затрат. Оптимальные решения вводятся с (дополнительными) затратами, равными 0. Система принимает решение о лечении с минимальным средним значением. расходы. Таким образом, это пример агента, ориентированного на затраты.

Матрица стоимости в бинарном случае

Чтобы лучше понять матрицу затрат и управление рисками, мы ограничим Система ЛЕКСМЕД к двузначному решению между диагнозом аппендицит и НСАП. В качестве возможных методов лечения только операция с вероятностью p1 и амбулаторное лечение. наблюдение с вероятностью p2 может быть выбрано. Таким образом, матрица затрат представляет собой матрицу 2 × 2. матрица вида

$$\begin{matrix} 0 & k2 \\ k1 & 0 \end{matrix}.$$

Два нуля на диагонали обозначают правильную решающую операцию в случае аппендицита и амбулаторное наблюдение по поводу NSAP. Параметр k2 означает ожидаемые затраты, которые возникают, когда пациент без воспаленного аппендикса прооперирован. Эта ошибка называется ложным срабатыванием. С другой стороны, решение

Machine Translated by Google

амбулаторное наблюдение в случае аппендицита является ложноотрицательным. Вектор вероятности $(p1, p2)^T$ теперь умножается на эту матрицу, и мы получаем вектор

$$(2 \text{ лиц изн}, 1 \text{ лиц изн})^T$$

со средними дополнительными затратами на два возможных лечения. Поскольку решение принимает во внимание только взаимосвязь двух компонентов, вектор можно умножить на любой скалярный коэффициент. Выбираем 1/k1 и получаем ((k2/k1)p2, p1).
Таким образом, здесь имеет значение только соотношение k = k2/k1 . Тот же результат дает более простая матрица затрат

$$\begin{matrix} 0 & k \\ 1 & 0 \end{matrix},$$

который содержит только переменную k. Этот параметр очень важен, поскольку он определяет управление рисками. Изменяя k, мы можем подогнать «рабочую точку» диагностической системы. При k система оказывается в чрезвычайно рискованной ситуации, потому что ни один пациент никогда не будет оперирован, в результате чего она дает не ложноположительные классификации, а много ложноотрицательных. В случае k = 0 условия обратные, и все больные оперируются.

7.3.6 Производительность

ЛЕКСМЕД предназначен для использования в медицинской практике или машине скорой помощи. Предпосылками к применению ЛЕКСМЕДа являются острые боли в животе в течение нескольких часов (но менее пяти дней). Кроме того, LEXMED (в настоящее время) специализируется на аппендиците, а это означает, что по другим заболеваниям система содержит очень мало информации.

В рамках проспективного исследования в 14-й больнице Нотельфера была создана репрезентативная база данных со 185 случаями. В нем находятся пациенты больницы, которые обратились в клинику после нескольких часов острой боли в животе и подозрении на аппендицит. У этих пациентов отмечаются симптомы и диагноз (подтвержденный по образцу ткани в случае операции).

Если пациентов отпускали домой (без операции) после пребывания в течение нескольких часов или 1–2 дней без особых жалоб или без них, то впоследствии по телефону спрашивали, остается ли у пациента бессимптомным или был ли установлен положительный диагноз в последующее лечение.

Чтобы упростить представление и сделать лучшее сравнение с аналогичными исследованиями, LEXMED был ограничен двумя значениями различия между аппендицитом и НСАП, как описано в разд. 7.3.5. Теперь k варьируется от нуля до бесконечности, и для каждого значения k чувствительность и специфичность измеряются в сравнении с тестовыми данными.
Меры чувствительности

$$P(\text{классифицированный положительный} \mid \text{положительный}) = \frac{|\text{положительные и классифицированные как положительные}|}{|\text{положительный}|},$$

Machine Translated by Google

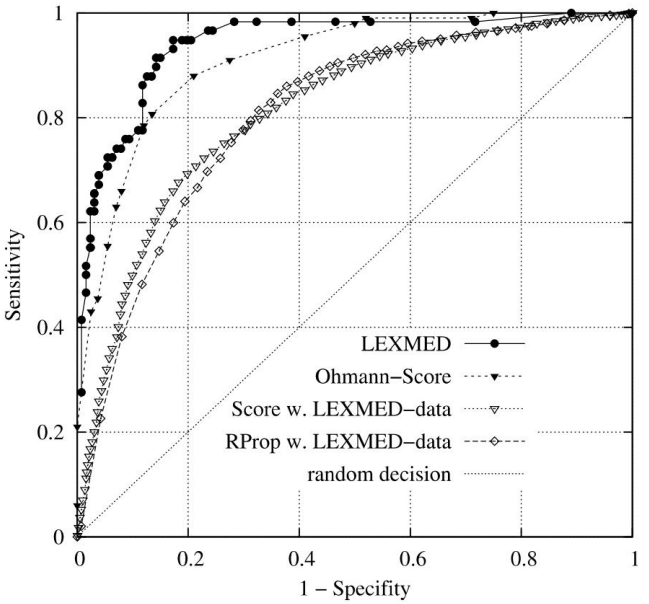


Рис. 7.10 ROC-кривая из LEXMED в сравнении с показателем Омана и двумя дополнительными моделями

то есть относительная доля правильно идентифицированных положительных случаев. Он показывает, насколько чувствительна диагностическая система. Специфика, с другой стороны, измеряет

$$P(\text{классифицированный отрицательный} \mid \text{отрицательный}) = \frac{|\text{отрицательные и классифицированные как отрицательные}|}{|\text{отрицательный}|},$$

то есть относительная доля правильно идентифицированных отрицательных случаев.

Мы приводим результаты чувствительности и специфичности на рис. 7.10 для 0 k < . Эта кривая обозначается ROC-кривой или рабочей характеристикой приемника. Прежде чем мы перейдем к анализу качества LEXMED, несколько слов о значении ROC-кривой. Линия, делящая диаграмму пополам по диагонали, нарисована для ориентации.
Все точки на этой прямой соответствуют случайному решению. Например, точка (0,2, 0,2) соответствует специфичности 0,8 при чувствительности 0,2. Мы можем довольно легко прийти к этому, классифицируя новый случай, не глядя на него, с вероятностью 0,2 для положительного и 0,8 для отрицательного. Таким образом, каждая система диагностики, основанная на знаниях, должна генерировать ROC, которая явно лежит выше диагонали.

Экстремальные значения на кривой ROC также интересны. В точке (0, 0) все три кривые пересекаются. Соответствующая система диагностики классифицировала бы все случаи как отрицательные. Другое экстремальное значение (1, 1) соответствует системе, которая решает сделать операцию каждому пациенту и, таким образом, имеет чувствительность 1. Мы могли бы назвать ROC-кривую характеристической кривой для двухзначных диагностических систем. Идеальная диагностическая система должна иметь характеристическую кривую, состоящую только из точек (0, 1), и, таким образом, иметь 100% специфичность и 100% чувствительность.

Machine Translated by Google

Теперь давайте проанализируем кривую ROC. При чувствительности 88% ЛЕКСМЕД достигает специфичности 87% (k = 0,6). Для сравнения приведена общеизвестная шкала Оманна для аппендицита [OMYL96, ZSR+99]. Поскольку LEXMED почти везде выше или левее шкалы Омана, его среднее качество диагностики явно выше. Это неудивительно, потому что оценки просто слишком слабы для моделирования интересных предложений. В разд. 8.6 и в упражнении 8.16 на стр. 219 мы покажем, что оценки эквивалентны частному случаю наивного Байеса, то есть предположению, что все симптомы попарно независимы, когда известен диагноз. Однако при сравнении LEXMED с баллами следует отметить, что для оценки Омана использовалась статистически репрезентативная база данных, а для LEXMED использовалась нерепрезентативная база данных, дополненная экспертными знаниями.
Чтобы получить представление о качестве данных LEXMED по сравнению с данными Ohmann, была рассчитана линейная оценка с использованием метода наименьших квадратов (см. раздел 9.4.1), который также используется для сравнения. Кроме того, нейронная сеть была обучена на данных LEXMED с помощью алгоритма RProp (см. раздел 9.5). Сила объединения данных и экспертных знаний четко отображается в разнице между кривой LEXMED и кривыми системы оценок и алгоритма RProp.

7.3.7 Области применения и опыт

ЛЕКСМЕД не должен заменять мнение опытного хирурга. Однако, поскольку специалист не всегда доступен в клинических условиях, запрос LEXMED предлагает существенное второе мнение. Особенно интересным и целесообразным является применение системы в клинической скорой помощи и для врачей общей практики.

Способность LEXMED к обучению, которая позволяет учитывать дополнительные симптомы, дополнительные данные пациента и дополнительные правила, также открывает новые возможности в клинике. Для особо редких групп, трудно поддающихся диагностике, например детей до шести лет, ЛЕКСМЕД может использовать данные педиатров или другие специальные базы данных, чтобы помочь даже опытным хирургам.

Помимо непосредственного использования в диагностике, LEXMED также поддерживает меры по обеспечению качества. Например, страховые компании могут сравнивать качество диагностики больниц с качеством экспертных систем. При дальнейшем развитии матрицы затрат, созданной в ЛЕКСМЕД (с согласия врачей, страховых компаний и пациентов), качество диагностики врачей, компьютерной диагностики и других медицинских учреждений станет проще сравнивать.

ЛЕКСМЕД указал на новый способ построения автоматических диагностических систем. Используя язык теории вероятностей и алгоритм MaxEnt, индуктивно статистически полученные знания объединяются со знаниями от экспертов и из литературы. Подход, основанный на вероятностных моделях, является теоретически элегантным, общеприменимым и дал очень хорошие результаты в небольшом исследовании.

ЛЕКСМЕД применяется на практике в больнице Nothelfer № 14 в Вайнгартене с 1999 года и очень хорошо себя зарекомендовал. Он также доступен на сайте www.lexmed.de, разумеется, без гарантии. Его качество диагностики сравнимо с качеством опытного хирурга и, таким образом, лучше, чем у среднего врача общей практики или неопытного врача в поликлинике.

Machine Translated by Google

Несмотря на этот успех, стало очевидным, что очень трудно продавать такую систему на коммерческой основе в немецкой медицинской системе. Одна из причин этого заключается в том, что не существует свободного рынка, который мог бы продвигать более высокое качество (в данном случае более совершенные диагнозы) с помощью механизмов отбора. Кроме того, в медицине время для широкого использования интеллектуальных методов еще не наступило — даже в 2010 году. Одной из причин этого могут быть консервативные учения в этом отношении на факультетах немецких медицинских школ.

Еще одной проблемой является стремление многих пациентов к личному совету и уходу со стороны врача, а также опасение, что с внедрением экспертных систем пациент будет общаться только с машиной. Однако этот страх совершенно необоснован. Даже в долгосрочной перспективе медицинские экспертные системы не могут заменить врача.
Однако их можно использовать так же, как лазерную хирургию и магнитно-резонансную томографию, с пользой для всех участников. С момента создания первой медицинской компьютерной диагностической системы де Домбаля в 1972 г. прошло почти 40 лет. Еще неизвестно, будет ли медицина ждать еще 40 лет, пока компьютерная диагностика не станет признанным медицинским инструментом.

7.4 Рассуждения с помощью байесовских сетей

Одна проблема, связанная с рассуждениями с использованием вероятности на практике, уже указывалась в разд. 7.1. Если используются d переменных X1,...,Xd с n значениями каждая, то распределение вероятностей соответствующие d общих значений. Это означает, что в худшем случае использует n памяти, а время вычислений для определения заданных вероятностей экспоненциально растет с увеличением числа переменных.

На практике приложения обычно очень структурированы, а дистрибутив содержит много избыточности. Это означает, что его можно значительно уменьшить соответствующими методами. Использование байесовских сетей доказало здесь свою эффективность и является одним из методов ИИ, которые успешно применяются на практике. Байесовские сети используют знания о независимости переменных для упрощения модели.

7.4.1 Независимые переменные

В простейшем случае все переменные попарно независимы, и это тот случай, когда

$$P(X1,...,Xd) = P(X1) \cdot P(X2) \cdots P(Xd).$$

Таким образом, все записи в распределении могут быть рассчитаны по значениям d $P(X1), \ldots, P(Xd)$. Однако интересные приложения обычно невозможно смоделировать, поскольку условные вероятности становятся тривиальными[13].

$$P(A|B) = \frac{П(A, Б)}{П(Б)} \text{ "=" } \frac{П(A)\,П(Б)}{П(Б)} = P(A)$$

13 В наивном байесовском методе предполагается независимость всех атрибутов, и этот метод был успешно применен к классификации текстов (см. раздел 8.6).

Machine Translated by Google

все условные вероятности сводятся к априорным вероятностям. Ситуация становится более интересной, когда только часть переменных является независимой или независимой при определенных условиях. Для рассуждений в ИИ зависимости между переменными оказываются важными и должны использоваться.

Мы хотели бы обрисовать рассуждения с байесовскими сетями на простом и очень наглядном примере Дж. Перла [Pea88], который стал широко известен благодаря [RN10] и теперь является базовым знанием ИИ.

Пример 7.7 (Пример с сигнализацией) Неженатый Боб установил в своем доме систему сигнализации для защиты от грабителей. Боб не слышит будильник, когда работает в офисе. Поэтому он попросил двух своих соседей, Джона в соседнем доме слева и Мэри в доме справа, позвонить ему в его офис, если они услышат его тревогу. Через несколько лет Боб знает, насколько надежны Джон и Мэри, и моделирует их поведение при вызове с использованием условной вероятности следующим образом.14

$$P(J|Al) = 0{,}90 \quad P(M|Al) = 0{,}70, \quad P(J|\neg Al) = 0{,}05 \quad P(M|\neg Al) = 0{,}01.$$

Поскольку Мэри плохо слышит, она не слышит сигнал тревоги чаще, чем Джон. Однако Джон иногда путает тревогу в доме Боба с тревогами в других домах. Тревога срабатывает при краже со взломом, но также может быть вызвана (слабым) землетрясением, что может привести к ложной тревоге, потому что Боб хочет знать о кражах со взломом только в своем офисе. Эти отношения моделируются

$$P(Al|Bur,Ear) = 0{,}95,$$

$$P(Al|Bur,\neg Ear) = 0{,}94,$$

$$P(Al|\neg Bur,Ear) = 0{,}29,$$

$$P(Al|\neg Bur,\neg Ear) = 0{,}001,$$

а также априорные вероятности P (Bur) = 0,001 и P (Ear) = 0,002. Эти две переменные независимы, потому что землетрясение не строят планы на основе привычек грабителей, и, наоборот, землетрясение невозможно предсказать, поэтому у грабителей нет возможности соответствующим образом устанавливать свой график.

Теперь к этой базе знаний делаются запросы. Например, Боба может интересовать P (Bur|J M), P (J|Bur) или P (M|Bur). То есть он хочет знать, насколько чутко переменные J и M реагируют на сообщение о взломе.

14Двоичные переменные J и M обозначают два события «Джон звонит» и «Мэри звонит» соответственно, Al — «звуки тревожной сирены», Bur — «кража со взломом» и Ear — «землетрясение».

Machine Translated by Google

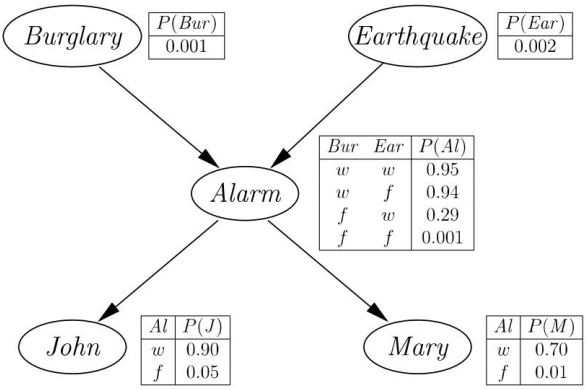


Рис. 7.11 Байесовская сеть для примера тревоги с соответствующими CPT

7.4.2 Графическое представление знаний в виде байесовской сети

Мы можем значительно упростить практическую работу, графически представив знания, сформулированные в виде условной вероятности. На рис. 7.11 показана байесовская сеть для примера тревоги. Каждый узел в сети представляет собой переменную, а каждое направленное ребро — утверждение условной вероятности. Ребро от Al до J, например, представляет два значения P (J |Al) и P (J |¬Al), которые задаются в виде таблицы, так называемой CPT (таблицы условной вероятности). CPT узла перечисляет все условные вероятности переменной узла, обусловленные всеми узлами, соединенными входящими ребрами.

Изучая сеть, мы можем спросить себя, почему в нее не включены другие ребра, кроме четырех нарисованных. Два узла Bur и Ear не связаны, поскольку переменные независимы. Все остальные узлы имеют родительский узел, что немного усложняет рассуждения. Сначала нам понадобится понятие условной независимости.

7.4.3 Условная независимость

Аналогично независимости случайных величин даем

Определение 7.6. Две переменные A и B называются условно независимыми при заданном C, если

$$P(A, B|C) = P(A|C) \cdot P(B|C).$$

Это уравнение верно для всех комбинаций значений всех трех переменных (то есть для распределения), которые мы видим в обозначениях. Теперь посмотрим на узлы J и M в примере с тревогой, которые имеют общий родительский узел Al. Если Джон и Мэри

Machine Translated by Google

независимо реагировать на аварийный сигнал, то две переменные J и M независимы при заданном Al, то есть:

$$P(J,M|Al) = P(J|Al) \cdot P(M|Al).$$

Если значение Al известно, например, из-за срабатывания тревоги, то переменные J и M независимы (при условии Al = w). Из-за условной независимости двух переменных J и M ребро между этими двумя узлами не добавляется. Однако J и M не являются независимыми (см. упражнение 7.11 на стр. 159).

Совершенно аналогична связь между двумя переменными J и Bur, потому что Джон реагирует не на кражу со взломом, а на тревогу. Это может быть, например, из-за высокой стены, которая загораживает ему обзор собственности Боба, но он все еще может слышать сигнал тревоги. Таким образом, J и Bur независимы при заданных Al и

$$P(J,Bur|Al) = P(J|Al) \cdot P(Bur|Al).$$

При тревоге переменные J и Ear, M и Bur, а также M и Ear также независимы. Для вычислений с условной независимостью полезны следующие характеристики, эквивалентные приведенному выше определению:

Теорема 7.5 Следующие уравнения попарно эквивалентны, что означает, что каждое отдельное уравнение описывает условную независимость переменных A и B при заданном C.

$$P(A, B|C) = P(A|C) \cdot P(B|C), \quad (7,16)$$

$$P(A|B,C) = P(A|C), \quad (7,17)$$

$$P(B|A,C) = P(B|C). \quad (7,18)$$

Доказательство. С одной стороны, используя условную независимость (7.16), мы можем заключить, что

$$P(A, B, C) = P(A, B|C)P(C) = P(A|C)P(B|C)P(C).$$

С другой стороны, правило произведения дает нам

$$P(A, B, C) = P(A|B,C)P(B|C)P(C).$$

Таким образом, $P(A|B,C) = P(A|C)$ эквивалентно (7.16). Аналогично получаем (7.18) , переставляя местами A и B в этом выводе.

Machine Translated by Google

7.4.4 Практическое применение

Теперь мы снова обратимся к примеру с тревогой и покажем, как байесовскую сеть на рис. 7.11 на стр. 146 можно использовать для рассуждений. Боба интересует, например, чувствительность двух его тревожных репортеров Джона и Мэри, то есть P (J |Bur) и P (M |Bur). Однако значения P (Bur|J) и P (Bur|M), а также P (Bur|J,M) для него еще важнее. Начнем с P (J |Bur) и вычислим

$$P(J \mid Bur) = \frac{P(Дж, Бур)}{П(Бур)} \overset{"="}{} \frac{P(J, Bur, Al) + P(J, Bur, \neg Al)}{П(Бур)} \tag{7.19}$$

и

$$P(J,Bur,Al) = P(J \mid Bur,Al)P(Al \mid Bur)P(Bur) = P(J \mid Al)P(Al \mid Bur)P(Bur), \tag{7.20}$$

где для последних двух уравнений мы использовали правило произведения и условную независимость J и Bur при данном Al. Подставляя в (7.19), получаем

$$\begin{aligned} P(J \mid Бур) &= P\frac{P(J \mid Al)P(Al \mid Bur)P(Bur) + P(J \mid \neg Al)P(\neg Al \mid Bur)P(Bur)}{(Бур)} \\ &= P(J \mid Al)P(Al \mid Bur) + P(J \mid \neg Al)P(\neg Al \mid Bur). \end{aligned} \tag{7.21}$$

Здесь P (Al |Bur) и P (¬Al |Bur) отсутствуют. Поэтому мы вычисляем

$$\begin{aligned} P(Ал \mid Бур) &= \frac{П(Ал, Бур)}{П(Бур)} \overset{"="}{} \frac{P(Al, Bur, Ear) + P(Al, Bur, \neg Ear)}{П(Бур)} \\ &\overset{"="}{} \frac{P(Al \mid Bur,Ear)P(Bur)P(Ear) + P(Al \mid Bur,\neg Ear)P(Bur)P(\neg Ear)}{П(Бур)} \\ &= P(Al \mid Bur,Ear)P(Ear) + P(Al \mid Bur,\neg Ear)P(\neg Ear) \\ &= 0{,}95 \cdot 0{,}002 + 0{,}94 \cdot 0{,}998 = 0{,}94 \end{aligned}$$

а также P (¬Al |Bur) = 0,06 и подставляем это в (7.21) , что дает результат

$$P(J \mid Bur) = 0{,}9 \cdot 0{,}94 + 0{,}05 \cdot 0{,}06 = 0{,}849.$$

Аналогично вычисляем P (M |Bur) = 0,659. Теперь мы знаем, что Джон вызывает около 85% всех краж со взломом, а Мэри — около 66% всех краж со взломом. Вероятность того, что они оба уравняют, вычисляется в силу условной независимости как

$$P(J,M \mid Bur) = P(J \mid Bur)P(M \mid Bur) = 0{,}849 \cdot 0{,}659 = 0{,}559.$$

Однако более интересна вероятность звонка от Джона или Мэри.

Machine Translated by Google

$$P(J \quad M|Bur) = P(\neg(\neg J, \neg M)|Bur) = 1 \quad P(\neg J, \neg M|Bur)$$
$$= 1 \quad P(\neg J|Bur)P(\neg M|Bur) = 1 \quad 0{,}051 = 0{,}948.$$

Таким образом, Боб получает уведомление примерно о 95% всех краж со взломом. Теперь, чтобы вычислить $P(Bur|J)$, мы применяем формулу Байеса, которая дает нам

$$P(Бур|J) = \frac{P(J \mid Бур)\,P(Бур)}{P(Дж)} \overset{"="}{} \frac{0{,}849 \cdot 0{,}001 =}{0{,}052} 0{,}016.$$

Очевидно, только около 1,6% всех звонков от Джона на самом деле связаны со взломом. Поскольку вероятность ложных тревог для Мэри в пять раз меньше, при $P(Bur|M) = 0{,}056$ мы имеем значительно более высокую достоверность при звонке от Мэри.
Боб должен серьезно беспокоиться о своем доме только в том случае, если они оба звонят, потому что $P(Bur|J,M) = 0{,}284$ (см. упражнение 7.11 на стр. 159).

В (7.21) на стр. 148 мы показали с

$$P(J|Bur) = P(J|Al)P(Al|Bur) + P(J|\neg Al)P(\neg Al|Bur)$$

как мы можем «вставить» новую переменную. Это соотношение в общем случае выполняется для двух переменных A и B при введении дополнительной переменной C и называется обусловленным:

$$P(A|B) = \sum_c P(A|B,C=c)P(C=c|B).$$

Если, кроме того, A и B условно независимы при заданном C, эта формула упрощается до

$$P(A|B) = \sum_c P(A|C=c)P(C=c|B).$$

7.4.5 Программное обеспечение для байесовских сетей

Мы дадим краткое введение в два инструмента на примере будильника. Мы уже знакомы с системой PIT. Мы вводим значения из CPT в синтаксисе PIT в онлайн-окно ввода на сайте www.pit-systems.de. После ввода, показанного на рис. 7.12 на стр. 150 , получаем ответ:

P([Эйнбрух=t] | [Джон=t] И [Мэри=t]) = 0,2841.

Хотя PIT не является классическим байесовским сетевым инструментом, он может принимать произвольные условные вероятности и запросы в качестве входных данных и вычислять правильные результаты. Можно показать [Sch96], что при вводе CPT или эквивалентных правил принцип MaxEnt подразумевает ту же условную независимость и, следовательно, те же ответы, что и байесовская сеть. Таким образом, байесовские сети являются частным случаем MaxEnt.

Machine Translated by Google


```
1 var Тревога{t,f}, Взлом{t,f}, Землетрясение{t,f}, Джон{t,f}, Мэри{t,f};
2
3 P([Землетрясение=t]) = 0,002; 4 P([Кража
со взломом=t]) = 0,001; 5 P([Тревога=t] |
[Кража со взломом=t] И [Землетрясение=t]) = 0,95; 6 P([Тревога=t] | [Кража со взломом=t]
И [Землетрясение=f]) = 0,94; 7 P([Тревога=t] | [Кража со взломом=f] И [Землетрясение=t])
= 0,29; 8 P([Тревога=t] | [Кража со взломом=f] И [Землетрясение=f]) = 0,001; 9 P([Джон=t]
| [Тревога=t]) = 0,90; 10 P([Джон=t] | [Тревога=f]) = 0,05; 11 P([Мария=t] | [Тревога=t]) =
0,70; 12 P([Мария=t] | [Тревога=f]) = 0,01; 13 14
QP([Кража со взломом=t] | [Джон=t] И [Мэри=t]);
```

Рис. 7.12 Ввод PIT для примера тревоги

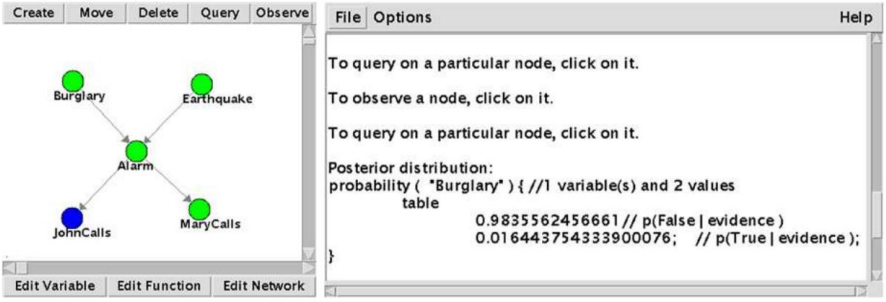


Рис. 7.13 Пользовательский интерфейс JavaBayes: слева — графический редактор, справа — консоль, где в качестве вывода выдаются ответы

Далее мы рассмотрим JavaBayes [Coz98], классическую систему, также свободно доступную в Интернете, с графическим интерфейсом, показанным на рис. 7.13. С помощью графического сетевого редактора можно манипулировать узлами и ребрами, а также редактировать значения в CPT. Кроме того, значения переменных могут быть назначены с помощью «Наблюдения», а значения других переменных могут быть вызваны с помощью «Запроса». Затем ответы на запросы появляются в окне консоли.

Профессиональная коммерческая система Hugin значительно мощнее и удобнее. Например, Хьюгин может использовать непрерывные переменные в дополнение к дискретным переменным. Он также может изучать байесовские сети, то есть полностью автоматически генерировать сеть из статистических данных (см. раздел 8.5).

7.4.6 Разработка байесовских сетей

Компактная байесовская сеть очень понятна и значительно более информативна для читателя, чем полное распределение вероятностей. Кроме того, требуется гораздо меньше

Machine Translated by Google

Память. Для переменных v1,...,vn с |v1|,..., |vn| различные значения каждый, распределение имеет в общей сложности

$$\prod_{i=1}^{n} |v_i| \quad 1$$

самостоятельные записи. В примере с тревогой все переменные являются бинарными. Таким образом, для всех переменных |vi | = 2, и в распределении 25 1 = 31 независимый элемент. Чтобы вычислить количество независимых записей для байесовской сети, необходимо определить общее количество всех записей всех CPT. Для узла vi с ki родительскими узлами имеет ei1,...,eiki, ассоциированный CPT

$$(|v_i| \quad 1) \prod_{j=1}^{k_i} |e_{ij}|$$

записи. Тогда все CPT в сети вместе имеют

$$\sum_{i=1}^{n} (|v_i| \quad 1) \prod_{j=1}^{k_i} |e_{ij}| \tag{7.22}$$

записи.15 Для примера тревоги результатом будет

$$2 + 2 + 4 + 1 + 1 = 10$$

независимые записи, которые однозначно описывают сеть. Сравнение сложности памяти между полным распределением и байесовской сетью становится более ясным, если мы предположим, что все n переменных имеют одинаковое число b значений и каждый узел имеет k родительских узлов. Тогда (7.22) можно упростить, и все КПТ вместе имеют n(b 1)bk элементов. Полное распределение содержит b не может быть усиления, только тогда, когда среднее $\sum_{i=1}^{n}$ 1 запись. Значимое количество родительских узлов намного меньше, чем количество переменных. Это означает, что узлы связаны только локально.

Благодаря локальному соединению сеть становится модульной, что, как и в программной инженерии, приводит к снижению сложности. В примере с тревогой узел тревоги отделяет узлы Bur и Ear от узлов J и M. Мы также ясно видим это в примере с LEXMED .

15Для узла без предков произведение в этой сумме пусто. Для этого мы подставляем значение 1, потому что CPT для узлов без предков содержит с его априорной вероятностью ровно одно значение.

Machine Translated by Google

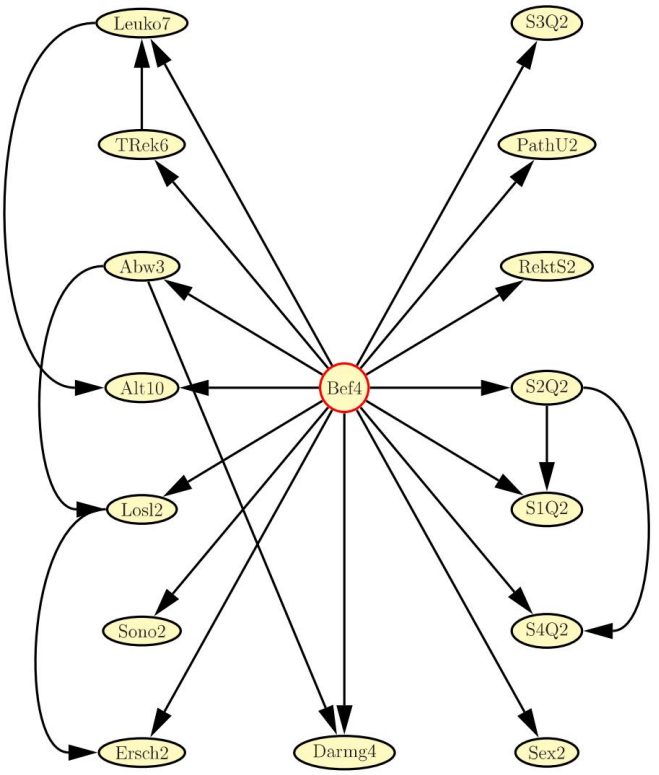


Рис. 7.14 Байесовская сеть для приложения LEXMED

LEXMED как байесовская сеть Система
LEXMED , описанная в разд. 7.3 также можно смоделировать как байесовскую сеть. Сделав внешние тонко нарисованные линии направленными (придав им стрелки), граф независимости на рис. 7.8 на стр. 137 можно интерпретировать как байесовскую сеть. Получившаяся сеть показана на рис. 7.14.

В разд. 7.3.2 размер дистрибутива для LEXMED рассчитывался как значение 20 643 839. С другой стороны, байесовская сеть может быть полностью описана всего 521 значением. Это значение можно определить, введя переменные с рис. 7.14 в (7.22) на стр. 151. В порядке (Лейко, ТРек, Гуа, Возраст, Реб, Соно, Тапп, БоуС, Пол, П4К, П1К, П2К, РекП , Урин, Р3Q, Diag4) вычисляем

$$|\text{Эйдж}| = \sum_{\text{я}=1}^{\text{н}} (|\text{ви}| - 1) \prod_{j=1}^{\text{ки}} = 6 \cdot 6 \cdot 4 + 5 \cdot 4 + 2 \cdot 4 + 9 \cdot 7 \cdot 4 + 1 \cdot 3 \cdot 4 + 1 \cdot 4 + 1 \cdot 2 \cdot 4$$
$$+ 3 \cdot 3 \cdot 4 + 1 \cdot 4 + 1 \cdot 4 \cdot 2 + 1 \cdot 4 \cdot 2 + 1 \cdot 4 + 1 \cdot 4 + 1 \cdot 4$$
$$+ 1 \cdot 4 + 1 = 521.$$

Machine Translated by Google

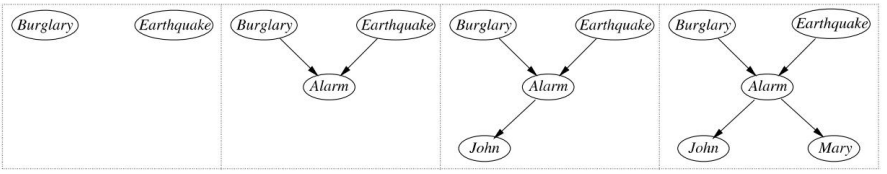


Рис. 7.15 Пошаговое построение сети сигнализации с учетом причинно-следственной связи

Этот пример демонстрирует, что практически невозможно создать полноценный дистрибутив для реальных приложений. С другой стороны, байесовская сеть с 22 ребрами и 521 значением вероятности по-прежнему управляема.

Причинность и структура сети Построение
байесовской сети обычно происходит в два этапа.

1. Проектирование сетевой структуры. Этот этап обычно выполняется вручную и будет описан ниже.
2. Ввод вероятностей в CPT: вручную вводить значения в случае многих переменных очень утомительно. Если (как, например, в случае с LEXMED) доступна база данных, этот шаг можно автоматизировать путем оценки записей CPT путем подсчета частот.

Теперь опишем построение сети на примере тревоги (см. рис. 7.15). В начале мы знаем две причины кражи со взломом и землетрясение и два симптома Джон и Мэри. Однако, поскольку Джон и Мэри не реагируют напрямую на грабителя или землетрясение, а только на тревогу, уместно добавить это как дополнительную переменную, которую Боб не может наблюдать. Процесс добавления ребер начинается с причин, то есть с переменных, не имеющих родительских узлов. Сначала выбираем «Кража со взломом» , а затем «Землетрясение». Теперь мы должны проверить, не зависит ли Землетрясение от Взлома. Это дано, и, таким образом, преимущество от Взлома до Землетрясения не добавляется . Поскольку Тревога напрямую зависит от Взлома и Землетрясения, эти переменные выбираются следующими, и к Тревоге добавляется ребро как от Взлома , так и от Землетрясения . Затем мы выбираем Джона. Поскольку Тревога и Джон не являются независимыми, к Джону добавляется ребро от тревоги. То же верно и для Марии. Теперь мы должны проверить, является ли Джон условно независимым от Взлома с учетом Тревоги. Если это не так, то нужно вставить еще одно ребро от Взлома до Джона. Мы также должны проверить, нужны ли ребра от Землетрясения до Джона и от Взлома или Землетрясения до Мэри. Из-за условной независимости эти четыре ребра не нужны. Ребра между Джоном и Мэри также не нужны, потому что Джон и Мэри условно независимы с учетом Тревоги.

Структура байесовской сети сильно зависит от выбранного порядка переменных. Если порядок переменных выбран так, чтобы отражать причинно-следственную связь, начиная с причин и заканчивая диагностическими переменными, то результатом будет простая сеть. В противном случае сеть может содержать значительно больше ребер.
Такие некаузальные сети часто очень сложны для понимания и имеют более высокую

Machine Translated by Google

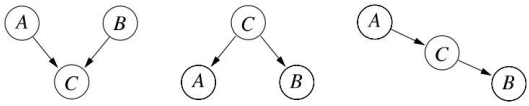


Рис. 7.16 Между A и B нет ребра, если они независимы (слева) или условно независимы (в середине, справа)

сложность для рассуждений. Читатель может обратиться к упражнению 7.11 на стр. 159 для лучшего понимания.

7.4.7 Семантика байесовских сетей

Как мы видели в предыдущем разделе, в байесовскую сеть не добавляется ребро между двумя переменными A и B, когда A и B независимы или условно независимы при заданной третьей переменной C. Эта ситуация представлена на рис. 7.16 .

Теперь мы требуем, чтобы в байесовской сети не было циклов, и мы предполагаем, что переменные пронумерованы так, что ни одна переменная не имеет меньшего номера, чем любая предшествующая ей переменная. Это всегда возможно, когда сеть не имеет циклов16. Тогда при использовании всех условных зависимостей имеем

$$P(X_n | X_1,\ldots,X_{n-1}) = P(X_n | \text{Родители}(X_n)).$$

Таким образом, это уравнение является предположением о том, что произвольная переменная X_i в байесовской сети условно независима от своих предков при заданных родителях. Несколько более общее предложение, изображенное на рис. 7.17 на стр. 155 , можно сформулировать компактно. как

> Теорема 7.6. Узел в байесовской сети условно независим от всех узлов, не являющихся преемниками, при заданных его родителях.

Теперь мы можем значительно упростить цепное правило ((7.1) на стр. 120):

$$P(X_1,\ldots,X_n) = \prod_{i=1}^{n} P(X_i | X_1,\ldots,X_{i-1}) = \prod_{i=1}^{n} P(X_i | \text{Родители}(X_i)). \quad (7.23)$$

Используя это правило, мы могли бы, например, написать (7.20) прямо на странице 148 .

$$P(J,Bur,Al) = P(J | Al)P(Al | Bur)P(Bur).$$

16Если, например, три узла X1, X2, X3 образуют цикл, то существуют ребра (X1, X2), (X2, X3) и (X3, X1) , где X1 имеет X3 в качестве преемника.

Machine Translated by Google

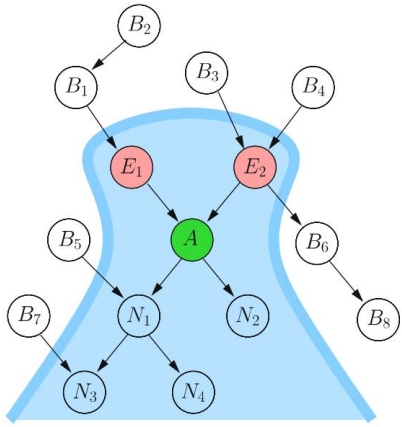


Рис. 7.17 Пример условной независимости в байесовской сети. Если заданы родительские узлы E1 и E2 , то все узлы, не являющиеся преемниками B1,...,B8, не зависят от A.

Теперь мы знаем самые важные концепции и основы байесовской сети. работает. Подытожим их [Jen01]:

Определение 7.7 Байесовская сеть определяется: • Набором переменных и набором направленных ребер между этими переменными. • Каждая переменная имеет конечное число возможных значений. • Переменные вместе с ребрами образуют направленный ациклический граф (DAG).

DAG — это граф без циклов, то есть без путей вида (A,... , A).

• Для каждой переменной A CPT (т. е. таблица условных вероятностей P(A |Родители(A))) дано.

Две переменные A и B называются условно независимыми при заданном C, если P(A, B |C) = P(A |C) · P(B |C) или, что то же самое, если P(A |B,C) = P(A |C). Помимо основных правил вычисления вероятностей, также верны следующие правила:

Теорема Байеса: $P(A|B) = \frac{P(B|A)\cdot P(A)}{P(B)}$

Маргинализация: P (B) = P (A, B) + P (¬A, B) = P (B |A) · P (A) + P (B |¬A) · P (¬A)

Кондиционирование: $P(A|B) = \sum_c P(A|B,C=c)P(C=c|B)$

Переменная в байесовской сети условно независима от всех переменных, не являющихся последующими, с учетом ее родительских переменных. Если X1,...,Xn 1 не являются потомками Xn, то P(Xn |X1,...,Xn 1) = P(Xn | Parents(Xn)). Это условие должно соблюдаться при построении сети.

При построении байесовской сети переменные должны быть упорядочены в соответствии с причинно-следственными связями. Сначала причины, затем скрытые переменные и, наконец, переменные диагностики.

Цепное правило: $P(X_1,...,X_n) = \prod_{i=1}^{n} P(X_i \mid \text{Родители}(X_i))$

Machine Translated by Google

В [Pea88] и [Jen01] для байесовских сетей введен термин d-разделение, из которого следует теорема, аналогичная теореме 7.6 на стр. 154 . Мы воздержимся от введения этого термина и тем самым придем к несколько более простому, хотя и не столь чистому в теоретическом отношении представлению.

7.5 Резюме

Путем, отражающим длительную, устойчивую тенденцию к вероятностным системам, мы ввели вероятностную логику для рассуждений с неопределенными знаниями. После знакомства с языком — пропозициональной логикой, дополненной вероятностями или вероятностными интервалами — мы выбрали естественный, хотя и необычный подход с использованием метода максимальной энтропии в качестве точки входа и показали, как мы можем моделировать немонотонные рассуждения с помощью этого метода. Затем были введены байесовские сети как частный случай метода MaxEnt.

Почему байесовские сети являются частным случаем MaxEnt? При построении байесовской сети делаются предположения о независимости, которые для метода MaxEnt не нужны. Кроме того, при построении байесовской сети все CPT должны быть полностью заполнены, чтобы можно было построить полное распределение вероятностей.
В противном случае рассуждение ограничено или невозможно. С другой стороны, с MaxEnt разработчик может сформулировать все имеющиеся в его распоряжении знания в виде вероятностей. Затем MaxEnt завершает модель и генерирует распределение. Даже если (например, при опросе эксперта) доступны только расплывчатые предложения, это можно соответствующим образом смоделировать. Утверждение типа «Я почти уверен, что A истинно». например, можно смоделировать, используя P (A) [0,6, 1] в качестве интервала вероятности. При построении байесовской сети необходимо указать конкретное значение вероятности, при необходимости путем угадывания. Однако это означает, что эксперт или разработчик вводят в систему специальную информацию. Еще одним преимуществом MaxEnt является возможность формулировать (почти) произвольные предложения. Для байесовских сетей CPT должны быть заполнены.

Свобода, которой обладает разработчик при моделировании с помощью MaxEnt, может быть недостатком (особенно для новичка), поскольку, в отличие от байесовского подхода, не обязательно ясно, какие знания следует моделировать. При моделировании байесовскими сетями подход достаточно ясен: по причинно-следственным зависимостям, от причин к следствиям, одно ребро за другим вводится в сеть путем проверки условной независимости17. В конце все КПН заполняются значениями .

Однако возможны следующие интересные комбинации двух методов: мы начинаем с построения сети по байесовской методологии, вводим соответственно все ребра и затем заполняем КПТ значениями. Если определенные значения для CPT недоступны, то они могут быть заменены интервалами или другими формулами вероятностной логики. Естественно, такая сеть — или, лучше сказать, набор правил — не

17Это тоже не всегда так просто.

Machine Translated by Google

имеет особую семантику байесовской сети. Затем он должен быть обработан и завершен системой MaxEnt.

Однако возможность использовать MaxEnt с произвольными наборами правил имеет и обратную сторону. Подобно ситуации в логике, такие наборы правил могут быть противоречивыми. Например, два правила P (A) = 0,7 и P (A) = 0,8 несовместимы. В то время как система MaxEnt PIT, например, может распознать несоответствие, если не может дать подсказку о том, как устранить проблему.

Мы представили медицинскую экспертную систему LEXMED, классическое приложение для рассуждений с неопределенными знаниями, и показали, как ее можно смоделировать и реализовать с помощью MaxEnt и байесовских сетей, и как эти инструменты могут заменить хорошо зарекомендовавший себя, но слишком слабый линейный скоринг. системы, применяемые в медицине.[18]

На примере LEXMED мы показали, что можно построить экспертную систему для рассуждений в условиях неопределенности, которая способна обнаруживать (извлекать) знания из данных в базе данных. Мы подробнее рассмотрим методы обучения байесовских сетей в гл. 8, после того как были заложены необходимые основы для машинного обучения.

Сегодня байесовские рассуждения представляют собой большую независимую область, которую мы можем здесь описать лишь вкратце. Мы полностью отказались от обработки непрерывных переменных. Для случая нормально распределенных случайных величин существуют процедуры и системы. Однако для произвольных распределений вычислительная сложность представляет собой большую проблему. В дополнение к направленным сетям, которые в значительной степени основаны на причинно-следственных связях, существуют также ненаправленные сети. С этим связана дискуссия о значении и полезности причинности в байесовских сетях. Заинтересованному читателю предлагается ознакомиться с отличными учебниками, такими как [Pea88, Jen01, Whi96, DHS01], а также с материалами ежегодной конференции Ассоциации по изучению неопределенностей в искусственном интеллекте (AUAI) (www.auai.org).

7.6 Упражнения

Упражнение 7.1. Докажите утверждение из теоремы 7.1 на стр. 117.

Упражнение 7.2 Садовод Макс хочет статистически проанализировать свой ежегодный урожай гороха. Для каждого сорванного стручка гороха он измеряет его длину xi в сантиметрах и вес yi в граммах. Он делит горох на два класса: хороший и плохой (пустые стручки). Измеренные данные (xi, yi)

хороший горох:	x 122334456 y 234455666	плохой горох:	x 4667 г 2233

18В разд. 8.6 и в упражнении 8.16 на стр. 219 мы покажем, что оценки эквивалентны наивному байесовскому методу для особого случая, то есть предположению, что все симптомы условно независимы при данном диагнозе.

Machine Translated by Google

(а) По данным вычислите вероятности P (y > 3 |Класс = хорошо) и
P (y 3 | Класс = хорошо). Затем используйте формулу Байеса, чтобы определить P (класс =
хорошо | y > 3) и P (класс = хорошо | y 3).

(б) Какая из вероятностей, вычисленных в подзадаче (а), противоречит утверждению
«Все хорошие горошины тяжелее 3 граммов»?

Упражнение 7.3. Вы должны предсказать дневную погоду, используя несколько простых
значения погоды с утра этого дня. Классический расчет вероятности
для этого требуется полная модель, которая приведена в следующей таблице.

Небо	Бар	Prec	P (небо, бар, Prec)
Прозрачный	Поднимается		0,40
Прозрачный	Сухой	Поднимается	Дождь 0,07
Прозрачный	Падение	насухо	0,08
Прозрачный	Падающий	дождь	0.10
Bewö lkt	Rising	Dry	Bewö lkt 0,09
Rising	Raining	0.11	
Bewö lkt	Falling	Dry	0.03

Небо: небо чистое или
облачно утром
Бар: барометр поднимается или
падение утром
Prec: Дождь или сухо в
полдень

а) Сколько событий в распределении этих трех переменных?

(b) Вычислите P (Prec = сухое | небо = чистое, Bar = поднимающееся).

(c) Вычислите P (Prec = дождь |Sky = облачно).

г) Что бы вы сделали, если бы в таблице не было последней строки?

Упражнение 7.4 В телевизионной викторине участник должен выбрать между тремя
закрытые двери. За одной дверью ждет приз: автомобиль. За обоими другими
двери козлы. Участник выбирает дверь, например номер один. Хозяин, который
знает, где машина, открывает другую дверь, например, номер три, и появляется коза.
Теперь участнику предоставляется возможность выбрать между двумя оставшимися
двери (одна и две). Что является лучшим выбором с его точки зрения? остаться с
дверь, которую он изначально выбрал, или переключиться на другую закрытую дверь?

Упражнение 7.5 Используя метод множителей Лагранжа, покажите, что без явного
ограничений, равномерное распределение p1 = p2 =···= pn = 1/n представляет максимальную энтропию.
Не забывайте о постоянно присутствующем неявном ограничении p1 + p2 +···+
pn = 1. Как мы можем показать тот же результат, используя безразличие?

Упражнение 7.6. Используйте систему PIT (www.pit-systems.de) для расчета MaxEnt.
решение для P (B) при ограничении P (A) = α и P (B |A) = β. Какой недостаток PIT по сравнению с ручным
расчетом вы здесь заметили?

Упражнение 7.7. Учитывая ограничения P (A) = α и P (A B) = β, вручную вычислите P (B), используя
метод MaxEnt. Используйте PIT, чтобы проверить свое решение.

Machine Translated by Google

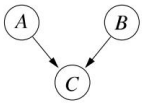
Упражнение 7.8. Учитывая ограничения из (7.10), (7.11), (7.12): p1 + p2 = α, p1 + p3 = γ , p1 + p2 + p3 + p4 = 1. Покажите, что p1 = αγ, p2 = α (1 γ), p3 = γ (1 α), p4 = (1 α)(1 γ) представляет собой максимум энтропии при этих ограничениях.

Упражнение 7.9 Вероятностный алгоритм вычисляет вероятность p того, что входящее электронное письмо является спамом. Чтобы классифицировать электронные письма по классам удаления и
чтения, к результату применяется матрица затрат. (a) Приведите матрицу затрат (матрица 2 × 2) для спам-фильтра. Предположим, что удаление сообщения электронной почты стоит пользователю 10 центов, а потеря сообщения электронной почты стоит 10
долларов (сравните это с примером 1.1 на странице 11 и упражнением 1.7 на странице 14). (b) Показать, что в случае матрицы 2 × 2 применение матрицы затрат эквивалентно применению порога вероятности спама, и определить порог.

Упражнение 7.10. Дана байесовская сеть с тремя бинарными переменными A, B, C и P (A) = 0,2, P (B) = 0,9, а также CPT, показанный ниже:

(a) Вычислите P (A|B). (b) Вычислите P (C|A).

АД (K) wf 0,1	
ww 0,2	
вперед 0,9	
вперед 0,4	



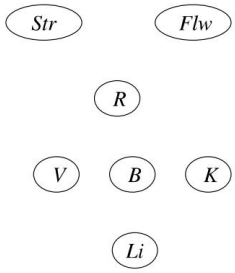
Упражнение 7.11 Для примера тревоги (Пример 7.7 на стр. 145) рассчитайте следующие условные вероятности: (a) Рассчитайте
априорные вероятности P (Al), P (J), P (M). (b) Рассчитайте P (M|Bur),
используя правило произведения, маргинализацию, цепное правило и условная независимость. (c)
Используйте формулу Байеса для расчета P (Bur|M). (d)
Вычислите P (Al||J,M) и P (Bur||J,M). (e) Покажите, что
переменные J и M не являются независимыми. (f) Проверьте все
свои результаты с помощью JavaBayes и PIT (см. [Ert11] для демонстрации программы).
(g) Разработайте байесовскую сеть для примера тревоги, но с измененным порядком переменных M, Al, Ear, Bur, J . Согласно семантике байесовских сетей, должны быть нарисованы только необходимые ребра. (Подсказка: приведенный здесь порядок переменных НЕ отражает причинно-следственную связь. Таким образом, будет трудно интуитивно определить условную независимость.)
(h) В исходной байесовской сети примера тревоги узлы землетрясения удалены. Какие CPT это меняет? (Почему именно эти?)
(i) Рассчитайте CPT аварийного узла в новой сети.

Упражнение 7.12 Необходимо создать диагностическую систему для велосипедного фонаря с динамо-машиной, используя байесовскую сеть. Переменные в следующей таблице даны.

Machine Translated by Google

Сокр.	Это означает, что	Ценности
Ли	Лайт находится в	т/ф
ул.	уличном состоянии.	сухой, мокрый, заснеженный
Износ маховика Flw Dynamo т/ф		
р	Динамо скользящее т/ф	
В	Динамо показывает напряжение t/f	
Б	Лампочка в норме т/ф	
К	Кабель в порядке т/ф	

Следующие переменные попарно независимы:
Улица, Полет, Б, К. Далее: (R, B), (R, K), (V , B),
(V , K) независимы, и следующее уравнение
держит:



$$P(Li \mid V, R) = P(Li \mid V)$$

$$P(V \mid R, Str) = P(V \mid R)$$

$$P(V \mid R, Flw) = P(V \mid R)$$

(a) Нарисуйте все ребра в графе (принимая во внимание причинность).

(b) Внесите все недостающие CPT в график (таблицу условных вероятностей). Свободно вставляйте правдоподобные значения вероятностей.

в) Показать, что сеть не содержит ребер (Стр, Ли).

(d) Вычислите P (V |Str = snowcovered).

ВБК П(Ли)
ттт 0,99
ттф 0,01
т фут 0,01
т фф 0,001
фут т 0,3
фут 0,005
фт 0,005
0

Machine Translated by Google

8 Машинное обучение и интеллектуальный анализ данных

Если мы определим ИИ, как это сделано в книге Элейн Рич [Ric83]:

> Искусственный интеллект — это наука о том, как заставить компьютеры делать то, что в момент, люди лучше.

а если учесть, что обучаемость компьютера особенно уступает этому человека, то следует, что исследование механизмов обучения и разработка алгоритмов машинного обучения является одним из важнейших направлений ИИ.

Существует также спрос на машинное обучение с точки зрения программного обеспечения. разработчик, который программирует, например, поведение автономного робота. структура разумного поведения может стать настолько сложной, что очень трудно или даже невозможно запрограммировать близко к оптимальному, даже с современными языками высокого уровня, такими как PROLOG и Python.1 Даже используются алгоритмы машинного обучения сегодня программировать роботов так же, как учатся люди (см. главу 10 или [BCDS08, RGH+06]), часто в гибридной смеси запрограммированного и усвоенного поведение.

Задача этой главы — описать наиболее важные алгоритмы машинного обучения и их приложения. Тема будет представлена в этом разделе, а затем важными фундаментальными алгоритмами обучения в следующих разделах. Параллельно с этим будут выстраиваться теория и терминология. Глава завершится подведением итогов и обзор различных алгоритмов и их приложений. В этой главе мы ограничимся рассмотрением алгоритмов обучения с учителем и без учителя. Нейронные сети, как важный класс алгоритмов обучения, будут рассмотрены в гл. 9.
Благодаря своему особому месту и важной роли для автономных роботов арматура обучению также будет посвящена отдельная глава (глава 10).

Что такое обучение? Изучение лексики иностранного языка, или технического термины, или даже запоминание стихотворения может быть трудным для многих людей. Для компьютеров,

1Python — это современный язык сценариев с очень читаемым синтаксисом, мощными типами данных и обширными стандартными библиотеками, которые можно использовать для этой цели.

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по информатике,
DOI 10.1007/978-0-85729-299-5_8, © Springer-Verlag London Limited 2011

Machine Translated by Google

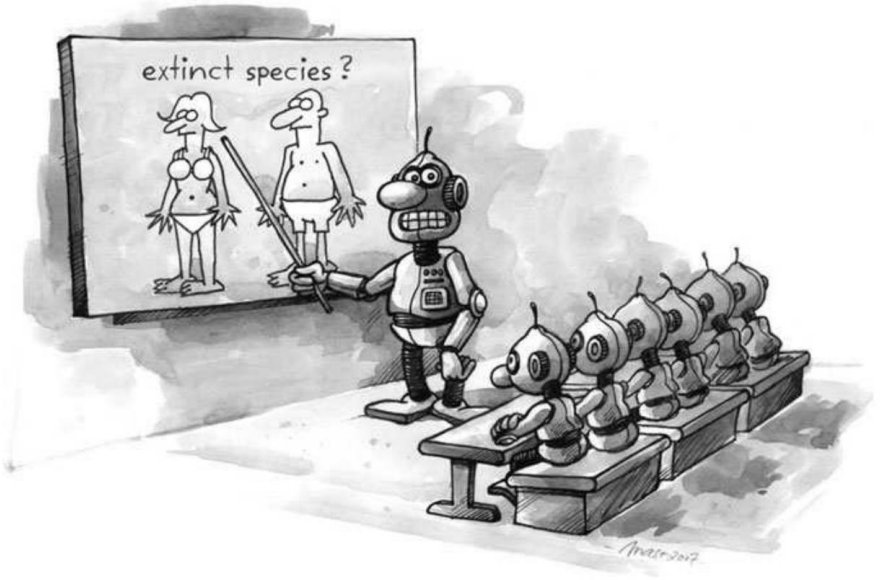


Рис. 8.1 Обучение с учителем. . .

однако эти задачи довольно просты, потому что они не более чем сохранение текста в файле. Таким образом, запоминание неинтересно для ИИ. Напротив, приобретение математических навыков обычно не происходит путем запоминания. Для сложения натуральных чисел это вообще невозможно, так как для каждого из слагаемых в сумме x + y имеется бесконечно много значений. Для каждой комбинации двух значений x и y пришлось бы хранить тройку (x, y, x + y), что невозможно. Для десятичных чисел это совершенно невозможно. Это ставит вопрос: как мы изучаем математику? Ответ звучит так: Учитель объясняет процесс, а ученики практикуют его на примерах до тех пор, пока не перестанут делать ошибки на новых примерах. После 50 примеров ученик (надеюсь) понимает сложение. То есть всего после 50 примеров он может применить выученное к бесконечно большому количеству новых примеров, которых до этого момента не видел. Этот процесс известен как генерализация. Начнем с простого примера.

Пример 8.1. Фермер, выращивающий фрукты, хочет автоматически разделить собранные яблоки на товарные классы A и B. Сортировочное устройство оснащено датчиками для измерения двух характеристик, размера и цвета, а затем решает, к какому из двух классов относится яблоко. Это типичная задача классификации. Системы, способные разбивать векторы признаков на конечное число классов, называются классификаторами.

Для настройки машины яблоки собираются специалистом вручную, то есть классифицируются. Затем два измерения вносятся вместе с меткой их класса в таблицу (Таблица 8.1 на стр. 163). Размер указывается в виде диаметра в сантиметрах, а цвет — в виде числового значения от 0 (для зеленого) до 1 (для красного).

Machine Translated by Google

Таблица 8.1 Данные обучения для агента сортировки яблок

Размер [см]	8	8 6 3. . .
Цвет	0,1 0,3 0,9 0,8 . . .	
Товарный класс	B	A A B . . .



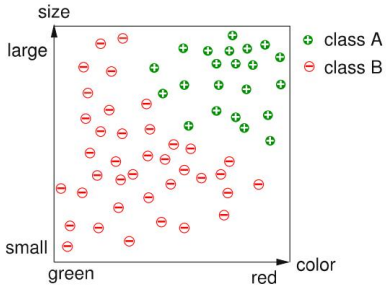
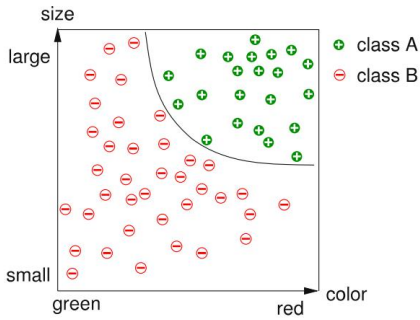


Рис. 8.2 Оборудование для сортировки яблок компании BayWa в Крессбронне и некоторые яблоки, классифицированные по товарным классам A и B в функциональном пространстве (Фото: BayWa)

Рис. 8.3 Кривая , нарисованная на диаграмме, разделяет классы и затем может быть применена к произвольным новым яблокам



Визуализация данных представлена в виде точек на диаграмме рассеяния справа на рис. 8.2.

Задача машинного обучения состоит в создании функции из собранных классифицированных данных, которая вычисляет значение класса (A или B) для нового яблока на основе двух характеристик размера и цвета. На рис. 8.3 такая функция показана разделительной линией, проведенной через диаграмму. Все яблоки с вектором признаков в левом нижнем углу линии помещаются в класс B, а все остальные — в класс A.

В этом примере по-прежнему очень просто найти такую разделительную линию для двух классов. Явно более трудная и, главное, гораздо менее наглядная задача, когда классифицируемые объекты описываются не двумя, а многими признаками. На практике обычно используется 30 и более признаков. Для n признаков задача состоит в том, чтобы найти n - 1-мерную гиперплоскость в n-мерном пространстве признаков , которая как можно лучше разделяет классы. «Хорошее» деление означает, что процент ошибочно классифицированных объектов как можно меньше.

Machine Translated by Google



Рис. 8.4 Функциональная структура обучающего агента для сортировки яблок (слева) и в целом (справа)

Классификатор сопоставляет вектор признаков со значением класса. Здесь он имеет фиксированное, обычно небольшое, количество альтернатив. Желаемое отображение также называется целевой функцией. Если целевая функция не отображается на конечную область, то это не классификация, а задача аппроксимации. Такой аппроксимационной задачей является определение рыночной стоимости акций по заданным характеристикам. В следующих разделах мы представим несколько обучающих агентов для обоих типов отображений.

Агент обучения Мы можем формально описать агент обучения как функцию, которая отображает вектор признаков в значение дискретного класса или, вообще говоря, в действительное число. Эта функция не запрограммирована, а возникает или изменяется на этапе обучения под влиянием данных обучения. На рис. 8.4 такой агент представлен в примере сортировки яблок. Во время обучения агент получает уже классифицированные данные из Таблицы 8.1 на стр. 163. После этого агент составляет как можно более точное отображение вектора признаков в значение функции (например, класс товаров).

Теперь мы можем попытаться подойти к определению термина «машинное обучение». Том Митчелл [Mit97] дает такое определение:

Машинное обучение — это изучение компьютерных алгоритмов, улучшающих автоматизацию. через опыт.

Опираясь на это, мы даем

Определение 8.1 Агент является обучающимся агентом, если он улучшает свою производительность (измеряемую с помощью подходящего критерия) на новых, неизвестных данных с течением времени (после того, как он увидел множество обучающих примеров).

Важно проверить способность алгоритма обучения к обобщению на неизвестных данных, на тестовых данных. В противном случае каждая система, которая только что сохранила тренировочные данные, будет работать оптимально, просто вызывая сохраненные данные. Агент обучения характеризуется следующими
терминами: Задача: задачей алгоритма обучения является изучение отображения. Это может быть, например, сопоставление размера и цвета яблока с его товарным классом, а также сопоставление 15 симптомов пациента с решением об удалении аппендикса.

Machine Translated by Google



Рис. 8.5 Интеллектуальный анализ данных

Переменный агент (точнее, класс агентов): здесь мы должны решить, с каким алгоритмом обучения будем работать. Если это было выбрано, таким образом определяется класс всех обучаемых функций.

Данные обучения (опыт): данные обучения (выборка) содержат знания, которые алгоритм обучения должен извлечь и изучить. При выборе обучающих данных необходимо убедиться, что это репрезентативная выборка для изучаемой задачи.

Тестовые данные: важны для оценки того, может ли обученный агент хорошо обобщать обучающие данные на новые данные.

Критерий эффективности: для устройства для сортировки яблок количество правильно классифицированных яблок. Он нужен нам для проверки качества агента. Знать показатель производительности обычно намного проще, чем знать функцию агента. Например, легко измерить производительность (время) бегуна на 10 000 метров. Однако это вовсе не означает, что судья, измеряющий время, может бежать так же быстро. Судья знает только, как измерить производительность, но не «функцию» агента, чью производительность он измеряет.

Что такое интеллектуальный анализ данных? Задача обучающей машины извлекать знания из обучающих данных. Часто разработчик или пользователь хотят, чтобы обучающаяся машина сделала извлеченные знания понятными для людей. Еще лучше, если разработчик может даже изменить знания. Процесс индукции деревьев решений в разд. 8.4 является примером такого метода.

Аналогичные проблемы связаны с электронным бизнесом и управлением знаниями. Здесь возникает классическая проблема: от действий посетителей его веб-портала,

Machine Translated by Google

владелец интернет-бизнеса хотел бы создать отношения между
характеристики покупателя и класс товаров, которые ему интересны
клиент. Тогда продавец сможет размещать рекламу для конкретных клиентов. Этот
красноречиво демонстрируется на сайте www.amazon.com, где покупателю рекомендуются товары,
аналогичные тем, что он видел при предыдущем посещении. Во многих
В областях рекламы и маркетинга, а также в управлении взаимоотношениями с клиентами (CRM) начинают
использоваться методы интеллектуального анализа данных. Всякий раз, когда большие суммы
данных доступны, можно попытаться использовать эти данные для анализа клиентов
предпочтения для показа рекламы, ориентированной на клиента. Развивающееся поле
обучения предпочтениям посвящена этой цели.

Процесс получения знаний из данных, а также их представление
и приложение, называется интеллектуальным анализом данных. Используемые методы обычно принимаются
из статистики или машинного обучения и должен быть применим к очень большим
объемы данных по разумной цене.

В контексте получения информации, например, в Интернете или в интрасети, интеллектуальный анализ текста играет все более важную роль. Типичные задачи включают в себя поиск похожего текста в поисковой системе или классификацию текстов, которые, например,
применяется в спам-фильтрах для электронной почты. В разд. 8.6.1 мы представим широко распространенный наивный байесовский алгоритм классификации текста. Относительно новая задача для интеллектуальный анализ данных — это извлечение структурной, статической и динамической информации из графовые структуры, такие как социальные сети, сети трафика или интернет-трафик.

Поскольку две описанные задачи машинного обучения и интеллектуального анализа данных формально очень похожи, основные методы, используемые в обеих областях, по большей части идентичны. Поэтому при описании алгоритмов обучения не будет различия.
сделано между машинным обучением и интеллектуальным анализом данных.

Благодаря огромному коммерческому влиянию методов интеллектуального анализа данных в настоящее время множество сложных оптимизаций и целая линейка мощных систем интеллектуального анализа данных, которые предлагают большую палитру удобных инструментов для извлечения знаний из данные. Такая система представлена в разд. 8.8.

8.1 Анализ данных

Статистика предоставляет несколько способов описания данных с помощью простых параметров. От из них мы выбираем несколько, которые особенно важны для анализа обучения
данные и протестировать их на подмножестве данных LEXMED из разд. 7.3. В этом примере
набор данных, симптомы x1,...,x15 у N = 473 пациентов, кратко описанные в Таблице 8.2 на стр. 167, а также метка класса, то есть диагноз (аппендицит
положительные/отрицательные) — перечислены. Пациент номер один, например, описан
вектор

$$x_1 = (26, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 37{,}9, 38{,}8, 23100, 0, 1)$$

Machine Translated by Google

Таблица 8.2 Описание переменные x1,...,x16. Немного другой формализация использовалась в Табл. 7.2 на стр. 133

Вар. число	Описание	Ценности
1	Возраст	Непрерывный
2	Пол (1=мужской, 2=женский)	1, 2
3	Квадрант боли 1	0, 1
4	Квадрант боли 2	0, 1
5	Квадрант боли 3	0, 1
6	Квадрант боли 4	0, 1
7	Местная мышечная защита	0, 1
8	Генерализованная мышечная защита	0, 1
9	Отскок нежности	0, 1
10	Боль при постукивании	0, 1
11	Боль во время ректального исследования	0, 1
12	Осевая температура	Непрерывный
13	Ректальная температура	Непрерывный
14	Лейкоциты	Непрерывный
15	Сахарный диабет	0, 1
16	Аппендицит	0, 1

и пациент номер два

$$^{2}x = (17, 2, 0, 0, 1, 0, 1, 0, 1, 1, 0, 36{,}9, 37{,}4, 8100, 0, 0)$$

У пациента номер два значение лейкоцитов $x_{14}^{2} = 8100$.

Для каждой переменной xi ее среднее значение $\bar{x}_i$ определяется как

$$\bar{x}_{я} := \frac{1}{H} \sum_{p=1}^{H} x_{я}^{п}$$

и стандартное отклонение si как мера его среднего отклонения от среднего значение как

$$си := \sqrt{\frac{1}{H-1} \sum_{p=1}^{H} (x_{я}^{опыт} - \bar{x}_i)^2}.$$

Вопрос о том, являются ли две переменные xi и xj статистически зависимыми (коррелированными), важен для анализа многомерных данных. Например, ковариация

$$\sigma_{ij} = \frac{1}{H-1} \sum_{p=1}^{H} (x_{я}^{опыт} - \bar{x}_i)(x_{дж}^{п} - \bar{x}_j)$$

Machine Translated by Google

Таблица 8.3 Матрица корреляции для 16 переменных аппендицита, измеренных в 473 случаях

1.	0,009	0,14	0,037	0,096	0,12	0,018	0,051	-0,034	-0,041	0,034	0,037	0,05	-0,037	0,37	0,012
0,009	1.	0,0074	0,019	0,06	0,063	0,17	0,0084	-0,17	-0,14	-0,13	-0,017	-0,034	-0,14	0,045	0,2
0,14	-0,0074	1.	0,55	0,091	0,24	0,13	0,24	0,045	0,18	0,028	0,02	0,045	0,03	0,11	0,045
0,037	0,019	0,55	1.	0,24	0,33	0,051	0,25	0,074	0,19	0,087	0,11	0,12	0,11	0,14	-0,0091
0,096	0,06	0,091	0,24	1.	0,059	0,14	0,034	0,14	0,049	0,057	0,064	0,058	0,11	0,017	0,14
0,12	0,063	0,24	0,33	0,059	1.	0,071	0,19	0,086	0,15	0,048	0,11	0,12	0,063	0,21	0,053
0,018	0,17	0,13	0,051	0,14	0,071	1.	0,16	0,4	0,28	0,2	0,24	0,36	0,29	0,0001	0,33
0,051	0,0084	0,24	0,25	0,034	0,19	0,16	1.	0,17	0,23	0,24	0,19	0,24	0,27	0,083	0,084
0,034	0,17	0,045	0,074	0,14	0,086	0,4	0,17	1.	0,53	0,25	0,19	0,27	0,27	0,026	0,38
0,041	0,14	0,18	0,19	0,049	0,15	0,28	0,23	0,53	1.	0,24	0,15	0,19	0,23	0,02	0,32
0,034	0,13	0,028	0,087	0,057	0,048	0,2	0,24	0,25	0,24	1.	0,17	0,17	0,22	0,098	0,17
0,037	0,017	0,02	0,11	0,064	0,11	0,24	0,19	0,19	0,15	0,17	1.	0,72	0,26	0,035	0,15
0,05	0,034	0,045	0,12	0,058	0,12	0,36	0,24	0,27	0,19	0,17	0,72	1.	0,38	0,044	0,21
0,037	0,14	0,03	0,11	0,11	0,063	0,29	0,27	0,27	0,23	0,22	0,26	0,38	1.	0,051	0,44
0,37	0,045	0,11	0,14	0,017	0,21	0,0001	0,083	0,026	0,02	0,098	0,035	0,044	0,051	1.	0,0055
0,012	0,2	0,045	0,0091	0,14	0,053	0,33	0,084	0,38	0,32	0,17	0,15	0,21	0,44	-0,0055	1.

дает информацию об этом. В этой сумме слагаемое возвращает положительную запись для
р-й вектор данных точно тогда, когда отклонения i-й и j-й компонент от
средние оба имеют одинаковый знак. Если они имеют разные знаки, то запись
отрицательный. Следовательно, ковариация σ12,13 двух различных значений лихорадки должна быть равна
вполне однозначно положительный.

Однако ковариация также зависит от абсолютного значения переменных,
что затрудняет сравнение значений. Чтобы можно было сравнить степень
зависимости в случае нескольких переменных, поэтому мы определяем корреляцию
коэффициент

$$\text{Кий} = \frac{\sigma ij}{\text{си} \cdot \text{сж}}$$

для двух значений xi и xj , что представляет собой не что иное, как нормализованную ковариацию. Матрица
K всех коэффициентов корреляции содержит значения от 1 до 1, является симметричным,
и все его диагональные элементы имеют значение 1. Корреляционная матрица для всех 16
переменных приведены в таблице 8.3.

Эта матрица становится несколько более читаемой, когда мы представляем ее как плотность
сюжет. Вместо числовых значений элементы матрицы на рис. 8.6 на стр. 169
заполнены серыми значениями. На правой диаграмме показаны абсолютные значения. Таким образом
мы можем очень быстро увидеть, какие переменные демонстрируют слабую или сильную зависимость. Мы
можно видеть, например, что переменные 7, 9, 10 и 14 демонстрируют самую сильную корреляцию с
переменной класса аппендицит и поэтому более важны для
диагноз, чем другая переменная. Мы также видим, однако, что переменные 9 и 10
сильно коррелированы. Это может означать, что одно из этих двух значений потенциально
достаточно для диагностики.

Machine Translated by Google

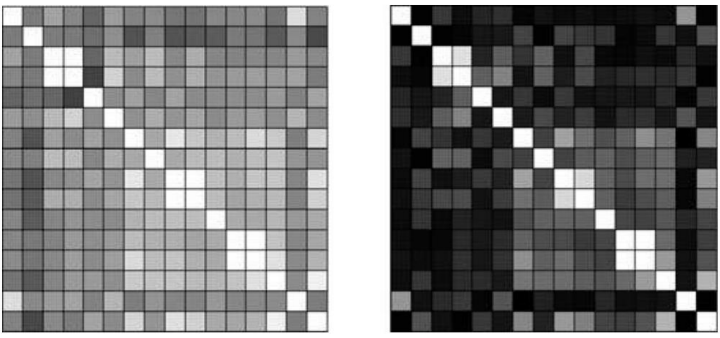
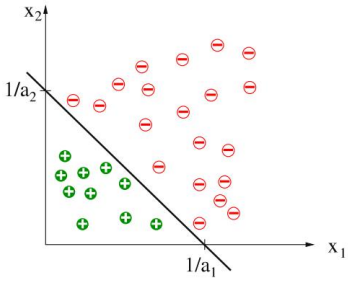


Рис. 8.6 Корреляционная матрица в виде частотного графика. На левой диаграмме темный цвет соответствует отрицательному, а светлый — положительному. На правом изображении были перечислены абсолютные значения. Здесь черный означает Kij 0 (некоррелированный), а белый |Kij | 1 (сильно коррелирует)

Рис. 8.7. Линейно разделимый двумерный набор данных. Уравнение разделительной прямой: a1x1 + a2x2 = 1



8.2 Персептрон, линейный классификатор

В примере классификации сортировки яблок между двумя классами на рис. 8.3 на стр. 163 проведена изогнутая разделительная линия. Более простой случай показан на рис. 8.7. Здесь двумерные обучающие примеры могут быть разделены прямой линией. Мы называем такой набор обучающих данных линейно разделимым. В n измерениях для разделения необходима гиперплоскость. Это представляет собой линейное подпространство размерности n - 1.

Поскольку каждая (n 1)-мерная гиперплоскость в уравнении R н можно описать с помощью

$$\sum_{i=1}^{n} a_i x_i = \theta$$

имеет смысл определить линейную отделимость следующим образом.

Machine Translated by Google

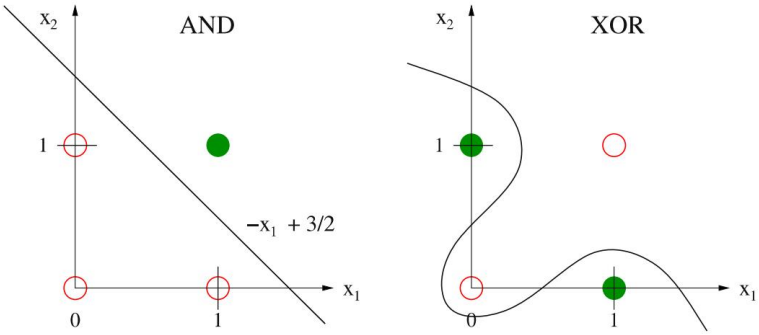


Рис. 8.8. Булева функция AND линейно разделима, а XOR — нет (ˆ= true, ˆ= ложь)

Определение 8.2 Два множества M1 R , н и M2 R н называются линейно разделимыми если существуют вещественные числа a1,...,an, θ с

н н
aixi > θ для всех x M1 и aixi θ для всех x M2.
я=1 я=1

Значение θ обозначает порог.

На рис. 8.8 мы видим, что функция И является линейно разделимой, а функция исключающее ИЛИ — нет. Для И, например, линия x1 + 3/2 разделяет истинную и неверную интерпретацию формулы x1 x2. Напротив, функция XOR не имеет прямой линии разделения. Очевидно, что в этом отношении функция XOR имеет более сложную структуру, чем функция AND.

С персептроном мы представляем очень простой алгоритм обучения, который может

арные линейно отделимые множества.

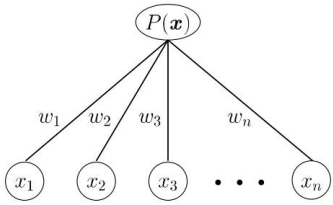
Определение 8.3. Пусть w = (w1,...,wn) R — н — весовой вектор и x R {0, н входной вектор. Персептрон представляет собой функцию P : R н 1}, что соответствует следующему правилу :

$$P(x) = \begin{cases} 1, \text{ если } wx = \sum_{i=1}^{n} w_i x_i > 0, \\ 0 \text{ еще.} \end{cases}$$

Персептрон [Рос58, МП69] — очень простой алгоритм классификации. Это эквивалентно двухслойной нейронной сети с активацией пороговой функцией, показанной на рис. 8.9 на стр. 171. Как показано в гл. 9 каждый узел в сети представляет собой нейрон, а каждое ребро — синапс. Однако пока мы будем только

Machine Translated by Google

Рис. 8.9 Графическое представление персептрона в виде двухслойной нейронной сети



рассматривайте персептрон как обучающий агент, то есть как математическую функцию, которая отображает вектор признаков в значение функции. Здесь входные переменные xi обозначены признаками.

Как видно из формулы $\sum_{i=1} w_i x_i > 0$, все точки x выше гиперплоскости $\sum_{i=1} w_i x_i = 0$ классифицируются как положительные (P(x)=1), а все остальные — как отрицательные (P(x) = 0). Разделяющая гиперплоскость проходит через начало координат, поскольку θ = 0.

Мы воспользуемся небольшой уловкой, чтобы показать, что отсутствие произвольного порога не означает ограничения мощности. Однако сначала мы хотим ввести простой алгоритм обучения для персептрона.

8.2.1 Правило обучения

С обозначениями M+ и M- для наборов положительных и отрицательных шаблонов обучения соответственно правило обучения персептрона читается как [MP69]

```
PERCEPTRONLEARNING[M+,M  ] w =
произвольный вектор действительных чисел
Повторить
    Для всех x   M+
        Если wx   0, то w = w + x
    Для всех x   M
        Если wx > 0, то w = w   x
Пока все x   M+   M   не будут правильно классифицированы
```

Перцептрон должен выводить значение 1 для всех x M+. По определению 8.3 на стр. 170 это верно, когда wx > 0. Если это не так, то x добавляется к вектору весов w, в результате чего вектор весов изменяется точно в правильном направлении.

Мы видим это, когда применяем персептрон к измененному вектору w + x, потому что

$$2\ (w + x) \cdot x = wx + x \qquad \cdot$$

Если этот шаг повторять достаточно часто, то в какой-то момент значение wx станет положительным, как и должно быть. Аналогично, мы видим, что для отрицательных обучающих данных

Machine Translated by Google

персептрон вычисляет все меньшее значение

$$(w \quad x) \cdot x = wx \quad x \quad ^2$$

который в какой-то момент становится отрицательным.2

Пример 8.2 Персептрон должен быть обучен на множествах M+ = {(0, 1,8), (2, 0,6)} и M = {(1,2, 1,4), (0,4, 1)}. w = (1, 1) использовался в качестве начального вектора весов. Данные обучения и линия, определяемая весовым вектором wx = x1 + x2 = 0, показаны на рис. 8.10 на стр. 173 на первом рисунке в верхнем ряду. Кроме того, вектор весов нарисован пунктирной линией. Поскольку wx = 0, это ортогонально прямой.

На первой итерации цикла алгоритма обучения единственное ложное пример классифицированного обучения (-1,2, 1,4), потому что

$$(\quad 1,2, 1,4) \cdot \quad _{1\,1} \quad = 0,2 > 0.$$

В результате w = (1, 1)(1,2, 1,4) = (2,2, 0,4), как показано на втором изображении в верхнем ряду на рис. 8.10 на странице 173. Другие изображения показывают , как после всего пять изменений, разделительная линия проходит между двумя классами. Таким образом, персептрон правильно классифицирует все данные. Мы ясно видим в примере, что каждая неправильно классифицированная точка данных из M+ «тянет» весовой вектор w в своем направлении, а каждая неправильно классифицированная точка из M «толкает» весовой вектор в противоположном направлении.

Было показано [MP69], что персептрон всегда сходится для линейного разделения. пахотные данные. У нас есть

Теорема 8.1. Пусть классы M+ и M линейно отделимы гиперплоскостью wx = 0. Тогда PERCEPTRONLEARNING сходится для каждой инициализации вектора w. Перцептрон P с рассчитанным таким образом весовым вектором делит классы M+ и M , то есть:

$$P(x) = 1 \quad x \quad M+$$

и

$$P(x) = 0 \quad x \quad M \quad .$$

Как мы можем ясно видеть в примере 8.2, персептроны, как определено выше, не могут делить произвольные линейно разделимые множества, а только те, которые делятся прямой, проходящей через начало координат, или в R^n — гиперплоскостью, проходящей через начало координат, потому что постоянный член θ равен отсутствует в уравнении $\sum_{i=1}^{n} w_i x_i = 0$.

2Осторожно! Это не доказательство сходимости правила обучения персептрона. Это показывает только то, что персептрон сходится, когда набор обучающих данных состоит из одного примера.

Machine Translated by Google

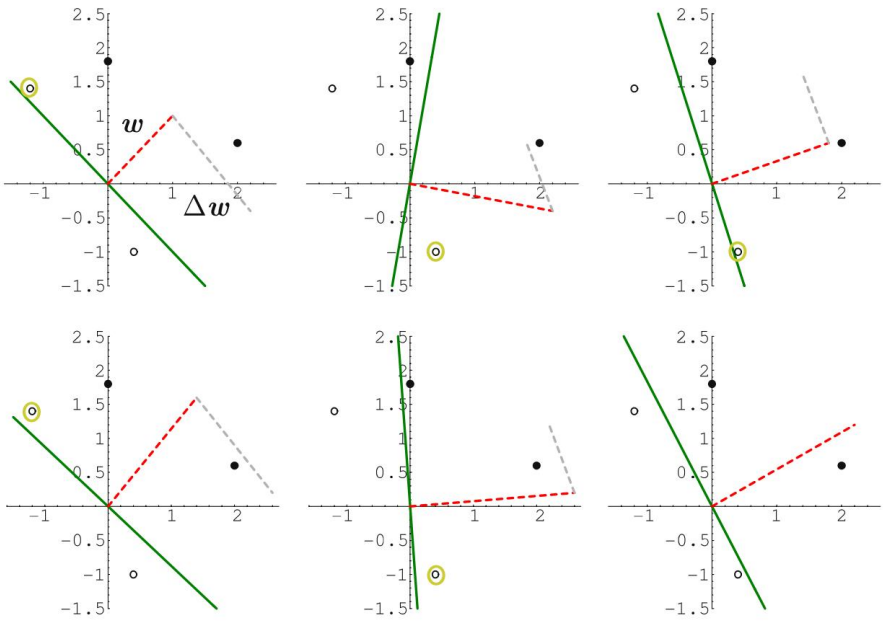


Рис. 8.10. Применение правила обучения персептрона к двум положительным (•) и двум отрицательным ( ) точкам данных. Сплошная линия показывает текущую разделительную линию wx = 0. Ортогональная пунктирная линия представляет собой весовой вектор w, а вторая пунктирная линия — добавляемый вектор изменения w = x или w = x, который вычисляется из текущей активной точки данных. окруженный серым цветом

С помощью следующего трюка мы можем сгенерировать постоянный член. Мы сохраняем последнюю компоненту xn входного вектора x постоянной и присваиваем ей значение 1. Теперь вес wn = θ работает как порог, потому что

$$\sum_{i=1}^{n} w_i x_i = \sum_{i=1}^{n-1} w_i x_i - \theta > 0 \quad \sum_{i=1}^{n-1} w_i x_i > \theta.$$

Такое постоянное значение xn = 1 на входе называется единицей смещения. Поскольку связанный вес вызывает постоянное смещение гиперплоскости, термин «смещение» подходит хорошо.

При применении алгоритма обучения персептрона к вектору обучающих данных добавляется бит с постоянным значением 1. Заметим, что вес wn, или порог θ , усваивается в процессе обучения.

Теперь было показано, что персептрон $P\theta : R^{n-1} \to \{0, 1\}$

$$P\theta(x_1,\ldots,x_{n-1}) = \begin{cases} 1 \text{ если } \sum_{i=1}^{n-1} w_i x_i > \theta, \\ 0 \text{ еще} \end{cases} \tag{8.1}$$

с произвольным порогом можно смоделировать с помощью персептрона $P : R^{n-1} \to \{0, 1\}$ с порог 0. Если сравнить (8.1) с определением линейно разделимого, то мы увидим, что оба утверждения эквивалентны. Подводя итог, мы показали, что:

Machine Translated by Google

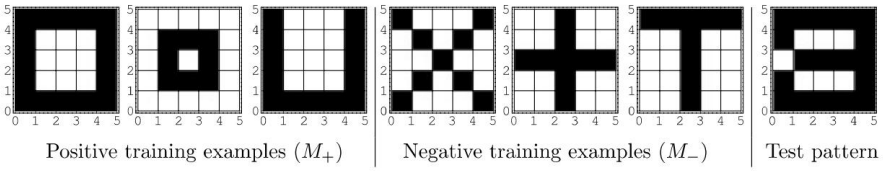
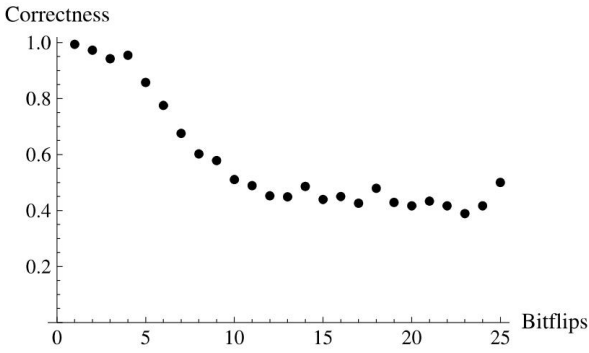


Рис. 8.11. Шесть паттернов, используемых для обучения. Весь правый шаблон является одним из 22 тестовых шаблонов для первого шаблона с последовательностью из четырех инвертированных битов.

Рис. 8.12 Относительная правильность персептрона в зависимости от количества инвертированных битов в тестовых данных



Теорема 8.2 Функция f : R {0, 1} может быть представлена персептроном тогда и только тогда, когда два множества положительных и отрицательных входных векторов линейно разделимы.

Пример 8.3 Теперь мы обучаем персептрон с порогом на шести простых графических двоичных шаблонах, представленных на рис. 8.11, размером 5 × 5 пикселей каждый.

Данные обучения могут быть изучены с помощью PERCEPTRONLEARNING за четыре итерации по всем шаблонам. Образцы с переменным числом инвертированных битов, представленных как шум, используются для проверки способности системы к обобщению. Инвертированные биты в тестовом шаблоне в каждом случае идут последовательно один за другим. На рис. 8.12 процент правильно классифицированных образов представлен как функция количества ложных битов.

Примерно после пяти последовательных инвертированных битов корректность резко падает, что неудивительно, учитывая простоту модели. В следующем разделе мы представим алгоритм, который работает намного лучше в этом случае.

8.2.2 Оптимизация и перспективы

Как один из простейших алгоритмов обучения на основе нейронных сетей, двухслойный персептрон может разделять только линейно разделимые классы. В разд. 9.5 мы увидим, что многоуровневые сети значительно мощнее. Несмотря на простоту конструкции,

Machine Translated by Google

персептрон в представленном виде сходится очень медленно. Его можно ускорить путем нормализации вектора изменения веса. Формулы w = w ± x заменяются на w = w ± x/|x|. Таким образом, каждая точка данных имеет одинаковый вес во время обучения, независимо от ее значения.

Скорость сходимости сильно зависит от инициализации вектора w.
В идеале его вообще не нужно было бы менять, и алгоритм сходился бы после одной итерации. Мы можем приблизиться к этой цели, используя эвристическую инициализацию

$$w0 = \frac{x -}{x \quad M+} \quad \frac{\text{Икс,}}{x \quad M-}$$

которые мы рассмотрим более подробно в упражнении 8.5 на стр. 217.

Если мы сравним формулу персептрона с методом оценки, представленным в разд. 7.3.1, мы сразу видим их эквивалентность. Кроме того, персептрон, как простейшая модель нейронной сети, эквивалентен наивной байесовской сети, простейшему типу байесовской сети (см. упражнение 8.16 на стр. 219). Таким образом, очевидно, что несколько весьма различных алгоритмов классификации имеют общее происхождение.

В гл. 9 мы познакомимся с обобщением персептрона в виде алгоритма обратного распространения, который может разделять нелинейно разделимые множества за счет использования нескольких слоев и обладает лучшим правилом обучения.

8.3 Метод ближайших соседей

Для персептрона знания, имеющиеся в обучающих данных, извлекаются и сохраняются в сжатом виде в весах wi . Тем самым информация о данных теряется. Однако это именно то, что требуется, если система должна обобщать обучающие данные на новые данные. Обобщение в данном случае является трудоемким процессом, целью которого является нахождение компактного представления данных в виде функции, которая максимально хорошо классифицирует новые данные.

Запоминание всех данных простым их сохранением — это совсем другое. Здесь обучение чрезвычайно простое. Однако, как было сказано ранее, сохраненные знания не так легко применимы к новым, неизвестным примерам. Такой подход совершенно не годится, например, для обучения катанию на лыжах. Новичок никогда не станет хорошим лыжником, просто просматривая видео с хорошими лыжниками. Очевидно, что при автоматическом выполнении обучающих движений такого типа происходит нечто подобное тому, что происходит в случае перцептрона. После достаточно длительной практики знания, хранящиеся в обучающих примерах, трансформируются во внутреннее представление в мозгу.

Однако есть удачные примеры запоминания, при которых возможно и обобщение. При диагностике сложного случая врач может попытаться вспомнить подобные случаи из прошлого. Если у него здоровая память, то он может наткнуться на это дело, поискать его в своих файлах и, наконец, прийти к похожему диагнозу. При таком подходе врач должен хорошо чувствовать сходство, чтобы помнить наиболее похожий случай. Если он нашел это, то он должен спросить себя, достаточно ли это похоже, чтобы оправдать такой же диагноз.

Machine Translated by Google

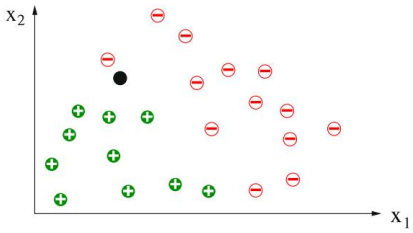


Рис. 8.13. В этом примере с отрицательными и положительными обучающими примерами метод ближайшего соседа группирует новую точку, отмеченную черным, в отрицательный класс

Что означает сходство в формальном контексте, который мы конструируем? Мы представляем обучающие выборки, как обычно, в многомерном пространстве признаков и определяем: Чем меньше их расстояние в пространстве признаков, тем более похожи два примера.

Применим теперь это определение к простому двумерному примеру на рис. 8.13. Здесь ближайший сосед черной точки является отрицательным примером. Таким образом, он отнесен к отрицательному классу.

Расстояние d(x, y) между двумя точками x R , н и y R н может быть например измеренное евклидовым расстоянием

$$d(x, y) = |x \quad y| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

Поскольку помимо этой метрики расстояния есть много других, имеет смысл подумать об альтернативах для конкретного приложения. Во многих приложениях одни функции важнее других. Поэтому часто имеет смысл по-разному масштабировать признаки с помощью весов wi . Затем формула читается

$$dw(x, y) = |x \quad y| = \sqrt{\sum_{i=1}^{n} w_i (x_i - y_i)^2}.$$

Следующая простая программа классификации ближайших соседей ищет в обучающих данных ближайшего соседа t для нового примера s, а затем классифицирует s точно так же , как t.

```
БЛИЖАЙШИЙ СОСЕД[M+,M ,s] t =
    argminx M+ M {d(s, x)}
    Если t M+ , то возврат («+»)
                Иначе Возврат(„–”)
```

3 Функционалы argmin и argmax определяют, аналогично min и max, минимум или максимум множества или функции. Однако вместо того, чтобы возвращать значение максимума или минимума, они выдают позицию, то есть аргумент, в котором появляется экстремум.

Machine Translated by Google

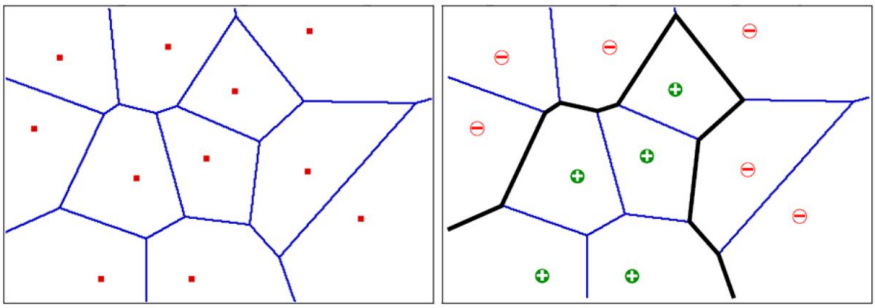


Рис. 8.14. Набор точек вместе с их диаграммой Вороного (слева) и разделительной линией, сгенерированной для двух классов M+ и M

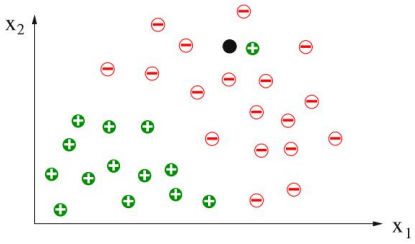


Рис. 8.15. Метод ближайшего соседа относит новую точку, отмеченную черным цветом , к неправильному (положительному) классу, поскольку ближайший сосед, скорее всего, классифицирован неправильно

В отличие от персептрона, метод ближайшего соседа не генерирует линию, разделяющую точки обучающих данных. Однако воображаемая линия, разделяющая два класса, определенно существует. Мы можем сгенерировать это, сначала сгенерировав так называемую диаграмму Вороного. На диаграмме Вороного каждая точка данных окружена выпуклым многоугольником, который, таким образом, определяет окрестности вокруг нее. Диаграмма Вороного обладает тем свойством, что для произвольной новой точки ближайшим соседом среди точек данных является точка данных, которая находится в той же окрестности. Если диаграмма Вороного для набора обучающих данных определена, то несложно найти ближайшего соседа для новой точки, подлежащей классификации. Затем членство в классе будет взято у ближайшего соседа.

На рис. 8.14 мы ясно видим, что метод ближайшего соседа значительно мощнее, чем персептрон. Он способен корректно реализовать сколь угодно сложные разделительные линии (в общем случае: гиперплоскости). Однако здесь есть опасность. Одна ошибочная точка может при определенных обстоятельствах привести к очень плохим результатам классификации. Такой случай имеет место на рис. 8.15 при классификации черной точки. Метод ближайшего соседа может классифицировать это неправильно. Если черная точка находится непосредственно рядом с положительной точкой, которая является выбросом из положительного класса, то она будет классифицироваться как положительная, а не отрицательная, как здесь предполагалось. Ошибочная подгонка к случайным ошибкам (шум) называется переподгонкой.

Machine Translated by Google


```
K-БЛИЖАЙШИЙ СОСЕД(M+,M   ,s)
    V = {k ближайших соседей в M+     M   }
    Если          |M+    V | > |M       V | Затем Возврат ("+")
    ИначеЕсли |M+    V | < |M       V | Затем Возврат(„-”)
    Еще                                    Возврат(Случайный("+", "-"))
```

Рис. 8.16. АЛГОРИТМ K-БЛИЖАЙШИХ СОСЕДЕЙ

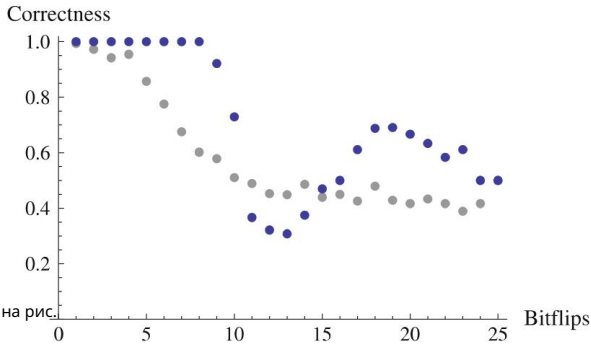


Рис. 8.17. Относительная правильность классификации ближайших соседей в зависимости от количества инвертированных битов. Структура кривой с минимумом в 13 и максимумом в 19 связана с особой структурой обучающих данных. Для сравнения значения персептрона из Примера 8.3 на стр. 174 показаны на рис. серый

Для предотвращения ложных классификаций из-за одиночных выбросов рекомендуется несколько сгладить поверхность деления. Этого можно добиться, например, с помощью алгоритма K-NEARESTNEIGHBOR на рис. 8.16, который принимает решение о большинстве среди k ближайших соседей.

Пример 8.4 Теперь применим NEARESTNEIGHBOR к примеру 8.3 на стр. 174. Поскольку мы имеем дело с двоичными данными, мы используем расстояние Хэмминга в качестве метрики расстояния4 . В качестве тестового примера мы снова используем модифицированные обучающие примеры с n последовательными инвертированными битами в каждом. На рис. 8.17 процент правильно классифицированных тестовых примеров показан как функция числа инвертированных битов b. До восьми инвертированных битов все шаблоны идентифицируются правильно. После этого количество ошибок быстро увеличивается. Это неудивительно, потому что тренировочный образец номер 2 на рис. 8.11 на странице 174 из класса M+ имеет расстояние Хэмминга 9 до двух тренировочных примеров, номера 4 и 5 из другого класса. Это означает, что тестовый образец, скорее всего, близок к образцам другого класса. Совершенно ясно мы видим, что классификация ближайших соседей превосходит персептрон в этом приложении до восьми ложных битов.

4Расстояние Хэмминга между двумя битовыми векторами — это количество различных битов двух векторов.

Machine Translated by Google

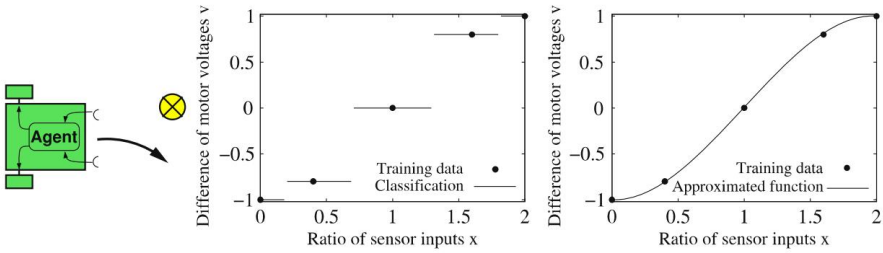


Рис. 8.18. Обучающийся агент, который должен избегать света (слева), представленный в виде классификатора (в центре) и в виде аппроксимации (справа)

8.3.1 Два класса, много классов, приближение

Классификация ближайших соседей также может применяться к более чем двум классам. Как и в случае с двумя классами, класс классифицируемого вектора признаков просто устанавливается как класс ближайшего соседа. Для метода k ближайших соседей класс должен быть определен как класс с наибольшим количеством членов среди k ближайших соседей.

Если количество классов велико, то, как правило, уже не имеет смысла использовать алгоритмы классификации, так как размер необходимых обучающих данных быстро растет с количеством классов. Кроме того, при определенных обстоятельствах важная информация теряется при классификации многих классов. Это станет ясно на следующем примере.

Пример 8.5 Предполагается, что автономный робот с простыми датчиками, подобными транспортным средствам Брайтенберга, представленным на рис. 1.1 на странице 2, учится уходить от света. Это означает, что он должен как можно оптимальнее научиться преобразовывать значения своих датчиков в сигнал рулевого управления, который управляет направлением движения. Робот оснащен двумя простыми датчиками света на передней стороне. Из сигналов двух датчиков (с sl для левого и sr для правого датчика) вычисляется соотношение x = sr/sl .

Чтобы управлять электродвигателями двух колес от этого значения x, разности v = Ur -Ul двух напряжений Ur и Ul левого и правого двигателей соответственно.
Теперь задача обучающегося агента состоит в том, чтобы избежать светового сигнала. Следовательно, он должен изучить 5- отображение f, которое вычисляет «правильное» значение v = f (x).

Для этого проведем эксперимент, в котором для нескольких измеренных значений x найдем как можно более оптимальное значение v. Эти значения нанесены в виде точек данных на рис. 8.18 и должны служить обучающими данными для агента обучения. При классификации ближайших соседей каждая точка в пространстве признаков (то есть на оси x) классифицируется точно так же, как ее ближайший сосед среди обучающих данных. Тогда функция управления двигателями представляет собой ступенчатую функцию с большими скачками (рис. 8.18, середина). Если нам нужны более тонкие шаги, мы должны предоставить соответственно больше обучающих данных. На

5 Чтобы сделать пример простым и читабельным, вектор признаков x был намеренно оставлен размерный.

Machine Translated by Google

с другой стороны, мы можем получить непрерывную функцию, если аппроксимируем гладкую функцию по пяти точкам (рис. 8.18 на странице 179 справа). Требование непрерывности функции f приводит к очень хорошим результатам даже без дополнительных точек данных

Для аппроксимации функций в точках данных существует множество математических методов, таких как полиномиальная интерполяция, сплайн-интерполяция или метод наименьших квадратов. Применение этих методов становится проблематичным в более высоких измерениях. Особая трудность в ИИ заключается в том, что необходимы безмодельные методы аппроксимации. То есть хорошая аппроксимация данных должна быть сделана без знания особых свойств данных или приложения. Очень хорошие результаты были достигнуты здесь с помощью нейронных сетей и других аппроксиматоров нелинейных функций, которые представлены в гл. 9.

Метод k ближайших соседей можно просто применить к задаче аппроксимации. В алгоритме K-NEARESTNEIGHBOR после определения множества V = {x1, x2,..., xk } среднее значение функции k ближайших соседей

$$f(x) = \frac{1}{k} \sum_{i=1}^{k} \phi(\text{кси}) \tag{8.2}$$

вычисляется и принимается как аппроксимация f^ для вектора запроса x. Чем больше становится k, тем более гладкой становится функция f^.

8.3.2 Расстояние имеет значение

При практическом применении как дискретных, так и непрерывных вариантов метода k ближайших соседей часто возникают проблемы. Когда k становится большим, обычно существует больше соседей с большим расстоянием, чем соседей с малым расстоянием. Таким образом, при вычислении f^ преобладают соседи, находящиеся далеко. Чтобы предотвратить это, k соседей взвешиваются таким образом, чтобы более удаленные соседи оказывали меньшее влияние на результат. При мажоритарном решении в алгоритме K-БЛИЖАЙШИЙ СОСЕД «голоса» взвешиваются с весом

$$\text{ви} = \frac{1}{1 + \alpha d(x, xi)^2}, \tag{8.3}$$

которая убывает пропорционально квадрату расстояния. Константа α определяет скорость убывания весов. Уравнение (8.2) теперь заменяется на

$$f(x) = \frac{\text{ки=1}\text{ж}\phi(\text{xi})}{\text{ки=1 ш}}.$$

Для равномерно распределенной концентрации точек в пространстве признаков это гарантирует, что влияние точек асимптотически стремится к нулю по мере увеличения расстояния. Таким образом, становится возможным использовать многие или даже все обучающие данные для классификации или аппроксимации заданного входного вектора.

Machine Translated by Google

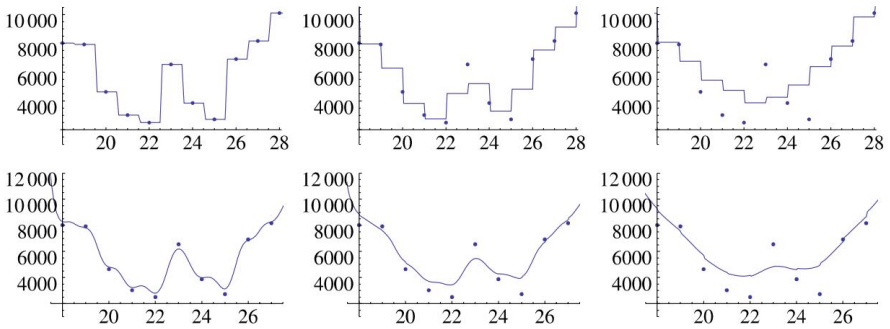


Рис. 8.19. Сравнение метода k-ближайших соседей (верхний ряд) с k = 1 (слева), k = 2 (в центре) и k = 6 (справа) с его взвешенным по расстоянию вариантом (нижний ряд) с α = 20 (слева), α = 4 (в центре) и α = 1 (справа) в одномерном наборе данных

Чтобы получить представление об этих методах, на рис. 8.19 метод k-ближайших соседей (в верхнем ряду) сравнивается с его взвешенной по расстоянию оптимизацией. Благодаря усреднению оба метода могут обобщать или, другими словами, компенсировать шумы, если число соседей для k-ближайшего соседа или параметр α заданы соответствующим образом. Диаграммы хорошо показывают, что метод взвешенного расстояния дает гораздо более гладкую аппроксимацию, чем метод k-ближайшего соседа. По качеству аппроксимации этот очень простой метод может хорошо конкурировать со сложными алгоритмами аппроксимации, такими как нелинейные нейронные сети, машины опорных векторов и гауссовы процессы.

Существует множество альтернатив весовой функции (также называемой ядром), приведенной в (8.3) на стр. 180. Например, можно использовать функцию Гаусса или аналогичную колоколообразную функцию. Для большинства приложений результаты не очень разумны при выборе ядра. Однако параметр ширины α, который для всех этих функций необходимо задавать вручную, оказывает большое влияние на результаты, как показано на рис. 8.19. Чтобы избежать такой неудобной ручной адаптации, были разработаны методы оптимизации для автоматической установки этого параметра [SA94, SE10].

8.3.3 Время вычислений

Как упоминалось ранее, обучение выполняется во всех вариантах метода ближайшего соседа путем простого сохранения всех обучающих векторов вместе с их метками (значениями класса) или значением функции f(x). Таким образом, нет другого алгоритма обучения, который учится так же быстро. Однако ответ на запрос о классификации или аппроксимации вектора x может стать очень дорогим. Простое нахождение k ближайших соседей для n обучающих данных требует затрат, которые растут линейно с n. Для классификации или аппроксимации есть дополнительная стоимость, линейная по k. Таким образом, общее время вычислений растет как Θ (n + k). Для больших объемов обучающих данных это может привести к проблемам.

Machine Translated by Google

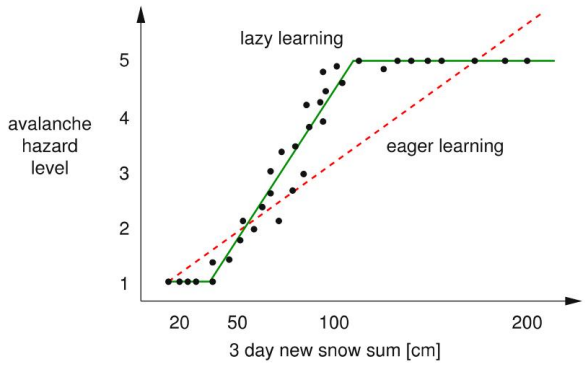


Рис. 8.20. Для определения лавинной опасности функция аппроксимируется по обучающим данным. Здесь для сравнения локальная модель (сплошная линия) и глобальная модель (штриховая линия)

8.3.4 Резюме и перспективы

Поскольку на этапе обучения представленных методов ближайшего соседа ничего не происходит, такие алгоритмы также обозначаются как ленивое обучение, в отличие от энергичного обучения, при котором этап обучения может быть дорогостоящим, но применение к новым примерам очень эффективно. Персептрон и все другие нейронные сети, обучение дереву решений и обучение байесовских сетей — это активные методы обучения. Поскольку ленивым методам обучения требуется доступ к памяти со всеми обучающими данными для аппроксимации нового ввода, их также называют обучением на основе памяти.

Для сравнения этих двух классов процессов обучения возьмем в качестве примера задачу определения текущей лавинной опасности по количеству свежевыпавшего снега в определенном районе Швейцарии6. На рис. 8.20 внесены определенные экспертами значения , которые мы хотим использовать в качестве обучающих данных. Во время применения алгоритма энергичного обучения, который выполняет линейную аппроксимацию данных, вычисляется пунктирная линия, показанная на рисунке. Из-за ограничения на прямую линию ошибка относительно велика и составляет максимум около 1,5 уровня опасности. При ленивом обучении ничего не вычисляется до поступления запроса о текущем уровне опасно Затем ответ вычисляется из нескольких ближайших соседей, то есть локально. Это может привести к кривой, показанной на рисунке, которая составлена из отрезков прямой и показывает гораздо меньшие ошибки. Преимуществом ленивого метода является его локальность. Приближение берется локально из текущего нового уровня снега, а не глобально. Таким образом, для принципиально равных классов функций (например, линейных функций) ленивые алгоритмы лучше.

Методы ближайших соседей хорошо подходят для всех проблемных ситуаций, в которых требуется хорошая локальная аппроксимация, но не предъявляются высокие требования к быстродействию системы. Упомянутый здесь предсказатель лавин, который запускается один раз в день, является таким приложением. Методы ближайшего соседа не подходят, когда описание знаний, извлеченных из данных, должно быть понятно человеку, что сегодня имеет место во многих приложениях интеллектуального анализа данных (см. раздел 8.4) . В

6 Общее количество снегопадов за три дня на самом деле является важной характеристикой для определения уровня опасности. Однако на практике используются дополнительные атрибуты [Bra01]. Используемый здесь пример упрощен.

Machine Translated by Google

Особенность	Запрос	Корпус из корпуса
Дефектная часть:	Задние фары	Передний свет
Модель велосипеда:	Марин Пайн Маунтин	ВСФ Т400
Год:	1993 г.	2001 г.
Источник питания:	Аккумулятор в	Динамо
Состояние лампы:	порядке	хорошо
Состояние светового кабеля:	?	хорошо
	Решение	
Диагноз:	?	Отсутствует передний электрический контакт
Ремонт:	?	Установить передний электрический контакт

Рис. 8.21 Пример простой диагностики для запроса и соответствующего случая из базы случаев

В последние годы эти методы обучения на основе памяти становятся популярными, и различные улучшенные варианты (например, локально взвешенная линейная регрессия) применил [Cle79].

Чтобы иметь возможность использовать описанные методы, должны быть доступны обучающие данные. в виде векторов целых или действительных чисел. Таким образом, они не подходят для приложения, в которых данные представлены символически, например, как первый логические формулы порядка. Сейчас мы кратко обсудим это.

8.3.5 Рассуждение на основе прецедентов

В рассуждении на основе прецедентов (CBR) метод ближайшего соседа распространяется на символические описания проблем и их решения. CBR используется для диагностики технических проблем в обслуживании клиентов или для телефонных горячих линий. Показанный пример на рис. 8.21 о диагностике выключения велосипедного фонаря иллюстрирует этот тип ситуация.

Ищется решение по запросу клиента с неисправным задним велосипедом свет. В правом столбце приведен случай, аналогичный запросу в среднем столбце. Это вытекает из базы прецедентов, которая соответствует обучающим данным в ближайшем метод соседа. Если мы просто возьмем наиболее похожий случай, как мы это делаем в ближайшем метод соседа, то мы закончим тем, что попытаемся отремонтировать передний фонарь, когда задний фонарь разбит. Таким образом, нам необходимо обратное преобразование решения обнаруженной аналогичной задачи обратно в запрос. Самые важные шаги в решении к случаю CBR выполняются на рис. 8.22 на стр. 184. Преобразование в этом пример прост: задний свет отображается на передний свет.

Какими бы красивыми и простыми ни казались эти методы в теории, на практике построение систем диагностики CBR является очень сложной задачей. Три главные трудности являются:

Моделирование Предметы приложения должны быть смоделированы в формальном контексте. Здесь логическое однообразие, известное нам из гл. 4, представляет трудности. Может разработчик предсказывает и отображает все возможные частные случаи и варианты проблемы?

Machine Translated by Google

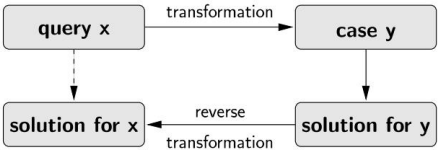


Рис. 8.22. Если для случая x найден аналогичный случай y, то для получения решения для x необходимо определить преобразование и применить его обратное к найденному случаю y

Сходство Поиск подходящей метрики сходства для символических, нечисловых признаков тур.

Преобразование Даже если подобный случай будет найден, пока неясно, как должно выглядеть отображение преобразования и обратное ему.

Действительно, сегодня существуют практические системы CBR для диагностических приложений. Однако по указанным причинам они остаются далеко позади людей-экспертов по производительности и гибкости. Интересной альтернативой CBR являются байесовские сети, представленные в гл. 7. Часто символическое представление задачи также может быть достаточно хорошо отображено в дискретные или непрерывные числовые признаки. Тогда можно успешно использовать упомянутые индуктивные методы обучения, такие как деревья решений или нейронные сети.

8.4 Обучение дереву решений

Обучение дереву решений — чрезвычайно важный алгоритм для ИИ, потому что он очень мощный, но также простой и эффективный для извлечения знаний из данных.
По сравнению с двумя уже представленными алгоритмами обучения он имеет важное преимущество. Извлеченные знания не только доступны и могут использоваться как функция черного ящика, но и могут быть легко поняты, интерпретированы и проконтролированы людьми в форме удобочитаемого дерева решений. Это также делает изучение дерева решений важным инструментом для интеллектуального анализа данных.

Мы обсудим функцию и применение обучения дерева решений с использованием алгоритма C4.5 . C4.5 был представлен в 1993 году австралийцем Россом Куинланом и является усовершенствованием своего предшественника ID3 (Iterative Dichotomiser 3, 1986). Он находится в свободном доступе для некоммерческого использования [Qui93]. Дальнейшее развитие, которое работает еще эффективнее и может учитывать затраты на решения, — это C5.0 [Qui93].

Система CART (Деревья классификации и регрессии, 1984 г.), разработанная Лео Брейманом [BFOS84], работает аналогично C4.5. Имеет удобный графический интерфейс, но стоит очень дорого.

Двадцатью годами ранее, в 1964 г., Дж. Сонквист и Дж. Морган представили систему CHAID (Chi-square Automatic Interaction Detectors), которая может автоматически генерировать деревья решений. У него есть примечательная особенность, заключающаяся в том, что он останавливает рост дерева до того, как оно станет слишком большим, но сегодня это уже не актуально.

Machine Translated by Google

Таблица 8.4 Переменные для задачи классификации лыж

Переменная	Ценить	Описание
Лыжи (целевая переменная)	Да нет	Должен ли я ехать на ближайший горнолыжный курорт при достаточном количестве снега?
Солнце (особенность)	Да нет	Сегодня есть солнце?
Snow_Dist (функция)	100, >100	Расстояние до ближайшего горнолыжного курорта с хорошие снежные условия (больше/меньше 100 км)
Выходные (особенность)	Да нет	Сегодня выходные?

Также интересен инструмент интеллектуального анализа данных KNIME (Konstanz Information Miner), который имеет дружественный пользовательский интерфейс и, используя библиотеку WEKA Java, также делает Возможна индукция деревьев решений. В разд. 8.8 мы представим KNIME.

Теперь мы сначала покажем на простом примере, как можно построить дерево решений. из обучающих данных, чтобы затем проанализировать алгоритм и применить его к более комплексный пример LEXMED для медицинской диагностики.

8.4.1 Простой пример

Преданный лыжник, живущий рядом с высокой сьеррой, красивым горным хребтом в Калифорнии, хочет, чтобы дерево решений помогло ему решить, стоит ли ехать. свою машину на горнолыжный курорт в горах. Таким образом, у нас есть проблемная лыжа с двумя классами да/нет на основе переменных, перечисленных в таблице 8.4.

На рис. 8.23 на стр. 186 показано дерево решений для этой проблемы. Дерево решений представляет собой дерево, внутренние узлы которого представляют признаки (атрибуты). Каждое ребро обозначает значение атрибута. В каждом листовом узле задается значение класса.

Данные, использованные для построения дерева решений, приведены в таблице 8.5 на стр. 186. Каждая строка в таблице содержит данные за один день и, таким образом, представляет образец. При ближайшем рассмотрении мы видим, что ряд 6 и ряд 7 противоречат друг другу. Таким образом, ни один алгоритм детерминированной классификации не может правильно классифицировать все данные. Таким образом, количество ложно классифицированных данных должно быть 1. Дерево на рис. 8.23 на стр. 186, таким образом, оптимально классифицирует данные.

Как создается такое дерево из данных? Чтобы ответить на этот вопрос, мы будем в сначала ограничимся дискретными атрибутами с конечным числом значений. Поскольку количество атрибутов также конечно, и каждый атрибут может встречаться не более одного раза на пути, существует конечное число различных деревьев решений. Простой и понятный алгоритм построение дерева просто генерировало бы все деревья, затем для каждого дерева вычислялось бы количество ошибочных классификаций данных, а в конце выбрать дерево с минимальным количеством ошибок. Таким образом, у нас был бы даже оптимальный алгоритм (в смысле ошибок для обучающих данных) для обучения дерева решений.

Очевидным недостатком этого алгоритма является неприемлемо большое время вычислений, так как количество признаков становится несколько больше. Мы будем теперь разработайте эвристический алгоритм, который, начиная с корня, рекурсивно строит

Machine Translated by Google

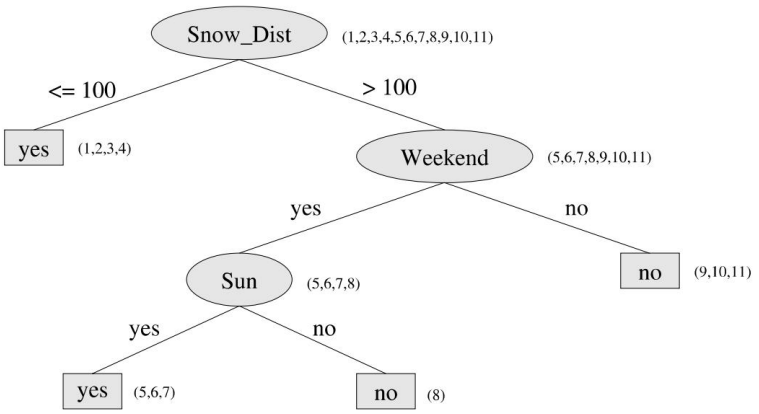


Рис. 8.23. Дерево решений для задачи классификации лыж. В списках справа от узлов даны номера соответствующих обучающих данных. Обратите внимание, что для листовых узлов солнечно = да только два из трех примеров классифицированы правильно

Таблица 8.5 Набор данных для проблема классификации лыж

День	Snow_Dist	Выходные	Солнце	Катание на лыжах
1	100	Да	Да	Да
2	100	Да	Да	Да
3	100	Да	Нет	да
4	100	Нет	Да	Да
5	>100	Да	Да	Да
6	>100	Да	Да	Да
7	>100	Да	да	нет
8	>100	Да	Нет	нет
9	>100	Нет	да	нет
10	>100	Нет	да	нет
11	>100	Нет	Нет	нет

дерево решений. Сначала атрибут с наибольшим информационным приростом (Snow_Dist) . выбирается для корневого узла из набора всех атрибутов. Для каждого значения атрибута (100, >100) в дереве есть ветвь. Теперь для каждой ветки этот процесс повторяется рекурсивно. При генерации узлов всегда выбирается атрибут с наибольшим информационным приростом среди еще не использованных атрибутов, т.е. дух жадной стратегии.

8.4.2 Энтропия как метрика информационного содержания

Описанный нисходящий алгоритм построения дерева решений, на каждом шаг выбирает атрибут с наибольшим приростом информации. Теперь мы представляем

Machine Translated by Google

энтропия как метрика информационного содержания набора тренировочных данных D. Если мы посмотрим только на бинарную переменную лыжи в приведенном выше примере, то D можно описать как

$$D = (\text{да}, \text{да}, \text{да}, \text{да}, \text{да}, \text{да}, \text{нет}, \text{нет}, \text{нет}, \text{нет}, \text{нет})$$

с предполагаемыми вероятностями

$$p1 = P(\text{да}) = 6/11 \text{ и } p2 = P(\text{нет}) = 5/11.$$

Здесь мы, очевидно, имеем распределение вероятностей p = (6/11, 5/11). В общем, для задачи n-класса это читается

$$p = (p1,\ldots,pn)$$

с

$$\sum_{i=1}^{n} pi = 1.$$

Чтобы представить информативность распределения, мы наблюдаем два крайних случая. Сначала пусть

$$p = (1, 0, 0, \ldots, 0). \tag{8.4}$$

То есть первое из n событий обязательно произойдет, а все остальные — нет. Таким образом, неопределенность относительно исхода событий минимальна. Напротив, для равномерного распределения

$$p = \frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n} \tag{8.5}$$

неопределенность максимальна, потому что ни одно событие нельзя отличить от других. Здесь Клод Шеннон задался вопросом, сколько битов потребуется для кодирования такого события. В определенном случае (8.4) нулевые биты необходимы, потому что мы знаем, что случай i = 1 всегда имеет место. В случае равномерного распределения (8.5) имеется n равновероятных возможностей. Для двоичного кодирования здесь требуется log2 n бит. Поскольку все отдельные вероятности равны pi = 1/n, для этого кодирования необходимы логарифмические $\frac{1}{1\ pi}$ 2 бита.

В общем случае p = (p1,...,pn), если вероятности элементарных событий отклоняются от равномерного распределения, то вычисляется математическое ожидание H для количества битов. С этой целью мы взвесим все значения $\frac{1}{1\ pi}$ log2 = log2 pi с их вероятностями и получим

$$H = \sum_{i=1}^{n} pi(-\log_2 pi) = \sum_{i=1}^{n} pi \log_2 pi .$$

Чем больше битов нам нужно для кодирования события, тем выше неопределенность результата. Поэтому мы определяем:

Machine Translated by Google

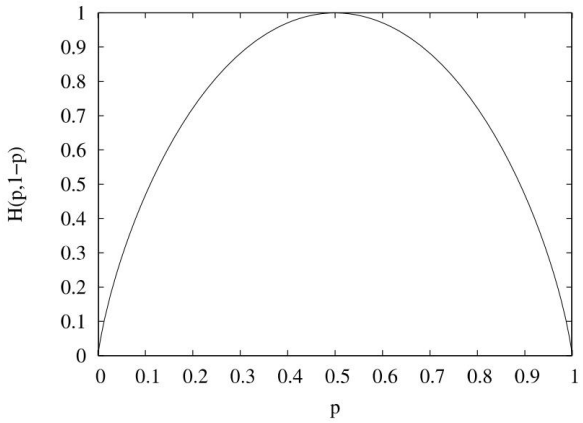


Рис. 8.24. Функция энтропии для случая двух классов. Мы видим максимум при p = 1/2 и симметрию относительно перестановки p и 1 p

Определение 8.4. Энтропия H как метрика неопределенности распределения вероятностей определяется формулой[7]

$$H(p) = H(p_1,\ldots,p_n) := \sum_{i=1}^{n} p_i \log_2 p_i .$$

Подробный вывод этой формулы можно найти в [SW76]. Если подставить некое событие p = (1, 0, 0,..., 0), то 0 log2 0, получится неопределенное выражение. Мы решаем эту задачу с помощью определения 0 log2 0 := 0 (см. упражнение 8.9 на стр. 218).

Теперь мы можем вычислить H (1, 0,..., 0) = 0. Покажем, что энтропия в гиперкубе [0, 1]^n при ограничении $\sum_{i=1}^{n} p_i = 1$ принимает максимальное значение $\log_2 n$ (равномерным распределением $\frac{1}{n},\ldots,\frac{1}{n}$). В случае события с двумя возможными исходы, соответствующие двум классам, результат

$$H(p) = H(p_1, p_2) = H(p_1, 1 - p_1) = (p_1 \log_2 p_1 + (1 - p_1)\log_2 (1 - p_1)).$$

Это выражение показано как функция p1 на рис. 8.24 с максимумом при p1 = 1/2.

Поскольку каждому классифицированному набору данных D присваивается распределение вероятностей p^D путем оценки вероятностей класса, мы можем расширить понятие энтропии на данные с помощью

[7] В (7.9) на стр. 124 для определения энтропии используется натуральный логарифм, а не log2 . Поскольку здесь, как и в случае с методом MaxEnt, только сравниваются энтропии, эта разница не играет роли. (см. упражнение 8.11 на стр. 218).

Machine Translated by Google

Определение

$$H(D) = H(p).$$

Теперь, поскольку информационное содержание I(D) набора данных D должно быть противоположным неопределенности. Таким образом, мы определяем:

Определение 8.5 Информационное содержание набора данных определяется как

$$I(D) := 1 \quad H(D). \tag{8.6}$$

8.4.3 Получение информации

Если мы применим формулу энтропии к примеру, результат будет

$$H(6/11, 5/11) = 0{,}994$$

При построении дерева решений набор данных дополнительно подразделяется по каждому новому атрибуту. Чем больше атрибут повышает информативность распределения за счет разделения данных, тем лучше этот атрибут. Соответственно определяем:

Определение 8.6 Прирост информации G(D, A) за счет использования атрибута A определяется разностьюсредней информативности набора данных D = D1 D2 ⋯ Dn, деленной на n-значение атрибута A и информативность I(D) неразделенного набора данных, что дает

$$G(D, A) = \sum_{i=1}^{n} \frac{|Di|}{|D|} + I(Di) \quad I(D).$$

С учетом (8.6) отсюда получаем

$$G(D, A) = \sum_{i=1}^{n} \frac{|Di|}{|D|} I(Di) - I(D) = \sum_{i=1}^{n} \frac{|Di|}{|D|} (1 - H(Di)) - (1 - H(D))$$

$$= 1 - \sum_{i=1}^{n} \frac{|Di|}{|D|} H(Di) - 1 + H(D)$$

$$= H(D) - \sum_{i=1}^{n} \frac{|Di|}{|D|} + H(Di). \tag{8.7}$$

Machine Translated by Google

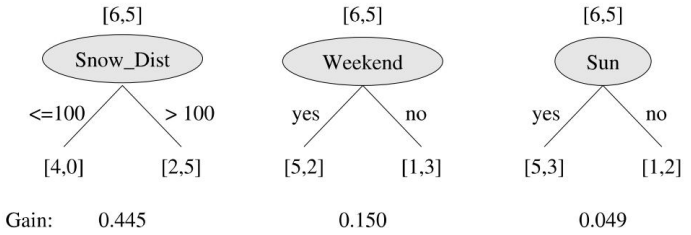


Рис. 8.25 Расчетный коэффициент усиления для различных атрибутов отражает, приводит ли разделение данных по соответствующему атрибуту к лучшему разделению классов. Чем больше распределения, генерируемые атрибутом, отклоняются от равномерного распределения, тем выше прирост информации.

Применительно к нашему примеру для атрибута Snow_Dist это дает

$$G(D, \text{Snow_Dist}) = H(D) - \frac{4}{11}H(D \le 100) + \frac{7}{11}H(D > 100)$$

$$= 0{,}994 - \frac{4}{11} \cdot 0 + \frac{7}{11} \cdot 0{,}863 = 0{,}445.$$

Аналогично получаем

$$G(\text{Дневные}) = 0{,}150$$

и

$$G(D, \text{Солнце}) = 0{,}049.$$

Атрибут Snow_Dist теперь становится корневым узлом дерева решений. Ситуация с выбором этого атрибута еще раз поясняется на рис. 8.25.

Два значения атрибута 100 и >100 генерируют два ребра в дереве, которые соответствуют подмножествам D 100 и D>100. Для подмножества D 100 классификация однозначна да. таким образом, дерево заканчивается здесь. В другой ветке D>100 однозначного результата нет. Таким образом, алгоритм повторяется рекурсивно. Из двух все еще доступных атрибутов, Sun и Weekend, необходимо выбрать лучший. Мы рассчитываем

$$G(D>100, \text{Дневные}) = 0{,}292$$

и

$$G(D>100, \text{Солнце}) = 0{,}170.$$

Таким образом, узлу назначается атрибут Weekend . Для Weekend = no дерево завершается решением Ski = no. Расчет усиления здесь возвращает значение 0. Для Weekend = yes, Sun дает прирост 0,171. Затем построение дерева прекращается, так как больше нет доступных атрибутов, хотя пример номер 7 классифицируется ложно. Готовое дерево уже знакомо по рис. 8.23 на стр. 186.

Machine Translated by Google

8.4.4 Применение C4.5

Дерево решений, которое мы только что сгенерировали, также может быть сгенерировано с помощью C4.5. Данные тренировки сохраняются в файле данных ski.data в следующем формате:

```
<=100, да, да, да <=100, да, да, да
<=100, да, нет, да <=100, нет, да, да
>100, да, да, да >100, да, да , да >100,
да, да, нет >100, да, нет, нет >100,
нет, да, нет >100, нет, да, нет >100,
нет,




                        нет нет
```

Информация об атрибутах и классах хранится в файле ski.names.
(строки, начинающиеся с «|», являются комментариями):

```
 |Классы: нет: не кататься на лыжах, да: кататься на лыжах |

нет да.
 |
 |Атрибуты |

Снег_Расстояние:          <=100,>100.
Выходные:                 нет да.
Солнце:                   нет да.
```

Затем C4.5 вызывается из командной строки Unix и генерирует дерево решений, показанное ниже, которое отформатировано с использованием отступов. Параметр -f предназначен для имени входного файла, а параметр -m указывает минимальное количество точек обучающих данных, необходимых для создания новой ветви в дереве. Поскольку количество точек обучающих данных в этом примере чрезвычайно мало, здесь уместна опция -m 1. Для больших наборов данных следует использовать значение не менее -m 10.

Machine Translated by Google


```
unixprompt> c4.5 -f ski -m 1

C4.5 [версия 8] генератор дерева решений

                                                Ср 23 Авг 10:44:49 2010
--------------------------------------------

    Параметры:
        Основа файла <ski>
        Разумный тест требует 2 ветвей с >=1 случаем

Считать 11 случаев (3 атрибута) из ski.data

Дерево решений:

Snow_Dist = <=100: ja (4.0)
Snow_Dist = >100:
 | Выходные = нет: нет (3.0)
 | Выходные = да:
 | | Солнце = нет: нет (1.0)
 | | Солнце = да: да (3,0/1,0)

Упрощенное дерево решений:

Snow_Dist = <=100: да (4.0/1.2)
Snow_Dist = >100: нет (7.0/3.4)

Оценка на обучающих данных (11 пунктов):

        Перед обрезкой                  После обрезки
        ----------------                -----------------------------
        Размер      Ошибки              Размер      Ошибки          Оценивать

          7         1(9,1%)               3         2(18,2%)        (41,7%) <<
```

Кроме того, дается упрощенное дерево только с одним атрибутом. Это дерево, которое был создан путем обрезки (см. раздел 8.4.7), будет важен для увеличения количества обучающих данных. В этом маленьком примере это пока не имеет особого смысла. Ошибка Также указана скорость для обоих деревьев на обучающих данных. Цифры в скобках после решений укажите размер базового набора данных и количество ошибок. Например, строка Sun = yes: yes (3.0/1.0) в верхнем дереве указывает что для этого листового узла Sun = yes существуют три обучающих примера, один из которых ложно классифицированы. Таким образом, пользователь может прочитать из этого, является ли решение статистически обоснованным и/или достоверным.

На рис. 8.26 на стр. 193 теперь мы можем представить схему алгоритма обучения . для построения дерева решений.

Теперь мы знакомы с основами автоматической генерации решений. деревья . Однако для практического применения необходимы важные расширения . Мы представит их с помощью уже знакомого приложения LEXMED .

Machine Translated by Google


```
СОЗДАТЬ ДЕРЕВО РЕШЕНИЙ(данные, узел)
Amax = Атрибут с максимальным информационным приростом.
Если G(Amax) = 0,
    то Узел становится конечным узлом с наиболее частым классом в данных. В
    противном случае присвойте узлу атрибут
        Amax. Для каждого значения a1,...,an Amax
                                        сгенерируйте последующий узел: K1,...,Kn
        Разделите данные на D1,...,Dn с Di = {x   Data|Amax(x) = ai}
        Для всех i   {1,...,n}
            Если все x   Di принадлежат одному классу Ci
                Затем сгенерируйте листовой узел Ki класса Ci
                Иначе GENERATEDECISIONTREE(Di, Ki)
```

Рис. 8.26 Алгоритм построения дерева решений

8.4.5 Обучение диагностике аппендицита

В исследовательском проекте LEXMED была разработана экспертная система для диагностики аппендицита на основе базы данных пациентов [ES99, SE00]. Система, работающая по методу максимальной энтропии, описана в разд. 7.3

Теперь мы используем базу данных LEXMED для создания дерева решений для диагностики аппендицита с C4.5. Симптомы, используемые в качестве атрибутов, определены в файле app.names:

```
|Определение классов и атрибутов| | Классы 0 =

отрицательный аппендицит | 1=положительный
аппендицит 0,1. | |Атрибуты|


Возраст:

непрерывный.
Пол_(1=м___2=ж): 1,2.
Боль_Квадрант1_(0=нет__1=да): 0,1.
Боль_Квадрант2_(0=нет__1=да): 0,1.
Боль_Квадрант3_(0=нет__1=да): 0,1.
Боль_Квадрант4_(0=нет__1=да): 0,1.
Локальная _охрана_(0=нет__1=да): 0,1.
Обобщенная _охрана_(0=нет_1=да): 0,1.
Отскок_нежность_(0=нет__1=да): 0,1.
Боль_при_постукивании_(0=нет__1=да): 0,1.
```

Machine Translated by Google


```
Боль_во время _ректального_обследования _(0=нет_1=да): 0,1.
Temp_axial: непрерывный.
Temp_rectal: непрерывный.
Лейкоциты: постоянно.
Сахарный диабет_(0=нет_1=да): 0,1
```

Мы видим, что помимо многих бинарных признаков, таких как различные болевые симптомы, также встречаются постоянные симптомы, такие как возраст и лихорадка. В следующем файле обучающих данных, app.data, в каждой строке описывается случай. В первой строке пациент 19 лет с болью в третьем квадранте (справа внизу, где аппендикс), двумя показателями лихорадки 36,2 и 37,8°C, лейкоцитами 13400 и положительным диагнозом, что есть воспаленный аппендикс.

```
19,1,0,0,1,0,1,0,1,1,0,362,378,13400,0,1 13,1,0,0,1,0,1,0,1,1,1,383,385,18100
,0,1 32,2,0,0,1,0,1,0,1,1,0,364,374,11800,0,1 18,2,0,0,1,1,0,0,0,0
,0,362,370,09300,0,0 73,2,1,0,1,1,1,0,1,1,1,376,380,13600,1,1 30,1,1,1,1,1,0,1
,1,1,1,377,387,21100,0,1 56,1,1,1,1,1,0,1,1,1,0,390,?,14100,0,1 36,1,0,0,1
,0,1,0,1,1,0,372,382,11300,0,1 36,2,0,0,1,0,0,0,1,1,1,370,379,15300,0,1 33,1,0
,0,1,0,1,0,1,1,0,367,376,17400,0,1 19,1,0,0,1,0,0,0,1,1,0,361,375,17600,0,1 12
,1,0,0,1,0,1,0,1,1,0,364,370,12900,0,0




...
```

Не вдаваясь в детали базы данных, важно отметить, что в базу данных включены только пациенты, у которых при поступлении в стационар возникло подозрение на аппендицит, а затем они были прооперированы. Мы видим в седьмой строке, что C4.5 также может иметь дело с пропущенными значениями. Данные содержат 9764 случая.

```
unixprompt> c4.5 -f приложение -u -m 100

C4.5 [версия 8] генератор деревьев решений
                                                    Ср 23 августа 13:13:15 2006
----------------------------------------


Считать 9764 случая (15 атрибутов) из app.data

Дерево решений:
```

Machine Translated by Google


```
Лейкоциты <= 11030:
| Rebound_tenderness = 0:
| | Temp_rectal > 381 : 1 (135,9/54,2)
| | Temp_rectal <= 381 :
| | | Local_guarding = 0:0 (1453,3/358,9)
| | | Local_guarding = 1:
| | | | Пол_(1=м___2=ж) = 1:1 (160,1/74,9)
| | | | Пол_(1=м___2=ж) = 2:0 (286,3/97,6)
| Rebound_tenderness = 1:
| | Лейкоциты <= 8600:
| | | Temp_rectal > 378 : 1 (176,0/59,4)
| | | Temp_rectal <= 378 :
| | | | Пол_(1=м___2=ж) = 1:
| | | | | Local_guarding = 0:0 (110,7/51,7)
| | | | | Local_guarding = 1:1 (160,6/68,5)
| | | | Пол_(1=м___2=ж) = 2:
| | | | | Возраст <= 14 : 1 (131,1/63,1)
| | | | | Возраст > 14 : 0 (398,3/137,6)
| | Лейкоциты > 8600:
| | | Пол_(1=м___2=ж) = 1:1 (429,9/91,0)
| | | Пол_(1=м___2=ж) = 2:
| | | | Local_guarding = 1:1 (311,2/103,0)
| | | | Local_guarding = 0:
| | | | | Temp_rectal <= 375 : 1 (125,4/55,8)
| | | | | Temp_rectal > 375 : 0 (118,3/56,1)
Лейкоциты > 11030:
| Отскок_нежность = 1:1 (4300,0/519,9)
| Rebound_tenderness = 0:
| | Лейкоциты > 14040 : 1 (826,6/163,8)
| | Лейкоциты <= 14040:
| | | Боль_при постукивании = 1:1 (260,6/83,7)
| | | Боль при постукивании = 0:
| | | | Local_guarding = 1:1 (117,5/44,4)
| | | | Local_guarding = 0:
| | | | | Temp_axial <= 368 : 0 (131,9/57,4)
| | | | | Temp_axial > 368 : 1 (130,5/57,8)

Упрощенное дерево решений:

Лейкоциты > 11030 : 1 (5767,0/964,1)
Лейкоциты <= 11030:
| Rebound_tenderness = 0:
| | Temp_rectal > 381 : 1 (135,9/58,7)
| | Temp_rectal <= 381 :
| | | Local_guarding = 0:0 (1453,3/370,9)
| | | Local_guarding = 1:
| | | | Пол_(1=м___2=ж) = 1:1 (160,1/79,7)
| | | | Пол_(1=м___2=ж) = 2:0 (286,3/103,7)
| Rebound_tenderness = 1:
| | Лейкоциты > 8600 : 1 (984,7/322,6)
| | Лейкоциты <= 8600:
```

Machine Translated by Google


```
. . . | Temp_rectal > 378 : 1 (176,0/64,3)
. . . | Temp_rectal <= 378 :
. . . . | Пол_(1=м___2=ж) = 1:
. . . . | | Local_guarding = 0:0 (110,7/55,8)
. . . . | | Local_guarding = 1:1 (160,6/73,4)
. . . . | Пол_(1=м___2=ж) = 2:
. . . . | | Возраст <= 14 : 1 (131,1/67,6)
. . . . | | Возраст > 14 : 0 (398,3/144,7)

Оценка на обучающих данных (9764 элемента):

До обрезки После обрезки
---------- ----------
Ошибки размера Оценка ошибок размера
37 2197(22,5%) 21 2223(22,8%) (23,6%) <<

Оценка на тестовых данных (4882 позиции):

До обрезки После обрезки
---------- ----------
Ошибки размера Оценка ошибок размера
37 1148(23,5%) 21 1153(23,6%) (23,6%) <<

 (a) (b) <-классифицируется как
---
758 885 (a): класс 0
268 2971 (б): класс 1
```

8.4.6 Непрерывные атрибуты

В деревьях, сгенерированных для диагностики аппендицита, есть узел Лейкоциты.
> 11030, что явно происходит от непрерывного атрибута Лейкоциты
установив пороговое значение на уровне 11030. Таким образом, C4.5 создал двоичный атрибут Лейкоциты
> 11030 из непрерывного атрибута Лейкоциты.

Порог $\Theta D,A$ для признака A определяется по следующему алгоритму:
для всех значений v, которые встречаются в обучающих данных D, бинарный атрибут A>v равен
генерируется и рассчитывается его информационный прирост. Затем порог $\Theta D,A$ устанавливается равным
значение v с максимальным приростом информации, то есть:

$$\Theta D,A = \text{argmax}_v \{G(D, A > v)\}.$$

Для такого атрибута, как количество лейкоцитов или возраст пациента, решение, основанное на
бинарная дискретизация, по-видимому, слишком неточна. Тем не менее нет необходимости
дискретизировать лучше, потому что каждый непрерывный атрибут проверяется на каждом вновь сгенерированном
узел и, таким образом, может повторяться в одном дереве с разным порогом $\Theta D,A$. Таким образом
в конечном итоге мы получаем очень хорошую дискретизацию, точность которой соответствует задаче.

Machine Translated by Google

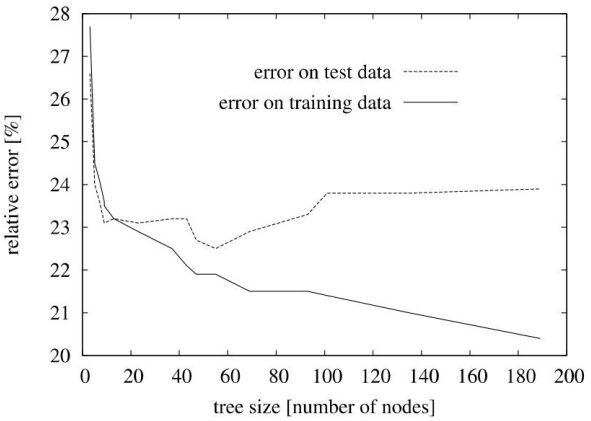


Рис. 8.27 Кривая обучения C4.5 на данных аппендицита. Мы я сно видим переоснащение деревье вс более чем 55 узлами.

8.4.7 Обрезка —спиливание дерева

Со времен Аристотеля было установлено, что из двух научных теорий, одинаково хорошо объясняющих одну и ту же ситуацию, предпочтение отдается более простой. Этот закон экономики, также известный как бритва Оккама, имеет большое значение для машинного обучения и интеллектуального анализа данных.

Дерево решений —это теория для описания обучающих данных. Другая теория описания данных —это сами данные. Если дерево классифицирует все данные без каких-либо ошибок, но гораздо более компактно и, следовательно, более понятно человеку, то оно предпочтительнее в соответствии с бритвой Оккама. То же верно и для двух деревье в решений разного размера. Таким образом, целью каждого алгоритма построения дерева решений должно быть создание наименьшего возможного дерева решений для данной частоты ошибок. Среди всех деревье в с фиксированной частотой ошибок всегда следует выбирать наименьшее дерево.

До сих пор мы не давали точного определения термину «частота ошибок». Как уже неоднократно упоминалось, важно, чтобы обученное дерево не просто запоминало обучающие данные, а хорошо обобщало. Чтобы проверить способность дерева к обобщению, мы делим доступные данные на набор обучающих данных и набор тестовых данных. Тестовые данные скрыты от алгоритма обучения и используются только для тестирования. Если доступен большой набор данных, например данные об аппендиците, то мы можем, например, использовать две трети данных для обучения, а оставшуюся треть —для тестирования.

Помимо лучшей понятности, у бритвы Оккама есть еще одно важное обоснование: способность к обобщению. Чем сложнее модель (в данном случае дерево решений), тем больше подробностей представлено, но в той же степени модель менее переносима на новые данные. Это соотношение показано на рис. 8.27. Деревья решений различных размеров были обучены на данных об аппендиците. На графике приведены ошибки классификации как на обучающих данных, так и на тестовых данных. Частота ошибок в обучающих данных монотонно уменьшается с размером дерева. При размере дерева до 55 узлов частота ошибок в тестовых данных также снижается. Однако если дерево растет дальше, то частота ошибок снова начинает увеличиваться! Этот эффект, который мы уже видели в методе ближайшего соседа, называется переоснащением.

Machine Translated by Google

Мы дадим это понятие, важное почти для всех процессов обучения, общее определение взято из [Mit97]:

> Определение 8.7. Пусть задан конкретный алгоритм обучения, то есть агент обучения. Мы называем агента A подогнанным к обучающим данным, если есть другой агент A', чья ошибка на обучающих данных больше, чем у A, но чья ошибка на всем распределении данных меньше, чем ошибка A.

Как мы теперь можем найти эту точку минимальной ошибки на тестовых данных? Наиболее очевидный алгоритм называется перекрестной проверкой. При построении дерева параллельно измеряется ошибка на тестовых данных. Как только ошибка значительно возрастает, сохраняется дерево с минимальной ошибкой. Этот алгоритм использует упомянутой ранее системой CART.

C4.5 работает несколько иначе. Во-первых, с помощью алгоритма GENERATEDECI SIONTREE, показанного на рис. 8.26 на стр. 193, создается дерево, которое обычно является переобученным. Затем, используя отсечение, он пытается отсечь узлы дерева до тех пор, пока ошибка на тестовых данных, оцениваемая по ошибке на обучающих данных, не начнет расти8. Подобно построению дерева, это также жадный алгоритм. Это означает, что после сокращения узла его нельзя вставить повторно, даже если позже это окажется лучше.

8.4.8 Отсутствующие значения

Часто в обучающих данных отсутствуют отдельные значения атрибутов. В наборе данных LEX MED встречается следующая запись:

56, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 390, ?, 14100, 0, 1,

при котором отсутствует одно из значений лихорадки. Тем не менее такие данные можно использовать при построении дерева решений. Мы можем присвоить атрибуту наиболее частое значение из всего набора данных или наиболее частое из всех точек данных того же класса. Еще лучше заменить отсутствующее значение атрибута распределением вероятностей всех значений атрибута и разбить обучающий пример на ветви в соответствии с этим распределением. Кстати, это является причиной появления нецелочисленных значений в выражениях в круглых скобках рядом с конечными узлами дерева C4.5.

Пропуски значений могут возникать не только при обучении, но и при классификации. Они обрабатываются так же, как и во время обучения.

---

8 Было бы лучше использовать ошибку непосредственно на тестовых данных. По крайней мере, когда объем обучающих данных достаточен для обоснования отдельного набора тестов.

Machine Translated by Google

8.4.9 Резюме

Изучение деревьев решений — излюбленный подход к задачам классификации. Причинами этого являются его простота применения и скорость. В наборе данных, состоящем примерно из 10 000 точек данных LEXMED с 15 атрибутами в каждой, для обучения C4.5 требуется около 0,3 секунды. Это очень быстро по сравнению с другими алгоритмами обучения.

Однако для пользователя также важно, чтобы дерево решений как изученная модель могла быть понята и потенциально изменена. Также нетрудно автоматически преобразовать дерево решений в серию операторов if-then-else и, таким образом, эффективно встроить его в существующую программу.

Поскольку для построения дерева, а также во время отсечения используется жадный алгоритм, деревья в целом не оптимальны. Обнаруженное дерево решений обычно имеет относительно небольшую частоту ошибок. Однако потенциально существует лучшее дерево, потому что эвристический жадный поиск C4.5 предпочитает маленькие деревья и атрибуты с высоким приростом информации в верхней части дерева. Для атрибутов со многими значениями представленная формула прироста информации показывает недостатки. Альтернативы этому приведены в [Mit97].

8.5 Изучение байесовских сетей

В разд. 7.4 показано, как построить байесовскую сеть вручную. Теперь мы введем алгоритмы индукции байесовских сетей. Подобно описанному ранее процессу обучения, байесовская сеть автоматически генерируется из файла, содержащего обучающие данные. Этот процесс обычно разбивается на две части.

1. Изучение структуры сети: для заданных переменных топология сети создается на основе обучающих данных. Этот первый шаг намного сложнее, и позже ему будет уделено более пристальное внимание.
2. Изучение условных вероятностей. Для известных сетевых топологий CPT должны быть заполнены значениями. Если доступно достаточно обучающих данных, все необходимые условные вероятности можно оценить, подсчитав частоты в данных.
   Этот шаг можно относительно легко автоматизировать.
   Теперь мы объясним, как байесовские сети обучаются, используя простой алгоритм из [Jen01].

8.5.1 Изучение структуры сети

При разработке байесовской сети (см. раздел 7.4.6) необходимо учитывать причинно-следственную зависимость переменных, чтобы получить простую сеть хорошего качества. Разработчик-человек полагается на фоновые знания, недоступные машине. Поэтому эту процедуру нельзя легко автоматизировать.

Поиск оптимальной структуры байесовской сети можно сформулировать как классическую задачу поиска. Пусть задан набор переменных V1,...,Vn и файл с обучающими данными. Ищем набор направленных ребер без циклов между узлами

Machine Translated by Google

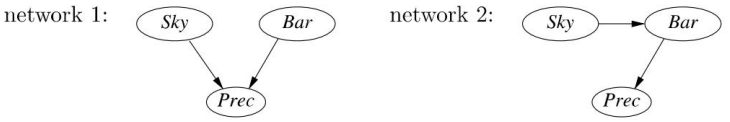


Рис. 8.28 Две байесовские сети для моделирования примера прогноза погоды из упражнения 7.3 на стр. 158

V1,...,Vn, то есть ориентированный ациклический граф (DAG), максимально точно воспроизводящий исходные данные.

Сначала мы наблюдаем пространство поиска. Количество различных DAG растет более чем экспоненциально с увеличением количества узлов. Для пяти узлов имеется 29281 и для девяти узлов около 1015 различных DAG [MDBM00]. Таким образом, неосведомленный комбинаторный поиск (см. п. 6.2) в пространстве всех графов с заданным набором переменных безнадежен, если число переменных растет. Поэтому необходимо использовать эвристические алгоритмы. Это ставит вопрос о функции оценки для байесовских сетей. Можно измерить ошибку классификации сети во время применения к набору тестовых данных, как это делается, например, в C4.5 (см. раздел 8.4). Однако для этого вероятности, рассчитанные байесовской сетью, должны быть сопоставлены с решением.

Прямое измерение качества сети можно провести по распределению вероятностей. Мы предполагаем, что до построения сети по данным мы могли определить (оценить) распределение. Затем начинаем поиск в пространстве всех DAG, оценивая по данным значение CPT для каждой DAG (т. е. для каждой байесовской сети), на основании чего вычисляем распределение и сравниваем его с известным данные. Для сравнения распределений нам, очевидно, понадобится метрика расстояния.

Давайте рассмотрим пример прогноза погоды из упражнения 7.3 на стр. 158 с тремя переменными Sky, Bar, Prec и распределением

P(Небо, Бар, Прек) = (0,40, 0,07, 0,08, 0,10, 0,09, 0,11, 0,03, 0,12).

На рис. 8.28 представлены две байесовские сети, которые мы сейчас сравним по их качеству. Каждая из этих сетей делает предположение о независимости, которое подтверждается тем, что мы определяем распределение сети, а затем сравниваем его с исходным распределением (см. упражнение 8.15 на стр. 219).

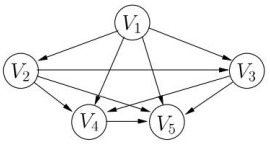
Поскольку для постоянных предопределенных переменных распределение явно представлено вектором постоянной длины, мы можем вычислить евклидову норму разности двух векторов как расстояние между распределениями. Мы определяем

$$dq(x, y) = \sum_{i} (x_i - y_i)^2$$

как сумму квадратов расстояний компонент вектора и вычислить расстояние dq (Pa,P) = 0,0029 распределения Ра сети 1 от исходного распределения. Для сети 2 мы вычисляем dq (Pb, P) = 0,014. Очевидно, что сеть 1 является лучшим приближением к распределению. Часто вместо квадратного расстояния

Machine Translated by Google

Рис. 8.29 Максимальная сеть с пятью переменными и ребер (Vi, Vj), удовлетворяющих условию i<j



так называемое расстояние Кульбака-Лейблера

$$dk(x, y) = \sum_i yi(\log 2\, yi - \log 2\, xi),$$

используется метрика теории информации. Сего помощью мы вычисляем dk(Pa,P) = 0,017 и dk(Pb,P) = 0,09 и приходим к тому же выводу, что и раньше. Следует ожидать, что сети с большим количеством ребер аппроксимируют распределение лучше, чем сети с небольшим количеством ребер. Если все ребра всети построены, то она становится очень запутанной и создает риск переобучения, как это имеет место во многих других алгоритмах обучения. Чтобы избежать переобучения, мы придаем небольшим сетям больший вес с помощью эвристической функции оценки.

$$f(N) = \text{Size}(N) + w \cdot dk(PN, P).$$

Здесь Size(N) —количество записей в CPT, а P N —распределение сети N. w —весовой коэффициент, который необходимо подобрать вручную.

Алгоритм обучения для байесовских сетей, таким образом, вычисляет эвристическую оценку f(N) для множества различных сетей, а затем выбирает сеть с наименьшим значением. Как упоминалось ранее, трудность состоит в сокращении пространства поиска для искомой топологии сети. В качестве простого алгоритма можно, начиная с (например, причинного) упорядочения переменных V1,...,Vn, включать в граф только те ребра, для которых i<j . Начнем с максимальной модели, удовлетворяющей этому условию. Эта сеть показана на рис. 8.29 для пяти упорядоченных переменных.

Теперь, например, в духе жадного поиска (ср. п. 6.3.1), одно ребро за другим удаляются до тех пор, пока значение f не перестанет уменьшаться .

Этот алгоритм не подходит для больших сетей в такой форме. Причинами этого являются большое пространство поиска, ручная настройка веса w и необходимое сравнение с целевым распределением P, потому что они могут просто стать слишком большими или доступный набор данных может быть слишком маленьким.

На самом деле исследования по обучению байесовских сетей все еще идут полным ходом, и существует большое количество предложенных алгоритмов, например алгоритм EM (см. раздел 8.7.2), методы Монте-Карло с цепями Маркова и выборка Гиббса [DHS01, Jor99, Jen01, HTF09]. Помимо представленного здесь пакетного обучения , при котором сеть генерируется один раз из всего набора данных, существуют также инкрементные алгоритмы, в которых каждый отдельный новый случай используется для улучшения сети. Также существуют реализации этих алгоритмов, такие как Hugin (www.hugin.com) и Bayesware (www.bayesware.com).

Machine Translated by Google

8.6 Наивный байесовский классификатор

На рис. 7.14 на стр. 152 диагноз аппендицита был смоделирован в виде байесовской сети. Поскольку направленные ребра начинаются в диагностическом узле и ни одно из них не заканчивается, для ответа на диагностический запрос необходимо использовать формулу Байеса. Для симптомов S1,...,Sn и k-значного диагноза D со значениями b1,...,bk вычисляем вероятность

$$P(D|S1,...,Sn) = P\frac{(S1,...,Sn|D)\cdot P(D)}{(S1,...,Sn)},$$

для постановки диагноза с учетом симптомов пациента. В худшем случае, т. е. при отсутствии независимых переменных, все комбинации всех симптомов и D необходимо было бы определить для всех 20 643 840 вероятностей распределения P(S1,...,Sn,D). Для этого потребуется огромная база данных. В случае с байесовской сетью LEXMED количество необходимых значений (в CPT) сокращено до 521. Однако сеть можно упростить еще больше, если предположить, что все переменные симптомов условно независимы при данном D, то есть:

$$P(S1,...,Sn|D) = P(S1|D)\cdot\cdot\cdot\cdot\cdot P(Sn|D).$$

Затем байесовская сеть для аппендицита упрощается до звезды, показанной на рис. 8.30 на стр. 203.

Таким образом, мы получаем формулу

$$S1,...,Sn) = \frac{P(D)n\ P(Si|D)\ i=1\ P(D|}{П(C1,...,Cн)}. \qquad (8.8)$$

Вычисленные вероятности преобразуются в решение простым наивным байесовским классификатором, который выбирает максимальное P(D = di |S1,...,Sn) из всех значений di в D. То есть он определяет

$$dНаивный\text{-}Байес = argmaxi\ \{1,...,k\}\ P(D = di\ |S1,...,Sn).$$

Поскольку знаменатель в (8.8) постоянный, его можно опустить при максимизации, что приведет к наивной формуле Байеса

$$dНаивный\text{-}Байес = argmaxi\ \{1,...,k\}\ P(D = di)\ \prod_{j=1}^{н} P(Sj\ |D).$$

Поскольку у некоторых узлов теперь меньше предков, количество значений, необходимых для описания распределения LEXMED в CPT, уменьшается, согласно (7.22) на стр. 151, до

$$6\cdot 4 + 5\cdot 4 + 2\cdot 4 + 9\cdot 4 + 3\cdot 4 + 10\cdot(1\cdot 4) + 1 = 141.$$

Machine Translated by Google

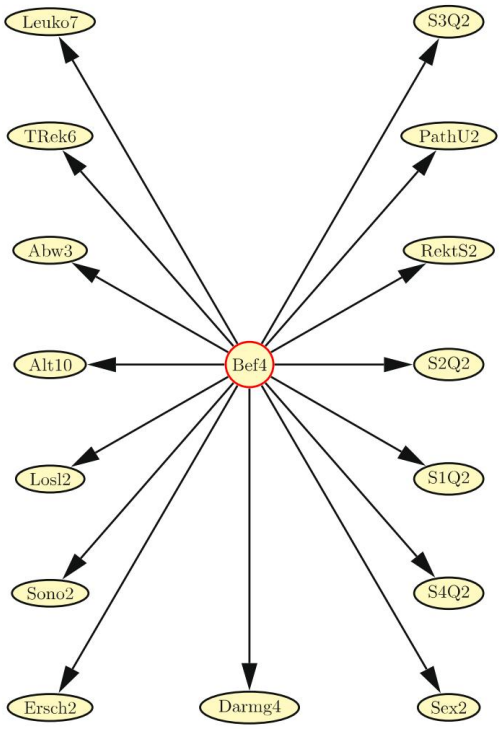


Рис. 8.30 Байесовская сеть для приложения LEXMED в предположении, что все симптомы условно независимы при данном диагнозе

Для медицинской диагностической системы, такой как LEXMED, такое упрощение неприемлемо. Но для задач со многими независимыми переменными наивный байесовский подход частично или даже очень хорошо подходит, как мы увидим на примере текстовой классификации.

Между прочим, наивная байесовская классификация столь же выразительна, как и линейная система оценок, описанная в разд. 7.3.1 (см. упражнение 8.16 на стр. 229). То есть, все оценки разделяющее в основе предположение, что все симптомы условно независимы с учетом диагноза. Тем не менее, счетыдо сих пор используются в медицине. Несмотря на то, что она была создана на основе более качественной репрезентативной базы данных, шкала Омана по сравнению с LEXMED на рис. 7.10 на стр. 142 имеет худшее качество диагностики. Его ограниченная выразительность, безусловно, является причиной этого. Например, как и в случае наивного Байеса, невозможно смоделировать зависимости между симптомами баллы.

Оценка вероятностей Если мы посмотрим на наивную формулу Байеса в (8.8) на стр. 202, мы увидим, что все выражение становится равным нулю, как только один из множителей P (Si |D) в правой части становится равным нулю Теоретически здесь нет ничего плохого. На практике, однако, это может привести к очень неприятным эффектам, если P (Si |D) малы, потому что они оцениваются путем подсчета частот и их подстановки.

Machine Translated by Google

в

$$P(Si = x|D = y) = \frac{|Si = x \quad D = y|}{|D = y|}.$$

Предположим, что для переменных Si : P (Si = x|D = y) = 0,01 и что имеется 40 обучающих случаев с D = y. Тогда с большой вероятностью обучающего случая с Si = x и D = y нет, и мы оцениваем P (Si = x|D = y) = 0. Для другого значения D = z предположим, что отношения расположены аналогично, но оценка приводит к значения м больше нуля для всех P (Si = x|D = z). Таким образом, всегда предпочтительнее значение D = z, которое не отражает фактического распределения вероятностей. Поэ тому при оценке вероятностей используется формула

$$П(A|B) \quad \frac{|A \quad B|}{|Б|} \text{ "=" } \frac{наб}{нБ}$$

заменена на

$$П(A|B) \quad \frac{нАБ + т.п.}{нБ + м},$$

где p = P (A) —априорная вероятность для A, а m —константа, которую можно свободно выбирать и которая известна как «эквивалентный размер данных». Чем больше становится m, тем больше вес априорной вероятности по сравнению со значением, определенным из измеренной частоты.

8.6.1 Классификация текста с помощью наивного Байеса

Наивный байесовский метод сегодня очень успешен и эффективен в классификации текстов. Его основным и в то же время очень важным применением является автоматическая фильтрация электронной почты нежелаемые и нежелательные или спам-письма. В спам-фильтрах, таких как SpamAssassin [Sch04], среди прочих методов используется наивный байесовский классификатор, который учится отделять нужные электронные письма от спама. SpamAssassin —это гибридная система, которая выполняет начальную фильтрацию с использованием черного и белого списков. Черные списки —это списки заблокированных адресов электронной почты от отправителей спама, чьи электронные письма всегда удаляются , а белые списки —это списки с отправителями, чьи электронные письма всегда доставляются . После такой предварительной фильтрации оставшиеся письма классифицируются наивным байесовским классификатором по их фактическому содержанию, то есть по тексту . Обнаруженное значение класса затем оценивается по баллу вместе с другими атрибутами из заголовка электронной почты, такими как домен отправителя , тип MIME и т. д., а затем, наконец, фильтр

Здесь весьма важна способность к обучению наивного байесовского фильтра. Для этого пользователь должен сначала вручную классифицировать большое количество писем как желаемое или как спам. Затем фильтр обучается . Чтобы оставаться в курсе , фильтр необходимо регулярно переобучать. Для этого пользователь должен корректно классифицировать все письма, ошибочно классифицированные фильтром, то есть поместить их в соответствующие папки. Затем фильтр постоянно переобучается с помощью этих писем.

Machine Translated by Google

Помимо фильтрации спама, существует множество других приложений для автоматической классификации текстов. Важные приложения включают в себя фильтрацию нежелательных записей на дискуссионных интернет-форумах и отслеживание веб-сайтов с криминальным содержанием, таким как боевики или террористическая деятельность, детская порнография или расизм. Его также можно использовать для настройки поисковых систем в соответствии с предпочтениями пользователя, чтобы лучше классифицировать результаты поиска. В промышленной и научной среде поиск в масштабах всей компании в базах данных или в литературе находится на переднем плане исследований. Благодаря своей способности к обучению фильтр может адаптироваться к привычкам и пожеланиям каждого отдельного пользователя.

Мы представим применение наивного Байеса к анализу текста на коротком примере. обширный текст Алана Тьюринга из [Tur50]:

> «Мы можем надеяться, что машины со временем будут конкурировать с людьми во всех чисто интеллектуальных областях. Но с каких лучше начать? Даже это трудное решение. Многие люди думают, что лучше всего подойдет очень абстрактная деятельность, например, игра в шахматы. Можно также утверждать, что лучше всего снабдить машину лучшими органами чувств, которые можно купить за деньги, а затем научить ее понимать и говорить по-английски. Этот процесс может следовать обычному обучению ребенка. Будут указаны и названы вещи и т. д. Опять же, я не знаю, какой правильный ответ, но я думаю, что следует попробовать оба подхода».

Предположим, что тексты, подобные приведенному, следует разделить на два класса: «I» — интересные и «¬I — неинтересные. Предположим также, что существует база данных уже классифицированных текстов. Какие атрибуты следует использовать? В классическом подходе к построению байесовской сети мы определяем набор атрибутов, таких как длина текста, средняя длина предложения, относительная частота определенных знаков препинания, частота некоторых важных слов, таких как «я». , «машины» и т. д. При классификации с использованием наивного Байеса, напротив, выбирается удивительно примитивный алгоритм. Для каждой из n позиций слов в тексте определяется атрибут si. Все слова, встречающиеся в тексте, допускаются в качестве возможных значений для всех позиций si. Теперь для классов I и ¬I значе

$$P(I \mid s1,\ldots,sn) = c \cdot P(I) \prod_{i=1}^{n} P(si \mid I) \qquad (8.9)$$

и $P(\neg I \mid s1,\ldots,sn)$ должны быть рассчитаны, а затем выбран класс с максимальным значением. В приведенном выше примере со 113 словами это дает

$$P(I \mid s1,\ldots,sn) = c \cdot P(I) \cdot P(s1 = \text{«Мы»} \mid I) \cdot P(s2 = \text{«может»} \mid I) \cdots\cdots P(s113 = \text{«должен»} \mid I)$$

и

$$P(\neg I \mid s1,\ldots,sn) = c \cdot P(\neg I) \cdot P(s1 = \text{«Мы»} \mid \neg I) \cdot P(s2 = \text{«может»} \mid \neg I) \cdots\cdots P(s113 = \text{«должен»} \mid \neg I).$$

Обучение здесь достаточно простое. Условные вероятности $P(si \mid I)$, $P(si \mid \neg I)$ и априорные вероятности $P(I)$, $P(\neg I)$ должны быть просто вычислены. Теперь дополнительно предположим, что $P(si \mid I)$ не зависит от позиции в тексте. Этот

Machine Translated by Google

означает, например, что

$$P(s61 = «и» \mid I) = P(s69 = «и» \mid I) = P(s86 = «и» \mid I).$$

Таким образом, мы могли бы использовать выражение $P(и|I)$ с новой двоичной переменной и как вероятность появления «и» в произвольной позиции.

Реализация может быть несколько ускорена, если найти частоту ni каждое слово wi, которое встречается в тексте, и используйте формулу

$$P(I|s1,...,sn) = c \cdot P(I) \prod_{i=1}^{л} P(wi|I)^{ni} \quad (8.10)$$

что эквивалентно (8.9) на стр. 205. Обратите внимание, что индекс i в произведении соответствует только количеству l различных слов, встречающихся в тексте.

Несмотря на свою простоту, наивный байесовский алгоритм дает отличные результаты для классификации текстов. Спам-фильтры, работающие с наивным байесовским алгоритмом, обеспечивают уровень ошибок менее одного процента. Системы DSPAM и CRM114 могут быть даже настолько хорошо обучены, что неправильно классифицируют только одно из 7000 или 8000 писем соответственно. Это соответствует точности почти 99,99%.

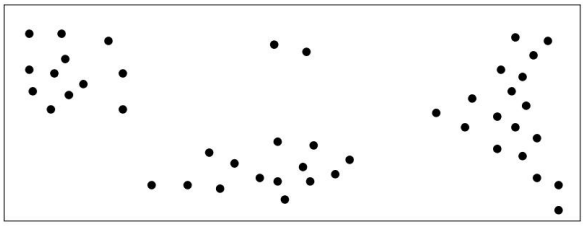
8,7 Кластеризация

Если мы будем искать в поисковой системе термин «марс», мы получим такие результаты, как «планета марс» и «конгломерат шоколада, кондитерских изделий и напитков», которые семантически совершенно разные. В наборе обнаруженных документов есть два заметно различающихся кластера. Google, например, по-прежнему выводит результаты в неструктурированном виде. Было бы лучше, если бы поисковая система разделяла кластеры и представляла их пользователю соответствующим образом, потому что пользователя обычно интересует только один из кластеров.

Отличие кластеризации от обучения с учителем заключается в том, что обучающие данные не помечены. Таким образом, предварительная структуризация данных руководителем отсутствует. Скорее, поиск структуры и есть весь смысл кластеризации. В пространстве обучающих данных должны быть найдены скопления данных, подобные тем, что показаны на рис. 8.31 на стр. 207 . В кластере расстояние между соседними точками обычно меньше, чем расстояние между точками разных кластеров. Поэтому выбор подходящей метрики расстояния для точек, т. е. для группируемых объектов и для кластеров, имеет принципиальное значение. Как и прежде, в дальнейшем мы предполагаем, что каждый объект данных описывается вектором числовых атрибутов.

Machine Translated by Google

Рис. 8.31. Простой двумерный пример с четырьмя четко разделенными кластерами



8.7.1 Метрики расстояния

Соответственно для каждого приложения различные метрики расстояния определяются для расстояния d между двумя векторами x и y в Rn: Наиболее распространенным является евклидово расстояние.

$$\text{де}(x, y) = \sqrt{\sum_{i=1}^{n} (xi - yi)^2}.$$

Несколько проще сумма квадратов расстояний

$$dq(x, y) = \sum_{i=1}^{n} (\text{кси} - \text{у и})^2,$$

что для алгоритмов, в которых сравниваются только расстояния, эквивалентно евклидову расстоянию (упражнение 8.19 на стр. 220). Также используются вышеупомянутые манхэттенские расстояния.

$$\text{дм}(x, y) = \sum_{i=1}^{n} |xi - yi|$$

а также расстояние максимальной составляющей

$$d(x, y) = \max_{i=1,\ldots,n} |xi - yi|$$

который основан на максимальной норме. Во время классификации текста нормированная проекция двух векторов друг на друга, то есть нормированное скалярное произведение

$$\frac{xy}{|x||y|}$$

часто вычисляется, где $|x|$ является евклидовой нормой x. Поскольку эта формула является метрикой подобия двух векторов, в качестве метрики расстояния обратная

$$\text{дс}(x, y) = \frac{|x||y|}{xy}$$

можно использовать или «>» и «<» можно поменять местами для всех сравнений. При поиске текста атрибуты x1,...,xn вычисляются аналогично наивному байесовскому методу как компоненты вектора x следующим образом. Для словаря с 50 000 слов значение xi

Machine Translated by Google

равно частоте i-го словарного слова в тексте. Поскольку обычно в таком векторе почти все компоненты равны нулю, при вычислении скалярного произведения почти все члены суммирования равны нулю. Используя такую информацию, можно значительно ускорить реализацию (упражнение 8.20 на стр. 220).

8.7.2 k-средних и алгоритм EM

Всякий раз, когда количество кластеров уже известно заранее, можно использовать алгоритм k-средних. Как следует из названия, k кластеров определяется их средним значением. Сначала случайным образом или вручную инициализируются k средних точек кластера μ1 ,...,μk . Затем повторно выполняются следующие два шага: • Классификация всех данных до их ближайшей средней точки кластера • Повторное вычисление средней точки кластера.
В качестве алгоритма получается следующая схема:

```
K-MEANS(x1,..., xn,k)
инициализируется μ1 ,...,μk (например, случайным образом
Повторить
    классификацию x1,..., xn до ближайшего μi
    пересчитать μ1 ,...,μk до
тех пор, пока не останется
изменений в μ1 ,...,μk Возврат(μ1 ,...,μk )
```

Вычисление середины кластера μ для точек x1,..., xl выполняется по формуле

$$м = \frac{1}{л} \sum_{я=1}^{л} xi .$$

Выполнение на примере показано на рис. 8.32 на стр. 209 для случая двух классов. Мы видим, как после трех итераций центры классов, которые сначала были выбраны случайным образом, стабилизируются. Хотя этот алгоритм не гарантирует сходимости, обычно он сходится очень быстро. Это означает, что количество шагов итерации обычно намного меньше количества точек данных. Его сложность равна O(ndkt), где n —общее количество точек, d —размерность пространства признаков, а t —количество шагов итерации.

Во многих случаях необходимость заранее указывать количество классов создает неудобное ограничение. Поэтому далее мы введем более гибкий алгоритм.

Однако перед этим мы упомянем алгоритм EM, который является непрерывным вариантом k-средних, так как он не делает жесткого распределения данных по классам, а скорее для каждой точки возвращает вероятность ее принадлежности различным классы

Machine Translated by Google

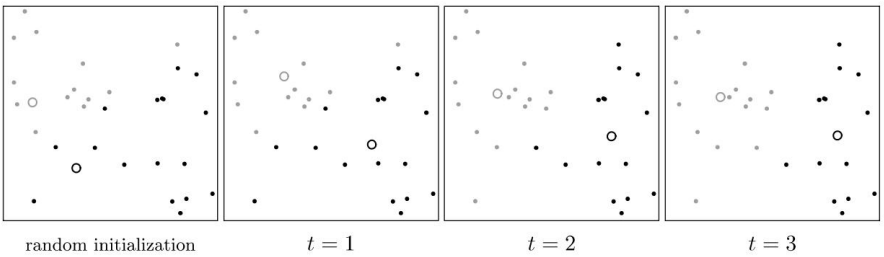


Рис. 8.32. k-средних с двумя классами (k = 2), примененными к 30 точкам данных. Крайний слева —набор данных с начальными центрами, а справа —кластер после каждой итерации. После трех итераций достигается сходимость

Здесь мы должны предположить, что тип распределения вероятностей известен. Часто используется нормальное распределение. Задача ЭМ-алгоритма состоит в определении параметров (среднего μi и ковариационных матриц Σi k многомерных нормальных распределений) для каждого кластера. Аналогично k-средним повторно выполняются два следующих шага:
Ожидание : для каждой
точки данных вычисляется вероятность P (Cj |xi) того, что она принадлежит каждому кластеру.

Максимизация : Используя вновь рассчитанные вероятности, пересчитываются параметры распределения .

Тем самым достигается более мягкая кластеризация , что во многих случаях приводит к лучшим результатам. Это чередование между ожиданием и максимизацией дало название алгоритму . Вдополнение к кластеризации, например, алгоритм EM используется для изучения байесовских сетей [DHS01].

8.7.3 Иерархическая кластеризация

В иерархической кластеризации мы начинаем с n кластеров, состоящих из одной точки каждый. Затем кластеры ближайших соседей объединяются до тех пор, пока все точки не будут объединены в один кластер или пока не будет достигнут критерий завершения . Получаем схему

```
ИЕРАРХИЧЕСКАЯ КЛАСТЕРИЗАЦИЯ(x1,..., xn,k)
инициализирует C1 = {x1},...,Cn = {xn}
Повторить
    Найти два кластера Ci и Cj с наименьшим расстоянием Объединить
    Ci и Cj до тех пор, пока
не будет достигнуто условие завершения
Возврат (дерево с кластерами)
```

Machine Translated by Google

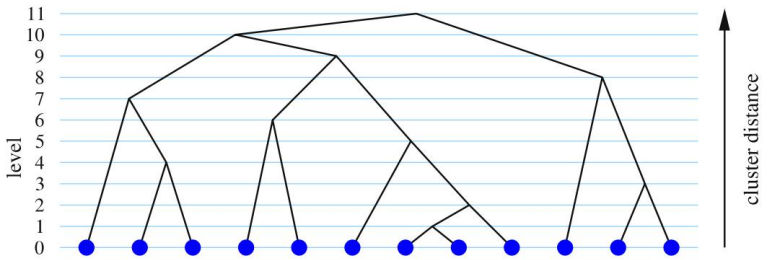


Рис. 8.33. При иерархической кластеризации на каждом шаге объединяются два кластера с наименьшим расстоянием

Условие завершения может быть выбрано, например, как желаемое количество кластеров или максимальное расстояние между кластерами. На рис. 8.33 этот алгоритм схематично представлен в виде бинарного дерева, в котором снизу вверх на каждом шаге, т. е. на каждом уровне, соединены два поддерева. На верхнем уровне все точки объединяются в один большой кластер.

Пока неясно, как рассчитываются расстояния между скоплениями. Действительно, в предыдущем разделе мы определили различные метрики расстояния для точек, но их нельзя использовать для кластеров. Удобной и часто используемой метрикой является расстояние между двумя ближайшими точками в двух кластерах Ci и Cj :

$$dmin(Ci, Cj) = \min d(x, y). \; x \in Ci, \; y \in Cj$$

Таким образом, мы получаем алгоритм ближайшего соседа, применение которого показано на рис. 8.34 на стр. 211. Дерево[9] 10. Мы видим, что этот алгоритм генерирует минимальный охват Кроме того, пример показывает, что два описанных алгоритма генерируют совершенно разные кластеры. Это говорит нам о том, что для графов с кластерами, которые четко не разделены, результат сильно зависит от алгоритма или выбранной метрики расстояния .

Для эффективной реализации этого алгоритма мы сначала создадим матрицу смежности, в которой сохраняются расстояния между всеми точками, что требует $O(n^2)$ времени и памяти. Если количество кластеров не имеет верхнего предела, цикл будет повторяться $n - 1$ раз, а асимптотическое время вычисления станет равным $O(n^3)$.

Чтобы вычислить расстояние между двумя кластерами, мы также можем использовать расстояние между двумя самыми дальними точками

$$dmax(Ci, Cj) = \max d(x, y) \; x \in Ci, y \in Cj$$

9 Алгоритм ближайшего соседа не следует путать с методом ближайшего соседа для классификация из разд. 8.3.

10 Минимальное остовное дерево — это ациклический неориентированный граф с минимальной суммой длин ребер.

Machine Translated by Google

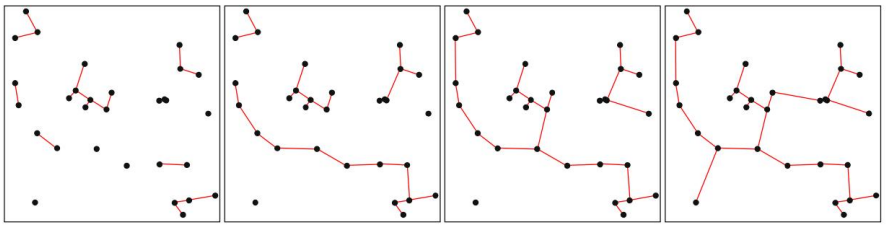


Рис. 8.34 Алгоритм ближайшего соседа, примененный к данным из рис. 8.32 на стр. 209 на разных уровнях с 12, 6, 3, 1 кластерами

и получить алгоритм самого дальнего соседа. В качестве альтернативы используется расстояние от средней точки кластера dμ(Ci, Cj) = d(μi,μj). Помимо алгоритма кластеризации представленные здесь, есть и многие другие, для ознакомления с которыми мы отправляем читателя к [DHS01] для дальнейшего изучения.

8.8 Интеллектуальный анализ данных на практике

Все алгоритмы обучения, представленные до сих пор, можно использовать в качестве инструментов для интеллектуального анализа данных. Для пользователю, однако, иногда довольно сложно привыкнуть к новому программному обеспечению инструменты для каждого приложения и, кроме того, поместить данные для анализа в соответствующий формат для каждого конкретного случая.

Ряд систем интеллектуального анализа данных решают эти проблемы. Большинство этих систем предлагают удобный графический пользовательский интерфейс с разнообразными инструментами для визуализации данных, для предварительной обработки, такой как манипулирование отсутствующими значениями, и для анализа. Для анализ, алгоритмы обучения, представленные здесь, используются, среди прочего.

Отдельного упоминания заслуживает все объемлющая библиотека Java с открытым исходным кодом WEKA. Он предлагает большое количество алгоритмов и упрощает разработку новых алгоритмов.

Свободно доступная система KNIME, которую мы кратко представим в следующий раздел предлагает удобный пользовательский интерфейс и все виды инструментов упомянутое выше. KNIME также использует модули WEKA. Кроме того, он предлагает простой способ управления потоком данных выбранных инструментов визуализации, предварительной обработки и анализа с помощью графического редактора. В то же время большое количество других систем предлагают весьма схожие функции, например, проект с открытым исходным кодом RapidMiner(www.rapidminer.com), система Clementine (www.spss.com/clementine), продаваемый SPSS, и аналитическая структура KXEN (www.kxen.com).

Machine Translated by Google

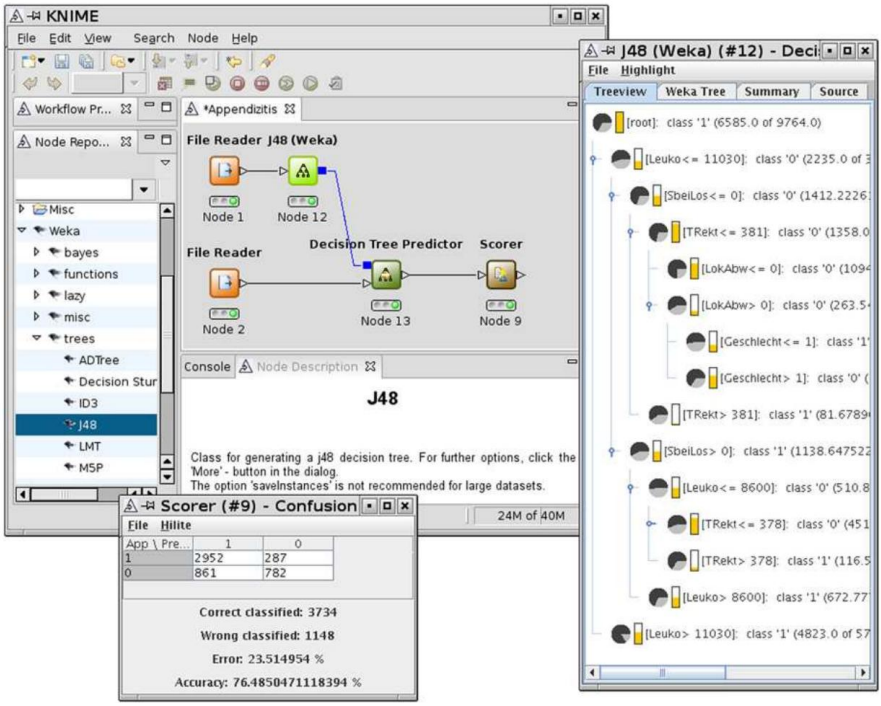


Рис. 8.35 Пользовательский интерфейс KNIME с двумя дополнительными представлениями, которые показывают дерево решений и матрицу путаницы

8.8.1 Инструмент интеллектуального анализа данных KNIME

Теперь, используя данные LEXMED, мы покажем, как извлекать знания из данных с помощью KNIME (Konstanz Information Miner, www.knime.org). Сначала мы создаем дерево решений, как показано на рис. 8.35. После создания нового проекта рабочий процесс строится графически. Для этого соответствующие инструменты просто достаются мышкой из репозитория узла и перетаскиваются в главное окно рабочего процесса.

Тренировочные и тестовые данные из файла C4.5 можно без проблем считывать с помощью двух узлов чтения файлов. Однако эти узлы также могут быть легко настроены для других форматов файлов. Боковой светофор под узлом показывает его статус (не готов, настроен, выполнен). Затем выбирается узел J48 из библиотеки WEKA [WF01], которая содержит Java-реализацию C4.5. Конфигурация для этого довольно проста. Теперь выбирается предикторный узел, который применяет сгенерированное дерево к тестовым данным. Он вставляет новый столбец в таблицу тестовых данных «Предсказание» с классификацией, сгенерированной деревом. Оттуда узел подсчета очков вычисляет матрицу путаницы, показанную на рисунке, которая дает количество правильно классифицированных случаев для обоих классов на диагонали, а также количество ложноположительных и ложноотрицательных точек данных.

Machine Translated by Google

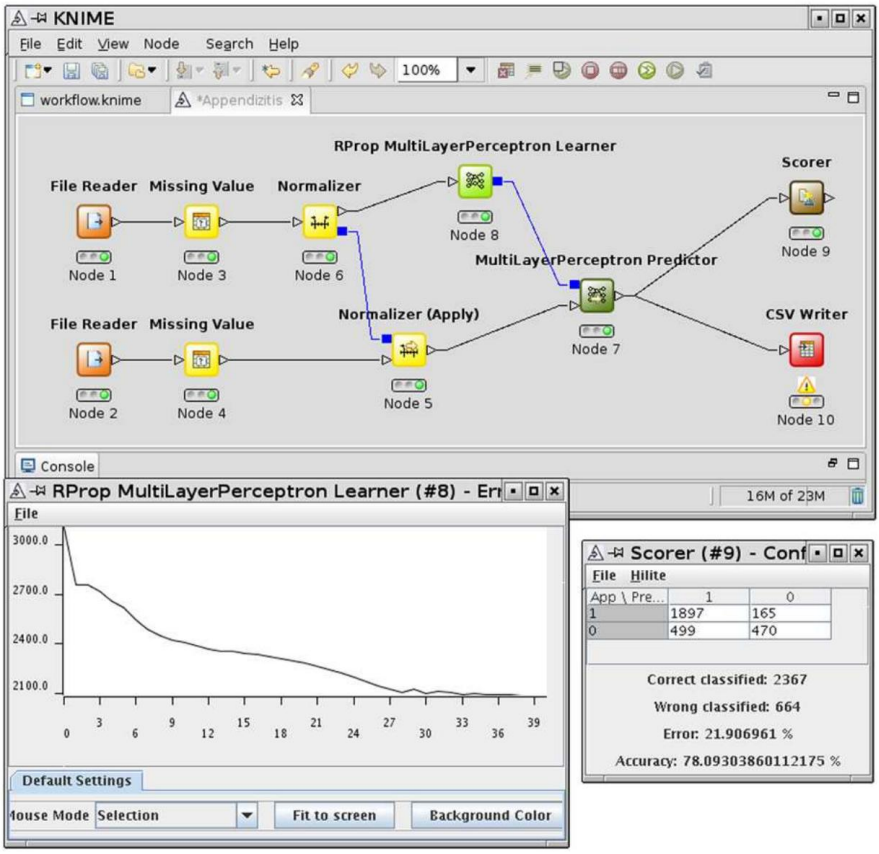


Рис. 8.36. Пользовательский интерфейс KNIME с окном рабочего процесса, кривой обучения и матрицей путаницы

После того, как поток полностью построен и все узлы настроены, произвольный узел может быть казнен. Он автоматически гарантирует выполнение узлов-предшественников, если необходимый. Узел J48 генерирует представление дерева решений, показанное справа. фигуры Это дерево идентично дереву, сгенерированному C4.5 в разд. 8.4.5, хотя здесь узел TRekt<=378 показан свернутым.

Для сравнения, проект обучения многослойного персептрона (см. раздел 9.5) показано на рис. 8.36. Это работает аналогично введенному ранее линейному персептрону, но также может разделять нелинейно разделимые классы Здесь течение несколько более сложный. Дополнительный узел необходим для обработки ошибок отсутствующих значений, когда предварительная обработка каждого из двух файлов. Мыустанавливаемего так, чтобыэтистроки были удалены Поскольку у нейронные сети не могут работать с произвольными значениями, значения всех переменных линейно масштабируются в интервале [0, 1] с помощью узла «нормализатор».

Machine Translated by Google

После применения обучающего модуля RProp улучшено обратное распространение (см. Разд. 9.5), мы можем проанализировать изменение во времени ошибки аппроксимации в показана кривая обучения. В матрице путаницы бомбардировщика выводит анализ тестовых данных. Узел «CSV Writer» в правом нижнем углу служит для экспорта файлы результатов, которые затем можно использовать извне для создания ROC-кривой, показанной на рис. Рис. 7.10 на стр. 142, для которого, к сожалению, нет инструмента KNIME (пока).

Подводя итог, можно сказать о KNIME (и подобных инструментах) следующее: для проекта с не слишком экзотическими требованиями к анализу данных, стоит работать с таким мощным верстаком для анализа данных. Пользователь уже сохраняет много времени на этапе предварительной обработки. Существует большой выбор легко используемых «инструментов общих», таких как классификаторы ближайших соседей, простые алгоритмы обучения байесовской сети, а также алгоритмы кластеризации k-средних (раздел 8.7.2). Оценка результатов, например перекрестную проверку, можно легко провести. Осталось Следует отметить, что помимо показанных существует множество других инструментов для визуализации данных. Кроме того, разработчики KNIME сделали расширение для Доступен KNIME, с помощью которого пользователь может программировать свои собственные инструменты на Java или Python.

Однако следует также упомянуть, что пользователь этого типа интеллектуального анализа данных система должна принести с собой прочные предварительные знания в области машинного обучения и использования методов интеллектуального анализа данных. Программное обеспечение само по себе не может анализировать данные, но в руке специалиста становится мощным инструментом извлечения знаний из данных.
Для новичка в увлекательной сфере машинного обучения такая система предлагает идеальная и простая возможность проверить свои знания на практике и сравнить различные алгоритмы Читатель может убедиться в этом в упражнении 8.21 на странице 220.

8.9 Резюме

Мы подробно рассмотрели несколько алгоритмов из устоявшейся области обучения с учителем, включая обучение по дереву решений, байесовские сети и ближайший алгоритм.
метод соседа. Эти алгоритмы стабильны и эффективно используются в различных приложениях и, таким образом, относятся к стандартному репертуару ИИ и интеллектуального анализа данных.
то же самое верно и для алгоритмов кластеризации, которые работают без «супервайзера» и можно найти, например, в приложениях поисковых систем. Обучение с подкреплением поскольку у другая область машинного обучения также не использует супервайзера. В отличие от обучения с учителем, когда учащийся получает правильные действия или ответы в качестве ярлыков. в обучающих данных, в обучении с подкреплением только время от времени поступает положительная или отрицательная обратная связь из окружающей среды В гл. 10 мы покажем, как это работает. Не такая сложная задача в полуконтролируемом обучении, молодой подобласти. машинного обучения, где только очень немногие из большого количества обучающих данных помечен.

Обучение с учителем в настоящее время является хорошо зарекомендовавшей себя область ю с большим количеством успешных приложений. Для контролируемого обучения данных с непрерывными метками может быть использован любой алгоритм аппроксимации функций. Таким образом, существует множество алгоритмов из различных областей математики и информатики. В разд. 9 мы представим различные типы нейронных сетей, алгоритмы наименьших квадратов и машины опорных векторов,

Machine Translated by Google

которые являются аппроксиматорами функций. В настоящее время гауссовыпроцессыочень популярны поскольку они очень универсальны и просты в применении и предоставляют пользователюоценку неопределенности выходных значений [RW06].

Следующая таксономия дает обзор наиболее важных алгоритмов обучения и их классификацию

Контролируемое обучение

- Ленивое обучение
  - метод k ближайших соседей (классификация + аппроксимация)
  - Локально взвешенная регрессия (аппроксимация)
  - Обучение на основе конкретных случаев (классификация + аппроксимация) •
  - Усердное обучение – Индукция деревьев решений (классификация)
  - Обучение байесовских сетей (классификация + аппроксимация)
  - Нейронные сети (классификация + аппроксимация)
  - Гауссовские процессы(классификация + аппроксимация)
  - вейвлеты сплайны радиальные базисные функции, . . .

Неконтролируемое обучение (кластеризация)

- Алгоритм ближайшего соседа •
- Алгоритм дальнего соседа • k-средних
- • Нейронные
- сети Обучение с
- подкреплением • Итерация
- значений • Q-
- обучение • TD-
- обучение • Методы
- градиента политик • Нейронные
- сети Однако сказанное

об обучении с учителем верно. , при работе с фиксированным набором известных атрибутов. Интересным, все еще открытым полем для интенсивных исследований является автоматический выбор признаков. В разд. 8.4, для обучения с деревьями решений мы представили алгоритм расчета прироста информации атрибутов, который сортирует атрибуты в соответствии с их релевантностью и использует только те, которые улучшают качество классификации. С помощью этого типа метода можно автоматически выбирать соответствующие атрибуты из потенциально большого базового набора.

Однако этот базовый набор необходимо выбирать вручную

До сих пор открытым и не вполне определенным является вопрос о том, как машина может находить новые атрибуты Представим себе робота, который должен собирать яблоки. Для этого он должен научиться различать спелые и незрелые яблоки и другие предметы
Традиционно мы определяли определенные атрибуты, такие как цвет и форма областей пикселей, а затем обучали алгоритм обучения , используя классифицированные вручную изображения .
Также возможно, что, например, нейронная сеть может быть обучена напрямую со всеми пикселями из изображения в качестве входных данных, что, однако, для высокого разрешения связано с серьезными проблемами времени вычислений. Здесь желательны подходы, которые автоматически предлагают соответствующие функции. Но это все еще научная фантастика.

Machine Translated by Google

Кластеризация обеспечивает один из подходов к выбору признаков. Перед обучением машины распознавания я блок мы запускаем кластеризацию данных. Для (контролируемого) обучения классов я блоки не я блоки входными данными больше не являются все пиксели, а только классы, найденные во время кластеризации, потенциально вместе с другими атрибутами.
Кластеризация в любом случае может быть использована для автоматического, творческого «обнаружения » признаков. Однако нея сно, имеют ли значение обнаруженные признаки.

Еще сложнее следующая задача: предположим, что видеокамера, используемая для распознавания я блок, передает только черно-белые изображения . Задача уже не может быть решена хорошо. Было бы неплохо, если бы машина занималась творчеством сама по себе, например, предлагая заменить камеру цветной камерой.
Сегодня это было бы слишком много.

Помимо специализированных работ по всем разделам машинного обучения , есть отличные учебники [Mit97, Bis06, Alp04, DHS01, HTF09]. Текущие результаты исследований можно найти в свободном доступе в Journal of Machine Learning Research (http://jmlr.csail.mit.edu), в журнале Machine Learning Journal, а также в материалах Международной конференции по машинному обучению(ICML).) Рекомендовано. Каждому разработчику алгоритмов обучения интересен репозиторий машинного обучения [DNM98] Калифорнийского университета в Ирвайне (UCI) с его большой коллекцией обучающих и тестовых данных для алгоритмов обучения и инструментов интеллектуального анализа данных. MLOSS, что означает программное обеспечение с открытым исходным кодом для машинного обучения , представляет собой отличный каталог ссылок на бесплатное программное обеспечение (www.mloss.org).

8.10 Упражнения

8.10.1 Введение

Упражнение 8.1

(а) Укажите задачу агента, который должен предсказать погоду на следующий день по измеренным значениям температуры, атмосферного давления и влажности. Погоду следует отнести к одному из трех классов: солнечная , облачная и дождливая .
б) Опишите структуру файла с обучающими данными для этого агента.

Упражнение 8.2. Покажите, что корреляционная матрица симметрична и что все диагональные элементы равны1.

8.10.2 Персептрон

Упражнение 8.3. Применение правила обучения персептрона к множествам

$$M+ = \{(0, 1,8), (2, 0,6)\} \text{ и } M = \{(1,2, 1,4), (0,4, 1)\}$$

из Примера 8.2 на стр. 172 и выдать результат значений весового вектора.

Machine Translated by Google

Упражнение 8.4. Имея таблицу справа с обучающими данными: (a) Используя график, покажите, что данные линейно разделимы (b) Вручную определить веса w1 и w2, а также порог Θ персептрона (с порогом), который правильно классифицирует данные.

номер	x1	x2	Класс
1	6	10	−
2	7	30	−
3	8	20	−
4	9	00	−
5	8	41	−
6	8	61	+
7	9	21	+
8	9	51	+

(c) Запрограммируйте правило обучения персептрона и примените вашу программу к таблице. Сравните обнаруженные веса с рассчитанными вручную

Упражнение 8.5

(a) Дайте визуальную интерпретацию эвристической инициализации.

$$w_0 = -\left(\frac{x_{i+} + x_{i-}}{2}\right)^T \frac{x_{i+} - x_{i-}}{\|x_{i+} - x_{i-}\|},$$

весового вектора, описанного в разд. 8.2.2.

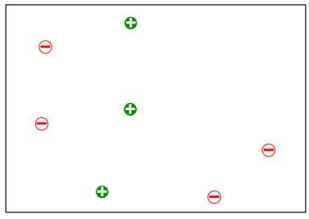
(b) Приведите пример линейно разделимого набора данных, для которого эта эвристика не дает разделительной линии.

8.10.3 Метод ближайших соседей

Упражнение 8.6

(a) Изобразите диаграмму Вороного для соседних наборов точек.

б) Затем нарисуйте линии деления классов.



Упражнение 8.7. Пусть дана таблица с обучающими данными из упражнения 8.4. В дальнейшем используйте манхэттенское расстояние d(a, b), определяемое как $d(a, b) = |a_1 - b_1| + |a_2 - b_2|$, чтобы определить расстояние d между двумя точками данных a = (a1, a2) и b = (b1, b2). (a) Классифицируйте вектор v = (8, 3.5) методом ближайшего соседа. (b) Классифицировать вектор v = (8, 3.5) методом k ближайших соседей для k = 2, 3, 5.

Упражнение 8.8

(a) Покажите, что в двумерном пространстве признаков разумно, как утверждается в (8.3) на стр. 180, взвешивать k ближайших соседей на величину, обратную квадрату расстояния.

(b) Почему взвешивание с использованием $w_i' = \frac{1}{1+\alpha d(x,x_i)}$ имеет меньше смысла?

Machine Translated by Google

8.10.4 Деревья решений

Упражнение 8.9. Покажите, что определение $0 \log_2 0 := 0$ разумно, другими словами, что функция $f(x) = x \log_2 x$ тем самым становится непрерывной в начале координат.

Упражнение 8.10. Определите энтропию для следующих распределений.

(a) (1, 0, 0, 0, 0)

(b) $\frac{1}{2}, \frac{1}{2}, 0, 0, 0$

(c) $\frac{1}{2}, \frac{1}{4}, \frac{1}{4}, 0, 0$

(d) $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16}$

(e) $\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}$

(f) $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \ldots$

Упражнение

8.11 (a) Покажите, что два разных определения энтропии из (7.9) на стр. 124 и определения 8.4 на стр. 188 отличаются только постоянным множителем, т. е. что

$$\sum_{i=1}^{n} p_i \log_2 p_i = c \sum_{i=1}^{n} p_i \ln p_i$$

и задайте константу c.

(b) Покажите, что для метода MaxEnt и для деревьев решений не имеет значения, какую из двух формул мы используем.

Упражнение 8.12. Разработайте дерево решений для набора данных D из упражнения 8.4 на стр. 217. (a)

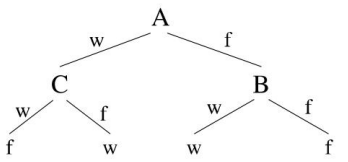
Рассматривайте оба атрибута как

дискретные. (b) Теперь считайте атрибут x2 непрерывным, а x1

дискретным. (c) Пусть C4.5 сгенерирует дерево с обоими вариантами. Используйте -m 1 -t 10 в

качестве параметров, чтобы получить разные предложения.

Упражнение 8.13. Учитывая следующее дерево решений и таблицы для обучающих и тестовых данных:



Тренировочные данные

ABC-класс
ттфт
тфт
ттф
ффф
ffff
фттт

Тестовые данные

ABC-класс
ттфт
тфт
ftff
ффф

(a) Укажите правильность дерева для обучающих и тестовых данных. (b)

Приведите формулу пропозициональной логики, эквивалентную

дереву. (c) Проведите обрезку дерева, нарисуйте получившееся дерево и укажите его правильность.

для обучающих и тестовых данных.

Machine Translated by Google

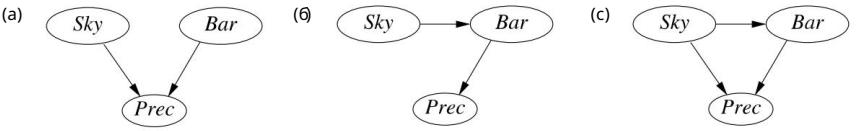
Упражнение 8.14

(a) При определении текущего атрибута алгоритм построения деревьев решений (рис. 8.26 на стр. 193) не исключает атрибуты, которые уже использовались выше по дереву. Несмотря на это, дискретный атрибут встречается в пути не более одного раза. Почему?

б) Почему непрерывные атрибуты могут встречаться несколько раз?

8.10.5 Изучение байесовских сетей

Упражнение 8.15. Используйте распределение, данное в упражнении 7.3 на стр. 158, и определите CPT для трех байесовских сетей:



(d) Определите распределение для двух сетей из (a) и (b) и сравните они с оригинальным дистрибутивом. Какая сеть «лучше»?

(e) Теперь определите распределение для сети (c). Что происходит с вами? Джустифай!

Упражнение 8.16. Покажите, что для бинарных переменных S1,...,Sn и бинарной переменной класса K линейная оценка вида

$$\text{решение} = \begin{cases} \text{положительно, если } w1S1 + \cdots + wnSn > \Theta, \text{ отрицательно} \\ \text{в противном случае} \end{cases}$$

одинаково выразительно по отношению к персептрону и наивному байесовскому классификатору, которые оба решают по формуле

$$\text{решение} = \begin{cases} \text{положительный, если } P(K|S1,...,Sn) > 1/2, \text{ отрицательный} \\ \text{в противном случае} \end{cases}$$

Упражнение 8.17. При реализации текстовой классификации с помощью наивного байесовского метода потеря значимости экспоненты может произойти быстро, потому что множители P (wi |K) (поя вляющиеся в (8.10) на стр. 206) обычно очень малы, что может привести к чрезвычайно малым значениям. Результаты Как мы можем смягчить эту проблему?

Упражнение 8.18. Напишите программу наивного байесовского анализа текста. Затем обучите и протестируйте его на текстовых тестах с помощью инструмента по вашему выбору. Подсчет частотыслов в тексте можно легко сделать в Linux с помощью команды

кошка <данные> | tr -d “[:punct:]” | tr -s «[:space:]» «\n» | сортировать | уникальный -ci

Machine Translated by Google

Получите данные Twenty Newsgroups Тома Митчелла в коллекции тестов машинного обучения UCI (репозиторий машинного обучения) [DNM98]. Там же вы найдете ссылку на наивную байесовскую программу для классификации текстов Митчелла.

8.10.6 Кластеризация

Упражнение 8.19. Покажите, что для алгоритмов, которые только сравнивают расстояния, применение строго монотонно возрастающей функции f к расстоянию не имеет значения. Другими словами, вы должны показать, что расстояние d1(x, y) и расстояние d2(x, y) := f (d1(x, y)) приводят к одному и тому же результату относительно отношения порядка.

Упражнение 8.20. Определите расстояния ds (скалярное произведение) следующих текстов друг от друга. x1: Мы

представим применение наивного Байеса к анализу текста в кратком

пример текста Алана Тьюринга из [Tur50].

x2: Мы можем надеяться, что машины со временем станут конкурировать с людьми во всех чисто интеллектуальных полях. Но с каких лучше начать?

x3: Опять же, я не знаю, каков правильный ответ, но я думаю, что оба подхода должны

пытался.

8.10.7 Интеллектуальный анализ данных

Упражнение 8.21. Используйте KNIME (www.knime.de) и (a)

Загрузите файл примера с данными Iris из каталога KNIME и поэкспериментируйте с различными

представления данных, особенно с диаграммами рассеяния.

(b) Сначала обучите дерево решений для трех классов, а затем обучите сеть RProp. (c) Загрузите данные об аппендиците на веб-сайт этой книги. Сравните качество классификации метода k

ближайших соседей с качеством сети RProp. Оптимизируйте k, а также количество скрытых

нейронов сети RProp. (d) Получите набор данных по вашему выбору из

коллекции данных UCI для интеллектуального анализа данных на http://kdd.ics.uci.edu или

для машинного обучения на http://mlearn.ics.uci.edu/MLRepository.html и поэкспериментируйте

с ним.

Machine Translated by Google

Нейронные сети

9

Нейронные сети представляют собой сети нервных клеток в мозге человека и животных. В человеческом мозгу около 100 миллиардов нервных клеток. Мы, люди, обязаны своим интеллектом и способностью обучаться различным двигательным и интеллектуальным способностям сложным механизмам и адаптивности мозга. На протяжении многих веков биологи, психологи и врачи пытались понять, как функционирует мозг. Примерно в 1900 году пришло революционное осознание того, что эти крошечные физические кирпичики мозга, нервные клетки и их связи, отвечают за осознание, ассоциации, мысли, сознание и способность к обучению.

Первый большой шаг к нейронным сетям в ИИ был сделан в 1943 году Мак Каллохом и Питтсом в статье под названием «Логическое исчисление идей, присущих нервной деятельности» [AR88]. Они первыми представили математическую модель нейрона как основного переключающего элемента мозга. Эта статья заложила основу для построения искусственных нейронных сетей и, следовательно, для этой очень важной области ИИ.

Мы могли бы рассматривать область моделирования и симуляции нейронных сетей как ветвь бионики[1] внутри ИИ. Почти все области ИИ пытаются воссоздать когнитивные процессы, такие как логика или вероятностные рассуждения. Однако инструменты, используемые для моделирования, а именно математика, язык программирования и цифровые компьютеры, имеют очень мало общего с человеческим мозгом. С искусственными нейронными сетями подход другой. Исходя из знаний о функциях естественных нейронных сетей, мы пытаемся моделировать, симулировать и даже реконструировать их в аппаратном обеспечении. Каждый исследователь в этой области сталкивается с увлекательной и захватывающей задачей сравнения результатов с человеческими способностями.

В этой главе мы попытаемся наметить историческую прогрессию, определив модель нейрона и его взаимосвязей, начиная с наиболее важных биологических открытий. Затем мы представим несколько важных и фундаментальных моделей: модель Хопфилда, две простые модели ассоциативной памяти и, что не менее важно, на практике алгоритм обратного распространения ошибки.

1Bionics занимается раскрытием «открытий живой природы» и ее инновационным преобразованием в технологии [Wik10].

В. Эртель, Введение в искусственный интеллект, Темы бакалавриата по компьютерным наукам, DOI 10.1007/978-0-85729-299-5_9, © Springer-Verlag London Limited 2011

Machine Translated by Google

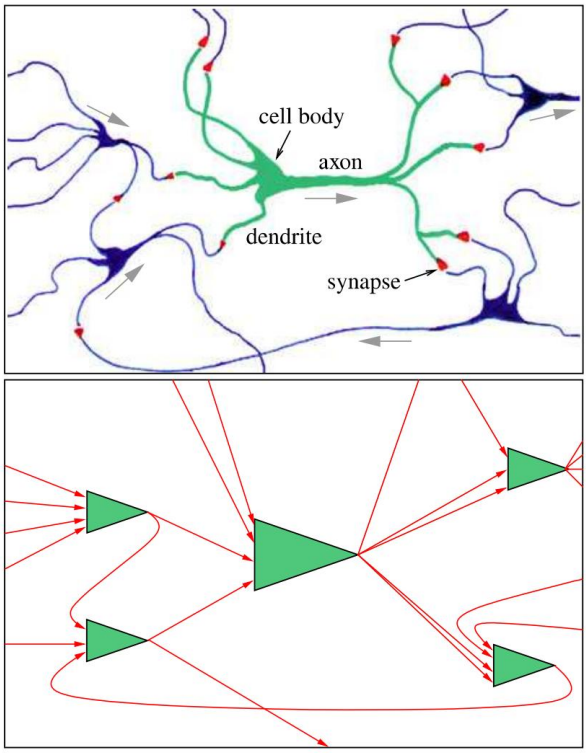


Рис. 9.1 Два этапа моделирования нейронной сети. Над биологической моделью под формальной моделью с нейронами и направленными связями между ними

9.1 От биологии к моделированию

Каждый из примерно 100 миллиардов нейронов в человеческом мозгу имеет, как показано в упрощенном представлении на рис. 9.1, следующую структуру и функцию Помимо тела клетки, у нейрона есть аксон, который может создавать локальные соединения с другими нейронами через дендриты Однако аксон может вырасти до метра в длину в виде нервного волокна, проходящего через тело.

Тело клетки нейрона может хранить небольшие электрические заряды подобно конденсатору или батарее. Это хранилище загружается поступающими электрическими импульсами от других нейронов. Чем больше поступает электрического импульса, тем выше напряжение. Если напряжение превышает определенный порог, нейрон срабатывает. Это означает, что он разгружает свой запас, посылая импульс по аксону и синапсам. Электрический ток делится и через синапсы достигает многих других нейронов, в которых происходит тот же процесс.

Теперь возникает вопрос о структуре нейронной сети. Каждый из примерно 1011 нейронов в мозгу связан примерно с 1000–10 000 других нейронов, что дает в общей сложности более 1014 соединений. Если мы далее учтем, что это гигантское количество через вышеупомянутых тонких соединений состоит из мягкой трехмерной ткани и что эксперименты над человеческим мозгом провести нелегко, то станет ясно, почему у нас нет подробной принципиальной схемы мозга. предположительно

Machine Translated by Google

мыник огда не сможем полность юпоня ть принц ипиаль нуюсхему нашего моз га, основьвая сь иск люн ит ель но на его огромных раз мерах.

С сегодня шней т оч к из рения уже не ст оит даже пыт ат ь ся сост авит ь полную принц ипиаль нуюсхему моз га, пот ому ч т о стр ук т ура моз га ад апт ивна. Он меня ет ся на лет у и приспособливает ся к дея т ель ност и ч еловек а и влия ниюок ружающей сред ы Ц ент раль ную роль з десь игр ают синапсы, соз д ающие свя з ь между нейронами. В т оч к е соед инения между двумя нейронами к ак быв ст реч ают ся два к абеля . Од нак о два вьвод а не обл ад ают ид еаль ной провод имост ь ю, ск орее имеет ся неболь шой з азор, ч ерез к от орый эл ек т роныне могут напря муюперепры нут ь . Эт от пробел з аполнен хим ич ск ими вещест вами, т ак наз ьваемыми нейрот рансмит т ерами. Они могут быт ь ионизированыприложенным напря жением, а з ат ем переносит ь з аря д ч ерез з азор. Провод имост ь эт ой щели з ависит от мног их парамет ров, например от к онц ент рац ии и хим ич ск ого сост ава нейрот рансмит т ера. Поуч ит ель но, ч т о функ ц ия моз га оч ень ч ут к о реаг ирует на из менения эт ой синапт ич ск ой свя з и, например, под влия ниемал к ого оля или д руг их нарк от ик ов.

К ак раб от ает обуч ение в т ак ой нейронной сет и? Уд ивит ель но з десь т о, ч т о ад апт ивны не сами ак т ивные ед иниц ы, а именно нейроны, а свя з и между ними, т о ест ь синапсы В ч аст ност и, эт о может изменит ь их провод имост ь . Мы з наем, ч т о синапс ст ановит ся силь нее от т ого, наск оль к о боль ший эл ек т рич ск ий т ок он д олжен нест и. Силь нее з десь оз нач ает , ч т о синапс имеет более выс ок уюпровод имост ь . Синапсы, к от орые исполь з ют ся , ч аст о получ ают все боль ший вес.
Для синапсов, к от орые исполь з ют ся неч аст о или вообще не ак т ивны, провод имост ь прод олж ает снижат ь ся . Эт о может д аже привест и к их от миранию

Все нейроныв моз гу раб от ают асинхронно и параллель но, но, по сравнениюс к омп ют ером, с оч ень низ к ой ск орост ь ю Время нервного имп ул са сост авля ет ок оло милл исек унд ы, ровно ст оль к о же, ск оль к о и время , необх од имое для т ранспорт а ионов ч ерез синапт ич ск ующель . В эт ом случ ае т ак т овая ч аст от а нейрона сост авля ет менее од ного к илогерц а и, т ак им образом, ниже, ч ем у современных к омп ют еров, в 106 раз . ст епень параллель ной обраб от к и в сет и нервных к лет ок .

Свя з ь с внешним миром осущест вля ет ся ч ерез ч увст вит ель нье нейроны, например, на сет ч ат к е глаза, или ч ерез нервные к лет к и с оч ень д линными ак сонами, к от орые т я нут ся от моз га к мышц ам, т ак им образом, могут вып олня т ь т ак ие дейст вия , к ак д вижение глаз а. ног а.

Од нак о д о сих пор нея сно, к ак обсужд аемые принц ипыд елают воз можным раз умное повед ение. К ак и мног ие исслед оват ели в обл аст и невр ологии, мыпыт аемся об ъя снит ь с помощь ю мод елирования прост ой мат емат ич ск ой мод ели, к ак ст ановя т ся воз можными к ог нит ивные з ад ач и, например распоз навание образ ов.

9.1.1 Мат емат ич ск ая мод ель

Снач ала мы з аменим непрерывнуюось времени ш к ал ой д иск рет ног о времени. Нейрон і вып олня ет след ующие вын исления на временном шаг е. «Заг руз к а» ак т иви-

Machine Translated by Google

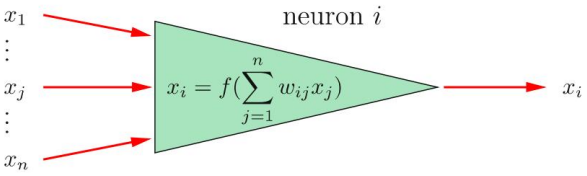


Рис. 9.2 Структура формального нейрона, который применяет функцию активации f к взвешенной сумме всех входов

потенциал достигается простым суммированием взвешенных выходных значений x1,...,xn всех входящих соединений по формуле

н

вид жхдж.

j=1

Эта взвешенная сумма рассчитывается большинством нейронных моделей. Затем к нему применяется функция активации f и результат

н

хи = е wij xj

j=1

передается соседним нейронам в виде выходных данных по синаптическим весам. На рис. 9.2 показан такой смоделированный нейрон. Для функции активации существует ряд возможностей. Самым простым является тождество: f(x) = x. Таким образом, нейрон вычисляет только взвешенную сумму входных значений и передает ее дальше. Однако это часто приводит к проблемам сходимости с нейронной динамикой, потому что функция f (x) = x неограничена, и значения функции могут выйти за все пределы с течением времени.

Напротив, очень хорошо ограничена пороговая функция (ступенчатая функция Хевисайда).

$$H(x) = \begin{cases} 0, & \text{если } x<\Theta, \\ 1 & \text{в противном случае.} \end{cases}$$

Затем весь нейрон вычисляет свой вывод как

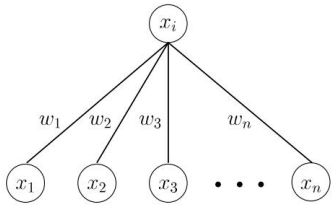
$$x_i = \begin{cases} 0 & \text{если } \sum_{j=1}^{n} w_{ij} x_j < \Theta, \\ 1 & \text{еще.} \end{cases}$$

Эта формула идентична (8.1) на стр. 173, другими словами, персептрону с порогом Θ (рис. 9.3 на стр. 225). Входные нейроны 1,...,n здесь имеют только функцию переменных, которые передают свои внешние заданные значения x1,...,xn без изменения.

Ступенчатая функция вполне разумна для бинарных нейронов, потому что активация нейрона в любом случае может принимать только значения ноль или единицу. Напротив, для непрерывного

Machine Translated by Google

Рис. 9.3 Нейрон со ступенчатой функцией работает как персептрон с порогом



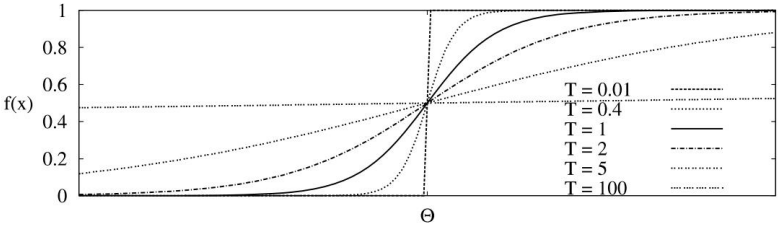


Рис. 9.4. Сигмовидная функция для различных значений параметра TT 0. . Мы видим, что в пределе Результаты ступенчатой функции

нейронов с активациями от 0 до 1, ступенчатая функция создает разрыв. Однако это можно сгладить сигмовидной функцией, такой как

$$\phi(x) = \frac{1}{1 + e^{\frac{x-\Theta}{T}}}$$

с графиком на рис. 9.4. Вблизи критической области около порога Θ эта функция ведет себя близко к линейной и имеет асимптотический предел. Сглаживание можно варьировать параметром T .

Моделирование обучения занимает центральное место в теории нейронных сетей. Как упоминалось ранее, одна из возможностей обучения состоит в укреплении синапса в зависимости от того, сколько электрических импульсов он должен передать . Этот принцип был постулирован Д. Хеббом в 1949 г. и известен как правило Хебба:

> Если между нейроном j и нейроном i существует связь wij и повторяющиеся сигналы посылаются от нейрона j к нейрону i, что приводит к одновременной активности обоих нейронов, то вес wij усиливается . Возможная формула для изменения веса wij :
>
> $$wij = \eta xixj$$
>
> с константой η (скорость юобучения), которая определяет размер отдельных шагов обучения .

Machine Translated by Google

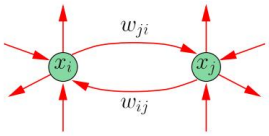


Рис. 9.5 Рекуррентные связи между двумя нейронами в сети Хопфилда

Существует множество модификаций этого правила, результатом которых становятся различные типы сетей или алгоритмов обучения. В следующих разделах мы познакомимся с некоторыми из них.

9.2 Сети Хопфилда

Глядя на правило Хебба, мы видим, что для нейронов со значениями от нуля до единицы веса могут только расти со временем. В соответствии с этим правилом нейрон не может ослабнуть или даже умереть. Это можно смоделировать, например, с помощью юконстанты затухания, которая ослабляет неиспользованный вес на постоянный коэффициент за временной шаг, например 0,99.

Совершенно иначе эта проблема решается в модели, представленной Хопфилдом в 1982 г. [Нор82]. Он использует бинарные нейроны, но с двумя значениями -1 для неактивных и 1 для активных. Используя правило Хебба, мы получаем положительный вклад в вес, когда два нейрона активны одновременно. Однако, если активен только один из двух нейронов, w_{ij} отрицателен.

На этой идее основана сеть Хопфилда, которые представляют собой прекрасный и наглядный пример автоассоциативной памяти. Паттерны могут храниться в автоассоциативной памяти. Для вызова сохраненного шаблона достаточно предоставить аналогичный шаблон. Затем магазин находит наиболее похожий сохраненный шаблон. Классическим применением этого является распознавание рукописного ввода.

На этапе обучения сети Хопфилда N двоично-кодированных шаблонов, сохраненных в векторы $q^1, \ldots, q^N$, должны быть изучены. Каждая компонента $q^k_j \in \{-1, 1\}$ вектора q^k представляет собой пиксель шаблона. Для векторов, состоящих из n пикселей, используется нейронная сеть с n нейронами, по одному на каждую позицию пиксель. Нейроны полностью связаны тем ограничением, что матрица весов симметрична и все диагональные элементы w_{ii} равны нулю. То есть между нейроном и самим собой нет никакой связи.

Полносвязная сеть включает в себя сложные петли обратной связи, так называемые рекуррентные сети (рис. 9.5).

N паттернов можно выучить, просто рассчитав все веса по формуле

$$w_{ij} = \frac{1}{n} \sum_{k=1}^{N} q^k_i q^k_j. \tag{9.1}$$

Эта формула указывает на интересную связь с правилом Хебба. Каждый шаблон, в котором пиксели i и j имеют одинаковое значение, вносит положительный вклад в вес w_{ij}.

Каждый другой паттерн вносит отрицательный вклад. Поскольку у каждого пикселя

Machine Translated by Google

соответствует нейрону, здесь усиливаются веса между нейронами, которые одновременно имеют одинаковое значение. Обратите внимание на это небольшое отличие от правила Хебба.

После сохранения всех шаблонов сеть можно использовать для распознавания образов. Задаем сети новый паттерн x и обновляем активации всех нейронов в асинхронном процессе по правилу

$$x_i = \begin{cases} 1, & \text{если } \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} x_j < 0, \\ 1 & \text{еще} \end{cases} \tag{9.2}$$

пока сеть не станет стабильной, то есть пока не перестанут меняться активации. В виде схемы программы это выглядит следующим образом:

```
ХОПФИЛДАССОЦИАТОР(q)
Инициализировать все нейроны
x = q
    Repeat i = Random(1, n)
    Обновить нейрон i согласно (9.2)
Пока x не сходится
Возврат (x)
```

9.2.1 Применение к примеру распознавания образов

Применим описанный алгоритм к простому примеру распознавания образов. Он должен распознавать цифры в поле размером 10× 10 пикселей. Таким образом, сеть Хопфилда состоит из 100 нейронов с общим количеством

$$\frac{100 \cdot 99}{2} = 4950$$

веса. Сначала обучаются шаблоны цифр 1, 2, 3, 4 на рис. 9.6 на стр. 228 выше. То есть веса рассчитываются по (9.1) на стр. 226. Затем мы вставляем шаблон с добавленным шумом и позволяем динамике Хопфилда работать до сходимости.
В строках со 2 по 4 на рисунке показаны пять снимков развития сети в процессе распознавания. При уровне шума 10 % все четыре выученных паттерна распознаются очень надежно. Шум выше примерно 20% алгоритм часто сходится к другим изученным шаблонам или даже к шаблонам, которые не были изучены. Несколько таких паттернов показаны на рис. 9.6 на стр. 228 ниже.

Теперь мы сохраняем цифры от 0 до 9 (рис. 9.7 на стр. 229 вверху) в той же сети и снова тестируем сеть с шаблонами, которые имеют случайное количество инвертированных пикселей около 10%. На рисунке мы сно видим, что итерация Хопфилда часто не сходится к наиболее похожему обученному состоянию даже для всего 10% шума. Очевидно

Machine Translated by Google

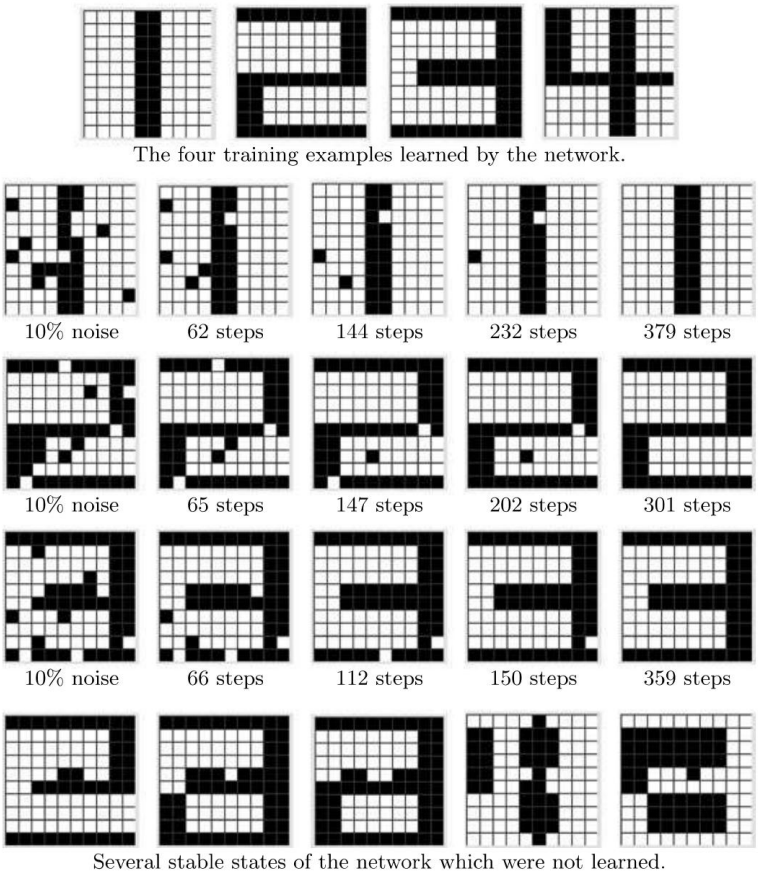


Рис. 9.6 Динамика сети Хопфилда. В строках 2, 3 и 4 мы можем легко увидеть, как сеть сходится, и изученный шаблон распознается примерно после 300–400 итераций. В последней строке показаны несколько устойчивых состояний, которые достигаются сетью, когда входной шаблон слишком сильно отклоняется от всех изученных шаблонов.

сеть может надежно сохранить и распознать четыре шаблона, но для десяти шаблонов ее объем памяти превышен. Чтобы лучше понять это, мы кратко рассмотрим теорию этой сети.

9.2.2 Анализ

В 1982 году Джон Хопфилд показал в [Hop82], что эта модель формально эквивалентна физической модели магнетизма. Маленькие элементарные магниты, так называемые спины, взаимно влияют друг на друга своими магнитными полями (см. рис. 9.8 на стр. 229). Если мы

Machine Translated by Google

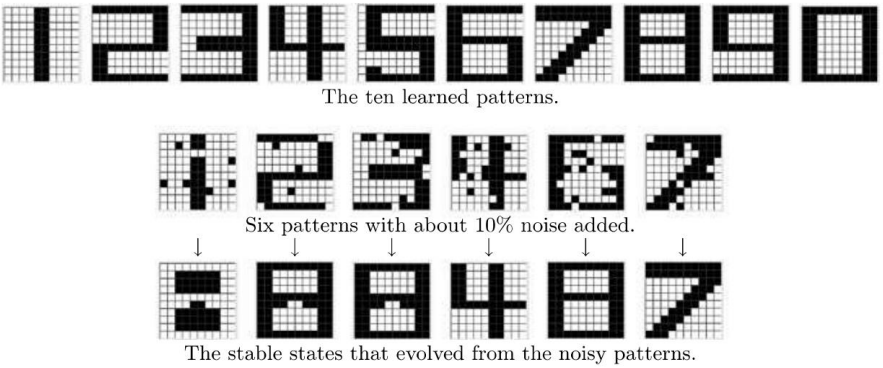


Рис. 9.7. Для десяти изученных состояний сеть показывает хаотичное поведение. Даже при небольшом шуме сеть сходится к неправильным паттернам или к артефактам

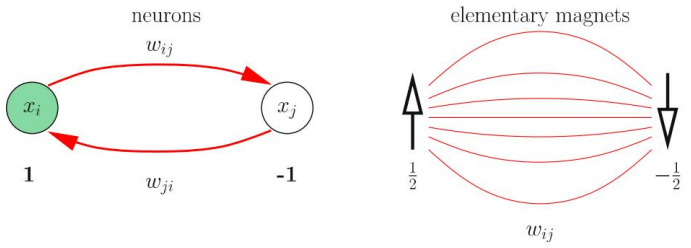


Рис. 9.8 Сравнение нейронной и физической интерпретации модели Хопфилда

наблюдаем два таких спина i и j , они взаимодействуют с постоянной величиной wij , и тогда полная энергия системы равна

$$E = -2 \frac{1}{\text{я,дж}} \text{вид}_{\text{ж}} x_i x_j .$$

К статье, и в физике wii = 0, потому что частицы не взаимодействуют друг с другом. Поскольку у физических взаимодействия симметричны, wij = wj i.

Физическая система, находящаяся в равновесии, переходит в (устойчивое) состояние с минимальной энергией и, таким образом, минимизирует E(x, y). Если такую систему привести в произвольное состояние, то она движется к состоянию минимальной энергии. Динамика Хопфилда, определенная в (9.2) на стр. 227 , точно соответствует этому принципу, потому что она обновляет состояние на каждой итерации таким образом, что из двух состояний 1 и 1 принимается то, которое имеет меньшую полную энергию Вклад нейрона i в общую энергию равен

$$\frac{1}{2} x_i \sum_{\text{дж=я}}^{\text{н}} \text{вид}_{\text{ж}} x_{\text{дж}} .$$

Machine Translated by Google

Если сейчас

$$\sum_{j=1}^{n} w_{ij} x_j < 0,$$

тогда $x_i = -1$ приводит к отрицательному вкладу в полную энергию, а $x_i = 1$ дает положительный вклад. При $x_i = -1$ сеть переходит в состояние с меньшей энергией, чем при $x_i = 1$. Аналогично можно утверждать, что в случае

$$\sum_{j=1}^{n} w_{ij} x_j \geq 0,$$

должно быть верно, что $x_i = 1$.

Если каждая отдельная итерация нейронной динамики приводит к уменьшению функции энергии, то полная энергия системы монотонно уменьшается со временем. Поскольку существует только конечное число состояний, сеть со временем переходит в состояние с минимальной энергией. Теперь у нас есть волнующий вопрос: что означают эти минимумы функции энергии?

Как мы видели в эксперименте по распознаванию образов, в случае нескольких выученных образов система сходится к одному из выученных образов. Выученные паттерны представляют собой минимумы энергетической функции в пространстве состояний. Однако если усвоено слишком много паттернов, то система сходится к минимумам, которые не соответствуют выученным паттернам. Здесь мы имеем переход от упорядоченной динамики к хаотической. один.

Хопфилд и другие физики исследовали именно этот процесс и показали, что фазовый переход действительно имеет место при критическом числе выученных паттернов.
Если количество выученных паттернов превышает это значение, то система переходит из упорядоченной фазы в хаотичную

В магнитной физике происходит такой переход от ферромагнитного режима, при котором все элементарные магнитыстремятся ориентироваться параллельно, к так называемому спиновому стеклу, в котором спины взаимодействуют хаотически. Более наглядным примером такого физического фазового перехода является таяние кристалла льда. Кристалл находится в состоянии высокого порядка, поскольку у молекулы H_2O строго упорядочены В жидкой воде, напротив, структура молекул растворена, а их положение более хаотично.

Затем в нейронной сети происходит фазовый переход от упорядоченного обучения и распознавания паттернов к хаотическому обучению и случае слишком большого количества паттернов, которые уже нельзя распознать наверняка. Здесь мы определенно видим параллели с эффектами, которые мы ранее испытывали сами.

Мы можем понять этот фазовый переход [RMS92], если мы поместим все нейроны а и вставим состояние шаблона, например q^1, из изученные веса из (9.1) на странице 226 , что актуально в терминах для обновления нейрона i. Это приводит к $\sum_{j=1, j\neq i}^{n} w_{ij} q_j$, в

Machine Translated by Google

$$\sum_{j=1,\, j\neq i}^{N} w_{ij} q_j \overset{\text{"="}}{=} \frac{1}{n} \sum_{j=1}^{N} \sum_{k=1,\, j\neq i}^{n} \xi_i^k \xi_j^k q_j \overset{\text{"="}}{=} \frac{1}{n} \sum_{j=1,\, j\neq i}^{N} \sum_{k=2}^{n} \xi_i^k \xi_j^k q_j \quad (\text{поскольку } \xi_i^1 \xi_j^1 q_j \approx q_i)$$

$$= q_i \left(1 + \frac{1}{n} \sum_{j=1}^{N} \sum_{k=2}^{n} \xi_i^k \xi_j^k q_j\right).$$

Здесь мы видим i-й компонент входного шаблона плюс сумму с (n 1)(N 1) элементами. Если все эти слагаемые статистически независимы, то мы можем описать сумму нормально распределенной случайной величиной со стандартным отклонением

$$\frac{1}{n} \sqrt{(n-1)(N-1)} \approx \sqrt{\frac{N-1}{n}}.$$

Статистическая независимость может быть достигнута, например, с помощью некоррелированных случайных закономерностей. Затем сумма генерирует шум, который является разрушительным до тех пор, пока N n, а это означает, что количество выученных паттернов остается намного меньше, чем количество нейронов. Если же N n, то влияние шума становится таким же большим, как и паттерн, и сеть реагирует хаотично. Более точный расчет фазового перехода дает критической точкой N = 0,146 n. Применительно к нашему примеру это означает, что для 100 нейронов можно сохранить до 14 паттернов. Однако, поскольку у закономерности в примере сильно коррелированы, критическое значение намного ниже, очевидно, между 0,04 и 0,1. Даже значение 0,146 намного меньше, чем емкость памяти традиционного списка памяти (упражнение 9.3 на стр. 255).

Сети Хопфилда в представленной форме хорошо работают только тогда, когда изученные шаблоны содержащие примерно 50% 1-битов. Если биты распределены очень асимметрично, то нейроны должны быть снабжены порогом [Roj96]. В физике это аналогично приложению внешнего магнитного поля, которое также вызывает асимметрию состояний со спином 1/2 и со спином -1/2.

9.2.3 Резюме и перспективы

Благодаря своему биологически правдоподобию, хорошо понятной математической модели и, прежде всего, впечатляющим симуляциям распознавания образов, модель Хопфилда способствовала волне интереса к нейронным сетям и росту нейроинформатики как важной ветви ИИ.2 В последствии было разработано много других сетевых моделей. С одной стороны, были исследованы сети без обратной связи, потому что их динамика значительно легче для понимания, чем рекуррентные сети Хопфилда. С другой стороны, были предприняты попытки улучшить емкость хранения данных в сетях, о чем мы поговорим в следующем разделе.

2Даже автора подхватила эта волна, унесшая его из физики в ИИ в 1987 году.

Machine Translated by Google

Особая проблема многих нейронных моделей уже проявилась в модели Хопфилда. Даже если существует гарантия сходимости, неясно, сойдется ли сеть в желаемое состояние или застрянет на локальном минимуме. Машина Больцмана с непрерывными значениями активации и вероятностным правилом обновления для ее сетевой динамики была разработана как попытка решить эту проблему. Используя параметр «температура», мы можем варьировать количество случайных изменений состояния и, таким образом, пытаться избежать локальных минимумов с целью найти устойчивый глобальный минимум. Этот алгоритм называется «имитация отжига». Отжиг — это процесс термической обработки металлов с целью сделать металл более прочным и «стабильным».

Модель Хопфилда осуществляет поиск минимума функции энергии в пространстве значений активации. Таким образом, он находит образец, сохраненный в весах и, таким образом, представленный в функции энергии. Динамика Хопфилда также может быть применена к другим функциям энергии, если матрица весов симметрична, а диагональные элементы равны нулю. Это было успешно продемонстрировано Хопфилдом и Танком на задаче коммивояжера [HT85, Zel94]. Задача здесь состоит в том, чтобы, зная n городов и их матрицу расстояний, найти кратчайший путь туда и обратно, который посещает каждый город ровно один раз.

9.3 Нейронная ассоциативная память

Традиционная память списка может в простейшем случае быть текстовым файлом, в котором строки цифр сохраняются строка за строкой. Если файл отсортирован построчно, то поиск элемента можно осуществить очень быстро за логарифмическое время даже для очень больших файлов.

Однако память списков также может использоваться для создания отображений. Например, телефонная книга представляет собой отображение набора всех введенных имен на набор всех телефонных номеров. Это сопоставление реализовано в виде простой таблицы, обычно сохраняемой в базе данных.

Аналогичная задача — контроль доступа в здание с помощью распознавания лиц. Здесь мы также могли бы использовать базу данных, в которой сохраняется фотография каждого человека вместе с именем человека и, возможно, другими данными. Затем камера на входе делает снимок человека и ищет в базе данных идентичную фотографию. Если фото найдено, то человек идентифицируется и получает доступ в здание. Однако в здании с такой системой управления не будет много посетителей, потому что вероятность того, что текущая фотография в точности совпадает с сохраненной фотографией, очень мала.

В этом случае недостаточно просто сохранить фото в таблице. Скорее, нам нужна ассоциативная память, способная не только присвоить нужное имя фотографии, но и любой из потенциально бесконечного набора «похожих» фотографий. Функциональность поиска сходства необходимо сгенерировать из конечного набора обучающих данных, а именно сохраненных фотографий, помеченных именами. Простым подходом для этого является метод ближайшего соседа, представленный в разд. 8.3. Во время обучения все фотографии просто сохраняются.

Для применения этой функции в базе данных должна быть найдена фотография, наиболее похожая на текущую. Для баз данных с большим количеством фотографий высокого разрешения этот процесс, в зависимости от используемой метрики расстояния, может потребовать очень большого времени вычислений и

Machine Translated by Google

поэтому не может быть реализован в этой простой форме. Поэтому вместо такого ленивого алгоритма мыпредпочтем тот, который передает данные в функцию, которая затем создает очень быструюассоциацию при ее применении.

Поиск подходящей метрики расстояния представляет собой еще одну проблему. Мыбы хотели, чтобыч еловека узнавали, даже если его лицо поя вляется в другом месте на фото (переводе), или если оно меньше, больше или даже повернуто. Уголобзора и освещение также могут различаться .

Именно здесь нейронные сети показывают свои сильные стороны Не требуя от разработчика думать о подходящей метрике сходства, они по-прежнему дают хорошие результаты Мыпредставим две простейшие модели ассоциативной памяти и начнем с модели Теувоо Кохонена, одного из пионеров в этой области.

Модель Хопфилда, представленнуюв предыдущей главе, было быслишком сложно использовать по двум причинам. Во-первых, это лишь автоассоциативная память, т. е. примерно тождественное отображение, сопоставляющее сходные объекты с выученным оригиналом. Во-вторых, сложной рекуррентной динамикой часто трудно управлять на практике. Поэтому сейчас мырассмотрим простые двухуровневые сети с прямой связью

9.3.1 Память корреляционной матрицы

В [Koh72] Кохонен представил ассоциативнуюмодель памяти, основаннуюна элементарной линейной алгебре. Это отображает векторызапроса $x^{\mu} \in \mathbb{R}^n$ в векторырезультатов $y \in \mathbb{R}^m$. Мы ищем матрицу W, которая правильно отображает набор обучающих данных.

$$T = \{(q^1, t^1), \ldots, (q^N, t^N)\}$$

с N запрос-ответ связывает все векторызапросов с их ответами.[3] То есть для $p = 1,\ldots,N$ должно быть так, что

$$t^p = W \cdot q^p, \tag{9.3}$$

или

$$t_i^p = \sum_{j=1}^{n} w_{ij} q_j^p \tag{9.4}$$

Для вычисления матричных элементов w_{ij} используется правило

$$w_{ij} = \sum_{p=1}^{N} t_i^p q_j^p \tag{9.5}$$

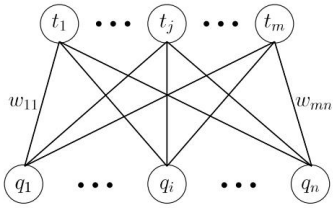
---

[3] Для четкого разграничения между обучающими данными и другими значениями нейрона в последующем обсуждении мыбудем называть вектор запроса q, а желаемый ответ — t (цель).

Machine Translated by Google

Рис. 9.9 Представление ассоциативной памяти

Кохонена в виде двухслойной нейронной сети



использует. Эти два линейных уравнения можно просто понимать как нейронную сеть, если мы определим, как на рис. 9.9, двухслойную сеть с q в качестве входного слоя и t в качестве выходного слоя. Нейроны выходного слоя имеют линейную функцию активации, а в качестве правила обучения используется (9.5), что в точности соответствует правилу Хебба.

Прежде чем мы покажем, что сеть распознает обучающие данные, нам понадобится следующее определение:

Определение 9.1 Два вектора x и y называются ортонормированными, если

$$T\ x \cdot y = \begin{cases} 1, \text{если } x = y, \\ 0 \text{ в противном случае.} \end{cases}$$

Таким образом

Теорема 9.1. Если все N векторов запроса q^p в обучающих данных ортонормированы, то каждый вектор q^p отображается в целевой вектор t^p путем умножения на матрицу w_{ij} из (9.5) на стр. 233.

Доказательство. Подставим (9.5) на стр. 233 в (9.4) на стр. 233 и получим

$$\begin{aligned}
(Вт \cdot к в^{п})_{я} &= \sum_{jj=1}^{н} w_{ij} q_{j}^{п} \\
&= \sum_{j=1}^{н} q_{j}^{п} \left(\sum_{г=p}^{Н} p_{кj} \right)_{i}^{p} q_{j}^{p} \\
&= \sum_{j=1}^{н} q_{j}^{п} \left(q_{j}^{п} + \sum_{г=p}^{Н} q_{j}^{п} \right)_{т_{я}}^{p} \\
&= т_{я}^{p} \sum_{j=1}^{н} д_{jq}^{п} j_{-}^{p} + \sum_{г=p}^{Н} т_{я}^{p} \sum_{j=1}^{н} д_{jq}^{p} j_{-}^{п} \\
&= т_{я}^{p} \underbrace{(q^{п})^{T}}_{=1} \underbrace{п q}_{} + т_{я}^{p} \sum_{г=p}^{Н} \underbrace{(q^{p})^{T}}_{=0} \underbrace{pp q}_{} = t_{i}
\end{aligned}$$

Machine Translated by Google

Таким образом, если векторы запроса ортонормированы, все шаблоны будут правильно сопоставлены с соответствующими целями. Однако ортонормированность является слишком сильным ограничением. В разд. 9.4 мы представим подход, преодолевающий это ограничение.

Поскольку линейные отображения непрерывны и инъективны, мы знаем, что отображение из векторов запроса в целевые векторы сохраняет сходство. Таким образом, аналогичные запросы сопоставляются с аналогичными целями из-за непрерывности. Однако в то же время мы знаем, что разные запросы сопоставляются с разными целями. Если сеть была обучена отображать лица в имена и если лицу присвоено имя Генри, то мы уверены, что на вход похожего лица будет выдан аналогичный «Генри», но сам «Генри» гарантированно не вычисляется. Если вывод можно интерпретировать как строку, то, например, это может быть «Genry» или «Gfnry». Чтобы получить наиболее похожий заученный случай, Кохонен использует двоичное кодирование для выходного нейрона. Вычисленный результат запроса округляется, если его значение не равно нулю или единице. Даже тогда у нас нет гарантии, что мы попадем в целевой вектор. В качестве альтернативы мы могли бы добавить последующее сопоставление вычисленного ответа с изученным целевым вектором с наименьшим расстоянием.

9.3.2 Псевдоинверсия

Существует другой подход, который мы можем использовать для вычисления весовой матрицы W: 1 думая обо всех векторах запросов как о столбцах матрицы $Q = (q_1 | \ldots | q_N)$ и аналогично целевые векторы как столбцы матрицы $T = (t_1 | \ldots | t_N)$, мы можем привести (9.3) на стр. 233 к форме

$$T = W \cdot Q. \tag{9.6}$$

Теперь попытаемся решить это уравнение относительно W. Формально инвертируем и получаем

$$W = T \cdot Q^{-1}. \tag{9.7}$$

Требованием для этого преобразования является обратимость Q. Для этого матрица должна быть квадратной и должна состоять из линейно независимых вектор-столбцов. То есть должно быть так, что $n = N$ и все n векторов запросов должны быть линейно независимыми. Это условие действительно неудобно, но все же не столь строго, как требуемая выше ортонормированность. Однако обучение по правилу Хебба имеет то преимущество, что мы можем просто применить его, даже если векторы запросов не являются ортонормированными, в надежде, что ассоциатор, тем не менее, будет работать сносно. Однако здесь это не так просто, ибо как инвертировать необратимую матрицу?

Матрица Q обратима тогда и только тогда, когда существует матрица Q^{-1} со свойством

$$Q \cdot Q^{-1} = I, \tag{9.8}$$

где I — единичная матрица. Если Q необратима (например, потому что Q неквадратная), то не существует матрицы Q^{-1} с таким свойством. Однако определенно существует матрица, которая близка к этому свойству. В этом смысле мы определяем

Machine Translated by Google

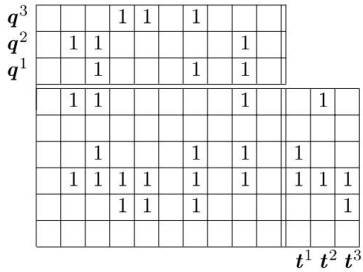


Рис. 9.10. Матрица W после сохранения трех пар $(q^1, t), (q^2, t), (q^3, t)$. Пустые поля соответствуют значению 0

Определение 9.2. Пусть Q — вещественная матрица размера $n \times m$. Матрица размера $m \times n$ Q^+ называется псевдообратной к Q, если она минимизирует

$$\|Q \cdot Q^+ - I\|.$$

Здесь $\|M\|$ — евклидова норма.

Теперь, используя

$$W = T \cdot Q^+ \tag{9.9}$$

мы можем рассчитать весовую матрицу, которая минимизирует перекрестные помехи (ошибки ассоциации) $\|T - W \cdot Q\|$. Существуют различные способы вычисления псевдообратного, например метод наименьших квадратов, который мы введем в разд. 9.4.1.

9.3.3 Бинарное правило Хебба

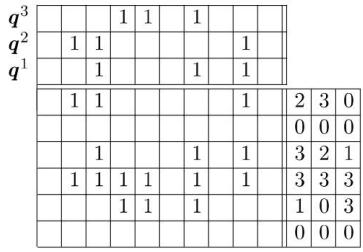
В контексте ассоциативной памяти было предложено так называемое бинарное правило Хебба. Это требует, чтобы шаблон был закодирован в двоичном виде. Это означает, что для всех образов q^p и t^p $\in \{0, 1\}^m$. Кроме того, сумма из (9.5) на стр. 233 равна $\{0, 1\}$ заменяется простым логическим ИЛИ, и мы получаем бинарное правило Хебба

$$w_{ij} = \bigvee_{p=1}^{N} t_i^p \wedge q_j^p. \tag{9.10}$$

Таким образом, весовая матрица также является двоичной, и матричный элемент w_{ij} равен единице, если он и только если хотя бы один из элементов $q_j^1, \ldots, q_j^N$ не равен нулю. Все остальные матрицы элементы равны нулю. Мысленно полагать, что при обучении здесь теряется много информации, потому что, когда элемент матрицы принимает значение 1 один раз, его нельзя изменить дополнительными шаблонами. На рис. 9.10 показано, как матрица заполняется единицами для примера с $n = 10$, $m = 6$ после изучения трех пар.

Machine Translated by Google

Рис. 9.11 Расчет произведения W q1 , W q2 , W q3



Чтобы получить сохраненные шаблоны, мы просто умножаем вектор запроса q на матрицу и посмотрите на результат W q. Проверим это на примере и получим рис. 9.11.

Мы видим, что в целевом векторе с правой стороны стоит значение 3 в том месте, где у изученного целевого вектора была единица. Правильные результаты будут получены при установке порогового значения 3. В общем случае мы выбираем количество единиц в векторе запроса в качестве порога. Таким образом, каждый выходной нейрон работает как персептрон, хотя и с переменным порогом.

Пока матрица весов разрежена, этот алгоритм работает хорошо. Однако, если сохраняется много разных паттернов, матрица становится все более и более плотной. В крайнем случае он содержит только единицы. Тогда, после установки порога, все ответы будут состоять только из единиц и больше не будут содержать никакой информации.

Этот случай возникает редко, пока количество битов, сохраняемых в матрице, не становится слишком большим. Матрица имеет размер mn элементов. Сохраняемая пара имеет m + n бит . Можно показать [Pal80], что количество запоминаемых паттернов Nmax определяется следующим условием:

$$\alpha = \frac{\text{количество сохраняемых битов}}{\text{количество элементов бинарной матрицы}}$$

$$\text{"="} \frac{(m+n)N_{max}}{mn}\ln 2 \quad 0{,}69. \tag{9.11}$$

Для памяти списка мы имеем α = 1. Ассоциативная память с двоичным правилом Хебба имеет максимальную эффективность памяти α = 0,69 по сравнению с α = 0,72 для ассоциативной памяти Кохонена и α = 0,292 для сетей Хопфилда [Pal80, Pal91]. Таким образом, емкость памяти бинарного правила Хебба на удивление высока по сравнению с моделью Кохонена с непрерывными нейронами.

Очевидно, что такая память «заполняется» менее быстро, когда вектор запроса и целевой вектор заполнены единицами редко. Это не единственная причина, по которой кодирование входных и выходных данных для ассоциативной памяти, как и для других нейронных сетей, очень важно для хорошей работы. Теперь мы продемонстрируем это на приложении этой памяти с соответствующим образом выбранными кодировками ввода и вывода.

Machine Translated by Google

9.3.4 Программа исправления правописания

В качестве приложения описанной ассоциативной памяти с бинарным правилом Хебба выберем программу, исправляющую ошибочные входные данные и отображающую их в сохраненные слова из словаря. Очевидно, что здесь потребуется автоассоциативная память. Однако, поскольку мы будем кодировать вектор запроса и целевой вектор по-разному, дело обстоит иначе. Для векторов запроса q мы выбираем парное кодирование. В алфавите из 26 символов имеется $26 \cdot 26 = 676$ упорядоченных пар букв. С 676 битами вектор запроса имеет по одному биту для каждой из возможных пар.

$$aa, ab, \ldots, az, ba, \ldots, bz, \ldots, za, \ldots, zz.$$

Если в слове встречается пара букв, то в соответствующее место будет вставлена единица. Например, для слова «генри» места для «га», «ан» и «нс» заполнены единицами. Для целевого вектора t зарезервировано 26 битов для каждой позиции в слове до максимальной длины (например, десять символов). Для i-й буквы алфавита на позиции j в слове устанавливается число битов $(j - 1) \cdot 26 + i$. Для слова «генри» установлены биты 8, 27, 66 и 97. Таким образом, для максимум 10 букв в слове целевой вектор имеет длину 260 бит.

Весовая матрица W, таким образом, имеет размер $676 \cdot 260$ битов = 199420 битов, что (9.11) на стр. 237 может хранить не более

$$n_{max} \approx 0{,}69 \frac{m_{in}}{130} - 676 + \frac{676 \cdot 260}{130 \cdot (m+n)}$$

слова. Имея 72 имени, мы экономим примерно половину этого количества и тестируем систему. Сохраненные имена и выходные данные программы для нескольких примеров ввода приведены на рис. 9.12 на стр. 239. Порог всегда инициализируется числом битов в закодированном запросе. Здесь это количество пар букв, следовательно, длина слова минус один. Затем его ступенчато уменьшают до двух. Мы могли бы дополнительно автоматизировать выбор порога, сравнивая со словарем каждую попытку порога и выводя найденное слово, когда сравнение завершается успешно.

Интересна реакция на двусмысленные входные данные «andr» и «johanne». В обоих случаях сеть создает сочетание двух подходящих сохраненных слов. Мы видим здесь важную силу нейронных сетей. Они способны создавать ассоциации с похожими объектами без явной метрики сходства. Однако, как и в случае с эвристическим поиском и принятием решений человеком, нет гарантии «правильного» решения.

Поскольку обучающие данные должны быть доступны в виде пар ввода-вывода для всех нейронных моделей, которые были введены до сих пор, мы имеем дело с обучением с учителем, что также имеет место для сетей, представленных в следующих разделах.

Machine Translated by Google

Сохраненные слова:

Агата, Агнес, Александр, Андреас, Андре, Анна, Аннемари, Аструд, Август, Бернхард, Бьорн, Катрин, Кристиан, Кристоф, Коринна, Корадо, Дитер, Элизабет, Эльвира, Эрдмут, Эрнст, Эвелин, Фабрицио, Франк, Франц, Джеффри, Георг, Герхард, Ханнелоре, Гарри, Герберт, Ингильт, Ирмгард, Ян, Йоханнес, Джонни, Юрген, Карин, Клаус, Людвиг, Луиза, Манфред, Мария, Марк, Маркус, Марлин, Мартин, Маттиас, Норберт, Отто, патриция, питер, филип, бросить, рейнхольд, рената, роберт, робин, сабина, себастьян, стефан, стефан, сильви, ульрих, ульрике, уте, уве, вернер, вольфганг, ксавьер

Ассоциации программы

шаблон ввода: Гарри

порог: 4, ответ: порог Гарри: 3, ответ: порог Гарри: 2, ответ: horryrrde

---

шаблон ввода: уте

порог: 2, ответ: уте

шаблон ввода: герхар

порог: 5, ответ: gerhard порог: 4, ответ: gerrarn
порог: 3, ответ: jerrrrd порог: 2, ответ: jurtyrrde

шаблон ввода: эгрхард

порог: 6, ответ: порог: 5, ответ:
порог: 4, ответ: gerhard порог: 3,
ответ: gernhrrd порог: 2, ответ: irrryrrde

шаблон ввода: andr

порог: 3, ответ: andrees порог: 2, ответ: anexenser

шаблон ввода: Андреес

порог: 6, ответ: порог: 5, ответ: порог
andree: 4, ответ: порог andrees: 3, ответ: mnnrens
порог: 2, ответ: morxsnssr

шаблон ввода: Джоанна

порог: 6, ответ: johnnnes порог: 5, ответ: johnnns
порог: 4, ответ: jornnnrse порог: 3, ответ: sorrnyrse
порог: 2, ответ: wtrrsyrse

шаблон ввода: johnnnes

порог: 6, ответ: joh порог: 5, ответ: johnnnes
порог: 4, ответ: johnnyes порог: 3, ответ: jonnnyes
порог: 2, ответ: jornsyrse

шаблон ввода: Джонни

порог: 7, ответ: порог: 6, ответ: joh
порог: 5, ответ: johnnyes порог: 4, ответ:
johnnyes порог: 3, ответ: johnnyes порог: 2, ответ:
johnnyes

Рис. 9.12. Применение программы исправления правописания к различным усвоенным или ошибочным данным. Правильные входы находятся с максимальным (то есть первым предпринятым) порогом. Для ошибочных вводов порог должен быть снижен для корректной ассоциации.

Machine Translated by Google

9.4 Линейные сети с минимальными ошибками

Правило Хебба, используемое в нейронных моделях, представленных до сих пор, работает с ассоциациями между соседними нейронами. В ассоциативной памяти это используется для изучения сопоставления векторов запросов с целями. Во многих случаях это работает очень хорошо, особенно когда векторы запросов линейно независимы. Если это условие не выполняется, например, когда доступно слишком много обучающих данных, возникает вопрос: как найти оптимальную весовую матрицу? Оптимальный означает, что он минимизирует среднюю ошибку.

Мы люди, способны учиться на ошибках. Правило Хебба не дает такой возможности. Алгоритм обратного распространения, описанный ниже, использует элегантное решение, известное из аппроксимации функций, для изменения весов таким образом, чтобы минимизировать ошибку обучающих данных.

Пусть N пар обучающих векторов

$$T = \{(q^{1}, t^{1}), \ldots, (q^{N}, t^{N})\}$$

быть задано с $q^{p} \in [0, 1]^{R}$ и $t^{p} \in [0, 1]^{M}$. Мы ищем функцию $f : [0, 1]^{R} \to [0, 1]^{M}$, что минимизирует квадрат ошибки

$$\sum_{p=1}^{N} (f(q^{p}) - t^{p})^{2}$$

на данные. Предположим сначала, что данные не содержат противоречий. То есть в обучающих данных нет вектора запроса, которому должны быть сопоставлены с двумя разными целями. В этом случае нетрудно найти функцию, минимизирующую квадрат ошибки. На самом деле существует бесконечно много функций, которые делают ошибку равной нулю. Определим функцию

$$f(q) = 0, \text{ если } q \notin \{q^{1}, \ldots, q^{N}\}$$

и

$$f(q^{p}) = t^{p} \quad p \in \{1, \ldots, N\}.$$

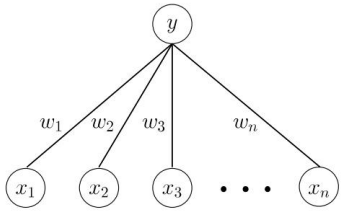
Это функция, которая даже обнуляет ошибку обучающих данных. Что еще мы могли бы хотеть? Почему нас не устраивает эта функция?

Ответ: потому что мы хотим построить интеллектуальную систему! Интеллектуальный означает, среди прочего, что изученная функция может хорошо обобщать обучающие данные на новые, неизвестные данные из того же репрезентативного набора данных. Другими словами, это означает: мы не хотим переобучения данных путем запоминания. Чего же тогда мы действительно хотим?

Нам нужна плавная функция, которая «выравнивает» пространство между точками. Разумными требованиями были бы непрерывность и возможность брать несколько производных. Поскольку даже при этих условиях остается бесконечно много функций, которые делают ошибку равной нулю, мы должны еще больше ограничить этот класс функций.

Machine Translated by Google

Рис. 9.13 Двухслойная сеть с выходом нейрон



9.4.1 Метод наименьших квадратов

Самый простой выбор — это линейное отображение. Начнем с двухслойной сети (рис. 9.13), в которой единственный нейрон у второго слоя вычисляет свою активацию, используя

$$y = e^{\sum_{i=1}^{n} w_i x_i}$$

с f(x) = x. Тот факт, что мы рассматриваем здесь только выходные нейроны, не представляет собой реального ограничения, поскольку у двухслойную сеть с двумя или более выходными нейронами всегда можно разделить на независимые сети с идентичными входными нейронами для каждого исходного выходного нейрона. Все веса подсетей независимы. Использование сигмовидной функции вместо линейной активации здесь не дает никаких преимуществ, поскольку у сигмоидальная функция является строго монотонно возрастающей и не меняет отношения порядка между различными выходными значениями.

Искомым является вектор w, который минимизирует квадрат ошибки

$$E(w) = \sum_{p=1}^{N} (w^T x^p - t^p)^2 = \sum_{p=1}^{N} \left(\sum_{i=1}^{n} w_i x_i^p - t^p \right)^2 .$$

В качестве необходимого условия минимума этой функции ошибок все частные производные должны быть равнынулю. Таким образом, мы требуем, чтобы для $j = 1,\ldots,n$:

$$\frac{\partial E}{\partial w_j} = 2 \sum_{p=1}^{N} \left(\sum_{i=1}^{n} w_i x_i^p - t^p \right) x_j^p = 0, \quad j$$

Умножение этого дает

$$\sum_{p=1}^{N} \left(\sum_{i=1}^{n} w_i x_i^p x_j^p - t^p x_j^p \right) = 0,$$

Machine Translated by Google

и замена сумм приводит к линейной системе уравнений

$$\sum_{i=1}^{n}\Big(\sum_{p=1}^{N} q_i^p q_j^p\Big) c_i = \sum_{p=1}^{N} t^p q_j^p,$$

который с

$$A_{ij} = \sum_{p=1}^{N} q_i^p q_j^p \quad \text{и} \quad b_j = \sum_{p=1}^{N} t^p q_j^p \tag{9.12}$$

можно записать в виде матричного уравнения

$$Ac = b. \tag{9.13}$$

Это так называемые нормальные уравнения всегда имеют хотя бы одно решение, а если A обратимо, то ровно одно. Кроме того, матрица A является положительно определенной, из чего следует, что найденное решение в единственном случае является глобальным минимумом. Этот алгоритм известен как метод наименьших квадратов.

Время расчета для настройки матрицы A растет с ростом $\Theta(N \cdot n^2)$ и время решения системы уравнений $O(n^3)$. Этот метод может быть очень просто расширен для включения нескольких выходных нейронов, потому что, как уже упоминалось, для двухслойных сетей с прямой связью выходные нейроны зависят друг от друга.

9.4.2 Применение к данным по аппендициту

В качестве приложения мы теперь определяем линейную оценку для диагностики аппендицита. Из данных проекта LEXMED, знакомых по разд. 8.4.5, мы используем метод наименьших квадратов для определения линейного отображения от симптомов к непрерывным переменным класса AppScore со значениями в интервале [0, 1] и получаем линейную комбинацию

$$\begin{aligned}\text{AppScore} = {} & 0{,}00085\,\text{Возраст} - 0{,}125\,\text{Пол} + 0{,}025\,\text{P1Q} + 0{,}035\,\text{P2Q} - 0{,}021\,\text{P3Q} \\ & - 0{,}025\,\text{P4Q} + 0{,}12\,\text{TensLoc} + 0{,}031\,\text{TensGlo} + 0{,}13\,\text{Losl} \\ & + 0{,}081\,\text{Conv} + 0{,}0034\,\text{RectS} + 0{,}0027\,\text{TAxi} + 0{,}0031\,\text{TRec} \\ & + 0{,}000021\,\text{Лейко} - 0{,}11\,\text{Диаб} - 1{,}83.\end{aligned}$$

Эта функция возвращает непрерывные переменные для AppScore, хотя фактическая переменная двоичного класса App принимает только значения 0 и 1. Таким образом, мы должны выбрать пороговое значение, как в случае с персептроном. Ошибка классификации оценки как функция порога показана на рис. 9.14 на стр. 243 для обучающих данных и тестовых данных. Мы сразу видим, что обе кривые почти одинаковы и имеют

Machine Translated by Google

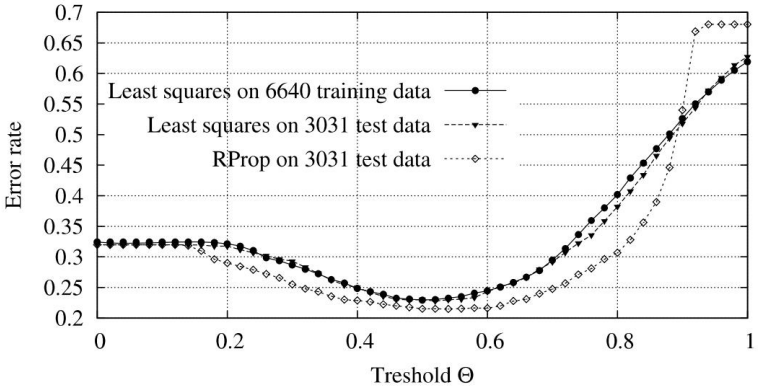


Рис. 9.14. Ошибка метода наименьших квадратов для обучающих и тестовых данных

их минимум при Θ = 0,5. По небольшой разнице двух кривых мы видим, что переобучение не является проблемой для этого метода, потому что модель очень хорошо обобщает тестовые данные.

Также на рисунке показан результат для нелинейной трехслойной сети RProp (раздел 9.5) с несколько меньшей ошибкой для пороговых значений от 0,2 до 0,9. Для практического применения полученной оценки и правильного определения порога Θ важно не только смотреть на ошибку, но и различать ее по типу (а именно ложноположительный и ложноотрицательный), как это делается в приложении LEXMED. на рис. 7.10 на стр. 142. На кривой ROC, показанной там, также показана рассчитанная здесь оценка. Мы видим, что простая линейная модель явно уступает системе LEXMED. Очевидно, что линейные приближения недостаточно эффективны для многих сложных приложений.

9.4.3 Дельта-правило

Метод наименьших квадратов, как и обучение персептрону и дереву решений, представляет собой так называемый алгоритм пакетного обучения, в отличие от пошагового обучения. При пакетном обучении все обучающие данные должны быть изучены за один прогон. Если добавляются новые обучающие данные, их нельзя просто выучить в дополнение к тому, что уже есть. Весь процесс обучения необходимо повторить с увеличенным набором. Эта проблема решается алгоритмами инкрементного обучения, которые могут адаптировать изученную модель к каждому дополнительному новому примеру. В алгоритмах, которые мы рассмотрим в следующем обсуждении, мы будем аддитивно обновлять веса для каждого нового обучающего примера по правилу

$$w_j = w_j + \Delta w_j .$$

Чтобы получить инкрементный вариант метода наименьших квадратов, мы пересматриваем вычисленные выше n частных производных функции ошибок

Machine Translated by Google

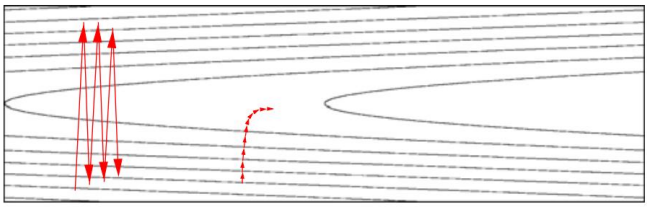


Рис. 9.15. Градиентный спуск при большом η (слева) и очень малом η (справа) в долину, пологого спускающуюся вправо. При большом η возникают колебания вокруг долины. Напротив, при слишком малом η сходимость в плоской долине происходит очень медленно.

$$\frac{E}{wj} = 2 \sum_{p=1}^{H} \left(\sum_{i=1}^{н} wiq_i^p - t_p \right) x_j^p.$$

Градиент

$$Э = \left(\frac{E}{w1}, \ldots, \frac{E}{вн} \right)$$

как вектор всех частных производных функции ошибок указывает направление наибольшего возрастания функции ошибок в n-мерном пространстве весов. Поэтому при поиске минимума мы будем следовать направлению отрицательного градиента. В качестве формулы заменывесов получаем

$$\eta wj = -\frac{E}{2\,wj} = \eta \sum_{p=1}^{H} \left(\sum_{i=1}^{н} wiq_i^p - t_p \right) q_j^p,$$

где скорость обучения η — свободно выбираемая положительная константа. Большое η ускоряет сходимость, но в то же время повышает риск колебаний вокруг минимумов или плоских долин. Поэтому оптимальный выбор η — непростая задача (см. рис. 9.15).
Большое η, например, η = 1, часто используется для начала, а затем постепенно уменьшается.

Заменив активацию

$$y^п = \sum_{i=1}^{н} вич_i^п$$

выходного нейрона для прикладного обучающего примера ^{п}, формула упрощена и получаем дельта-правило

$$wj = \eta \sum_{p=1}^{H} (t_p - y_p) q_{Дж.}^p$$

Machine Translated by Google

Рис. 9.16. Обучение двухслойной линейной сети с помощью дельта-правила. Обратите внимание, что изменения веса всегда происходят после применения всех тренировочных данных.

DELTALEARNING (Примерыобучения , η)

Инициализировать все веса wj случайным образом

Повторите ш= 0

Для всех (q p ,tp) TrainingExamples

Рассчитать вывод сети yw = w ^{n} = wpq p

+ η(tp y p)q p w = w + w

Пока w не сходится

DELTALEARNINGINCREMENTAL(TrainingExamples, η)

Инициализировать все веса wj случайным образом

Повторить

Для всех (q p ,tp) TrainingExamples

Вычислить вывод сети yw = w + ^{n} = wpq p

η(tp y p)q p

Пока w не сходится

Рис. 9.17 Инкрементальный вариант дельта-правила

Таким образом, для каждого обучающего примера разница между целевым t p и фактическим p шаблонов сети вычисляется для заданного входа q. После суммирования по выходу у всех веса затем изменяются пропорционально сумме. Этот алгоритм показан на рис. 9.16.

Мывидим, что алгоритм все еще является инкрементным, потому что изменения веса происходят только после того, как все обучающие примерыбыли применены один раз. Мыможем исправить этот недостаток, напрямую изменяя веса (инкрементальный градиент) после каждого обучающего примера (рис. 9.17), что, строго говоря, больше не является корректной реализацией дельта-правила.

9.4.4 Сравнение с персептроном

Правило обучения для персептронов, введенное в разд. 8.2.1, метод наименьших квадратов и дельта-правило могут использоваться для генерации линейных функций из данных. Однако для персептронов, в отличие от других методов, классификатор для линейно разделимых классов изучается посредством порогового решения . Однако два других метода генерируют линейную аппроксимацию данных. Как показано в разд. 9.4.2, классификатор может быть сгенерирован из линейного отображения , при желании, путем применения пороговой функции.

Machine Translated by Google

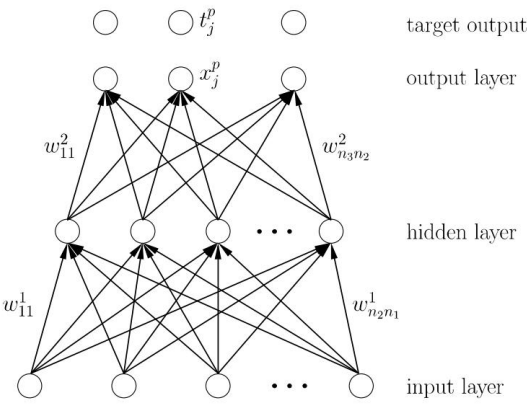


Рис. 9.18. Трехслойная сеть обратного распространения с n1 нейронами в первом, n2 нейронами во втором и n3 нейронами в третьем слое

Персептрон и дельта-правило представляют собой итерационные алгоритмы, для которых время до сходимости сильно зависит от данных. В случае линейно разделимых данных для персептрона можно найти верхний предел числа шагов итерации. Для дельта-правила, напротив, существует только гарантия неограниченной асимптотической сходимости [HKP91].

Для метода наименьших квадратов обучение состоит в построении и решении линейной системы уравнений для вектора весов. Таким образом, существует жесткое ограничение на время вычислений. Из-за этого метод наименьших квадратов всегда предпочтительнее, когда не требуется постепенное обучение.

9.5 Алгоритм обратного распространения

Теперь с помощью алгоритма обратного распространения мы представляем наиболее часто используемую нейронную модель. Причиной его широкого использования является его универсальная универсальность для задач произвольной аппроксимации. Алгоритм исходит непосредственно из правила возрастания дельты. В отличие от дельта-правила, оно применяет нелинейную сигмовидную функцию к взвешенной сумме входных данных в качестве функции активации. Кроме того, сеть обратного распространения может иметь более двух слоев нейронов. Алгоритм стал известен благодаря статье [RHR86] в легендарном сборнике PDP [RM86].

На рис. 9.18 типичная сеть обратного распространения с входным слоем, скрытым выходным слой, и отображается выходной слой. Поскольку у текущее выходное значение $_{\text{П}}$ нейрона слоя x_j сравнивается с целевым выходным значением t, они рисуются параллельно $_{\text{П}}$ j другу другу. Все нейроны, кроме входных, вычисляют свое текущее значение x_j по правилу

$$x_j = f\left(\sum_{i=1}^{н} w_{ji} x_i\right) \tag{9.14}$$

Machine Translated by Google

где n — количество нейронов в предыдущем слое. Мыиспользуемсигмовидную функцию

$$f(x) = 1 \frac{1}{+ e\text{-}x}.$$

По аналогии с правилом приращения дельтывеса изменяются пропорционально отрицательному градиенту квадратичной функции ошибки, суммированной по выходным нейронам.

$$Ep(w) = 2 \frac{1}{k \text{ выход}} (т_к п - x_к^p)^2$$

для тренировочного шаблона p:

$$pwj\,i = \eta \frac{Эп}{wj\,я}.$$

Для вывода правила обучения приведенное выше выражение заменяется на Ep. В выражении xk заменяется на (9.14) на стр. 246. В уравнении выходыхи нейронов следующего, более глубокого слоя встречаются рекурсивно и т. д. Примногократном применении цепного правила (см. [RHR86] или [Zel94]) получаемправило обучения обратного распространения

$$pwj\,ij = \eta \delta p \; ик с пи,$$

с

$$\delta_j^p = \begin{cases} ик с п (1 \quad x^п)(т п - x^p) & \text{если j — выходной нейрон,} \\ джп x (1 \quad x_j^{jp}) \; jp \; \delta ik wkj, & \text{если j — скрытый нейрон,} \end{cases}$$

который также обозначается обобщенным дельта-правилом. Для всех нейронов формула изменения веса wj i от нейрона i к нейрону j (см. рис. 9.19 на стр. 248) содержит, как и правило Хебба, член ηx $\text{стр}x_{\text{я}}$ Джк. Новый коэффициент $п$) создает симмуникатор (1 x j try, отсутствующий в правиле Хебба, между активациями 0 и 1 нейрона. Для выходных нейронов коэффициент (tp j . $- x_{\text{дж}}^p$) заботится об изменении веса пропорционален ошибке. Для скрытых нейронов значение δ j $п$ нейрона j вычисляется рекурсивно от всех изменений δ $п_к$ нейронов следующего более высокого уровня.

Все выполнение процесса обучения показано на рис. 9.20 на стр. 248.

После расчета выхода сети (прямого распространения) для обучающего примера вычисляется ошибка аппроксимации. Затем это используется во время обратного распространения для изменения весов в обратном направлении от слоя к слою. Затем весь процесс применяется ко всем обучающим примерам и повторяется до тех пор, пока веса не перестанут изменяться или не будет достигнуто ограничение по времени.

Machine Translated by Google

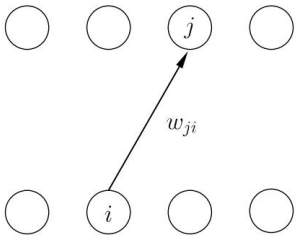


Рис. 9.19 Обозначение нейронов и весов для применения правила обратного распространения

ОБРАТНОЕ РАСПРОСТРАНЕНИЕ (Примерыобучения , η)
Инициализировать все веса wj случайными

значениями. Повторить для всех (q p ,t p)
для входного слоя 1. Применить TrainingExamples p
вектор запроса q 2. Прямое
распространение: Для всех слоев, начиная с первого
скрытого слоя вверх Для каждого нейрона слой
Рассчитать активацию xj = f (ni=1 wj ixi)
3. Расчет квадратичной ошибки Ep(w)
4. Обратное распространение: Для
всех уровней весов с последнего вниз Для каждого веса
wj i = wj i + ηδp
wj я
дж
x п
я

Пока w не сойдется или не будет достигнут предел времени

Рис. 9.20 Алгоритм обратного распространения

Если мыпостроимсеть хотя быс одним скрытым слоем, можно изучить нелинейные отображения . Без скрытых слоев выходные нейроныне мощнее линейного нейрона, несмотря на сигмовидную функцию . Причина этого в том, что сигмовидная функция строго монотонна. То же верно и для многоуровневых сетей, которые используют только линейную функцию в качестве функции активации, например, функцию идентификации.

Это связано с тем, что цепные выполнения линейных отображений являются линейными в совокупности.

Как и в случае с персептроном, класс функций, которые могут быть представлены также увеличиваются , если мыиспользуем переменную сигмовидную функцию

$$f(x) = 1 + \frac{1}{e(x \quad \Theta)}.$$

с порогом Θ. Это реализуется аналогично тому, как показано в разд. 8.2, в котором во входной слой и в каждый скрытый слой вставляется нейрон, активация которого всегда имеет значение единица и который связан с нейронами следующего по высоте уровня . Веса этих связей усваиваются нормально и представляют собой порог Θ нейронов-преемников.

Machine Translated by Google

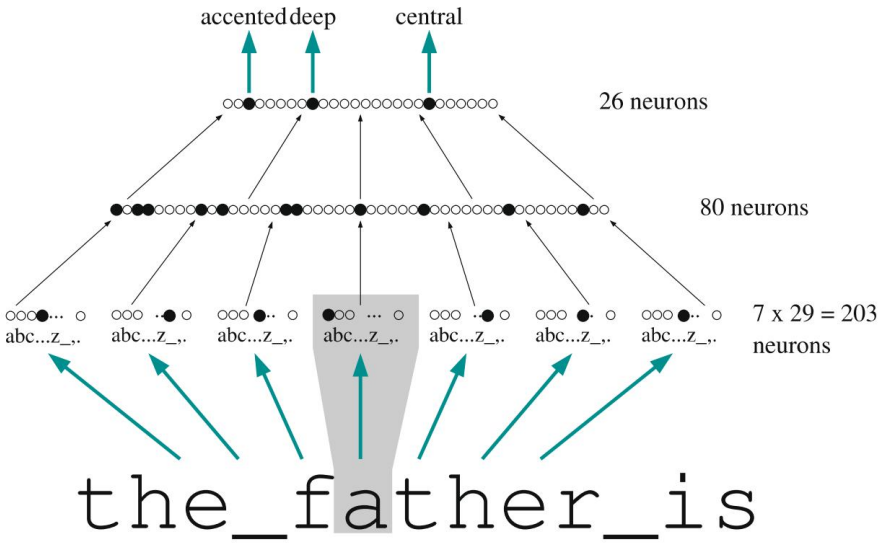


Рис. 9.21. Сеть NETtalk сопоставляет текст с его атрибутами произношения.

9.5.1 NETtalk: сеть учится говорить

Сейновски и Розенберг в 1986 г. весьма впечатляюще продемонстрировали, насколько способно обратное распространение [SR86]. Они построили систему, способную понятно читать вслух английский текст из текстового файла. Архитектура сети, показанная на рис. 9.21, состоит из входного слоя с 7 × 29 = 203 нейронами в котором закодированы текущая буква и три предыдущие буквы, а также три последующие буквы. Для каждой из этих семи букв зарезервировано 29 нейронов. символы «a...z ,.». Вход отображается на 26 выходных нейронов более 80. скрытые нейроны, каждый из которых соответствует определенной фонеме. Например, «а» в слове «отец» произносится глубоко, с ударением и посередине. Сеть прошла обучение с 1000 слов, которые применялись случайным образом одно за другим буква за буквой. Для каждой буквы вручную задавался целевой вывод по ее интонации. Чтобы преобразовать интонационные признаки в реальные звуки, часть системы синтеза речи Использовался DECtalk. Благодаря полной взаимосвязанности сеть содержит в общей сложности 203 × 80 + 80 × 26 = 18320 гирь.

Обучение системы проводилось на тренажере VAX 780 около 50 циклов. над всеми словами. Таким образом, примерно при 5 символах на слово в среднем около 5·50·1000 = Потребовалось 250 000 итераций алгоритма обратного распространения ошибки. При скорости около один символ в секунду, это означает примерно 69 часов вычислительного времени. разработчики наблюдали многие свойства системы, которые очень похожи на человеческие обучение. Сначала система может говорить только неразборчиво или использовать простые слова. С течением времени он продолжал улучшаться и, наконец, достиг 95% правильности произнесения букв.

Machine Translated by Google

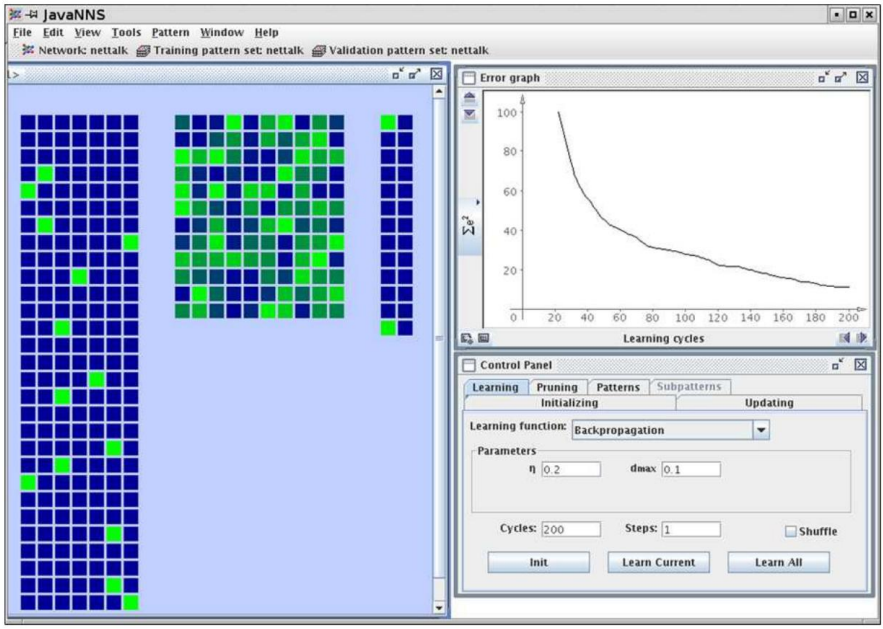


Рис. 9.22 Сеть NETtalk в JNNS. В левом окне слева направо мы видим 7 · 29 входных нейронов, 80 скрытых и 26 выходных нейронов. Из-за их большого количества веса в сети опущены. Вверху справа — кривая обучения, показывающая изменение квадрата ошибки с течением времени.

Для визуальных экспериментов с нейронными сетями рекомендуется Java Neural Network Simulator JNNS [Zel94]. Сеть NETtalk, загруженная и обученная с помощью JNNS, показана на рис. 9.22.

9.5.2 Изучение эвристики для средств доказательства теорем

В гл. В разделе 6 обсуждались алгоритмы эвристического поиска, такие как A -алгоритм и IDA -алгоритм. Чтобы добиться значительного сокращения пространства поиска, для реализации этих алгоритмов необходима хорошая эвристика. В разд. 4.1 была продемонстрирована проблема взрывающегося пространства поиска во время поиска доказательства доказательством теоремы. Эта проблема вызвана большим количеством возможных шагов вывода на каждом шаге.

Теперь мы попытаемся создать модули управления эвристическим доказательством, которые оценивают различные альтернативы для следующего шага, а затем выбирают альтернативу с лучшим рейтингом. В случае разрешения рейтинг доступных предложений может быть выполнен с помощью функции, которая на основе определенных атрибутов предложений, таких как количество литералов, сложность терминов и т. д., вычисляет значение для каждой пары разрешений. способные оговорки. В [ESS89, SE90] такая эвристика была изучена для средства доказательства теорем SETHEO с использованием обратного распространения ошибки. При изучении эвристики ат

Machine Translated by Google

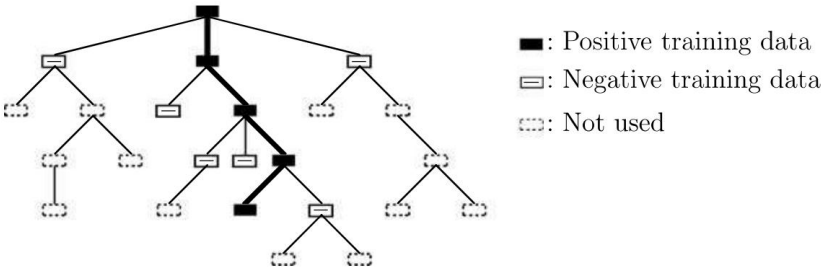


Рис. 9.23. Дерево поиска с успешным путем, узлы которого оценены как положительные, и отрицательно оцененными неуспешными ветвями

текущее предложение было сохранено в качестве обучающих данных для положительного класса для каждого найденного доказательства в каждом отделении во время поиска. На всех ответвления х, которые не привели к доказательству, атрибуты текущего предложения были сохранены как обучающие данные для отрицательного класса. Рисуется такое дерево с успешным путем и соответствующими рейтингами узлов. на рис. 9.23.

На этих данных была обучена сеть обратного распространения , а затем использована для оценки предложения в доказывающем. Для каждого предложения вычисляли, нормализовали и кодировали во входном нейроне 16 числовых признаков. Сеть обучалась на 25 скрытых нейронов и один выходной нейрон для класса (положительный/отрицательный).

Было показано, что количество попыток вывода может быть уменьшено во много раз. порядков для сложных задач , что в итоге сокращает время вычислений с часов до секунд. Тем самым стало возможным доказать теоремы которые без эвристики были недостаемы

9.5.3 Проблемы и улучшения

Обратному распространению уже 25 лет , и оно зарекомендовало себя в различных приложения х. например, в распознавании образов и в робототехнике. Однако его применение иногда проблематично. Особенно когда в сети многотысячные веса и необходимо изучить много обучающих данных, возникают две проблемы

Сеть часто сходится к локальным минимумам функции ошибок . Более того, обратное распространение часто сходится очень медленно. Это означает , что многие итерации в течение нужны все тренировочные шаблоны Для решения этих проблем было предложено множество усовершенствований. Как упоминалось в разд. 9.4.3, колебаний можно избежать , медленно уменьшая скорость обучения η, как показано на рис. 9.15 на странице 244.

Другой метод уменьшения колебаний - использование члена импульса, в то время как обновление весов, что гарантирует , что направление градиентного спуска не слишком резко меняется от одного шага к другому. Вот текущий вес к изменению pwj i(t) в момент времени t добавляется еще одна часть изменения pwj i(t 1) с предыдущего шага. Затем правило обучения меняется на

$$pwj\ i(t) = \eta\delta p \quad \text{Ик с} \quad \text{П} \quad + \gamma pwj\ i\ (t \quad 1)$$

Machine Translated by Google

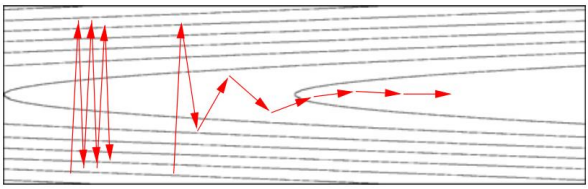


Рис. 9.24 Резкое направление изменения сглаживаются использование импульса срок. Итерация без член импульса (слева) в сравнение с итерацией с термин импульса (справа)

с параметром у между нулем и единицей. Это изображено в двухмерном пример на рис. 9.24.

Другая идея состоит в том, чтобы минимизировать линейную функцию ошибки вместо квадратичной. функция, сводящая проблему медленной сходимости к плоским долинам.

Градиентный спуск при обратном распространении в конечном счете основан на линейной аппроксимации функции ошибок. Алгоритм Quickprop, который использует квадратичное приближение к функции ошибок и, таким образом, обеспечивает более быструю сходимость, был создан Скотт Фалманн.

Путем умной унификации упомянутых улучшений и других эвристических приемов Мартин Ридмиллер добился дальнейшей оптимизации с помощью алгоритма RProp [RB93]. RProp заменил обратное распространение изменяется новым состоянием алгоритма аппроксимации нейронной сети с прямой связью искусства. В разд. 8.8 мы применили RProp к классификации данных аппендицита и добился ошибки, которая примерно того же размера, что и обученное дерево решений.

9.6 Машины опорных векторов

Нейронные сети с прямой связью только с одним слоем весов являются линейными. Линейность приводит к простым сетям и быстрому обучению с гарантированной сходимостью. Кроме того, опасность переобучения для линейных моделей невелика. Для многих приложений однако линейные модели недостаточно сильны, например, потому, что соответствующие классыне являются линейно разделимыми. Здесь используются многослойные сети, такие как обратное распространение, в результате чего локальные минимумы сходимость проблемы и может произойти переобучение.

Многообещающий подход, объединяющий преимущества линейных и нелинейных моделей, следует теории машин опорных векторов (SVM), которую мы будем грубо наметить, используя задачу двух классов.4

В случае двух линейно разделимых классов легко найти разделяющую гипер плоскости, например, с правилом обучения персептрона. Однако, как правило, бесконечно много таких плоскостей, как в двумерном примере на рис. 9.25 на стр. 253. Мы ищем плоскость, которая имеет наибольшее минимальное расстояние до обоих классы Эта плоскость обычно однозначно определяется несколькими точками в границной области. Эти точки, так называемые опорные векторы, находятся на одинаковом расстоянии от разделяющей поверхности.

4Машины опорных векторов не являются нейронными сетями. Однако из-за их исторического развития и математической связи с линейными сетями они обсуждаются здесь.

Machine Translated by Google

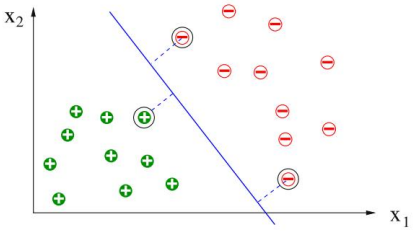


Рис. 9.25 Два класса с максимально разделяющей линией. Обведенные точки — опорные векторы

линия . Для нахождения опорных векторов существует эффективный алгоритм оптимизации. Интересно, что оптимальная разделяющая гиперплоскость определяется несколькими параметрами, а именно опорными векторами. Таким образом, опасность переоснащения мала.

Машиныопорных векторов применяют этот алгоритм к нелинейно разделимым задачам в двухэтапном процессе: на первом этапе к данным применяется нелинейное преобразование с тем свойством, что преобразованные данные являются линейно разделимыми. На втором этапе определяются опорные векторыв преобразованном пространстве.

Первый шаг очень интересен, но не совсем прост . На самом деле всегда можно сделать классы линейно разделимыми путем преобразования векторного пространства, если данные не содержат противоречий5. Такое разделение может быть достигнуто, например, введением нового (n + 1)-го измерения и Определение

$$x_{n+1} = \begin{cases} 1, \text{если } x \quad \text{класс 1,} \\ 0, \text{если } x \quad \text{класс 0.} \end{cases}$$

Однако эта формула не очень помогает , поскольку она неприменима к новым точкам неизвестного класса, подлежащим классификации. Таким образом, нам необходимо общее преобразование, максимально независимое от текущих данных. Можно показать , что такие общие преобразования существуют даже для границ деления классов произвольной формыв исходном векторном пространстве. В преобразованном пространстве данные становятся линейно разделимыми. Однако количество измерений нового векторного пространства растет экспоненциально с количеством измерений исходного векторного пространства.
Однако большое количество новых измерений не столь проблематично, поскольку при использовании опорных векторов разделяющая плоскость , как было сказано выше, определяется лишь несколькими параметрами.

Центральное нелинейное преобразование векторного пространства называется ядром, из-за чего машиныопорных векторов также известны как методыядра. Первоначальная теория SVM, разработанная для задач классификации, была расширена и теперь может использоваться и в задачах регрессии.

Используемая здесь математика очень интересна, но слишком обширна для первоначального введения . Чтобыглубже погрузиться в эту многообещающую молодую отрасль машинного обучения , мыотсылаем читателя к [SS02, Alp04] и [Bur98].

5 Точка данных противоречива, если она принадлежит обоимклассам.

Machine Translated by Google

9.7 Приложения

Помимо приведенных здесь примеров, существует бесчисленное множество приложений для работы с нейронными сетями во всех отраслях промышленности. Очень важной областью является распознавание образов во всех его формах, будь то анализ фотографий для распознавания людей или лиц, распознавание кожи кожи рыб по изображениям сонара, распознавание и классификация военной техники по скан радаров и многое другое. Но нейронные сети также можно научить распознавать устную речь и рукописный текст.

Нейронные сети используются не только для распознавания объектов и сцен. Их также обучают управлять простыми роботами с помощью данных датчиков, а также эвристически управлять поиском в компьютерах для игры в нарды и шахматы. Особый интерес в играх представляют техники обучения с подкреплением, разд. 10.8, например с нейронными сетями.

Нейронные сети уже давно, помимо статистических методов, успешно используются для прогнозирования фондовых рынков и оценки кредитоспособности клиентов банков.

Для многих из этих приложений также могут использоваться другие алгоритмы машинного обучения. Однако на сегодняшний день нейронные сети оказались более успешными и популярными, чем все другие алгоритмы машинного обучения. Однако историческое преобладание нейронных сетей постепенно отодвигается в пользу других методов из-за большого коммерческого успеха интеллектуального анализа данных и машин опорных векторов

9.8 Резюме и перспективы

С персептроном, дельта-правилом и обратным распространением мы представили самый важный класс нейронных сетей и их отношение к оценкам наивному Байесу, с одной стороны, но также и к методу наименьших квадратов. Затем мы увидели увлекательные сети Хопфилда, вдохновленные биологическими моделями, которыми, однако, трудно управлять на практике из-за их сложной динамики. На практике большее значение имеют различные модели ассоциативной памяти.

Во всех нейронных моделях информация хранится в распределении по множеству весов. Из-за этого отмирание нескольких нейронов в мозгу не оказывает заметного влияния на функцию мозга. Благодаря распределенному хранению данных сеть устойчива к небольшим сбоям. Это также связано со способностью распознавать шаблоны с ошибками, показанными в нескольких примерах.

Однако у распределенного представления знаний есть недостаток, заключающийся в том, что инженеру по знаниям трудно локализовать информацию. Практически невозможно проанализировать и понять множество весов в полностью обученной нейронной сети. Напротив, для дерева усвоенных решений легко понять полученные знания и даже представить их в виде логической формулы. Особенно выразительна и изящна логика предикатов, позволяющая формализовать отношения. Например, предикат бабушка (карен, клайд) легко понять. Такого рода отношения также можно изучить с помощью нейронных сетей, но невозможно даже локализовать «бабушкин нейрон» в сети. Поэтому до сих пор существуют проблемы с подключением нейронных сетей к системам обработки символов.

Machine Translated by Google

Из большого количества интересных нейромоделей многие здесь не поддаются обработке. Например, самоорганизующиеся карты, введенные Кохоненом, представляют собой биологически мотивированное отображение сенсорного слоя нейронов на второй слой нейронов со свойством адаптивности и сохранения сходства.

Проблема с представленными здесь сетями возникает во время поэтапного обучения. Если, например, полностью обученная сеть обратного распространения далее обучается с новыми шаблонами, то многие (и потенциально все) старые шаблоны быстро забываются. Чтобы решить эту проблему, Карпентер и Гроссберг разработали теорию адаптивного резонанса (APT), результатом которой стала целая линейка нейронных моделей.

В качестве дополнительной литературы рекомендуем учебники [Bis05, RMS92, Roj96, Zel94]. Те, кто заинтересован в чтении наиболее важных оригинальных работ в этой захватывающей области, могут обратиться к двум сборникам [AR88, APR90].

9.9 Упражнения

9.9.1 От биологии к моделированию

Упражнение 9.1. Покажите точечную симметрию сигмовидной функции относительно $(\theta, \frac{1}{2})$.

9.9.2 Сети Хопфилда

Упражнение 9.2. Используйте сетевой апплет Хопфилда по адресу http://www.cbu.edu/~pong/ai/hopfield/hopfieldapplet.html и проверьте объем памяти для коррелированных и некоррелированных шаблонов (со случайно установленными битами) с размером 10×10. размер сетки. Используйте шаблон с 10% шума для тестирования.

Упражнение 9.3. Сравните теоретический предел $N = 0{,}146\,n$ для максимального количества хранимых шаблонов, приведенный в разд. 9.2.2, с емкостью классической бинарной памяти того же размера.

9.9.3 Линейные сети с минимальными ошибками

Упражнение 9.4

(а) Напишите программу для обучения линейному отображению методом наименьших средних квадратов. Это довольно просто: нужно только составить нормальные уравнения по (9.12) на стр. 242, а затем решить систему уравнений.

(а) Примените эту программу к данным об аппендиците на веб-сайте этой книги и определите линейную оценку. Задайте ошибку как функцию порога, как показано на рис. 9.14 на стр. 243. (с) Теперь определите ROC-кривую для этой оценки.

Machine Translated by Google

9.9.4 Обратное распространение

Упражнение 9.5. Используя программу обратного распространения ошибки (например, в JNNS или KNIME), создайте сеть с восемью входными нейронами, восемью выходными нейронами и тремя скрытыми нейронами. Затем обучите сеть с помощью восьми парообучающихся с опорой, данных q, чтобы убедиться, что в m векторе q p-й бит равен единице, а все остальные биты равны нулю. Таким образом, перед сетью стоит простая задача изучения идентичного отображения. Такая сеть называется кодером 8-3-8. После процесса обучения наблюдайте за кодированием трех скрытых нейронов для ввода всех восьми изученных входных векторов. Что ты заметил? Теперь повторите эксперимент, уменьшив количество скрытых нейронов до двух, а затем до одного.

Упражнение 9.6 Чтобы показать, что сети обратного распространения могут разделять нелинейно разделимые множества, обучите функцию XOR. Обучающие данные для этого: ((0, 0), 0), ((0, 1), 1), ((1, 0), 1), ((1, 1), 0). (a) Создайте сеть с двумя
нейронами во входном слое и скрытом слое, и
один нейрон в выходном слое и обучить эту сеть.
(b) Теперь удалите скрытые нейроны и соедините входной слой напрямую с выходными нейронами.
Что вы наблюдаете?

Упражнение 9.7

(a) Покажите, что любая многослойная сеть обратного распространения с линейной функцией
активации обладает такой же мощностью, как и двухслойная. Для этого достаточно
показать, что последовательное выполнение линейных отображений
является линейным отображением. (b) Покажите, что двухслойная сеть обратного
распространения с любой строго монотонной функцией активации не более эффективна для
задач классификации, чем сеть без функции активации или с линейной функцией активации.

9.9.5 Машины опорных векторов

Упражнение 9.8 Два нелинейно разделимых двумерных набора обучающих данных M+ 2 2 = 1 и все + x
и M заданы. Все точки M+ лежат внутри единичной окружности x 2 2 2 1 R, что делает
точки в M находятся снаружи. Задайте преобразование координат
f : R для линейно разделимых данных. Дайте уравнение разделительной линии и нарисуйте два
пространства и точки данных.

Machine Translated by Google

Обучение с подкреплением

10

10.1 Введение

Все алгоритмы обучения, описанные до сих пор, кроме алгоритмов кластеризации, относятся к классу обучения с учителем. При обучении с учителем предполагается, что агент изучает отображение входных переменных в выходные переменные. Здесь важно, чтобы для каждого отдельного обучающего примера были предоставлены все значения как входных, так и выходных переменных. Другими словами, нам нужен учитель или база данных, в которой сопоставление, которое необходимо изучить, приблизительно определено для достаточного количества входных значений. Единственная задача алгоритма машинного обучения — отфильтровать шум из данных и найти функцию, которая хорошо аппроксимирует отображение даже между заданными точками данных.

При обучении с подкреплением ситуация иная и более сложная, потому что нет данных для обучения. Начнем с простого иллюстративного примера из робототехники, который затем используется как приложение для различных алгоритмов.

Обучение с подкреплением очень ценно в области робототехники, где задачи, которые необходимо выполнить, часто настолько сложны, что не поддаются кодированию в виде программ, а данные для обучения недоступны. Задача робота состоит в том, чтобы путем проб и ошибок (или успеха) выяснить, какие действия хороши в той или иной ситуации, а какие нет. Во многих случаях мы, люди, учимся очень похожим образом. Например, когда ребенок учится ходить, это обычно происходит без обучения, а просто через подкрепление. Успешные попытки ходить вознаграждаются продвижением вперед, а неудачные попытки наказываются часто болезненными падениями. Положительное и отрицательное подкрепление также являются важными факторами успешного обучения в школе и во многих видах спорта (см. рис. 10.1 на стр. 258).

В следующем примере изучается значительно упрощенная задача движения.

Пример 10.1 Робот, механизм которого состоит только из прямоугольного блока и руки с двумя шарнирами, дуги, показан на рис. 10.2 на стр. 258 (см. [KMK97]).
Единственными возможными действиями робота являются движения дуг вверх или вниз и дуг вправо или влево. Кроме того, мы разрешаем перемещение только фиксированных дискретных единиц (например, с шагом 10 градусов). Задача агента состоит в том, чтобы изучить политику

В. Эртель, Введение в искусственный интеллект, Темы бакалавриата по компьютерным наукам, DOI 10.1007/978-0-85729-299-5_10, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google



Рис. 10.1. «Может быть , в следующий раз мне следует нач ать поворач ивать ся немного ранше или двигать ся медленнее?» — Обуч ение на отриц ат ель ном под к реплении

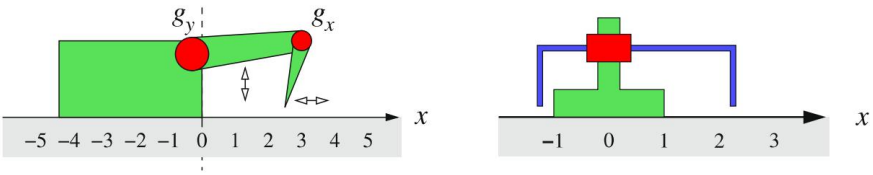


Рис. 10.2. Двигая обоими суст авами, ползающий робот в левой ч аст и изображения может двигат ь ся вперед и наз ад . Шагающий робот с правой ст ороны должен соот вет ст венно перемещат ь раму вверх и вниз или влево и вправо. Обрат ная свя зь для движения робот а положит ель на для движения вправо и отриц ат ель на для движения влево .

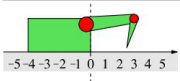
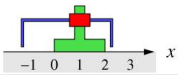
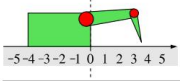
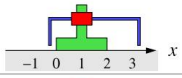
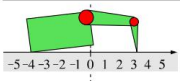
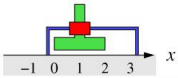
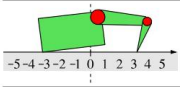
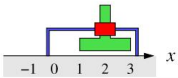
ч т о позволя ет ему двигат ь ся к ак можно быст рее вправо. Шагающий робот на рис. 10.2 работ ает аналогич но в том же двухмерном прост ранст ве сост оя ний.

Успешная последоват ель ност ь дейст вий пок аз ана в т аблиц е 10.1 на ст раниц е 259. Дейст вие в момент времени t = 2 приводит к т ому, ч т о нагруженная рук а перемещает т ело на одну ед иниц у д линывправо. Хорошие анимац ии эт ого примера можно найт и в [KMK97] и [Tok06].

Прежде ч ем мыперейдем к алгорит му обуч ения , мыдолжнысоот вет ст вующим образом смоделироват ь зад ач у мат емат ич еск и. Мыописываем сост оя ние робот а двумя переменными gx и gy для положения суст авов, к аждая из к от орых имеет к онеч ное ч исло д иск рет ных з нач ений. Т ак им образом, сост оя ние робот а к од ируется к ак век т ор (gx, gy). К олич ест во возможных совмест ных положений равно nx и ny или соот вет ст венно . Мыиспольз уем гориз онт аль ное положение т ела робот а, к от орое может принимат ь реаль ные з нач ения , для оц енк и дейст вий робот а. Движения вправо воз наг раждают ся

Machine Translated by Google

Таблица 10.1 Циклическая серия движений с систематическим движением вперед

Ползучий робот	Бегущий робот	Время	Состояние		Вознаграждение за действие	
		т	гы	gx	икс	в
		0	Вверх	Левый	0	Верно
		1	Вверх	Верно	0	Вниз
		2	Вниз вправо		0	Левый
		3	Вниз влево1			Вверх

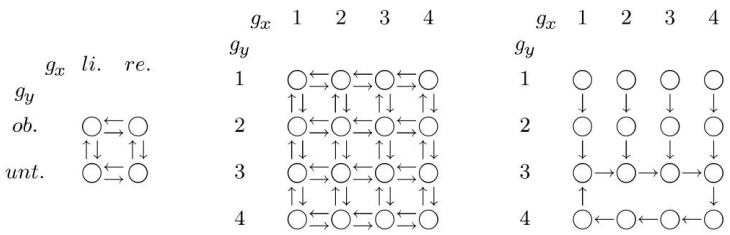


Рис. 10.3 Пространство состояний примера робота в случае двух возможных положений для каждого из суставов (слева) и в случае четырех горизонтальных и вертикальных положений для каждого (посередине). в правильное изображение дается оптимальная политика

с положительными изменениями x, а движения влево наказываются отрицательными изменениями .

На рис. 10.3 пространство состояний для двух вариантов этого показано в упрощенной форме1. В левом примере оба сустава имеют два положения , а в среднем примере они у каждого по четыре положения . Оптимальная политика представлена на рис. 10.3 справа.

10.2 Задача

Как показано на рис. 10.4 на стр. 260, мы различаем агента и его окружение. В момент времени t мир, включающий в себя агента и его окружение, описывается состоянием st S. Множество S является абстракцией реальных возможных состояний мира, потому что, с одной стороны, мир не может быть точно описан, а с другой стороны, агент часто имеет только неполную информацию о фактическом

1 Пространство движения руки, состоящее из дуг, визуализируется как прямоугольная сетка.

Machine Translated by Google

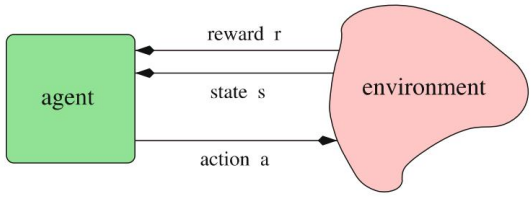


Рис. 10.4 Агент и его взаимодействие со средой

состояние из-за ошибок измерения. Затем агент выполняет действие в точке A в момент времени t. Это действие изменяет мир, таким образом, приводит к состоянию st+1 в момент времени t+1. Функция перехода состояний δ, определяемая средой, определяет новое состояние st+1 = δ(st, at). Агент не может повлиять на него.

Выполнив действие при, агент получает немедленное вознаграждение rt = r(st, at) (см. рис. 10.4). Немедленное вознаграждение rt = r(st, at) всегда зависит от текущего состояния и выполненного действия. r(st, at) = 0 означает, что агент не получает немедленной обратной связи за действие в. При обучении rt > 0 должно приводить к положительному, а rt < 0 к отрицательному подкреплению оценки действия в состоянии st. В частности, при обучении с подкреплением изучаются приложения, в которых в течение длительного времени не происходит немедленного вознаграждения. Шахматист, например, учится улучшать свою игру благодаря выигранным или проигранным партиям, даже если он не получает немедленного вознаграждения за все отдельные ходы. Здесь мы можем увидеть сложность присвоения вознаграждения в конце последовательности действий всем действиям в последовательности, которые привели к этой точке (проблема распределения кредитов).

В случае ползающего робота состояние состоит из положения двух суставов, то есть s = (gx, gy). Награда определяется пройденным расстоянием x.

Политика π : S A — это отображение состояний в действия. Цель обучения с подкреплением состоит в том, чтобы агент выучил оптимальную политику на основе своего опыта. Политика оптимальна, если она максимизирует вознаграждение в долгосрочной перспективе, то есть на многих этапах. Но что именно означает «максимальное вознаграждение»? Определяемость, или дисконтированное вознаграждение

$$B^{\pi}(st) = rt + \gamma\, rt+1 + \gamma^{2}\, p+2 + \cdots = \sum_{i=0} \gamma^{i}\, rt+i \qquad (10.1)$$

политики π, когда мы начинаем в начальном состоянии st. Здесь 0 γ < 1 — константа, которая гарантирует, что будущая обратная связь тем больше обесценивается, чем дальше в будущем она происходит. Немедленное вознаграждение rt взвешивается как самое сильное. Эта функция вознаграждения используется чаще всего. Альтернативой, которая иногда интересна, является среднее вознаграждение.

$$B^{\pi}(st) = \lim_{h \to \infty} \frac{1}{h} \sum_{i=0}^{h} rt+i \,. \qquad (10.2)$$

Политика π называется оптимальным, если для всех состояний s

$$B^{\pi}(c) \quad B^{\pi}(c). \qquad (10.3)$$

Machine Translated by Google

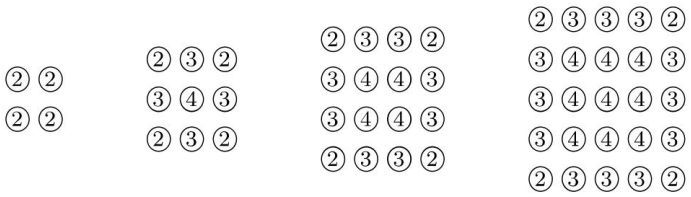


Рис. 10.5. Пространство состояний для примера со значениями 2, 3, 4, 5 для nx и ny. Количество возможных действий указано для каждого состояния в соответствующих кругах.

Таблица 10.2 Количество политик и для пространств состояний разного размера в примере

нкс, нью-йорк	Количество штатов	Количество политик
24 24 = 16		
39 243 44 = 5184		
4	16	243 84 4 2,7 × 107 243 124
5	25	9 2,2 × 1012

То есть он как минимум не уступает всем другим политикам в соответствии с заданным значением. функция. Для лучшей читаемости функция оптимального значения V π будет обозначаться V .

Обсуждаемые здесь агенты или их политики используют только информацию о текущем состоянии st для определения следующего состояния, а не предыдущую историю. Это оправдано, если награда за действие зависит только от текущего состояния и текущих действий.
Такие процессы называются марковскими процессами принятия решений (MDP). Во многих приложениях, особенно в робототехнике, фактическое состояние агента точно неизвестно, что еще больше затрудняет планирование действий. Причиной этого может быть зашумленный сигнал датчика. Мы называем такой процесс частично наблюдаемым марковским процессом принятия решений (POMDP).

10.3 Неинформированный комбинаторный поиск

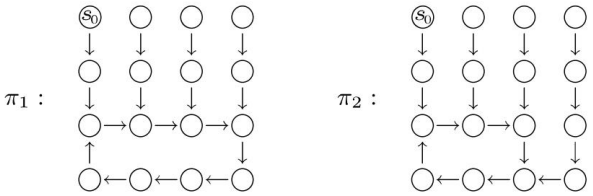
Простейшая возможность найти успешную политику — это комбинаторный перебор всех политик, как описано в гл. 6. Однако даже в простом примере 10.1 имеется очень много политик, из-за чего комбинаторный поиск связан с чрезвычайно высокими вычислительными затратами. На рис. 10.5 указано количество возможных действий для каждого состояния. Исходя из этого, количество возможных политик рассчитывается как произведение заданных значений, как показано в таблице 10.2.

Для произвольных значений nx и ny всегда есть четыре угловых узла с двумя возможными действиями, 2 (nx - 2) + 2 (ny - 2) краевых узла с тремя действиями и (nx - 2) (ny - 2) внутренние узлы. с четырьмя действиями. Таким образом, есть

4 2 3 2(nx 2)+2(ny 2) 4 (nx 2)(ny 2)

Machine Translated by Google

Рис. 10.6 Две разные политики для примера



разные политики для фиксированных nx и ny. Таким образом, количество политик растет экспоненциально с количеством состояний. В общем случае это верно, если в каждом состоянии имеется более одного возможного действия. Поэтому для практических приложений этот алгоритм бесполезен. Даже эвристический поиск, описанный в гл. 6, здесь нельзя использовать. Поскольку упрямое вознаграждение почти за все действия равно нулю, его нельзя использовать в качестве эвристической функции оценки.

Вычислительные затраты возрастают еще выше, если учесть, что (в дополнение к (s) должны быть перечисление всех политик), значение V^{π} рассчитаны для каждого сгенерированного (s) должны политика π и каждое начальное состояние s. Бесконечная сумма в V^{π} быть отсечены для использовать в практическом расчете; однако из-за экспоненциального уменьшения коэффициентов γ в (10.1) на стр. 260 это не представляет проблемы

В примере 10.1 на стр. 257 разница xt+1 xt может быть использована как немедленная награда за действие при, что означает, что каждое движение тела робота вправо вознаграждается 1, а каждое движение тела робота вправо левый называется -1. На рис. 10.6 показаны две политики. Здесь немедленное вознаграждение равно нулю везде, кроме нижней строки и пространства состояний. Левая политика π1 лучше в долгосрочной перспективе, потому что для длинных последовательностей действий средний прогресс на одно действие составляет 3/8 = 0,375 для π1 и 2/6 0,333 для π2. Если мы используем (10.1) (s), результатом будет следующая таблица с начальным состоянием s0 в точке
на стр. 260 для V^{π} вверху
слева и различных значений γ :

γ	0,9	0,8375	0,8
В π1 (c0)	2,52	1,156	В π2 0,77
(c0)	2,39	1,156	0,80

Здесь мы видим, что политика π1 лучше политики π2, когда gamma = 0,9, и наоборот, когда gamma = 0,8. Для γ 0,8375 обе политики одинаково хороши.
Мы можем сноситься, что более шее γ приводит к более шему временному горизонту для оценки политики.

10.4 Итерация значений и динамическое программирование

При простом подходе к перечислению всех политик выполняется много избыточной работы, потому что многие политики по большей части идентичны Они могут лишь незначительно отличаться. Тем не менее, каждая политика создается и оценивается полностью заново.
Это предполагает сохранение промежуточных результатов для частей политик и их повторное использование.

Machine Translated by Google

Этот подход к решению задач оптимизации был представлен как динамическое программирование Ричардом Беллманом еще в 1957 году [Bel57]. Беллман признал, что для оптимальной политики это случай, когда:

> Независимо от начального состояния st и первого действия при , все последующие решения, исходящие из каждого возможного последующего состояния st+1 , должны быть оптимальными.

На основе так называемого принципа Беллмана становится возможным найти глобально оптимальная политика за счет локальной оптимизации отдельных действий. Мы выведем это принцип для MDP вместе с подходящим алгоритмом итерации.

Желаемая оптимальная политика π что удовлетворяет (10.3) на стр. 260 и (10.1) на стр. 260. Перепишем два уравнения и получим

$$B(s_t) = \max_{a_t, a_{t+1}, a_{t+2}, \ldots} (r(s_t, a_t) + \gamma\, r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \cdots). \quad (10.4)$$

Поскольку непосредственная награда r(st, at) зависит только от st и at , но не от последующие состояния и действия, максимизация может быть распределена, что в конечном итоге приводит к следующей рекурсивной характеристике V :

$$\begin{aligned} B(s_t) &= \max_{a_t} [r(s_t, a_t) + \gamma \max_{a_{t+1}, a_{t+2}, \ldots} (r(s_{t+1}, a_{t+1}) + \gamma\, r(s_{t+2}, a_{t+2}) + \cdots)] \\ &= \max_{a_t} [r(s_t, a_t) + \gamma\, V(s_{t+1})]. \end{aligned} \quad (10.5)$$

Уравнение (10.5) получается в результате замены t t + 1 в (10.4). Написано несколько проще:

$$B(s) = \max_a [r(s, a) + \gamma\, V(\delta(s, a))]. \quad (10.6)$$

Из этого уравнения, как и из (10.1) на стр. 260, следует, что для вычисления V (s), им опосредованное вознаграждение добавляется к вознаграждению всех государств-преемников со скидкой фактор γ . Если B (δ(s, a)) известно, то V (s) явно результат простой локальной операции тимизация всех возможных действий a в состоянии s. Это соответствует Беллману. принцип, из-за которого уравнение (10.6) также называют уравнением Беллмана.

Оптимальная политика π (s) выполняет действие в состоянии s , которое приводит к максимальное значение B . Таким образом,

$$\pi(s) = \operatorname{argmax}_a [r(s, a) + \gamma\, V(\delta(s, a))]. \quad (10.7)$$

Из уравнения рекурсии (10.6) правило итерации для аппроксимации V следует прямым образом:

$$\hat{V}_{\text{Против)}}(s) = \max_a [r(s, a) + \gamma\, \hat{V}(\delta(s, a))]. \quad (10.8)$$

Для начала инициализируются приблизительные значения $\hat{V}(s)$ для всех состояний, например со значением ноль. Теперь $\hat{V}(s)$ многократно обновляется для каждого состояния рекурсивно возвращаясь к значению $\hat{V}(\delta(s, a))$ лучшего последующего состояния. Этот процесс расчет V называется итерацией значения и схематически показана на рис. 10.7.

Machine Translated by Google

Рис. 10.7 Алгоритм итерации значения

```
ОЦЕНКА()
Для всех s ∈ S
    V (c) = 0
Повторить
    Для всех s ∈ S
        V̂ (s) = maxa[r(s, a) + γ V̂ (δ(s, a))]
Пока V(s) не изменится
```

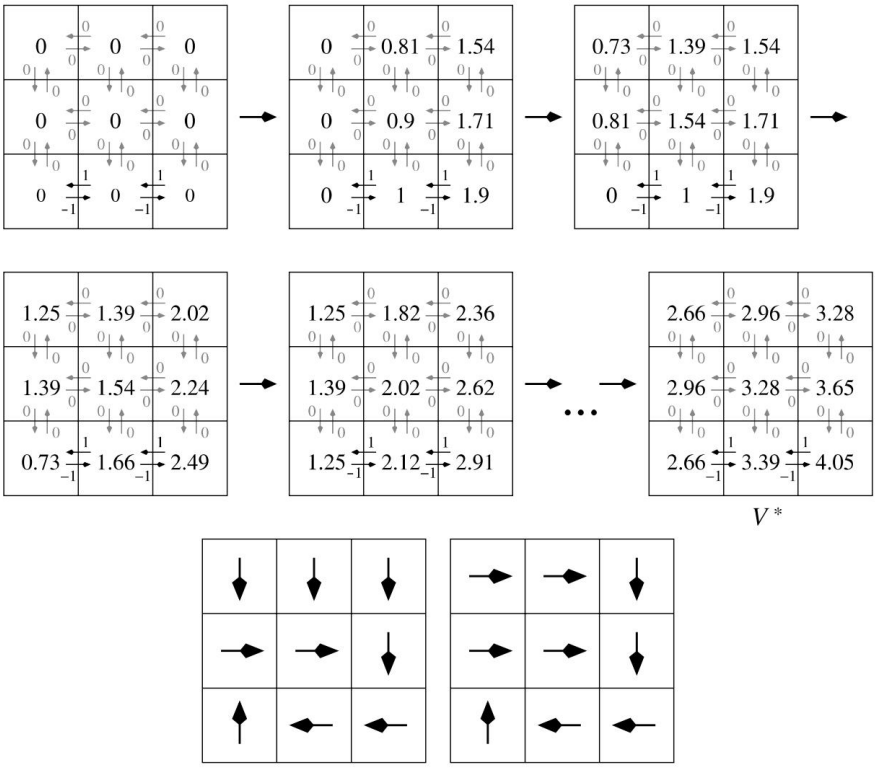


Рис. 10.8. Итерация значения в примере с 3 × 3 состояниями. Последние два изображения показывают два оптимальных политики. Цифры рядом со стрелками показывают немедленное вознаграждение r(s, a) за каждое действие.

на стр. 264. Можно показать, что итерация значения сходится к V [SB98]. Ан отличный анализ алгоритмов динамического программирования можно найти в [Sze10], где, основываясь на свойствах сжатия конкретных алгоритмов (например, итерации значений), сходимость может быть доказана с помощью теории Банаха с фиксированной точкой. рем.

Machine Translated by Google

На рис. 10.8 этот алгоритм применяется к примеру 10.1 на стр. 257 с γ = 0,9.
На каждой итерации состояния обрабатываются построчно снизу слева направо вверх.
Показаны несколько начальных итераций, а на втором изображении в нижнем ряду стабильные предельные значения для V .

Мы ясно видим прогресс обучения в этой последовательности. Агент повторно исследует все состояния, выполняет итерацию значения для каждого состояния и сохраняет политику, которая в виде табличной функции V эффективно , затем может быть дополнительно скомпилирована в виде используемая таблица π .

Кстати, чтобы найти оптимальную политику из V Было бы неправильно выбирать действие в состоянии st , которое приводит к состоянию с максимальным V, ценить. Корре отвечающим (10.7) на стр. 263, необходимо также добавить немедленное вознаграждение r(st, at) , поскольку мы ищем V на рис . 10.8 на (st), а не V (ст+1). Применительно к состоянию s = (2, 3) стр. 264 , это означает

$$\pi(2,3) = \operatorname{argmax}_{a \in \{left,right,up\}} [r(s,a) + \gamma V(\delta(s,a))]$$

$$= \operatorname{argmax}_{\text{влево, вправо, вверх}} \{1 + 0{,}9 \cdot 2{,}66,\ 1 + 0{,}9 \cdot 4{,}05,\ 0 + 0{,}9 \cdot 3{,}28\}$$

$$= \operatorname{argmax}_{\text{влево, вправо, вверх}} \{3{,}39,\ 2{,}65,\ 2{,}95\} = \text{слева}.$$

В (10.7) на стр. 263 мы видим, что агент в состоянии st должен знать немедленное вознаграждение rt и последующее состояние st+1 = δ(st, at) , чтобы выбрать оптимальное действие в .
Он также должен иметь модель функций r и δ. Поскольку для многих практических приложений это не так, нужны алгоритмы, которые также могут работать без знания r и δ. Раздел 10.6 посвящен такому алгоритму.

10.5 Обучаемый шагающий робот и его моделирование

Графический пользовательский интерфейс для простых экспериментов с обучением с подкреплением показан на рис. 10.9 на странице 266 [TEF09]. Пользователь может наблюдать за обучением с подкреплением для двухмерных пространств состояний разного размера. Для лучшего обобщения для сохранения состояния используются сети обратного распространения (см. раздел 10.8). Редактор обратной связи, показанный в правом нижнем углу, с помощью которого пользователь может вручную предоставить обратную связь об окружающей среде, особенно интересен для экспериментов. Не показано меню для настройки параметров итерации значения и обучения обратного распространения.

Помимо моделирования, специально для обучения [TEF09] были разработаны два небольших реальных ползающих робота с одинаковым двухмерным дискретным пространством состояний [TEF09]. Два робота показаны на рис. 10.10 на стр . 266 .

2Дополнительная информация и соответствующие источники о сканирующих роботах доступны на веб-сайте www. hs-weingarten.de/~ertel/kibuch.

Machine Translated by Google

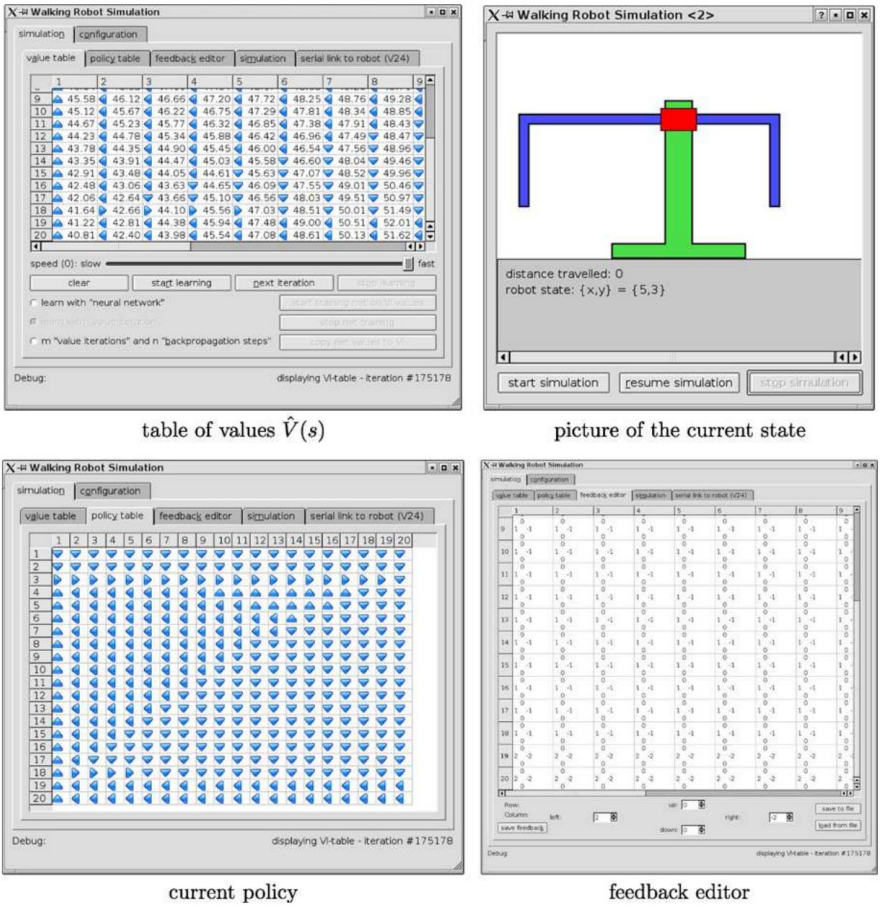


Рис. 10.9 Четыре разных окна симулятора шагающего робота



Рис. 10.10 Два варианта ползающего робота

Machine Translated by Google

атор. Сервоприводы управляются микроконтроллером или через беспроводной интерфейс прямо с ПК. Используя программное обеспечение для моделирования, матрицу обратной связи робота можно визуализировать на ПК. С помощью этой сохраненной обратной связи можно обучить политику на ПК (который вычисляет быстрее), а затем снова загрузить в робота и выполнить. Однако робот также может обучаться автономно. Для пространства состояний размером 5× 5 это занимает около 30 секунд.

Интересно наблюдать разницу между симуляцией и «настоящим» роботом. В отличие от симуляции, краулер изучает правила, при которых он никогда не отрывает руку от земли, но, тем не менее, движется вперед очень эффективно. Причина этого в том, что в зависимости от поверхности подземелья кончик «подмышки» может захватывать землю при движении назад, но проскальзывает при движении вперед. Этот эффект очень чувствительно воспринимается датчиками измерения расстояния и соответственно оценивается в процессе обучения.

Адаптивность робота приводит к удивительным эффектам. Например, мы можем наблюдать, как краулер, несмотря на неисправный сервопривод, который буксует под определенным углом, тем не менее учится ходить (больше похоже на ковыляние). Он даже способен адаптироваться к изменившимся ситуациям путем изменения политики. Весьма желательным эффектом является способность на заданных разногладких поверхностях (например, разных шероховатых коврах) выучить оптимальную политику для каждой из них. Также оказывается, что настоящий робот действительно очень хорошо адаптируется к небольшому пространству состояний размером 5 × 5.

Читатель может (за отсутствием настоящего робота) смоделировать различные поверхности или дефекты сервоприводов, варьируя значения обратной связи и затем наблюдая за результирующими политиками (упражнение 10.3 на стр. 276).

10.6 Q-обучение

Политика, основанная на оценке возможных состояний-последователей, явно непригодна, если у агента нет модели мира, то есть когда он не знает, к какому состоянию приводит возможное действие. В большинстве реальных приложений агент не может прибегать к такой модели мира. Например, робот, который должен захватывать сложные объекты, не может предсказать, будет ли объект надежно удерживаться в его захвате после захвата или останется на месте.

При отсутствии модели мира необходима оценка действия at, осуществляемого в состоянии st , даже если еще неизвестно, куда это действие ведет. Таким образом, теперь мы работаем с функцией оценки Q(st, at) для состояний и связанных с ними действий. С помощью этой функции выбор оптимального действия производится по правилу

$$\pi(s) = \operatorname*{argmax}_a Q(s, a). \tag{10.9}$$

Чтобы определить функцию оценки, мы снова используем пошаговое дисконтирование оценки для пар состояние-действие, которые происходят дальше в будущем, точно так же, как в (10.1) на стр. 260. Таким образом, мы хотим максимизировать $r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$.
Поэтому для оценки действия at в состоянии st определим по аналогии с (10.4) на

Machine Translated by Google

стр. 263:

$$Q(st, at) = \max_{at+1,\, at+2,\ldots} (r(st, at) + \gamma\, r(st+1, at+1) + \gamma^2 r(st+2, at+2) + \cdots). \quad (10.10)$$

Аналогично подходу для итерации значений, мы приводим это уравнение к простой рекурсивной форме с помощью

$$\begin{aligned} Q(st, at) &= \max_{at+1,\, at+2,\ldots} (r(st, at) + \gamma\, r(st+1, at+1) + \gamma^2 r(st+2, at+2) + \cdots) \\ &= r(st, at) + \gamma \max_{at+1,\, at+2,\ldots} (r(st+1, at+1) + \gamma\, r(st+2, at+2) + \cdots) \\ &= r(st, at) + \gamma \max_{at+1} (r(st+1, at+1) + \gamma \max_{at+2} (r(st+2, at+2) + \cdots)) \\ &= r(st, at) + \gamma \max_{at+1} Q(st+1, at+1) \\ &= r(s, a) + \gamma \max_{a'} Q(\delta(s, a), a'). \end{aligned} \quad (10.11)$$

В чем тогда преимущество по сравнению с итерацией значений? Старое уравнение лишь немного переписано, но это оказывается как раз правильным подходом к новому алгоритму. Вместо сохранения V теперь сохраняется функция Q , и агент может выбирать свои действия из функций δ и r без модели мира. Мы однако до сих пор нет процесса, который мог бы изучить Q напрямую, то есть без знания V. Из ·

рекурсивной формулировки Q(s, a) можно вывести итерационный алгоритм для определения Q(s, a) в прямой манере. Произвольно инициализируем таблицу Q(s, a) для всех состояний, например нулями, и итеративно выполняем

$$\hat{Q}(s, a) = r(s, a) + \gamma \max_{a'} \hat{Q}(\delta(s, a), a'). \quad (10.12)$$

Остается заметить, что функции r и δ нам неизвестны . Мы решаем эту проблему достаточно прагматично, позволяя агенту в его окружении в состоянии s выполнять действие a. Затем состояние-преемник, очевидно, представляет собой δ(s, a) , и агент получает свое вознаграждение от окружающей среды. Алгоритм, показанный на рис. 10.11 на стр. 269 , реализует этот алгоритм Q-обучения.

Применение алгоритма к примеру 10.1 на стр . 257 с γ = 0,9 и nx = 3, ny = 2 (то есть в сетке 2 × 3) показано в качестве примера на рис. 10.12 на стр . 269 . На первом рисунке все значения Q инициализированы нулями. На втором рисунке после первой последовательности действий четыре значения r , не равные нулю, становятся видимыми как значения Q. На последнем рисунке дана изученная оптимальная полити Следующая теорема, доказательство которой можно найти в [Mit97], показывает, что этот алгоритм сходится не только в примере, но и вообще.

Machine Translated by Google

Q-ОБУЧЕНИЕ ()

Для всех s S, a A

Q(s, a) = 0 (или случайно)

Повторить

Выбрать (например, случайным образом)

состояние

s Повторить Выбрать действие a и выполнить

его Получить награду r и новое состояние s

Q(s, a) := r(s, a) + γ maxa′ Q(s , a')

′ c := c

До тех пор, пока s не станет конечным состоянием или не будет достигнут предел времени

Пока Q^ не сходится

Рис. 10.11 Алгоритм Q-обучения

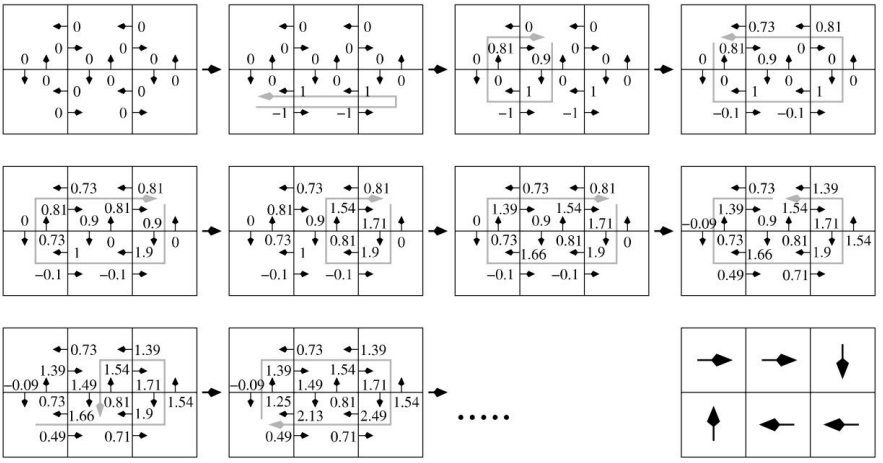


Рис. 10.12 Q-обучение применительно к примеру с nx = 3, ny = 2. Серыми стрелками отмечены действия, выполняемые на каждой картинке. Приведены обновленные значения Q. На последней картинке показана текущая политика, которая также является оптимальной.

Теорема 10.1. Пусть дан детерминированный MDP с ограниченным немедленным вознаграждением r(s, a) . Уравнение (10.12) на стр. 268 с 0 γ < 1 используется для обучения. Пусть Q^ n(s, a) будет значением Q(s, a) после n обновлений. Если каждую пару состояний-действий посещают бесконечно часто, то Q^ n(s, a) сходится к Q(s, a) для всех значений s и a при n .

Machine Translated by Google

Доказательство Поскольку каждый переход между состояниями и действиями происходит бесконечно часто, мы рассматриваем последовательные интервалы времени с тем свойством, что в каждом интервале все переходы между состояниями и действиями происходят хотя бы один раз. Покажем теперь, что максимальная ошибка для всех записей в таблице Qˆ уменьшается по крайней мере в γ раз на каждом из этих интервалов. Позволять

$$_{н} = \max_{с, а} |Qˆ n(s, a) \quad Q(s, a)|$$

– максимальная ошибка в таблице Qˆ n и с ′ = δ(s, a). Для каждой записи таблицы Qˆ n(s, a) мы вычисляем его вклад в ошибку через интервал как

$$|Qˆ n+1(s, a) \quad Q(s, a)| = r + γ\max_{′a} . Qˆ n(s′ , a′) \quad r + γ \max_{′a} Q(s' , a')$$

$$= γ\max_{a′} . Qn (s′ , a′) \quad \max_{a′} Q(s' , a')$$

$$γ \max_{′a} |Qˆ n(s′ , a′) \quad Q(s′ , a′)|$$

$$γ \max_{s′′,a′} |Qˆ n(s′′, a′) \quad Q(s′′, a′)| = γн .$$

Первое неравенство верно, поскольку для произвольных функций f и g

$$\max_{x} f (x) \quad \max_{Икс} g(x) \quad \max_{Икс} |f (x) \quad g(x)|$$

а второе неравенство верно, так как при дополнительном варьировании состояния s ′′ результирующий максимум не может стать меньше. Таким образом, было показано, что n+1 γ н. Поскольку ошибка на каждом интервале уменьшается не менее чем в γ раз , через k интервалов она не превосходит γ k0, и, как следствие, 0 ограничено. Поскольку каждое состояние посещается бесконечно много раз, существует бесконечно много интервалов и n сходится до нуля.

Согласно теореме 10.1 на стр. 269 Q-обучение сходится независимо от действий, выбранных при обучении. Это означает, что для сходимости не имеет значения, какие действия выбирает агент, лишь бы каждое из них выполнялось бесконечно часто.
Однако скорость сходимости, безусловно, зависит от того, какие пути выбирает агент во время обучения (см. раздел 10.7).

10.6.1 Q-обучение в недетерминированной среде

Во многих приложениях робототехники среда агента недетерминирована. Это означает, что реакция среды на действие a в состоянии s в два разных момента времени может привести к различным последующим состояниям и вознаграждениям. Такой недетерминированный марковский процесс моделируется вероятностной функцией перехода δ(s, a) и вероятностным немедленным вознаграждением r(s, a). Чтобы определить функцию Q , каждый раз ожидаемое значение должно вычисляться для всех возможных последующих состояний. Уравнение (10.11)

Machine Translated by Google

на стр. 268 , таким образом, обобщается на

$$Q(st, at) = E(r(s, a)) + \gamma \sum_{s'} P(s' | s, a) \max_{a'} Q(s', a'), \quad (10.13)$$

где $P(s' | s, a)$ — вероятность перехода из состояния s в последующее состояние s с действием a. К сожалению, нет гарантии сходимости для Q-обучения в недетерминированном случае, если мы будем действовать, как и раньше, в соответствии с (10.12) на стр. 268.
Это связано с тем, что при последовательных прогонах внешнего цикла алгоритма на рис. 10.11 на стр. 269 вознаграждение и последующее состояние могут быть совершенно разными для одного и того же состояния s и одного и того же действия a. Это может привести к чередующейся последовательности, которая перескакивает между несколькими значениями. Чтобы избежать таких сильных скачков значений Q , мы добавляем старое взвешенное значение Q в правую часть (10.12) на стр. 268. Это стабилизирует итерацию. Затем правило обучения читается

$$\hat{Q}_n(s, a) = (1 - \alpha_n)\hat{Q}_{n-1}(s, a) + \alpha_n [r(s, a) + \gamma \max_{a'} \hat{Q}_{n-1}(\delta(s, a), a')] \quad (10.14)$$

с изменяющимся во времени весовым коэффициентом

$$\alpha_n = \frac{1}{1 + b_n(s, a)}.$$

Значение $b_n(s, a)$ показывает, как часто действие a выполнялось в состоянии s на n-й итерации. При малых значениях b_n (т. е. в начале обучения) стабилизирующий член $\hat{Q}_{n-1}(s, a)$ не играет роли, так как мы хотим, чтобы процесс обучения шел быстро. Однако позже b_n становится больше и тем самым предотвращает чрезмерно большие скачки в последовательности значений $\hat{Q}$. При интегрировании (10.14) в Q-обучение значения $b_n(s, a)$ должны быть сохранены для всех пар состояние-действие. Этого можно добиться, расширив таблицу значений $\hat{Q}$.

Для лучшего понимания (10.14) мы упростим его, предположив, что $\alpha_n = \alpha$ является постоянной и преобразуя ее следующим образом:

$$\hat{Q}_n(s, a) = (1 - \alpha)\hat{Q}_{n-1}(s, a) + \alpha [r(s, a) + \gamma \max_{a'} \hat{Q}_{n-1}(\delta(s, a), a')]$$

$$= \hat{Q}_{n-1}(s, a) + \alpha [r(s, a) + \gamma \max_{a'} \hat{Q}_{n-1}(\delta(s, a), a') - \hat{Q}_{n-1}(s, a)].$$

TD-ошибка

Новое значение Q $\hat{Q}_n(s, a)$ может быть ясно представлено как старое $\hat{Q}_{n-1}(s, a)$ плюс α, умноженное на поправочный член, который совпадает с изменением значения Q на этом шаге. Поправочный член называется TD-ошибкой или ошибкой временной разности, а приведенное выше уравнение для изменения значения Q является частным случаем TD-обучения, важного класса алгоритмов обучения [SB98]. Для $\alpha = 1$ мы получаем Q-обучение, описанное выше. При $\alpha = 0$ значения $\hat{Q}$ совершенно не меняются. Таким образом, никакого обучения не происходит.

Machine Translated by Google

10.7 Разведка и разработка

Для Q-обучения до сих пор была дана только грубая схема алгоритма. Особенно
отсутствует описание выбора начального состояния каждый раз и действий, которые необходимо выполнить
во внутреннем цикле рис. 10.11 на стр. 269. Для выбора следующего действия есть две возможности. Среди
возможных действий
один может быть выбран случайным образом. В долгосрочной перспективе это приводит к равномерному исследованию
всех возможных действий или политик, но с очень медленной конвергенцией. Альтернатива
к этому относится использование ранее изученных значений Q^. Здесь агент всегда
выбирает действие с наибольшим значением Q^. Это приводит к относительно быстрой сходимости конкретной
траектории. Однако другие пути остаются непройденными на протяжении всего пути.
к концу. В крайнем случае мы можем получить неоптимальную политику. Поэтому в Теореме 10.1 на стр. 269
требуется, чтобы посещалась каждая пара состояние-действие.
бесконечно много раз. Рекомендуется использовать комбинацию разведки и
эксплуатация с высокой долей разведки в начале и уменьшать ее все больше и больше
больше со временем.

Выбор начального состояния также влияет на скорость обучения. Во-первых
На трех рисунках на рис. 10.12 на стр. 269 хорошо видно, что на первых итерациях изменяются только значения
Q в непосредственной близости от пар состояние-действие
немедленной наградой. Если начать дальше от такой точки,
ненужная работа. Это предполагает передачу априорных знаний о состоянии-действии
пары с немедленным вознаграждением в начальные состояния рядом с этими точками. В ходе
обучения, то можно выбрать более отдаленные начальные состояния.

10.8 Аппроксимация, обобщение и сходимость

Поскольку Q-обучение было описано до сих пор, таблица со всеми значениями Q явно
сохраняется в таблице. Это возможно только при работе с конечным пространством состояний с
конечное число действий. Однако если пространство состояний бесконечно, например, в случае
непрерывных переменных, то невозможно ни сохранить все значения Q, ни посетить
все пары состояние-действие во время обучения.

Тем не менее, существует простой способ использования Q-обучения и итерации значений непрерывных
переменных. Таблица Q(s, a) заменяется нейронной сетью, например
сеть обратного распространения с входными переменными s, a и значением Q в качестве цели
выход. Для каждого обновления значения Q нейронной сети предоставляется обучающая программа.
пример с (s, a) в качестве входных данных и Q(s, a) в качестве целевого вывода. В конце мы имеем конечное
представление функции Q(s, a). Поскольку у нас есть только конечное множество
обучающих примеров, но функция Q(s, a) определена для бесконечного множества входных данных, мы
таким образом, автоматически получают обобщение, если размер сети выбран соответствующим образом (см.
главу 9). Вместо нейронной сети мы также можем использовать другую контролируемую
алгоритм обучения или аппроксиматор функции, такой как машина опорных векторов или
гауссовский процесс.

Machine Translated by Google

Однако переход от конечного числа обучающих примеров к непрерывной функции в определенных ситуациях может стать очень дорогостоящим. Q-обучение с аппроксимацией функций может не сходиться, потому что теорема 10.1 на стр. 269 верна только в том случае, если каждая пара состояние-действие посещается бесконечно часто.

Однако проблемы сходимости также могут возникнуть в случае конечного числа пар состояние-действие, когда Q-обучение используется в POMDP. Q-обучение может быть применено — в обоих описанных вариантах — к детерминированным и недетерминированным марковским процессам (MDP). Для POMDP может случиться так, что агент, например, из-за зашумленных датчиков, воспринимает множество разных состояний как одно. Часто многие состояния в реальном мире целенаправленно сопоставляются с одним так называемым наблюдением. Результирующее пространство наблюдения становится намного меньше пространства состояний, благодаря чему обучение становится быстрее и можно избежать переобучения (см. раздел 8.4.7).

Однако, объединяя вместе несколько состояний, агент больше не может различать фактические состояния, и действие может привести его ко многим различным последующим состояниям, в зависимости от того, в каком состоянии он действительно находится. Это может привести к проблемам конвергенции для ценности. итерации или для Q-обучения. В литературе (например, в [SB98]) предлагается множество различных подходов к решению.

Также очень многообещающими являются так называемые методы улучшения политики и производные от них методы градиента политики, в которых значения Q не изменяются, а политика изменяется напрямую. В этой схеме в пространстве всех политик ищется политика, максимизирующая совокупное дисконтированное вознаграждение ((10.1) на стр. 260). Одна из возможностей добиться этого — следовать градиенту совокупного вознаграждения до максимума. Политика, найденная таким образом, явно оптимизирует совокупное вознаграждение. В [PS08] показано, что этот алгоритм может значительно ускорить обучение в приложениях с большими пространствами состояний, таких как те, которые используются для роботов-гуманоидов.

10.9 Приложения

Тем временем практическая полезность обучения с подкреплением неоднократно демонстрировалась. Из большого количества примеров этого кратко представим небольшую подборку.

TD-обучение вместе с сетью обратного распространения с 40–80 скрытыми нейронами очень успешно использовалось в TD-gammon, программе для игры в нарды [Tes95]. Единственной немедленной наградой за программу является результат в конце игры. Оптимизированная версия программы с опережением на два хода была обучена сама на себе в 1,5 млн партий. Он продолжал побеждать игроков и игроков мирового класса, а также трех лучших игроков-людей.

Есть много приложений в робототехнике. Например, в лиге футбольных симуляторов RoboCup лучшие футбольные команды роботов теперь успешно используют обучение с подкреплением [SSK05, Robb]. Балансировка шеста, которая относительно проста для человека, много раз успешно решалась с помощью обучения с подкреплением.

Впечатляющую демонстрацию способности роботов к обучению продемонстрировал Расс Тедрейк на IROS 2008 в своей презентации о модели самолета, которая учится приземляться в определенной точке, как птица, садящаяся на ветку [Ted08]. Потому что воздух

Machine Translated by Google

течения становятся очень турбулентными во время такого динамичного захода на посадку, соответствующее дифференциальное уравнение, уравнение Навье – Стокса, неразрешимо. Следовательно, приземление нельзя контролировать классическим математическим способом. Комментарий Тедрака по этому поводу:

> «Птицы не решают Навье-Стокса!»

Очевидно, что птицы могут научиться летать и приземляться даже без уравнения Навье-Стокса. Тедрейк показал, что теперь это возможно и для самолетов.

Сегодня также можно научиться управлять настоящим автомобилем всего за 20 минут, используя Q-обучение и аппроксимацию функций [RMD07]. Этот пример показывает, что реальные промышленные приложения, в которых несколько измерений должны быть сопоставлены с действиями, можно очень хорошо изучить за короткое время.

Настоящие роботы по-прежнему испытывают трудности с обучением в многомерных пространствах состояния-действия, потому что по сравнению с симуляцией настоящие роботы относительно медленно получают обратную связь от окружающей среды. Таким образом, из-за ограничений по времени невозможно реализовать многие миллионы необходимых тренировочных циклов. Здесь, кроме алгоритмов быстрого обучения, нужны методы, которые позволяют хотя бы части обучения происходить оффлайн, то есть без обратной связи со средой.

10.10 Проклятие размерности

Несмотря на успехи последних лет, обучение с подкреплением остается активной областью исследований в области ИИ, не в последнюю очередь потому, что даже лучшие алгоритмы обучения, известные сегодня, по-прежнему непрактичны для многомерных пространств состояний и действий из-за их гигантского времени вычислений. Эта проблема известна как «проклятие размерности».

В поисках решения этой проблемы ученые наблюдают за животными и людьми во время обучения. Здесь мы замечаем, что обучение в природе происходит на многих уровнях абстракции. Ребенок сначала осваивает простые моторные и языковые навыки на самом низком уровне. Когда они хорошо изучены, они сохраняются и позже могут быть вызваны в любое время и использованы. В переводе на язык информатики это означает, что каждая изученная способность инкапсулируется в модуль, а затем, на более высоком уровне, представляет собой действие. Используя такие сложные действия на более высоком уровне, пространство для действий значительно сокращается, и, таким образом, обучение ускоряется. Аналогичным образом можно абстрагировать состояния и, таким образом, уменьшить пространство состояний. Такое обучение на нескольких уровнях называется иерархическим обучением [BM03].

Другой подход к модуляризации обучения — это распределенное обучение или мультиагентное обучение [PL05]. При обучении двигательным навыкам робота-гуманоида необходимо одновременно управлять до 50 различными двигателями, что приводит к 50-мерному пространству состояний и 50-мерному пространству действий. Чтобы уменьшить эту гигантскую сложность, централизованное управление заменяется распределенным управлением. Например, каждый отдельный двигатель может иметь индивидуальное управление, которое управляет им напрямую, если возможно, независимо от других двигателей. В природе мы находим этот вид контроля у насекомых. Например, многочисленными ногами многоножки управляет не центральный мозг, а каждая пара ног имеет свой крошечный «мозг».

Machine Translated by Google

Подобно неосведомленному комбинаторному поиску, обучение с подкреплением имеет задачу найти лучшее из огромного количества политик. Задача обучения значительно упрощается, если у агента есть более или менее хорошая политика до начала обучения.
Тогда задачи многомерного обучения могут быть решены раньше. Но как найти такую начальную политику? Здесь есть две основные возможности.

Первая возможность — это классическое программирование. Программист предоставляет агенту политику, включающую программу, которую он считает хорошей. Затем происходит переключение, например, на Q-learning. Агент выбирает, по крайней мере в начале обучения, свои действия в соответствии с запрограммированной политикой и таким образом попадает в «интересные» области пространства состояния-действия. Это может привести к резкому сокращению пространства поиска обучения с подкреплением.

Если традиционное программирование становится слишком сложным, мы можем начать обучение робота или агента, заставив человека предписывать правильные действия. В простейшем случае это делается с помощью ручного дистанционного управления роботом. Затем робот сохраняет запрещенное действие для каждого состояния и обобщает, используя алгоритм обучения с учителем, такой как обратное распространение или обучение дерева решений. Это так называемое демонстрационное обучение [BCDS08, SE10], таким образом, также обеспечивает начальную политику для последующего обучения с подкреплением.

10.11 Резюме и перспективы

Сегодня у нас есть доступ к хорошо функционирующим и проверенным алгоритмам обучения для обучения наших машин. Однако задача инструктора или разработчика по-прежнему требует сложных приложений. Существует очень много возможностей, как структурировать обучение робота, и без экспериментов оно не будет успешным. На практике это экспериментирование может быть очень утомительным, потому что каждый новый учебный проект должен быть разработан и запрограммирован. Здесь необходимы инструменты, которые, помимо различных алгоритмов обучения, также предлагают тренеру возможность комбинировать их с традиционным программированием и демонстрационным обучением. Одним из первых инструментов такого рода является Teaching-Box [ESCT09], который в дополнение к обширной библиотеке программ также предлагает шаблоны для конфигурации учебных проектов и для связи между роботом и окружающей средой. Например, учитель-человек может дать роботу дополнительную обратную связь с клавиатуры или через речевой интерфейс в дополнение к обратной связи из окружающей среды.

Обучение с подкреплением — увлекательная и активная область исследований, которая будет все чаще использоваться в будущем. Все больше и больше систем управления роботами, а также другие программы будут учиться на основе обратной связи с окружающей средой. На сегодняшний день существует множество вариаций представленных алгоритмов, а также совершенно разные алгоритмы. Проблема масштабирования остается нерешенной. Для небольших пространств действий и состояний с небольшим количеством степеней свободы можно добиться впечатляющих результатов. Если число степеней свободы в пространстве состояний вырастет до 18, например, для простого робота-гуманоида, то обучение станет очень дорогим.

Machine Translated by Google

Для дальнейших основополагающих лекций мы рекомендуем компактное введение в обучение с подкреплением в книге Тома Митчелла [Mit97]. Стандартная работа Саттона и Барто [SB98] является тщательной и всеобъемлющей, как и обзорная статья Кельблинга, Литтмана и Мура [KLM96].

10.12 Упражнения

Упражнение 10.1

(a) Подсчитайте количество различных политик для n состояний и n действий. Таким образом, возможны переходы из каждого состояния в каждое состояние. (б) Как изменится количество политик в подзадаче (а), если пустые действия, т. е. действия из одного состояния в себя, не допускаются. (c) С помощью стрелочных диаграмм, подобных тем, что показаны на рис. 10.3 на стр. 259, укажите все политики для два состояния.

(d) Используя стрелочные диаграммы, укажите все политики без пустых действий для трех состояний.

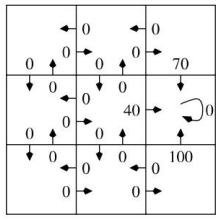
Упражнение 10.2. Использование итерации значения вручную в примере 10.1 на стр. 257 с nx = ny = 2.

Упражнение 10.3. Проведите различные эксперименты, используя симулятор итерации значений. (а) Установите симулятор итерации значений из [Tok06]. (b) Воспроизведите результаты упражнения 10.2 , сначала введя обратную связь с редактор обратной связи, а затем выполнить итерацию значения. (c) Моделируйте поверхности разной гладкости и наблюдайте, как изменяется политика. (d) Используя аналогичную матрицу обратной связи, постепенно увеличивайте пространство состояний примерно до 100 × 100 и подбирайте коэффициент дисконтирования γ таким образом, чтобы в результате получалась разумная политика.

Упражнение 10.4 . Покажите, что для примера расчета на рис. 10.8 на стр. 264) 4,05499. точное значение V (3, 3) = 1,9/(1 0,9 6

Упражнение 10.5

Проведите Q-обучение на сетке 3× 3 справа. Состояние в середине справа — это поглощающее целевое состояние.



Упражнение 10.6 Дана рука робота с n шарнирами (размерами) и ℓ дискретными состояниями на сустав. Возможны действия из каждого состояния в каждое состояние (если робот ничего не делает, это оценивается как (пустое) действие).

Machine Translated by Google

a) Приведите формулу количества состояний и количества действий в каждом состоянии.
и за количество полисов для робота.

(b) Создайте таблицу с количеством стратегий для n = 1, 2, 3, 4, 8 и ℓ =
1, 2, 3, 4, 10.

(c) Чтобы уменьшить количество возможных стратегий, предположим, что количество
возможных действий на сустав всегда равно 2 и что робот может перемещать только
один сустав за раз. Введите новую формулу для количества стратегий и создайте
соответствующую

таблицу. (d) С помощью вычисленного результата обосновать, что агент, который
действует автономно и адаптивно при n = 8 и l = 10, безусловно, может быть назван разумным.

Machine Translated by Google

Machine Translated by Google

Решения для упражнений 11

11.1 Введение

Упражнение 1.3. Многие известные процессы вывода, процессы обучения и т. д. являются NP-полными или даже неразрешимыми. Что это означает для ИИ?

Упражнение 1.4. Если проблема является NP-полной или может быть описана как «сложная», это означает, что есть случаи, в которых проблема не может быть решена за приемлемое время. Это так называемый худший случай. В некоторых приложениях приходится мириться с тем фактом, что в худшем случае эффективное решение невозможно. Это означает, что и в будущем будут практически значимые проблемы, которые в некоторых частных случаях неразрешимы.

Поэтому ИИ не найдет универсальной формулы и не построит супермашину, с помощью которой все проблемы станут решаемыми. Он скорее ставит перед собой задачу построения систем с более высокой вероятностью нахождения решения или с более высокой вероятностью нахождения быстрых оптимальных решений. Мы, люди, в повседневной жизни довольно хорошо справляемся с неоптимальными решениями. Причина проста — чрезмерная стоимость поиска оптимального решения. Например, мне требуется всего семь минут, чтобы найти дорогу из пункта А в пункт Б по карте в незнакомом городе. Самый короткий путь занял бы всего шесть минут. Однако поиск кратчайшего пути занял бы около часа. Доказательство оптимальности пути может обойтись еще дороже.

Упражнение

1.5 (a) Выход зависит не только от ввода, но и от содержимого памяти. Для входа x, в зависимости от содержимого памяти, выход может быть y1 или y2. Таким образом, он не уникален и, следовательно, не является функцией. (b) Если рассматривать содержимое памяти как дополнительный вход, то выход будет уникальным (поскольку агент детерминирован) и агент представляет собой функцию.

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по компьютерным
наукам, DOI 10.1007/978-0-85729-299-5_11, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google

Упражнение 1.6

(а) Скорость vx (t) = $\frac{x}{t}$ limt 0 $\frac{x(t) \quad x(t \quad t) =}{\tau}$ $\frac{x(\tau) - x(\tau - \tau)}{\tau}$. vy рассчитывается аналогично.

(t)2 (t)2 2x vx (t) vx (t t) (б) Ускорение ax(t) = vx(t) = limt 0 t2 x(t) x(t t) x(t t) x(t t) x(t 2t) "=" ay вычисляется аналогично. Также нужна позиция в три момента времени t 2t, t t, t. "=" x(t) 2x(t t)+x(t 2t) limt 0 .

Упражнение

1.7 (а) Затраты агента 1 = 11 · 100 центов + 1 · 1 цент = 1, 101 цент.

Затраты на 2 = 0 · 100 центов + 38 · 1 цент = 38 центов.

Следовательно, агент 2 экономит 1 101 цент - 38 центов = 1 063 цента. (b)

Прибыль агента 1 = 189 · 100 центов + 799 · 1 цент = 19 699 центов.

Прибыль агента 2 = 200 · 100 центов + 762 · 1 цент = 20 762 цента.

Таким образом, агент 2 имеет прибыль на 20 762 цента 19 699 центов = на 1 063 цента больше.

Если кто-то оценивает упущенную выгоду из-за ошибок, агент, основанный на полезности, принимает те же решения, что и агент, основанный на затратах.

11.2 Пропозициональная логика

Упражнение 2.1 С сигнатурой = {σ1, σ2,...,σn} и формулой грамматических переменных синтаксис логики высказываний можно определить следующим образом:

формула ::= σ1 |σ2 |··· |σn |w |f

| ¬формула |(формула)| формула формула

| формула формула |формула формула

| формула формула

Упражнение 2.2. Доказательство методом таблицы истинности.

Упражнение 2.3 (а) (¬A B) (¬B A) (b) (¬A B) (¬B A) (c) t

Упражнение 2.4 (а) выполнимо (б) верно (в) невыполнимо

Упражнение

2.6 (а) В упражнении 2.3 (c) уже было показано, что A (A B) B является тавтологией.

Таким образом, теорема дедукции гарантирует правильность правила вывода.

(b) Мы показываем методом таблицы истинности, что (A B) (¬B C) (A C) есть тавтология.

Machine Translated by Google

Упражнение 2.7. Применение правила разрешения к клаузулам (f B) и (¬B f) дает резольвенту (f f) (f). Теперь применим правило разрешения к предложениям B и ¬B и получим пустое предложение в качестве резольвенты. Поскольку (f B) B и (¬B f) ¬B, (f) (). На практике важно, что всякий раз, когда образуется пустое предложение, это происходит из-за противоречия.

Упражнение 2.8. Если KB содержит противоречие, то есть два предложения A и ¬A, которые позволяют вывести пустое предложение. Противоречие в KB явно остается в KB ¬Q. Следовательно, это также позволяет получить пустое предложение.

Упражнение 2.9 (a) (A B) (¬A ¬B) (b) (A B) (B C) (A C)

Упражнение 2.10 Формализация: Сообщник: A, Автомобиль: C, Ключ: K

WB (A C) [(¬A ¬K) (A K)] K

Преобразование в КНФ: (¬A ¬K) (A K) (¬K A) (¬A K)

Попробуйте доказать C и добавьте ¬C к набору предложений. Набор предложений CNF

(¬A C)1 (¬K A)2 (¬A K)3 (K)4 (¬C)5.

Доказательство разрешения : Res(2, 4):
(A)6 Res(1, 6): (C)7
Res(7, 5): ()8
Таким образом, C было показано.

Упражнение
2.11 (a) KB (A B) (¬B C), Q (A C)
KB ¬Q (A B)1 (¬B C)2 (¬A)3 (¬C)4
Доказательство разрешения : Res (1, 3): (B) 5
Res(2, 4): (¬B)6
Res(5, 6) : () (b)
¬(¬B (B ¬A) ¬A) (¬B)1 (B ¬A)2 (A)3
Доказательство разрешения : Res(1, 2): (¬A)4
Рез (3, 4): ()

Упражнение 2.12. Применяя эквивалентности из теоремы 2.1 на стр. 18, мы можем сразу доказать утверждения.

Упражнение 2.13

Res(8, 9): (C F E f)10
Res(3, 10): (F E f)11
Res(6, 11): (A B C E f)12
Res(1, 12): (B C E f)13
Res(2, 13): (C E f)14
Res(3, 14): (E f)15
Рез(5, 15) : ()

Machine Translated by Google

11.3 Логика предикатов первого порядка

Упражнение

3.1 (a) x мужчина(x)
¬женщина(x) (b) x y z отец(x, y) мужчина(x)
ребенок(y, x, z) (c) x y siblings(x, y) [(z отец(z, x) отец(z, y)) (z мать(z, x)
мать(z, y))] (d)
x y z родители(x, y, z) отец(x, z) мать(y, z) (e) x y
дядя(x, y) z u ребенок(y, z, u) братья и сестры(z, x) мужчина(x) (f) x
y предок(x, y) z ребенок(y, x, z) u v дочерний(u, x, v)
предок (и, y))

Упражнение

3.2 (a) x y z отец(y, x) мать(z, x) (b)
x y ребенок(y, x, z) (c)
x птица(x) мухи (x) (d)
x y z животное(x) животное(y) ест(x, y) ест(y, z) зерно(z) (e) x
животное(x) (y (ест(x, y) (растение(y) (животное(y) z растение(z) ест(y, z)
намного_меньше(y, x)))

Упражнение 3.3 x y zx = отец(y) мужчина(x) ребенок(y, x, z)

Упражнение
3.4 x y x<y y<x x = y, x
y x<y ¬y<x, x y z
x<y y<z x<z

Упражнение 3.5

(a) МГУ: x/f (z), u/f (y), терм: p(f (z), f (y)) (б) не
унифицируемый
(c) МГУ: x/ cos y, z/ 4 7 · cos y, терм: cos y = 4 7 · cos y (d) не
унифицируемый
(e) MGU: u/f (g(w, w), g(g(w, w), g(w, w)), g(g(g(w, w), g(w, w)), g(g(w, w), g(w, w)))), x/g(w, w),
y/g(g(w, w), g(w, w)), z/g(g(g(w, w), g(w, w)), g(g(w, w), g(w, w))) член: q(f (g(w, w), g(g(w,
w), g(w, w)), g(g(g(w, w) ,
g(w, w)), g(g(w, w), g(w, w)))), f (g(w, w), g(g(w, w), g(w, w)), g(g(g(w, w), g(w, w)), g(g(w,
w), g(w, w)))))

Упражнение

3.7 (a) Пусть задана невыполнимая формула p(x) ¬p(x) r(x). Мы выбираем пункт r(x)
в качестве SOS, поэтому противоречие не может быть
получено. (b) Если SOS уже невыполнимо, то никакого противоречия не может быть
получено. Если нет, то необходимы шаги разрешения между предложениями из
SOS и (KB ¬Q)\SOS.

Machine Translated by Google

(c) Если нет дополнения к литералу L в предложении K, то литерал L останется в каждом предложении, полученном с помощью разрешения из предложения K. Таким образом, пустое предложение не может быть получено из K или его резольвенты, как и любой будущая решимость.

Упражнение 3.8 ¬Q KB (e = n)1 (n · x = n)2 (e · x = x)3 (¬a = b)4 Доказательство: Dem(1, 2) : (e · x = e)5 Tra,Sym(3, 5) : (x = e)6 Dem(4, 6) : (¬e = b)7 Dem(7, 6) : (¬e = e)8 ( )

Здесь «Dem» означает демодуляцию. Пункт номер 6 был получен путем применения транзитивности и симметрии равенства в пунктах 3 и 5.

Упражнение 3.9. Входные файлы LOP:

а) a; b<-. <- a,b. b;c<-. <-b, в. c; a<-. <-c, a.

(b) <- бреет(бородка,X), бреет(X,X). бреет(колючка,X); бреет(X,X) <-.

в) e = n. <- a = b. m(m(X,Y),Z)=m(X,m(Y,Z)). m(e,X) = X. m(n,X) = n.

11.4 Ограничения логики

Упражнение

4.1 (a) Правильно: возьмем полное доказательство для PL1. С его помощью мы находим доказательство каждой истинной формулы за конечное время. Для всех невыполнимых формул я поступаю следующим образом: я применяю исчисление к ¬φ и показываю, что ¬φ истинно. Таким образом, φ ложно. Таким образом, мы можем доказать каждую истинную формулу из PL1 и опровергнуть любую ложную формулу. К сожалению, этот процесс непригоден для выполнимых формул.

Упражнение 4.2

(a) Он бреется тогда и только тогда, когда он не бреется. (противоречие) (b) Множество всех множеств, которые не содержат самих себя. Он содержит себя тогда и только тогда, когда если он не содержит себя.

11.5 ПРОЛОГ

Упражнение 5.1 PROLOG сигнализирует о переполнении стека. Причина в поиске в глубину PROLOG, который всегда выбирает первый унифицируемый предикат во входном файле.
С рекурсивными предикатами, такими как равенство, это вызывает непрерывную рекурсию.

Machine Translated by Google

Упражнение
5.2. write_path([H1,H2|T]): -- write_path([H2|T]), write_move(H2,H1). путь_записи([_X]).

```
write_move(состояние(X,W,Z,K), состояние(Y,W,Z,K)) :-
        написать('Фермер из'), написать(X), написать('в'), написать(Y), nl.
write_move(состояние(X,X,Z,K), состояние(Y,Y,Z,K)) :-
        write('Фермер и волк из '), write(X), write(' в '), write(Y),nl.
write_move(состояние(X,W,X,K), состояние(Y,W,Y,K)) :-
        написать('Фермер и коза из'),написать(X),написать(' в '), написать(Y), nl.
write_move(состояние(X,W,Z,X), состояние(Y,W,Z,Y)) :-
        написать('Фермер и капуста из'),написать(X),написать(' в '), написать(Y), nl.
```

Упражнение

5.3 (a) Требуется вывести найденные решения. Унификация в фактическом плане(Цель, Цель,Путь,Путь). позаботится об этом. (b) Условия вхождения в предложения предиката safe перекрывают друг друга. Возврат, вызванный сбоем, приводит к выполнению второй или третьей альтернативы safe, где явно будет найдено то же самое решение. Сокращение в конце первых двух безопасных предложений решает проблему. В качестве альтернативы все безопасные состояния могут быть заданы явно, например, safe(state(left,left,left,right)).

Упражнение 5.5 ? - один (10). один
(0) :- записать (1). один(N):- N1
есть N-1, один(N1), один(N1).

Упражнение

5.6 (a) При n1 = |L1|, n2 = |L2| верно, что Tappend(n1, n2) = Θ (n1). (б) При n = |L| верно, что Tnrev(n) = Θ (n2). (в) При n = |L| верно, что Taccrev(n) = Θ (n).

Упражнение 5.7 Для деревьев с символами во внутренних узлах мы можем использовать, например, термины a(b,c) и a(b(e,f,g),c(h),d). Деревья без символов: дерево(b,c) или дерево(дерево(e,f,g),дерево(h),d).

Упражнение 5.8 Программы SWI-PROLOG:
(a) fib(0,1). выдумка (1,1).
fib(N,R):-N1 есть N-1, fib(N1,R1), N2 есть N-2, fib(N2,R2), R есть R1 + R2. (b) T (n) = Θ (2 (c) :- динамическое fib/2. fib(0,1).
fib(1,1). H).

fib(N,R):-N1 есть N-1, fib(N1,R1), N2 есть N-2, fib(N2,R2), R есть R1 + R2, asserta(fib(N,R)).

(d) Если факты от fib(0,1) до fib(k,f ib(k)) уже были вычислены, то для вызова fib(n,X) верно, что

$$T(n) = \begin{cases} \Theta(1) & \text{если } n \quad k, \\ \Theta(n \quad k), \text{ если } n > k. \end{cases}$$

Machine Translated by Google

(e) Поскольку после первых n вызовов предиката fib все дальнейшие вызовы имеют прямые
доступ к фактам.

Упражнение 5.9

(а) Решение здесь не раскрывается.

(б)

```
начинать :-
    fd_domain([Британец, Швед, Датчанин, Норвежец, Немецкий],1,5),
    fd_all_ Different([Британец, Швед, Датчанин, Норвежец, Немецкий]),
    fd_domain([Чай, Кофе, Вода, Пиво, Молоко],1,5),
    fd_all_ Different([Чай, Кофе, Вода, Пиво, Молоко]),
    fd_domain([Красный, Белый, Зеленый, Желтый, Синий],1,5),
    fd_all_ Different ([Красный, Белый, Зеленый, Желтый, Синий]),
    fd_domain([Собака, Птица, Кошка, Лошадь, Рыба],1,5),
    fd_all_ Different([Собака, Птица, Кошка, Лошадь, Рыба]),
    fd_domain([Паллмолл, Данхилл, Мальборо, Уинфилд, Ротманнс],1,5),
    fd_all_ Different ([Паллмолл, Данхилл, Мальборо, Уинфилд, Ротманнс]),
    fd_labeling([британец, швед, датчанин, норвежец, немец]),
    Британец #= Красный, % Британец живет в красном доме
    Швед #= Собака, % У шведа нет собаки
    Датчанин #= Чай, % Датчанин любит пить чай
    Зеленый #= Белый - 1, % Зеленый дом находится слева от белого дома
    Зеленый #= Кофе, % Хозяин теплицы пьет кофе
    Pallmall #= Птица, % У человека, который курит Pallmall, есть птица
    Молоко #= 3, % Мужчина в среднем доме пьет молоко
    Желтый #= Dunhill, % Владелец желтого дома курит Dunhill
    Норвежец #= 1, % Норвежец живет в первом доме
    dist(Marlboro,Cat)#= 1, % Курильщик Marlboro живет рядом с котом
    dist(Horse,Dunhill) #= 1, % Мужчина с лошадью живет рядом с курильщиком Dunhill
    Winfield #= Beer, % Курильщик Winfield любит пить пиво
    dist(Norwegian,Blue) #= 1, % Норвежец живет рядом с синим домом
    Немецкий #= Rothmanns, % Немец курит Rothmanns
    dist(Marlboro,Water)#=1, % Сосед курильщика Marlboro пьет воду.
    write([британец, швед, датчанин, норвежец, немец]), nl,
    написать([Собака, Птица, Кошка, Лошадь, Рыба]), nl.
```

11.6 Поиск, игры и решение проблем

Упражнение 6.1

а) На последнем уровне находится $b^{г}$ узлы. Все предыдущие уровни вместе имеют

$$Nb(\text{дмакс}) = \sum_{i=0}^{d-1} b^{i} = \frac{b^{d} - 1}{b - 1} \approx b^{d-1}$$

узлы, если b становится большим. Потому что $b^{d} / b^{d} - 1 = b$ примерно b раз
много узлов на последнем уровне, как и на всех остальных уровнях вместе взятых.

(b) Для непостоянного коэффициента ветвления это утверждение уже не верно, как показывает следующий
контрпример: дерево, сильно разветвляющееся до уровня be fore last, за которым следует уровень с
постоянным коэффициентом ветвления, равным 1. , точно
столько же узлов на последнем уровне, сколько на позапрошлом уровне.

Упражнение 6.2

(а) На рис. 6.3 на стр. 86 структура дерева на втором уровне с его восемью
узлы повторяются. Таким образом, мы можем вычислить средний коэффициент ветвления bm из $b_{м}^{2} = 8$ в $bm = \sqrt{8}$.

Machine Translated by Google

(б) Здесь вычисление не так просто, потому что корневой узел дерева ветвится сильнее всех остальных. Однако мы можем сказать, что внутри дерева коэффициент ветвления ровно на 1 меньше, чем он был бы без проверки цикла. Таким образом, bm 8 1 1,8.

Упражнение

6.3 (a) Для среднего коэффициента ветвления число листовых узлов фиксировано. Напротив, для эффективного фактора ветвления число узлов во всем дереве фиксировано.

(b) Поскольку количество всех узлов в дереве обычно лучше измеряет время вычислений для поиска всего дерева, чем количество конечных узлов. (c) Для большого b согласно (6.1) на стр. 87 имеем n b¯d+1 /b¯ = b¯d , что дает
b¯ = d н.

Упражнение

6.4 (a) 3-головоломка: 4! = 24 состояния, 8-головоломка: 9! = 362 880 состояний, 15-головоломка: 16! = 20 922 789 888 000 штатов.

(b) Переместив пустой квадрат 12 раз по часовой стрелке, мы снова достигнем исходного состояния и, таким образом, создадим циклическое подпространство с 12 состояниями.

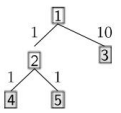
Упражнение 6.6 (a) Программа Mathematica:

```
0 BreadthFirstSearch[Node_, Goal_, Depth_] := Module[{i, NewNodes={}},
1 Для[i=1, i<= Длина[Узел], i++, 2
                  NewNodes = Join[ NewNodes, Successors[ Node[[i]] ] ];
3                 If[MemberQ[Successors[Node[[i]]], Goal],
4                         Возврат[{"Решение найдено", Глубина+1}], 0
5                   ]
6             ];
7 Если[Длина[НовыеУзлы] > 0, 8
                Поиск в ширину[Новые узлы, Цель, Глубина+1],
9               Возврат[{"Ошибка", Глубина}]
10          ]
11]
```

(b) Поскольку глубина пространства поиска не ограничена.

Упражнение

6.7 (a) При постоянной стоимости 1 стоимость всех путей на глубине d меньше стоимости всех путей на глубине d + 1. Поскольку все узлы на глубине d проверяются перед первым узлом на глубине d + 1, решение длины d гарантированно будет найдено раньше, чем решение длины d + 1. (b) Для соседнего дерева поиска
сначала генерируются узел номер 2 и узел решения номер 3 стоимости 10. Поиск прекращается. Узлы 4 и 5 со стоимостью пути 2 каждый не генерируются. Поэтому оптимальное решение не найдено.



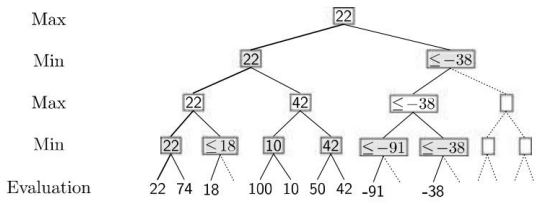
Machine Translated by Google

Упражнение 6.11 Поиск в глубину: новый узел должен иметь более низкий рейтинг, чем все открытые узлы. Поиск в ширину: новый узел должен иметь более высокий рейтинг, чем все открытые узлы.

Упражнение 6.12 Как допустимая эвристика, жена недооценивает расстояние до цели. Это может привести к тому, что они оба найдут кратчайший путь к цели, хотя и с большими усилиями. Однако это верно только в том случае, если дама всегда недооценивает расстояние.

Упражнение 6.14

(а)



11.7 Рассуждения с неопределенностью

Упражнение 7.1 |

1. P(Ω) = 2. Ом| = 1. |Ом|

P( ) = 1 P(Ω) = 0.

3. Для попарно исключающих событий A и B: P (A B) = |Ω| |A B| "=" |A|+|B| | = P (A)+ Ом|

П (Б).

4. Для двух дополнительных событий A и ¬A: P (A) + P (¬A) = P (A) + P (Ω |Ω A| | Ω| |A B| 5. P

A) = |A| | + (A "=" | = 1.

Ом| Б) Ом|

= |Ω| "=" |Ом| |A|+|B| |A B| = P (A) + P (B) - P (A B). |Ом| |Б| = П (B). |Ом|

6. Для A B: P (A) = |A| |

Ом|

7. Согласно определению 7.1 на стр. 115: A1 ··· An = Ω и |A1 ··· An| |Ом| н P (Аи) =

я=1

ни=1 |Ай | "=" н |Ай | "=" "=" |Ом| = 1. |

|Ом| я=1 |Ом| Ом|

Упражнение

7.2 (а)

$$P(y > 3 \mid \text{Класс} = \text{хорошо}) = 7/9 \; P(\text{Класс} = \text{хорошо}) = 9/13 \; P(y > 3) = 7/13,$$

$$P(\text{Класс} = \text{кишечник} \mid y > 3) = \frac{P(y > 3 \mid \text{Класс} = \text{хорошо}) \cdot P(\text{Класс} = \text{хорошо})}{P(y > 3)}$$

$$= \frac{7/9 \cdot 9/13}{1} = 7/13$$

Machine Translated by Google

$$P(\text{Класс} = \text{хорошо} \mid y \quad 3) = \frac{P(y \quad 3 \mid \text{Класс} = \text{хорошо}) \cdot P(\text{Класс} = \text{хорошо})}{P(y \quad 3)}$$

$$"=" \frac{9/2 \cdot 13/9}{13/6} "=" \frac{1}{3}.$$

(b) P (y > 3 | Класс = хорошо) = 7/9.

Упражнение 7.3

(а) восемь событий.

(б)

$$P(\text{Prec} = \text{сухо} \mid \text{Небо} = \text{ясно}, \text{Bar} = \text{поднимается})$$

$$"=" \frac{P(\text{Prec} = \text{сухо}, \text{Sky} = \text{ясно}, \text{Bar} = \text{поднимается})}{P(\text{небо} = \text{чистое}, \text{полоса} = \text{поднимается})} "=" \frac{0,4}{0,47} = 0,85.$$

(c) Отсутствует строка в таблице: Облачно Дождь 0,12

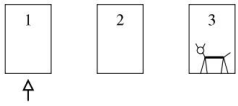
$$P(\text{Prec} = \text{дождь} \mid \text{Sky} = \text{облачно}) = \frac{P(\text{Prec} = \text{дождь}, \text{Sky} = \text{облачно})}{P(\text{небо} = \text{облачно})}$$

$$"=" \frac{0,23}{} = 0,66. \; 0,35$$

(d) Отсутствующая вероятностная мера равна 0,15. Принцип безразличия (Определение 7.5 на стр. 126) теперь требует, чтобы обеим отсутствующим строкам было симметрично присвоено значение 0,075.

Упражнение 7.4

Кандидат сначала выбрал дверь номер 1. Затем ведущий открыл дверь номер 3. Мы вводим аббревиатуру Ai для «Автомобиль находится за дверью номер i (до начала эксперимента) и Mi для «Ведущий (модератор) открывает дверь номер i» . Тогда P (A1) = P (A2) = P (A3) = 1/3, и мы вычисляем



$$P(A1 \mid M3) = \frac{П(M3 \mid A1)П(A1)}{П(M3)}$$

$$"=" \frac{П(M3 \mid A1)П(A1)}{P(M3 \mid A1)P(A1) + P(M3 \mid A2)P(A2) + P(M3 \mid A3)P(A3)} \; 1/2 \cdot 1/3 \; 1/2 \cdot 1/3 + 1 \cdot 1 /$$

$$"=" \frac{3 + 0 \cdot 1/3}{} "=" \frac{1/2 \cdot 1/3 =}{1/3, \; 1/2}$$

$$P(A2 \mid M3) = \frac{П(M3 \mid A2)П(A2)}{П(M3)} "=" \frac{1 \cdot 1/3 =}{2/3. \; 1/2}$$

Таким образом понятно, что лучше переключиться на другую дверь.

Machine Translated by Google

Упражнение 7.5. Функция Лагранжа имеет вид L = $\sum_{i=1}^{n}$ пи lnpi + λ(n i=1 пи 1).
приравнивание частных производных по pi и pj к нулю дает

$$\frac{L}{} = \ln p_i \quad 1 + \lambda = 0 \text{ и } \frac{L}{} = \ln p_j \quad 1 + \lambda = 0.$$

Вычитание этих двух уравнений дает pi = pj для всех i, j {1,...,n}. Таким образом, p1 = p2 =⋯= pn = 1/n.

Упражнение 7.6

Входные данные PIT: var A{t,f}, B{t,f}; P([A=t]) = 0,5; P([B=t] | [A=t]) = 0,94; КП([B=t]);	Выход ПИТ: Отчет о состоянии запросов Nr Истинное значение Вероятность Запрос 1 НЕ УКАЗАНО 7.200e-01 QP([B=t]);

Поскольку PIT работает численно, он не может выполнять символьные вычисления с помощью ра. рамметры α и β, а только конкретными цифрами.

Упражнение 7.7

Если: p1 = P (A, B), p2 = P (A, ¬B), p3 = P (¬A, B), p4 = P (¬A, ¬B)

$$\text{Ограничения: } p_1 + p_2 = \alpha \tag{11.1}$$

$$p_4 = 1 - p \tag{11.2}$$

$$p_1 + p_2 + p_3 + p_4 = 1 \tag{11.3}$$

Отсюда следует, что p3 = β α. Из (11.1) мы заключаем, основываясь на безразличии, что p1 = p2 = α/2. Таким образом, P (B) = p1 + p3 = α/2 + β α = β α/2 = P (A B) 1/2P (A). Соответствующая программа PIT читает

переменная A{t,f}, B{t,f}; P([A=t])
= 0,6; P([B=t] ИЛИ [A=t])
= 0,94; QP([B=t]).

Упражнение 7.8 Функция Лагранжа имеет вид

$$Л = -\sum_{я=1}^{4} p_i \ln p_i + \lambda_1(p_1+p_2 \quad \alpha) + \lambda_2(p_1+p_3 \quad \gamma) + \lambda_3(p_1+p_2+p_3+p_4 \quad 1). \tag{11.4}$$

Взяв частные производные по p1, p2, p3, p4 , получим

$$\frac{L}{} = \ln p_1 \quad 1 + \lambda_1 + \lambda_2 + \lambda_3 = 0, \tag{11,5}$$

Machine Translated by Google

$$\frac{L}{} = \ln p2 \quad 1 + \lambda 1 + \lambda 3 = 0, \quad p2 \tag{11.6}$$

$$\frac{L}{} = \ln p3 \quad 1 + \lambda 2 + \lambda 3 = 0, \quad p3 \tag{11.7}$$

$$\frac{L}{p4} = \ln p4 \quad 1 + \lambda 3 = 0 \tag{11.8}$$

и вычислить

$$(11.5)–(11.6): \ln p2 \quad \ln p1 + \lambda 2 = 0, \tag{11,9}$$

$$(11.7)–(11.8): \ln p4 \quad \ln p3 + \lambda 2 = 0, \tag{11.10}$$

откуда следует, что p3/p4 = p1/p2, что мы сразу подставляем в (7.12) на стр. 129:

$$\frac{p2p3}{п4} + p2 + p3 + p4 = 1$$

$$p2\ 1 + p4 \quad \frac{p3}{} + p3 + p4 = 1 \tag{11.11}$$

$$(7.10)–(7.11): p2 = p3 + \alpha \quad \gamma$$

что при замене p2 дает $(p3 + \alpha \quad \gamma)(1 + \frac{p3}{п4}) + p3 + p4 = 1$

$$(7.10) \text{ в } (7.12): \alpha + p3 + p4 = 1. \tag{11.12}$$

Таким образом, мы исключаем p4 в (11.8), что приводит к

$$(p3 + \alpha \quad \gamma)\ 1 + 1 \quad \frac{p3}{\alpha \quad p3} + 1 - a = 1 \tag{11.13}$$

$$(p3 + \alpha \quad \gamma)(1 \quad \alpha \quad p3 + p3) = \alpha(1 \quad \alpha \quad p3) \tag{11.14}$$

$$(p3 + \alpha \quad \gamma)(1 \quad \alpha) = \alpha(1 \quad \alpha \quad p3) \tag{11.15}$$

$$p3 + \alpha \quad \gamma \quad \alpha p3 \quad \alpha \quad ^2 + \alpha\gamma = \alpha \quad \alpha \quad 2 - ap3 \tag{11.16}$$

$$p3 = \gamma\ (1 \quad \alpha). \tag{11.17}$$

Из (11.12) следует, что p4 = (1 α)(1 γ), а из (11.11) получаем p2 = α(1 γ) и p1 = αγ.

Упражнение

7.9 (а)

$$M = \begin{matrix} 0 & 1000 \\ 10 & 0 \end{matrix}.$$

(б)

$$M \cdot \begin{matrix} п \\ 1 - p \end{matrix} \quad "=" \quad \begin{matrix} 1000 \cdot (1 \quad п) \\ 10 \cdot п \end{matrix}.$$

Machine Translated by Google

Поскольку решение между удалением или чтением определяется путем установления минимального из двух значений, достаточно сравнить два значения: 1000 (1 - p) < 10p, что дает p > 0,99. В общем, для матрицы $M = \begin{smallmatrix} 0 & k \\ 1 & 0 \end{smallmatrix}$ мы вычисляем порог k/(k + 1).

Упражнение

7.10 (a) Поскольку A и B независимы, P (A |B) = P (A) = 0,2. (b)
Применяя правило кондиционирования (стр. 155), мы получаем

$$P(C|A) = P(C|A, B)P(B)+P(C|A,\neg B)P(\neg B) = 0{,}2\cdot 0{,}9+0{,}1\cdot 0{,}1 = 0{,}19.$$

Упражнение 7.11

(а)

$$\begin{aligned}
P(Al) &= P(Al, Bur, Ear) + P(Al, \neg Bur, Ear) \\
&\quad + P(Al,Bur,\neg Ear) + P(Al,\neg Bur, \neg Ear) \\
&= P(Al|Bur,Ear)P(Bur,Ear) + P(Al|\neg Bur,Ear)P(\neg Bur,Ear) \\
&\quad + P(Al|Bур,\neg Ear)P(Bур,\neg Ear) \\
&\quad + P(Al|\neg Bur,\neg Ear)P(\neg Bur, \neg Ear) \\
&= 0{,}95 \cdot 0{,}001 \cdot 0{,}002 + 0{,}29 \cdot 0{,}999 \cdot 0{,}002 + 0{,}94 \cdot 0{,}001 \cdot 0{,}998 \\
&\quad + 0{,}001 \cdot 0{,}999 \cdot 0{,}998 \\
&= 0{,}00252,
\end{aligned}$$

$$\begin{aligned}
P(J) &= P(J,Al) + P(J,\neg Al) = P(J|Al)P(Al) + P(J|\neg Al)P(\neg Al) \\
&= 0{,}9 \cdot 0{,}0025 + 0{,}05 \cdot (1 \quad 0{,}0025) = 0{,}052,
\end{aligned}$$

$$\begin{aligned}
P(M) &= P(M,Al) + P(M,\neg Al) = P(M|Al)P(Al) + P(M|\neg Al)P(\neg Al) = 0{,}7 \cdot 0{,}0025 + \\
&\quad 0{,}01 \cdot (1 - 0{,}0025) = 0{,}0117.
\end{aligned}$$

(б)

$$\begin{aligned}
P(Al|Bur) &= P(Al|Bur,Ear)P(Ear) + P(Al|Bur,\neg Ear)P(\neg Ear) \\
&= 0{,}95 \cdot 0{,}002 + 0{,}94 \cdot 0{,}998 = 0{,}94002,
\end{aligned}$$

$$\begin{aligned}
P(M|Bur) &= P(M,Bur)/P(Bur) = [P(M,Al,Bur) + P(M,\neg Al,Bur)] \\
&\quad / P(Bур) \\
&= [P(M|Al)P(Al|Bur)P(Bur) + P(M|\neg Al)P(\neg Al|Bur)P(Bur)] \\
&\quad / P(Bур) \\
&= P(M|Al)P(Al|Bur) + P(M|\neg Al)P(\neg Al|Bur) \\
&= 0{,}7 \cdot 0{,}94002 + 0{,}01 \cdot 0{,}05998 = 0{,}659.
\end{aligned}$$

(с)

$$P(Bур|M) = \frac{P(M|Bур)P(Bур)}{П(M)} \text{ "=" } \frac{0{,}659 \cdot 0{,}001}{0{,}0117} = 0{,}056.$$

Machine Translated by Google

(г)

$$P(\text{Ал} \mid \text{Дж}, M) = \frac{P(\text{Ал}, \text{Дж}, M)}{П(\text{Дж}, M)} = \frac{P(\text{Ал}, \text{Дж}, M)}{P(\text{Al}, J, M) + P(\neg\text{Al}, J, M)}$$

$$= \frac{1}{1 + \frac{P(\neg\text{Al}, J, M)}{P(\text{Ал}, \text{Дж}, M)}} = \frac{1}{\frac{P(J \mid \neg\text{Al})P(M \mid \neg\text{Al})P(\neg\text{Al}) \; 1 + }{P(J \mid \text{Al})P(M \mid \text{Al})P(\text{Al})}}$$

$$= \frac{1}{1 + \frac{0{,}05 \cdot 0{,}01 \cdot 0{,}9975}{0{,}9 \cdot 0{,}7 \cdot 0{,}0025}} = 0{,}761,$$

$$P(J, M \mid \text{Bur}) = P(J, M \mid \text{Al})P(\text{Al} \mid \text{Bur}) + P(J, M \mid \neg\text{Al})P(\neg\text{Al} \mid \text{Bur})$$

$$= P(J \mid \text{Al})P(M \mid \text{Al})P(\text{Al} \mid \text{Bur})$$

$$+ P(J \mid \neg\text{Al})P(M \mid \neg\text{Al})P(\neg\text{Al} \mid \text{Bur}))$$

$$= 0{,}9 \cdot 0{,}7 \cdot 0{,}94 + 0{,}05 \cdot 0{,}01 \cdot 0{,}06 = 0{,}5922,$$

$$P(\text{Al} \mid \neg\text{Bur}) = P(\text{Al} \mid \neg\text{Bur}, \text{Ear})P(\text{Ear}) + P(\text{Al} \mid \neg\text{Bur}, \neg\text{Ear})P(\neg\text{Ear})$$

$$= 0{,}29 \cdot 0{,}002 + 0{,}001 \cdot 0{,}998 = 0{,}00158,$$

$$P(J, M \mid \neg\text{Bur}) = P(J, M \mid \text{Al})P(\text{Al} \mid \neg\text{Bur}) + P(J, M \mid \neg\text{Al})P(\neg\text{Al} \mid \neg\text{Bur})$$

$$= P(J \mid \text{Al})P(M \mid \text{Al})P(\text{Al} \mid \neg\text{Bur})$$

$$+ P(J \mid \neg\text{Al})P(M \mid \neg\text{Al})P(\neg\text{Al} \mid \neg\text{Bur}))$$

$$= 0{,}9 \cdot 0{,}7 \cdot 0{,}00158 + 0{,}05 \cdot 0{,}01 \cdot 0{,}998 = 0{,}00149,$$

$$P(J, M) = P(J, M \mid \text{Bur})P(\text{Bur}) + P(J, M \mid \neg\text{Bur})P(\neg\text{Bur})$$

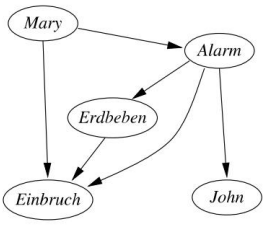
$$= 0{,}5922 \cdot 0{,}001 + 0{,}00149 \cdot 0{,}999 = 0{,}00208,$$

$$P(\text{Bur} \mid J, M) = \frac{P(J, M \mid \text{Bur})P(\text{Bur})}{П(\text{Дж}, M)} = \frac{0{,}5922 \cdot 0{,}001}{0{,}00208} = 0{,}284.$$

(д) $P(J)P(M) = 0{,}052 \cdot 0{,}0117 = 0{,}00061$, но $P(J, M) = 0{,}00208$ (см. выше).
Следовательно, $P(J)P(M) = P(J, M)$.

(f) См. разд. 7.4.5 на стр. 149.

(г)



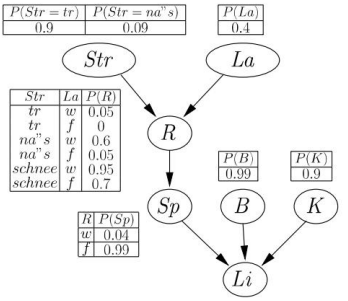
(h) Изменяется только CPT аварийного узла. Все остальные узлы не зависят от землетрясения или от землетрясений, выдаваемых по сигналу тревоги.

(i) Бур	П (Ал)
т	0,94
ф	0,0016

Возможны и другие решения из-за контрпродуктивного переменного порядка. Таким образом, мы учимся: Всегда используйте переменный порядок, учитывающий причинно-следственную связь!

Machine Translated by Google

Рис. 11.1 Байесовская сеть для велосипедного фонаря. CPT для Li указан в задаче заявление



Упражнение

7.12 (a), (б) См. рис.

11.1. (c) Ребро (Str, Li) отсутствует, так как Li и Str условно независимы при заданном V , т. е. P (Li |V , Str) = P (Li |V), так как состояние улицы не оказывает прямого влияния на освещенность. , а только динамо и напряжение, которое оно генерирует. (Условная независимость Li и Str при заданном V здесь не может быть показана расчетным путем на основе имеющихся данных для P (Li |V , Str) и P (Li |V), а может быть только утверждена.) (d) Для данного CPT:

$$P(R|Str) = P(R|Str,Flw)P(Flw) + P(R|Str,\neg Flw)P(\neg Flw)$$
$$= 0{,}95 \cdot 0{,}4 + 0{,}7 \cdot 0{,}6 = 0{,}8,$$
$$P(V|Str) = P(V|R)P(R|Str) + P(V|\neg R)P(\neg R|Str)$$
$$= 0{,}04 \cdot 0{,}8 + 0{,}99 \cdot 0{,}2 = 0{,}23.$$

Упражнение

8.1 (a) Агент — это функция A, которая отображает вектор (t1,t2,t3, d1, d2, d3, f1, f2, f3) , состоящий из трех значений температуры, давления и влажности, в класс значение k {солнечно, облачно, дождливо}. (b) Файл обучающих данных состоит из строки вида

ti1, ti2, ti3, di1, di2, di3, fi1, fi2, fi3, ki

для каждого элемента обучающих данных. Индекс я проходит по всем обучающим данным. Конкретный файл может, например, начинаться так:

17,17,17,980,983,986,63,61,50,солнечно
20,22,25,990,1014,1053,65,66,69, солнечно
20,22,18,990,979,929,52,60,61,дождь

Упражнение 8.2. Симметрию можно увидеть непосредственно, используя определение коэффициентов корреляции или ковариации. Замена i и j не меняет значения. Для

Machine Translated by Google

диагональные элементы вычисляем

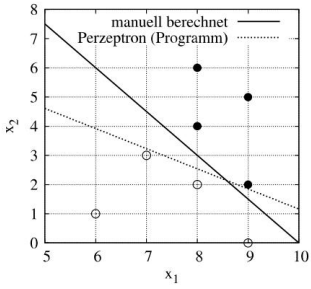
$$\text{Кии} = \frac{\sigma ii}{\text{си} \cdot \text{си}} \text{ "=" } \frac{\sum_{p=1}^{N} (xp_{\text{я}} - \bar{}xi)(xp_{\text{я}} - \bar{}xi)}{\sum_{(xp_{\text{я}} - \bar{}xi)}^{N} p=1} = 1.$$

Упражнение 8.3. Последовательность значений w такова

(1, 1), (2.2, 0.4), (1.8, 0.6), (1.4, 1.6), (2.6, 0.2), (2.2, 1.2)

Упражнение 8.4

(a)



(b) Проведение прямой линии на графике дает:

1,5x1 + x2 - 15 > 0.

После 442 итераций алгоритмы обучения персептрона с начальным вектором w = (1, 1, 1) возвращает:

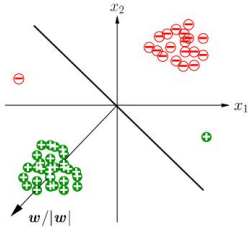
w = (11, 16, 129).

Это соответствует: 0,69x1 +x2 -8,06 > 0

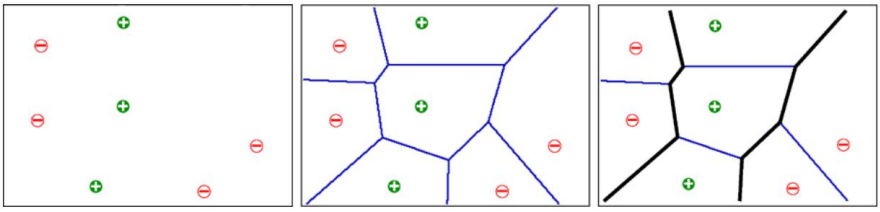
Упражнение 8.5

(a) Вектор x x указывает в «среднем» направлении всех положительных точек, а x M+ x M- указывает в «среднем» направлении всех отрицательных точек. Разность этих векторов указывает от отрицательных точек к положительным точкам. Тогда разделительная линия перпендикулярна этому разностному вектору.

(b) Облако положительных и отрицательных точек доминирует при вычислении w. Два выброса вряд ли играют здесь роль. Однако для определения разделительной линии они важны.



Упражнение 8.6



Machine Translated by Google

Упражнение

8.7 (а) Ближайшим соседом является (8, 4), таким образом, класс 1. (b) k = 2: класс не определен, поскольку один экземпляр класса 1 и один экземпляр класса 2. k = 3: решение 2:1 для класса 0. k = 5: решение 2:3 для класса 1.

Упражнение

8.8 (а) Чтобы иметь возможность делать общие утверждения, мы должны предположить, что точки равномерно распределены в пространстве признаков, что означает, что количество точек на площадь в целом примерно равно. Теперь вычислим площадь Ad узкого кольца шириной δd и расстоянием d вокруг точки x:

$$Ad = \pi(d + d)2 \quad \pi d2 = \pi(d2 + 2dd + d2) \quad \pi d2 \quad 2\pi dd.$$

Таким образом, общий вес всех точек в кольце ширины d и расстояния d пропорционален $dwi = d/(1 + \alpha d2) \quad 1/(\alpha d)$ при d . (b) Для этого взвешивания общий вес всех точек в кольце шириной d и = d/d = 1. Таким образом, каждое кольцо расстояние d будет пропорционально dw′ я ширины d будет иметь одинаковый вес, независимо от его расстояния до точки x. Это, конечно, не имеет смысла, потому что непосредственное окружение наиболее важно для приближения.

Упражнение 8.9 $\lim x \to 0 \; x \log 2 \, x = \lim x \to 0 \; \frac{\log 2 \, x}{1/x} = \lim x \to 0 \; \frac{1/(x \ln 2)}{-1/x2} = \lim x \to 0 \; \frac{x}{\ln 2}$ "=" где для второго уравнения использовалось правило Лопиталя.

Упражнение 8.10 (а) 0 (б) 1 (в) 1,5 (г) 1,875 (д) 2,32 (е) 2

Упражнение

8.11 (а) Из $\log 2 \, x = \ln x / \ln 2$ следует, что $c = 1/\ln 2 \quad 1{,}44$. (б) Поскольку обе функции энтропии отличаются только постоянным множителем, положение максимума энтропии не меняется. Экстремумы при ограничениях также сохраняют то же положение. Таким образом, фактор c не создает проблем для метода Max Ent. Для изучения деревьев решений сравниваются приросты информации различных атрибутов. Поскольку здесь имеет значение только порядок, а не абсолютное значение, фактор c здесь также не вызывает проблем.

Упражнение

8.12 (а) Для первого атрибута мы вычисляем

$$G(D, x1) = H(D) - \sum_{я=6}^{9} \frac{|Dx1=я|}{|Д|} H(Dx1=i)$$

$$= 1 - \frac{1}{8} H(Dx1=6) + \frac{1}{8} H(Dx1=7) + \frac{3}{8} H(Dx1=8) + \frac{3}{8} H(Dx1=9)$$

Machine Translated by Google

$$0 + \cdot 0{,}918 + \cdot 0{,}918 = 0{,}311$$

G(D, x2) = 0,75, поэтому выбирается x2 . Для x2 = 0, 1, 3, 4, 5, 6 решение очевидно. Для x2 = 2 выбирается x1 . Тогда дерево имеет вид

x2 = 0:0 (1/0) x2 = 1:0 (1/0)
x2 = 2:
| x1 = 6: 0 (0/0) | x1 = 7:0 (0/0) | x1 = 8:0 (1/0) | x1 = 9: 1 (1/0) x2 = 3: 0 (1/0) x2 = 4: 1 (1/0) x2 = 5: 1 (1/0) x2 = 6: 1 (1/0)

(b) Прирост информации для непрерывного атрибута x2 в качестве корневого узла:

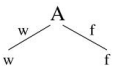
Порог Θ	012345
G(D, x2 Θ)	0,138 0,311 0,189 0,549 0,311 0,138

Поскольку G(D, x2 3) = 0,549 > 0,311 = G(D, x1), выбирается x2 3. При x2 3 классификация неоднозначна. Вычислим G(Dx2 3, x1) = 0,322, G(Dx2 3, x2 0) = 0,073, G(Dx2 3, x2 1) = 0,17, G(Dx2 3, x2 2) = 0,073. Таким образом, x1 выбран. При x1 = 9 классификация неоднозначна. Здесь очевидно, что решение G(D(x2 3,x1=9), x1) = 0, G(D(x2 3,x1=9), x2 1) = 1, x2 1 выбрано, и дерево снова имеет 100% правильность.

Дерево: x2
<= 3: | x1 = 6:0 (1/0) | x1 = 7:0 (0/0) | x1 = 8:0 (1/0) | x1 = 9: | | x2 <= 1: 0 (1/0) | | x2 > 1: 1 (1/0) x2 > 3: 1 (3/0)

Упражнение 8.13

(а) 100 % для обучающих данных, 75 % для тестовых данных, 80 % общей правильности. (б) (A ¬C) (¬A B) (в)



66,6 % правильности для обучающих данных
100 % правильности для тестовых данных
80 % общей правильности

Machine Translated by Google

Упражнение 8.14

(a) Уравнение (8.7) на стр. 189 для расчета прироста информации атрибута A читает

$$\text{Прирост информации (D, A)} = H(D) - \sum_{i=1}^{n} \frac{|Di|}{|D|} H(Di),$$

где n — количество значений текущего наблюдаемого атрибута. Если атрибут A проверяется где-то в поддереве как преемник значения aj , то только значение A = aj будет встречаться в наборе данных Dj и каждом из его подмножеств D′ . Таким образом, D′ = D′ и для всех k = j , $|D'_j|_k| = 0$, и мы получаем

$$\text{Прирост информации (D' , A)} = H(D') - \sum_{i=1}^{n} \frac{|D'_i|}{|D'|} H(D'_i) = H(D') - \frac{|D'|}{|D'|} H(D') = 0.$$

Таким образом, повторяющийся атрибут имеет нулевую информационную выгоду, из-за чего он больше не используется. (b) Из каждого непрерывного атрибута A генерируется бинарный атрибут $A>\Theta\ D,A$. Если ниже по дереву атрибут A снова дискретизируется с другим порогом $\Theta\ D',A$, то атрибут $A>\Theta\ D',A$ отличается от $A>\Theta\ D,A$. Если он имеет более высокий информационный прирост, чем все остальные атрибуты, он будет использоваться в дереве. Однако это также означает, что для нескольких вхождений непрерывного атрибута пороги должны быть разными.

Упражнение 8.15

(a) P (небо = ясное) = 0,65 P (столбик = поднимается) = 0,67

Небо	Бар	P (Prec = сухой | Sky, Bar)
Прозрачный	Повышение	0,85
Прозрачный	Падение	0,44
Облачно	Падение	0,45
Облачность	Падение	0,2

(b) P (небо = ясное) = 0,65

Бар	P (Prec = сухой |Bar)
Рост	0,73
Падение	0,33

Небо	P (Bar = рост | небо)
Прозрачный	0,72
Облачно	0,57

(c) Необходимые CPT для P (Prec |Sky,Bar) и P (Bar |Sky), а также P (Sky) уже известны.

(г) Па = (0,37, 0,065, 0,095, 0,12, 0,11, 0,13, 0,023, 0,092)
Pb = (0,34, 0,13, 0,06, 0,12, 0,15, 0,054, 0,05, 0,1)
P = (0,4, 0,07, 0,08, 0,1, 0,09, 0,11, 0,03, 0,12) (исходное распределение) Квадратичный расстояние: dq (Pa, P) = 0,0029, dq (Pb, P) = 0,014 расстояние Кульбака-Лейблера: dk (Pa, P) = 0,017, dk (Pb, P) = 0,09 Оба показателя расстояния показывают, что сеть (a) аппроксимирует распределение лучше, чем сеть (б). Это означает, что предположение, что Prec и Sky условно независимы при данном Bar , менее вероятно, чем предположение, что Sky и Bar независимы.

Machine Translated by Google

(д) Pc = (0,4, 0,07, 0,08, 0,1, 0,09, 0,11, 0,03, 0,12). Это распределение в точности равно исходному распределению P. Это неудивительно, поскольку в сети нет недостающих ребер. Это означает, что никакой независимости не было.

Упражнение 8.16 Мы можем сразу увидеть, что оценки и персептроны эквивалентны, сравнив их определения. Теперь об эквивалентности наивному байесовскому: сначала мы устанавливаем, что P (K|S1,...,Sn) > 1/2 эквивалентно P (K|S1,...,Sn) > P (¬K| S1 ,...,Sn) , так как P (¬K|S1,...,Sn) > 1 P (K|S1,...,Sn). На самом деле здесь мы имеем дело с бинарным наивным байесовским классификатором.

Применим логарифм к наивной формуле Байеса

$$P(K|S1,...,Sn) = P(S1,...,Sn) \frac{P(S1|K) \cdots P(Sn|K) \cdot P(K)}{},$$

и получить

$$\log P(K|S1,...,Sn) = \log P(S1|K) + \cdots + \log P(Sn|K) + \log P(K) \log P(S1,...,Sn). \quad (11.18)$$

Чтобы получить оценку, мы должны интерпретировать переменные S1,...,Sn как числовые переменные со значениями 1 и 0. Легко видеть, что

$$\log P(Si|K) = (\log P(Si = 1|K) \log P(Si = 0|K))Si + \log P(Si = 0|K).$$

Следует, что

$$\sum_{i=1}^{n} \log P(Si|K) = \sum_{i=1}^{n} (\log P(Si = 1|K) \log P(Si = 0|K))Si + \sum_{i=1}^{n} \log P(Si = 0|K).$$

Теперь определим $wi = \log P(Si = 1|K) \log P(Si = 0|K)$ и $c = \sum_{i=1}^{n} \log P(Si = 0|K)$ и упростить

$$\sum_{i=1}^{n} \log P(Si|K) = \sum_{i=1}^{n} WiSi + c.$$

Подставляя в (11.18), получаем

$$\log P(K|S1,...,Sn) = \sum_{i=1}^{n} wiSi + c + \log P(K) \log P(S1,...,Sn).$$

Machine Translated by Google

Для решения K должно быть так, согласно определению байесовского классификатора, что logP (K|S1,...,Sn) > log(1/2). Таким образом, должно быть так, что либо

$$\sum_{i=1}^{n} wiSi + c + \log P(K) \quad \log P(S1,...,Sn) > \log(1/2)$$

или это

$$\sum_{i=1}^{n} wiSi > \log 1/2 \quad c \quad \log P(K) + \log P(S1,...,Sn),$$

с помощью которого мы определили оценку с порогом $\Theta = \log 1/2 \quad c \quad \log P(K) + \log P(S1,...,Sn)$. Поскольку все преобразования можно отменить, мы также можем преобразовать любую оценку в байесовский классификатор. При этом была показана эквивалентность.

Упражнение 8.17 Логарифмирование (8.10) на стр. 206 приводит к

$$\log P(I|s1,...,sn) = \log c + \log P(I) + \sum_{i=1}^{l} ni \log P(wi|I).$$

Таким образом, очень маленькие отрицательные числа становятся умеренными отрицательными числами. Поскольку функция логарифма монотонно растет, для определения класса максимизируем по правилу

$$INaive\text{-}Bayes = \operatorname{argmax}_{I \in \{w,f\}} \log P(I|s1,...,sn).$$

Недостатком этого метода является несколько большее время вычислений на этапе обучения для больших текстов. При классификации время не увеличивается, так как значения logP (I | s1,...,sn) могут сохраняться при обучении.

Упражнение 8.19. Пусть f строго монотонно возрастает, то есть $\forall x,y \; x < y \quad f(x) < f(y)$. Если теперь d1(s, t) < d1(u, v), то, очевидно, d2(s, t) = f (d1(s, t)) < f (d1(u, v)) = d2(u, v). Поскольку обратная функция f также строго монотонна, верно и обратное, т. е. d2(s, t) < d2(u, v) \quad d1(s, t) < d1(u, v). Таким образом, показано, что d2(s, t) < d2(u, v) \quad d1(s, t) < d1(u, v).

Упражнение 8.20

$23 \cdot 26$ x1x2 = 4, и, следовательно, $ds(x1, x2) = \sqrt{6,1} = \sqrt{1\,4}$

x2x3 = 2, и, таким образом, $ds(x2, x3) = \sqrt{\frac{26 \cdot 20}{1,4}} = \sqrt{1\,4}$

$23 \cdot 20$ x1x3 = 1, поэтому $ds(x1, x3) = \sqrt{\frac{2}{1}} = \sqrt{21,4}$

Предложения 1 и 2 наиболее похожи по метрике расстояния ds .

Machine Translated by Google

Упражнение 8.21 Помощь при проблемах с KNIME: www.knime.org/forum

11.9 Нейронные сети

Упражнение 9.1. Мы хотим показать, что f (Θ + x) + f (Θ x) = 1.

$$f(\Theta + x) = 1 + e^{\frac{1}{\frac{\Theta x}{T}}} = \frac{T e^{\frac{\Theta x}{T}}}{1 + e^{\frac{\Theta x}{T}}}, \qquad f(\Theta - x) = 1 + e^{\frac{1}{\frac{\Theta x}{T}}},$$

$$f(\Theta + x) + f(\Theta x) = 1.$$

Упражнение 9.3 Каждый шаблон, сохраненный в сети, имеет размер n бит. Всего сеть имеет n(n 1)/2 веса. Если мы зарезервируем 16 битов на вес и определим двоичную память размером 16n(n 1)/2 как одинаково большую, то это, очевидно, может хранить N = 8n(n 1)/n = 4(n 1) шаблонов n битов. по размеру. Для больших n мы получаем N = 4n как предел.
Если мы возьмем частное α числа сохраняемых битов и количества доступных ячеек памяти, как в (9.11) на стр. 237, то мы получим значение 1 для списка памяти 2 и значение α = 0,146n /(16n(n 1)/2) 0,018 для сети Хопфилда. Таким образом, классическое хранилище имеет (для 16 бит на вес) емкость примерно в 55 раз выше.

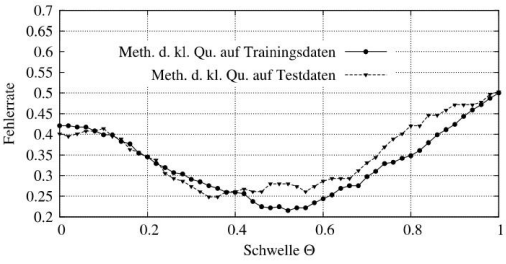
Упражнение 9.4

(а) Программа Mathematica для метода наименьших квадратов.

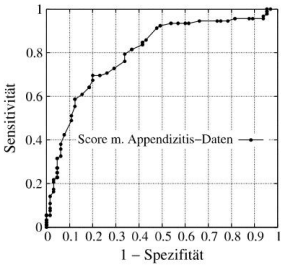
```
LeastSq[q_,a_] := Module[{Nq,Na,m,A,b,w}, Nq = длина[q]; m =
                  Длина[q[[1]]]; Na = длина [a]; If[Nq != Na, Print["Длина[q] != Длина[a]"]; Выход,
                  0]; A = Table[N[Sum[q[[p,i]] q[[p,j]], {p,1,Nq}]], {i,1,m}, {j,1,m} ]; b =
                  Таблица[N[Сумма[a[[p]]q[[p,j]], {p,1,Nq}]], {j,1,m}]; w = линейное решение[A,b]

]
LeastSq::usage = "LeastSq[x,y,f] вычисляет из векторов запросов q[[1]],..., q[[m]] таблицу коэффициентов w[[i]] для
      линейного отображения f[x] = Sum[w[[i]] x[[i]], {i,1,m}] с f[q[[p]]] = a[[p]]."
```

(б)



(с)



Упражнение 9.6 (а) Обучение работает без ошибок. (б) Обучение не работает без ошибок!

Machine Translated by Google

Упражнение

9.7 (a) Отображение f называется линейным, если для всех x, y, k выполняется равенство f (x + y) = f (x) + f (y) и f (kx) = kf (x) . Пусть теперь f и g — линейные отображения. Тогда f (g(x + y)) = f (g(x) + g(y)) = f (g(x)) + f (g(y)) и f (g(kx)) = f (кг(x)) = kf(g(x)). Таким образом, последовательное выполнение линейных отображений является линейным отображением.

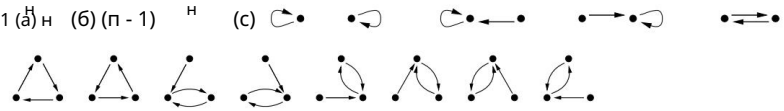
(b) Мы наблюдаем два произвольных выходных нейрона j и k. Каждый из двух представляет класс. Классификация осуществляется путем формирования сумма максимума из двух активаций. i wj ixi и netk = i wkixi — взвешенная значений ar. Пусть netj = расхождение в нейронах j и k. Кроме того, пусть netj > netk . Без функции активации выводится класс j. Теперь, если применяется строго монотонная функция активации f, в результате ничего не изменится, потому что из-за строго монотонной функции f (netj)>f(netk).

Упражнение 9.8 Преобразованное $\frac{2}{1}$, f2(x1, x2) = x $\frac{2}{2}$. Затем разделительная линия в пространство f1(x1, x2) = x имеет уравнение y1 + y2 = 1.

11.10 Обучение с подкреплением

Упражнение

10.1 (а) ^{н} н (б) (п - 1) ^{н} (c)



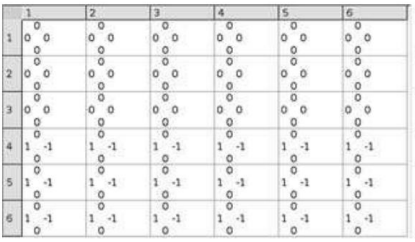
(г)

Упражнение 10.2. Повторение значения с построчным обновлением значений V^ от верхнего левого угла к нижнему правому дает

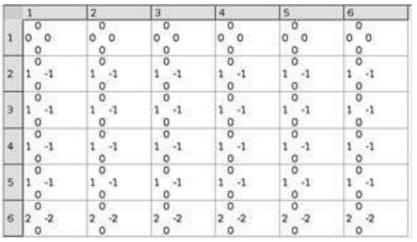
0	0	0,81	0,9	1,35	1,49	...	2,36 2,62
0	0	0,73	1	1,21	1,66		2,12 2,91

Упражнение 10.3

(c)



6×6 feedback for a completely smooth surface



6×6 feedback for a very heavy survace (like thick snow)

Machine Translated by Google

(d) Мы видим, что чем длиннее становится политика (т. е. чем больше шагов, например, имеет цикл циклической стратегии), тем ближе значение γ должно быть к 1, потому что большее значение γ увеличивает память. возможный. Однако итерация значений сходится намного медленнее.

Упражнение 10.4 Значение V выглядит (3, 3) справа внизу в матрице состояния изменяется как следующим образом:

$$B \quad (3,1) = 0{,}9\ B \qquad (2,1) = 0{,}9\ 2\ B \qquad (2,2) = 0{,}9\ 3\ B \qquad (2,3) = 0{,}9\ 4\ B \qquad (3,3). \tag{11.19}$$

Эта цепочка уравнений следует из (10.6), поскольку для всех заданных переходов состояний максимальное немедленное вознаграждение равно r(s, a) = 0, и это тот случай, когда

$$B \quad (s) = \max_a [r(s,a) + \gamma\ V \qquad (\delta(s,a))] = \gamma\ V \qquad (\delta(c,a)) = 0{,}9\ B \qquad (\delta(s,a)).$$

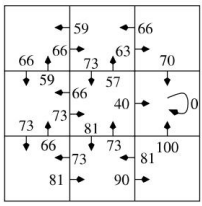
Из (10.6) также следует, что V (3, 2) = 1 + 0,9 B (3, 1), так как r(s, a) = 1 максимально. Точно так же верно, что B. (3, 3) = 1 + 0,9 B (3, 2) и круг закрывается. Два последних уравнения вместе (3, 3) = 1 + 0,9 (1 + 0,9 B (3, 1)). Из (11.19) следует, что V дает дают V (3, 1) = 0,9 4V (3, 3). Заменено в B (3, 3), это

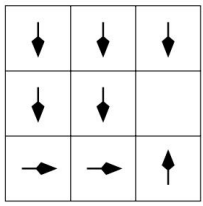
$$B \quad (3,3) = 1 + 0{,}9(1 + 0{,}9\ 5B \qquad (3,3)),$$

из чего следует утверждение.

Упражнение 10.5

Стабильные значения Q и оптимальная политика:





Упражнение

10.6 (a) Количество состояний. Количество действий на состояние = ℓ^{H} 1. Количество полигонов = ℓ cies = $(\ell n)^{Hn} = л^{п\,л}$.

(б)

	л = 1	л = 2	л = 3	л = 4	л = 10
п = 1	1	4	27	256	1010
п = 2 1		256	3,9 × 108	1,8 × 1019	10200
n = 31 1,7 × 107 n = 41 1,8 ×			4,4 × 1038	3,9 × 10115	103000
1019 3,9 × 10154 n = 81 3,2 × 10616 1,4 ×				3,2 × 10616	1040000
1025043 6,7 × 10315652					10800000000

Machine Translated by Google

(c) В каждом состоянии теперь есть 2n возможных действий. Таким образом, имеется (2n)ℓ ^{н} политики.

	л = 1	л = 2	л = 3	л = 4	л = 10
п = 1	2	4	8	16	1024
n = 2 4 256 n = 36 1,7 ×			2,6 × 109 4,3 × 109 1,0 ×		1,6 × 1060
106 n = 48 2,8 × 1014 1,8 ×			1021 6,3 × 1049 1,4 × 1073		1,4 × 10778
10308			1,6 × 10231 1,7 × 107900 1,6		7,9 × 109030
п = 8 16			× 1078913		1,8 × 10120411998

(d) 10120411998 различных политик никогда не могут быть исследованы комбинаторно, даже если все имеющихся компьютеров в мире должны были работать на них параллельно. Таким образом, необходимы «интеллектуальные» алгоритмы для нахождения оптимального или близкого к оптимальному политика.

Machine Translated by Google

Machine Translated by Google

Рекомендации

[Ада75] Э. В. Адамс. Логика условностей. Synthese Library, том 86. Рейдель, Дордрехт, 1975 год.

[Альп04] Э. Алпайдин. Введение в машинное обучение. Массачусетский технологический институт, Кембридж, 2004.

[90 апреля] Дж. Андерсон, А. Пеллионис и Э. Розенфельд. Нейрокомпьютинг (том 2): Направления для исследований. Массачусетский технологический институт, Кембридж, 1990.

[AR88] Дж. Андерсон и Э. Розенфельд. Нейрокомпьютинг: основы исследований. Массачусетский технологический институт Press, Cambridge, 1988. Сборник фундаментальных оригинальных статей.

[Бар98] Р. Бартак. Онлайн-руководство по программированию в ограничениях, 1998 г. http://kti.ms.mff.cuni.cz/~бартак/ограничения.

[BCDS08] А. Биллард, С. Калинон, Р. Диллманн и С. Шаал. Демонстрационное программирование роботов. В Б. Сицилиано и О. Хатиб, редакторы, Справочник по робототехнике, стр. 1371–1394. Спрингер, Берлин, 2008 г.

[Бел57] РЭ Беллман. Динамическое программирование. Издательство Принстонского университета, Принстон, 1957.

[Бер89] М. Беррондо. Fallgruben fü r Kopffü ssler. Фишер Ташенбух, Nr. 8703, 1989.

[BFOS84] Л. Брейман, Дж. Фридман, Р. А. Олшен и С. Дж. Стоун. Классификация и деревья регрессии. Уодсворт, Белмонт, 1984 г.

[Би682] В. Бибель. Автоматическое доказательство теорем. Виег, Висбаден, 1982 г.

[Bis05] КМ Бишоп. Нейронные сети для распознавания образов. Издательство Оксфордского университета, Лондон, 2005.

[Bis06] КМ Бишоп. Распознавание образов и машинное обучение. Спрингер, Нью-Йорк, 2006.

[BM03] А. Г. Барто и С. Махадеван. Последние достижения в иерархическом обучении с подкреплением. Discrete Event Syst., Специальный выпуск по обучению с подкреплением, 13:41–77, 2003.

[Бюстгальтер84] В. Брайтенберг. Транспортные средства - эксперименты в синтетической психологии. MIT Press, Кэмбридж, 1984.

[Бюстгальтер01] Б. Брабек. Computergestü tzte Regionale Lawinenprognose. Кандидатская диссертация, ETH Zü rich, 2001.

[Бюстгальтер11] И. Братко. Программирование PROLOG для искусственного интеллекта. Эддисон-Уэсли, Рединг, 4-е издание, 2011 г.

[Бри91] Британская энциклопедия. Британская энциклопедия, Лондон, 1991.

[Бур98] Си Джей Берджес. Учебник по машинам опорных векторов для распознавания образов. Данные Мин. Знай. Дисков., 2(2):121–167, 1998.

[CAD] CADE: Конференция по автоматизированной дедукции. www.cadeconference.org.

[Че83] П. Чизмен. Метод вычисления обобщенных байесовских значений вероятности для экспертные системы. В проц. 8-го междунар. Совместная конф. по искусственному интеллекту (IJCAI 83), 1983.

[Че85] П. Чизмен. В защиту вероятности. В проц. 9-го междунар. Совместная конф. на Искусственный интеллект (IJCAI-85), 1985 г.

[CL73] CL Чанг и RC Ли. Символическая логика и механическое доказательство теорем. Академическое издательство, Орландо, 1973.

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по информатике,
DOI 10.1007/978-0-85729-299-5, © Springer-Verlag London Limited 2011

Machine Translated by Google

[Кле79] WS Кливленд. Надежная локально взвешенная регрессия и сглаживающие диаграммы рассеяния. Варенье. Стат. Assoc., 74(368):829–836, 1979.

[CLR90] Т. Кормен, гл. Лейзерсон и Р. Ривест. Введение в алгоритмы. Массачусетский технологический институт Пресс, Кембридж, 1990.

[CM94] В. Ф. Клоксин и К. С. Меллиш. Программирование на Прологе. Шпрингер, Берлин, 4-я улица издание 1994 года.

[Coz98] ФГ Козман. Javabayes, байесовские сети в Java, 1998. www.cs.cmu.edu/~ javabayes.

[dD91] FT де Домбаль. Диагностика острой боли в животе. Черчилль Ливингстон, Лондон, 1991 г.

[dDLS+72] FT de Dombal, DJ Leaper, JR Staniland, AP McCann и JC Horrocks. Компьютерная диагностика острой боли в животе. бр. Мед. Дж., 2:9–13, 1972.

[Ди11] Проект DeepQA, 2011 г. http://www.research.ibm.com/deepqa/deepqa.shtml.

[DHS01] Р.О. Дуда, П.Е. Харт и Д.Г. Сторк. Классификация узоров. Уайли, Нью-Йорк, 2001.

[Dia04] Д. Диаз. ПРОЛОГ ГНУ. Universität Paris, 2004. Aufl. 1.7, для версии GNU Prolog 1.2.18, http://gnu-prolog.inria.fr.

[DNM98] CL Blake, DJ Newman, S. Hettich и CJ Merz. Репозиторий UCI машины учебные базы данных, 1998 г. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[Эде91] Э. Эдер. Относительная сложность исчисления первого порядка. Виег, Висбаден, 1991.

[Лось93] К. Элькан. Парадоксальный успех нечеткой логики. В материалах Одиннадцатой национальной конференции по искусственному интеллекту (AAAI-93), стр. 698–703. Массачусетский технологический институт Пресс, Кембридж, 1993.

[Эрт93] В. Эртель. Parallelesuche mit randomisiertem Wettbewerb inInferenzsystemen. DISKI, Band 25. Infix, St. Augustin, 1993. Диссертация, Технический университет. Мюнхен.

[Эрт11] В. Эртель. Искусственный интеллект, 2011. www.hs-weingarten.de/~ertel/aibook. Домашняя страница этой книги с материалами, демонстрационными программами, ссылками, литературой, исправлениями и т. д.

[ES99] В. Эртель и М. Шрамм. Объединение данных и знаний с помощью MaxEnt-оптимизации вероятностных распределений. В ПКДД'99 (3-я Европейская конференция по принципам и Практика обнаружения знаний в базах данных). LNCS, том 1704, стр. 323–328. Спрингер, Прага, 1999.

[ESCT09] В. Эртель, М. Шнайдер, Р. Кубек и М. Токич. Учебный ящик: универсальный система обучения роботов. В материалах 14-й Международной конференции по передовой робототехнике (ICAR 2009), 2009 г. www.servicerobotik.hs-weingarten.de/учебная коробка.

[ESS89] В. Эртель, Дж. Шуман и Ч. Саттнер. Изучение эвристики для средства доказательства теорем используя обратное распространение. В J. Retti и K. Leidlmair, редакторы, 5. Österreichische Искусственный интеллект-Tagung. Informatik-Fachberichte, том 208, страницы 87–95. Спрингер, Берлин, 1989.

[Подходит96] М. Фитинг. Логика первого порядка и автоматическое доказательство теорем. Шпрингер, Берлин, 1996.

[FNA+09] Д. Ферруччи, Э. Найберг, Дж. Аллан, К. Баркер, Э. Браун, Дж. Чу-Кэрролл, А. Чикколо, P. Duboue, J. Fan, D. Gondek et al. К открытому выдвижению вопроса системы ответов. Технический отчет IBM RC24789, Йорктаун-Хайтс, Нью-Йорк, 2009 г. http://www.research.ibm.com/deepqa/question_answering.shtml.

[FPP07] Д. Фридман, Р. Пизани и Р. Пурвс. Статистика. Нортон, Нью-Йорк, 4-е издание, 2007.

[Фра05] К. Фрейн. Теория компьютерного шахматного программирования, 2005. www.frayn.net/beowulf/теория.html.

[Свободный97] Э. Фрейдер. В погоне за Святым Граалем. Ограничения, 2(1):57–61, 1997.

[ФС97] Б. Фишер и Дж. Шуман. Setheo идет на разработку программного обеспечения: применение atp к повторному использованию программного обеспечения. На конференции по автоматизированной дедукции (CADE-14). ЛНКС, том 1249, страницы 65–68. Springer, Берлин, 1997 г. http://ase.arc.nasa.gov/people/schumann/publications/papers/cade97-reuse.html.

Machine Translated by Google

[GW08] Р. С. Гонсалес и Р. Е. Вудс. Цифровая обработка изображений. Пирсон/Прентис Холл, Река Аппер-Сэдл / Принстон, 2008 г.

[Göd31a] К. Гёдель. Diskussion zur Grundlegung der Mathematik, Erkenntnis 2. Monatshefte Мат. Phys., 32(1):147–148, 1931.

[Göd31b] К. Гёдель. Überformal unentscheidbare Sätze der Principia Mathematica und ver wandter Systeme I. Monatshefte Math. Phys., 38(1):173–198, 1931.

[HKP91] Дж. Герц, А. Крог и Р. Палмер. Введение в теорию нейронных вычислений. Аддисон-Уэсли, чтение, 1991.

[Hon94] Б. Хончик. Теория и практика аппендэктомии. Мабузе, Франкфурт-на-Майне, 1994.

[Хоп82] Джей Джей Хопфилд. Нейронные сети и физические системы с возникающими коллективными вычислительными способностями. проц. Натл. акад. науч. США, 79:2554–2558, 1982. Перепечатка в [AR88], страницы 460–464.

[HT85] Джей Джей Хопфилд и Д. У. Танк. «Нейронный» расчет решений в оптимизации проблемы. биол. Cybern., 52(3):141–152, 1985. Спрингер.

[HTF09] Т. Хасти, Р. Тибширани и Дж. Фридман. Элементы статистического обучения: данные Добыча полезных ископаемых, вывод и прогнозирование. Springer, Berlin, 3-е издание, 2009 г. Электронная версия: http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

[Джей57] ЭТ Джейнс. Теория информации и статистическая механика. физ. Обр., 1957.

[Джей03] ЭТ Джейнс. Теория вероятностей: логика науки. Издательство Кембриджского университета, Кембридж, 2003 г.

[Джен01] ФВ Дженсен. Байесовские сети и графы решений. Спрингер, Берлин, 2001.

[Jor99] М. И. Джордан, редактор. Обучение графическим моделям. Массачусетский технологический институт, Кембридж, 1999.

[Кал01] Я Кальман. Автоматизированное мышление с помощью OTTER. Ринтон, Парамус, 2001. www-unix.mcs.anl.gov/AR/otter/index.html.

[Кан89] Т. Кейн. Максимальная энтропия в вероятностной логике Нильссона. В проц. 11-го Международный Совместная конф. по искусственному интеллекту (IJCAI-89), 1989 г.

[KK92] Дж. Н. Капур и Х. К. Кесаван. Принципы оптимизации энтропии с приложениями. Академическая пресса, Сан-Диего, 1992.

[KLM96] Л.П. Кельблинг, М.Л. Литтман и А.П. Мур. Обучение с подкреплением: опрос. Дж. Артиф. Интел. Рез., 4:237–285, 1996. www-2.cs.cmu.edu/afs/cs/project/jair/pub/том4/kaelbling96a.pdf.

[KMK97] Х. Кимура, К. Миядзаки и С. Кобаяши. Обучение с подкреплением в POMDP с приближением функции. На 14-й Международной конференции по машинному обучению, стр. 152–160. Морган Кауфманн, Сан-Матео, 1997 г. http://sysplan.nams.kyushu-u.ac.jp/gen/papers/JavaDemoML97/robodemo.html.

[Ко72] Т. Кохонен. Корреляционная матрица памяти. IEEE транс. Вычисл., С-21(4):353–359, 1972 г. Перепечатка в [AR88], стр. 171–174.

[Лар00] ФД Лараме. Шахматное программирование, части 1–6, 2000 г. www.gamedev.net/reference/программирование/возможности/шахматы1.

[Le999] LEXMED — обучающая экспертная система для медицинской диагностики, 1999 г. www.lexmed.de.

[Lov78] Д. У. Лавленд. Автоматическое доказательство теорем: логическая основа. Северная Голландия, Амстердам, 1978 г.

[LSBB92] Р. Летц, Дж. Шуман, С. Байерл и В. Бибель. SETHEO: средство доказательства теорем высокой производительности. Дж. Автомат. Reason., 8(2):183–212, 1992. www4.informatik.tu-muenchen.de/~letz/seteo.

[МакК] В. МакКьюн. Автоматизированные дедуктивные системы и группы. www-unix.mcs.anl.gov/AP/другие.html. См. также www-formal.stanford.edu/clt/ARS/systems.html.

[МакД82] Дж. Макдермотт. R1: конфигуратор компьютерных систем на основе правил. Артиф. Интел., 19:39–88, 1982.

[MDBM00] Г. Мелансон, И. Дютур и Г. Буск-Мелу. Случайная генерация дагов для графическое рисование. Технический отчет INS-R0005, Голландский исследовательский центр математических и компьютерных наук (CWI), 2000 г. http://ftp.cwi.nl/CWIreports/INS/INS-R0005.pdf.

Machine Translated by Google

[Мит97] Т. Митчелл. Машинное обучение. McGraw-Hill, Нью-Йорк, 1997. www-2.cs.cmu.edu/~tom/mlbook.html.

[MP69] М. Мински и С. Пейперт. Персептроны. MIT Press, Кембридж, 1969.

[Новый00] М. Новорожденный. Автоматизированное доказательство теорем: теория и практика. Шпрингер, Берлин, 2000.

[Нет86] Н. Дж. Нильссон. Вероятностная логика. Артиф. Интел., 28(1):71–87, 1986.

[Ноль98] Н. Нильссон. Искусственный интеллект — новый синтез. Морган Кауфманн, Сан-Матео, 1998.

[NPW02] Т. Нипков, Л. С. Полсон и М. Венцель. Изабель/ХОЛ — Помощник по проверке Логика высшего порядка. LNCS, том 2283. Springer, Берлин, 2002. www.cl.cam.ac.uk/Research/HVG/Изабель.

[NS61] А. Ньюэлл и Х.А. Саймон. Gps, программа, имитирующая человеческое мышление. В Х. Биллинг, редактор Lernende Automaten, страницы 109–124. Ольденбург, Мюнхен, 1961.

[NSS83] А. Ньюэлл, Дж. К. Шоу и Х. А. Саймон. Эмпирические исследования с логикой теория машины: пример эвристики. В J. Siekmann и G. Wrightson, редакторы, Автоматизация рассуждений 1: классические статьи по вычислительной логике 1957–1966, страницы 49–73. Springer, Берлин, 1983. Erstpublikation: 1957.

[OFY+95] К. Оманн, К. Франке, К. Ян, М. Маргулис, М. Чан, ван П.Дж. Элк, Ф.Т. Домбал и Х.Д. Рёрер. Диагностика острого аппендицита. Чирург, 66:135–141, 1995.

[OMYL96] К. Оманн, В. Мустакис, К. Ян и К. Ланг. Оценка автоматических методов приобретения знаний в диагностике острой боли в животе. Артиф. Интел. Мед., 8:23–36, 1996.

[OPB94] К. Оманн, К. Платен и Г. Беленький. Computerunterstü tze Diagnose bei akuten Баухшмерцен. Чирург, 63:113–123, 1994.

[Приятель80] Г. Палм. Об ассоциативной памяти. биол. Cybern., 36:19–31, 1980.

[Приятель91] Г. Палм. Возможности памяти локальных правил синаптической модификации. Concepts Neu rosci., 2(1):97–128, 1991. MPI Tü bingen.

[Горох84] Дж. Перл. Эвристика, интеллектуальные стратегии поиска для решения компьютерных проблем. Аддисон-Уэсли, Рединг, 1984.

[Горох88] Дж. Перл. Вероятностное мышление в интеллектуальных системах. Сети правдоподобного вывода. Морган Кауфманн, Сан-Матео, 1988 год.

[PL05] Л. Панайт и С. Люк. Совместное мультиагентное обучение: современное состояние. Автон. Агенты Multi-Agent Syst., 11(3):387–434, 2005.

[PS08] Дж. Питерс и С. Шаал. Усиление обучения двигательным навыкам с помощью градиентов политики. Нейронная сеть, 21(4):682–697, 2008 г. http://www-clmc.usc.edu/publications/P/peters-NN2008.pdf.

[PS09] Г. Полиа и С. Слоан. Как решить: новый аспект математического метода. Иши, Маунтин-Вью, 2009 г.

[Qui93] Дж. Росс Куинлан. C4.5: Программы для машинного обучения. Морган Кауфманн, Сан Матео, 1993. Загрузка C4.5: http://www.rulequest.com/Personal, загрузка C5.0: http://www.rulequest.com/download.html.

[RB93] М. Ридмиллер и Х. Браун. Прямой адаптивный метод для более быстрого обратного распространения обучение: алгоритм rprop. В материалах международной конференции IEEE по нейронным сетям, страницы 586–591, 1993.

[RGH+06] М. Ридмиллер, Т. Габель, Р. Хафнер, С. Ланге и М. Лауэр. Мозговые штурмовики: Entwurfsprinzipien lernfähiger autonomer Roboter. Информ.-Спектр, 29(3):175–190, 2006.

[RHR86] Д. Е. Румельхарт, Г. Э. Хинтон и Р. Дж. Уильямс. Изучение внутренних представлений путем распространения ошибок. В [RM86].

[Ric83] Э. Рич. Искусственный интеллект. Макгроу-Хилл, Нью-Йорк, 1983 г.

[86 ринггитов] Д. Румельхарт и Дж. Макклелланд. Параллельная распределенная обработка, том 1. MIT Пресса, Кембридж, 1986.

Machine Translated by Google

[96 рингитов] В. Рёддер и К.-Х. Мейер. Согласованная обработка знаний с максимальной энтропией ДУХОМ. В KI-96 (Германская национальная конференция по ИИ), Дрезден, 1996 г.

[RMD07] М. Ридмиллер, М. Монтемерло и Х. Далкамп. Учимся водить настоящий автомобиль в 20 минут. В FBIT '07: Proceedings of the 2007 Frontiers in the Convergence of Биологические науки и информационные технологии, страницы 645–650. Компьютерное общество IEEE, Вашингтон, округ Колумбия, 2007 г.

[RMS92] Х. Риттер, Т. Мартинес и К. Шульten. Нейронные вычисления и самоорганизация Карты. Аддисон-Уэсли, Рединг, 1992.

[RN10] С. Рассел и П. Норвиг. Искусственный интеллект: современный подход. Прентис Холл, Нью-Йорк, 3-е издание, 2010 г. 1-е издание: 1995 г. http://aima.cs.berkeley.edu.

[Роба] Официальный сайт Робокап. www.robocup.org.

[Робб] Футбольный симулятор robocup. http://sserver.sourceforge.net.

[Роб65] Дж. А. Робинсон. Машинно-ориентированная логика, основанная на принципе разрешения. Дж. АКМ, 12(1):23–41, 1965.

[Рой96] Р. Рохас. Нейронные сети: систематическое введение. Спрингер, Берлин, 1996.

[Рос58] Ф. Розенблатт. Персептрон: вероятностная модель хранения информации и организация в мозгу. Психол. Rev., 65:386–408, 1958. Перепечатка в [AR88], стр. 92–114.

[Рос09] СМ Росс. Введение в вероятность и статистику для инженеров и ученых. Академическая пресса, Сан-Диего, 2009 г.

[RW06] CE Rasmussen и CKI Williams. Гауссовские процессы для машинного обучения. MIT Press, Кембридж, 2006 г. Интернет-версия: http://www.gaussianprocess.org/ gpml/главы/.

[СА94] С. Шаал и К. Г. Аткесон. Жонглирование роботами: реализация на основе памяти обучение. Система управления IEEE. Маг., 14(1):57–71, 1994.

[Сэм59] ЭЛ Самуэль. Некоторые исследования машинного обучения с использованием игры в шашки. IBM Дж., 1(3):210–229, 1959.

[SB98] Р. Саттон и А. Барто. Обучение с подкреплением. Массачусетский технологический институт, Кембридж, 1998. www.cs.ualberta.ca/~sutton/book/the-book.html.

[SB04] Дж. Зикманн и Ч. Бензмюллер. Омега: математика, поддерживаемая компьютером. В КИ 2004: Достижения в области искусственного интеллекта. ЛНАИ, том 3238, страницы 3–28. Спрингер, Берлин, 2004 г. www.ags.uni-sb.de/~omega.

[Ш96] М. Шрамм. Indifferenz, Unabhängigkeit и максимальная энтропия: Eine wahrscheinlichkeitstheoretische Semantik fü r Nicht-Monotones Schließen. Dissertationen zur Informatik, Band 4. CS, Мюнхен, 1996.

[Sch01] Дж. Шуман. Автоматизированное доказательство теорем в программной инженерии. Спрингер, Берлин, 2001.

[Sch02] С. Шульц. E — умный доказатель теорем. J. AI Commun., 15 (2/3): 111–126, 2002. www4.informatik.tu-muenchen.de/~schulz/WORK/eprover.html.

[Sch04] А. Шварц. СпамУбийца. O'Рейли, Кембридж, 2004 г. Домашняя страница SpamAssassin: http://spamassassin.apache.org.

[SE90] Ч. Зутнер и В. Эртель. Автоматическое получение эвристики, направляющей поиск. В 10-м Междунар. конф. по автоматизированной дедукции. ЛНАИ, том 449, страницы 470–484. Спрингер, Берлин, 1990 год.

[SE00] М. Шрамм и В. Эртель. Рассуждения с вероятностями и максимальной энтропией: система PIT и ее применение в LEXMED. К. Индерфурт и др., редактор, Operations Research Proceedings (SOR'99), стр. 274–280. Спрингер, Берлин, 2000.

[SE10] М. Шнайдер и В. Эртель. Робот учится на демонстрации с местным гауссовцем регресс процесса. В материалах Международной конференции IEEE/RSJ по Интеллектуальные роботы и системы (IROS'10), 2010.

[SET09] Т. Сегаран, К. Эванс и Дж. Тейлор. Программирование семантической сети. O'Рейли, Кэмбридж, 2009 г.

[Шо76] Э. Х. Шортлифф. Компьютерные медицинские консультации, MYCIN. Северная Голландия, Нью-Йорк, 1976 год.

Machine Translated by Google

[Spe97] Тайны разума. Специальный выпуск. Научный американец Inc., 1997.

[Spe98] Изучение интеллекта. Scientific American Presents, том 9. Scientific American Инк., 1998.

[SR86] TJ Sejnowski и CR Розенберг. NETtalk: параллельная сеть, которая учится читать вслух. Технический отчет JHU/EECS-86/01. Технический отчет Университета Джона Хопкинса по электротехнике и компьютерным наукам, 1986 г. Перепечатка в [AR88], страницы 661–672.

[SS02] С. Шёлькопф и А. Смола. Обучение с помощью ядер: машины опорных векторов, регуляризация, оптимизация и не только. Массачусетский технологический институт, Кембридж, 2002.

[SS06] Г. Сатклифф и К. Саттнер. Состояние ЦАСК. AI Commun., 19(1):35–48, 2006. Домашняя страница CASC: www.cs.miami.edu/~tptp/CASC.

[CCK05] П. Стоун, Р. Саттон и Г. Кульманн. Обучение с подкреплением для робокапа по футболу. Адаптивное поведение, 2005. www.cs.utexas.edu/~pstone/Papers/bib2html-ссылки/AB05.pdf.

[Ste07] Дж. Стюарт. Многомерное исчисление. Брукс Коул, Флоренция, 2007 г.

[SW76] К. Э. Шеннон и У. Уивер. Математические основы теории информации. Ольденбург, Мюнхен, 1976 год.

[Размер10] К. Сепешвари. Алгоритмы обучения с подкреплением. Морган и Клейпул, Сан Рафаэль, 2010. Черновик доступен в Интернете: http://www.ualberta.ca/~szepesva/RLBook.HTML.

[Тед08] Р. Тедрейк. Управление обучением при промежуточных числах Рейнольдса. В мастерской по: Robotics Challenges for Machine Learning II, Международная конференция по интеллектуальным роботам и системам (IROS 2008), Ницца, Франция, 2008 г.

[TEF09] М. Токич, В. Эртель и Дж. Фесслер. Краулер, демонстратор в классе для обучения с подкреплением. В материалах 22-й Международной Флоридской искусственной Конференция Общества разведки (FLAIRS 09). AAAI Press, Менло-Парк, 2009.

[Тес95] Г. Тезауро. Обучение временной разнице и td-gammon. коммун. АКМ, 38(3), 1995. www.research.ibm.com/massive/tdl.html.

[Ток06] М. Токич. Entwicklung eines Lernfähigen Laufroboters. Diplomarbeit Hochschule Ravensburg-Weingarten, 2006. Inklusive Simulationssoftware verfü gbar auf www.hs-weingarten.de/~ertel/kibuch.

[Тур37] А. М. Тьюринг. О вычислимых числах, с приложением к проблеме навоза Entschei. проц. Лонд. Мат. Соц., 42(2), 1937.

[Тур50] А. М. Тьюринг. Вычислительная техника и интеллект. Разум, 59: 433–460, 1950.

[vA06] Л. в. Ан. Игры с целью. IEEE вычисл. Mag., 96–98, июнь 2006 г. http://www.cs.cmu.edu/~biglou/ieee-gwap.pdf.

[Вэй66] Дж. Вейценбаум. ELIZA — компьютерная программа для изучения общения человека и машины на естественном языке. коммун. АКМ, 9(1):36–45, 1966.

[WF01] И. Виттен и Э. Франк. Сбор данных. Hanser, Мюнхен, 2001 г. Библиотека DataMining Java Library WEKA: www.cs.waikato.ac.nz/~ml/weka.

[Whi96] Дж. Уиттакер. Графические модели в прикладной многомерной статистике. Уайли, Нью-Йорк, 1996.

[Ви] У. Видеманн. PhilLex, Лексикон философии. www.phillex.de/paradoxa.htm.

[Wie04] Дж. Вилмейкер. SWI-ПРОЛОГ 5.4. Университет Амстердама, 2004. www.swi-prolog.org.

[Wik10] Википедия, бесплатная энциклопедия, 2010 г. http://en.wikipedia.org.

[Победить] П. Уинстон. Демонстрация игры. http://www.ai.mit.edu/courses/6.034f/gamepair.HTML. Java-апплет для минимаксного и альфа-бета-поиска.

[Зел94] А. Целль. Моделирование Neuronaler Netze. Аддисон-Уэсли, Рединг, 1994. Описание. SNNS и JNNS: www-ra.informatik.uni-tuebingen.de/SNNS.

[ZSR+99] А. Зильке, Х. Ситтер, Т. А. Рампп, Э. Шефер, К. Хассе, В. Лоренц и М. Ротмунд. Überprü fung eines diagnostischen Scoresystems (Ohmann-Score) для смерти острый аппендицит. Чирург, 70:777–783, 1999.

Machine Translated by Google

Индекс

A

Априорная вероятность, 122

A -алгоритм, 98, 250

Действие, 86, 87, 260

Функция активации, 224, 234

Актуатор, 9

Теория адаптивного резонанса, 255

Допустимо, 98

Агент, 3, 9, 257, 259–261, 265, 267, 268, 270, 272, 273

- автономных, 8
- стоимостных, 12
- стоимостных, 140
- распределенных,
- 8 целевых, 9
- аппаратных,
- 9
- интеллектуальных, 9
- обучение,
- 8, 12, 164
- рефлекторных, 9
- программных, 9 утилитарных, 12 с памятью, 9

Агенты, распределенные, 12

ИИ, 1

Аварийный пример, 145

Альфа-бета обрезка, 103

И ветки, 70

И/или дерево, 70

Аппендицит, 121, 131

Приближение, 164, 180

APT, 255

Искусственный интеллект, 1

Ассоциативная память, 232

Атрибут, 106, 185

Автоассоциативная память, 226, 233, 238

Б

Нарды, 108, 273

Обратное распространение, 246, 265, 275

- правило обучения, 247

Возвращение, 69, 91

Обратная цепочка, 27

Пакетное обучение, 201, 243

Формула Байеса, см. теорему Байеса,
122, 155.

формула Байеса, см. теорему Байеса

Байесовская сеть, 7, 10, 64, 115, 144, 146, 184

- обучение, 157, 199

BDD, см. диаграмму бинарных решений

Уравнение

- Беллмана, 263
- принцип, 263

Узел смещения, 173

Бинарная диаграмма решений, 29

Машина Больцмана, 232

Наука о мозге, 3

Автомобиль Брайтенберга, 1, 8, 179

Коэффициент ветвления, 83,
87 средний,
84 эффективный, 87

Встроенный предикат, 76

C

C4.5, 184, 200

Исчисление,
20 Gentzen, 40
естественное рассуждение,
40

последовательное,

40 CART,

184, 198 CASC, 48

прецедентная база, 183

прецедентная аргументация, 183 CBR,

см. прецедентная

аргументация Факторы уверенности , 114 Цепное

правило для

байесовских сетей, 120, 154, 155 Chatterbot, 5 Checkers, 102, 103, 108

В. Эртель, Введение в искусственный интеллект, Темы бакалавриата по компьютерным наукам, DOI 10.1007/978-0-85729-299-5, © Springer-Verlag London Limited 2011

Machine Translated by Google

Шахматы, 102, 103, 105, 108
Церковь, Алонсо, 5
Классификация, 162
Классификатор, 162, 164, 245
Пункт, 21 -
головной, 26
определенный, 26
Закрытая формула, 32
CLP, см. Программирование логики ограничений
Кластер, 206
Кластеризация, 206, 216
иерархическая, 209
КНФ, 21 год
Когнитивная наука, 3
Дополнительный, 23
Полный, 20
Компьютерная диагностика, 144
Заключение, 26
Условная вероятность, таблица 119, см.
CPT
Условно независимые, 146, 147, 155
Кондиционирование, 149, 155
Матрица путаницы, 11, 212
Соединение, 16, 21
Конъюнктивная нормальная форма, 21
Коннекционизм, 7
Последовательный, 23
Константа, 32 года
Программирование логики ограничений, 78
Проблема удовлетворения ограничений, 78
Корреляция,
коэффициент 137,
матрица 168, 216
Функция оценки стоимости, 96
Функция стоимости, 87, 98
Матрица затрат, 136, 139
КПП, 146, 149, 156, 199, 202
Кредитное поручение, 107, 260
Перекрестная проверка, 198
Перекрёстные помехи, 236
ЦСП, 78
Проклятие размерности, 274
Вырезать, 71 год

Д

D-разделение, 156
ДАГ, 155, 200
ДАИ, 8
Интеллектуальный анализ данных, 8, 165, 166, 182, 184, 197
Де Морган, 37 лет
Решение, 139
Дерево решений, 13, 185
индукция, 165, 185
обучение, 184, 275
Логика по умолчанию, 63
Правило по умолчанию, 63
Дельта-правило, 243–245
обобщенное, 247
Демодуляция, 46
Теория Демпстера – Шефера, 115
График зависимости, 137
Ограничение по глубине, 91
Деривация, 20
Детерминированный, 89, 102
Система диагностики, 132
Разъединение, 16, 21
Метрика расстояния, 207
Распределенный искусственный интеллект, 8
Распределенное обучение, 274
Распределение, 118, 134
Динамическое программирование, 263

Е

Электронное обучение, 5
Стремление к обучению, 182, 215
Элементарное событие, 115
Элиза, 5 лет
Алгоритм ЭМ, 201, 208
Энтропия,
максимум 188, 115, 122
Окружающая среда, 9, 12
непрерывные, 12
детерминированные,
12 дискретные,
12 недетерминированные,
12 наблюдаемые, 12
частично наблюдаемые, 12
Уравнение, направленное, 47
Эквивалентность, 16
Функция оценки, 103
Событие, 115
Экспертная система, 131, 144

Ф

Факт, 26
Факторизация, 23, 45
Ложноотрицательный, 141
Ложное срабатывание, 140
Алгоритм самого дальнего соседа, 211
Особенность, 106, 162, 171, 185
Художественное пространство, 163
Решатель ограничений конечной области, 79
Предложение первого порядка, 32
Цепочка вперед, 27
Проблема с рамой, 63
Свободные переменные, 32

Machine Translated by Google

Символ функции, 32
Нечеткая логика, 8, 13, 29, 64, 115

г

Гауссовский процесс, 181, 215, 272
Общий решатель проблем, 10
Обобщение, 162
Генетическое программирование, 75
Иди, 102, 103, 108
Гол, 28
стек, 28
штат, 87
Гёдель
Курт, 5
теорема о полноте, 5, 41 теорема о
неполноте, 5, 60
GPS, 10
Градиентный спуск, 244
Жадный поиск, 96, 97, 201
Земельный срок, 41

час

Проблема с остановкой, 5
Правило Хебба, 225, 234, 247,
двоичное, 236
Эвристика, 46, 57, 94
Функция эвристической оценки, 95, 98
Иерархическое обучение, 274
Сеть Хопфилда, 226, 227, 237
Оговорка Хорна, 26, 73
Хьюгин, 150

я

ID3, 184
IDA -алгоритм, 100, 250
Немедленная награда, 260
Значение, 16
Пошаговый градиентный спуск, 245
Инкрементное обучение, 243
Независимая, 119
условно, 146, 147, 155
Безразличие, 126
Безразличные переменные, 126
Логическая машина, 42
Механизм вывода, 12
Информативность, 189
Прирост информации, 186, 189, 215
Входное разрешение, 46
Интерпретация, 16, 33
Итеративное углубление, 92, 100

дж

Джавабайес, 150

К

К метод ближайшего соседа, 178, 180
К-значит, 208
Ядро, 181, 253
Метод ядра, 253 тыс.
Метод ближайшего соседа, 180
НОЖ, 185, 211, 212
Знания, 12 базы,
12, 18, 145
последовательных,
23 инженера, 9, 12
инженерии, 12
источников, 12

л

Предположение Лапласа, 117
Вероятности Лапласа, 117
Закон экономики, 197
Ленивое обучение, 182, 215
Обучение, 161
пакет, 243
путем демонстрации, 275
распределенных,
274 иерархических,
274 инкрементных, 201,
243 машинных,
136 многоагентных,
274 подкрепленных, 89, 107, 214
полуконтролируемых, 214
контролируемых, 161, 206, 238
ТД-, 273
Агент по обучению, 164 года
Фаза обучения, 164
Скорость обучения, 225, 244
Метод наименьших квадратов, 143, 241, 242, 245
ЛЕКСМЕД, 115, 122, 131, 193
Ограниченные ресурсы, 95
Линейное приближение, 245
Линейно отделимые, 169, 170
ГУБЫ, 68
ЛИСП, 6, 10
Буквенный,
21 дополнительный, 23
Локально взвешенная линейная регрессия, 183
Логика
нечеткая, 29, 64, 115
высшего порядка, 59, 60
вероятностная, 13, 29
Теоретик логики, 6, 10
Логически верно, 17

Machine Translated by Google

М

Машинное обучение, 134, 161
Манхэттенское расстояние, 100, 207
Предельное распределение, 121
Маргинализация, 120, 121, 123, 155
Марковский процесс принятия решений,
9, 273 недетерминированный,
270 частично наблюдаемый, 261
Марковские процессы принятия решений, 261
Материальный смысл, 16, 115, 128
MaxEn t, 115, 126, 129, 131, 134, 149, 156 раздача, 126

МДП, 261, 263, 273
детерминированный,
269 недетерминированный, 270
Запоминание, 161
Обучение на основе памяти, 182, 183
МГУ, 44
Минимальное остовное дерево, 210
Модель, 17 лет
Модус поненс, 22, 30, 114, 125
Импульс, 251
Монотонный, 61
Мультиагентное обучение, 274
Мультиагентные системы, 10
МИКИН, 10, 114, 132

Н

Наивный байесовский классификатор, 143, 144, 157, 166,
175, 202, 204, 219,
298, 202, 204
Наивный реверс, 74
Уравнение Навье – Стокса, 274
Классификация
ближайших соседей,
метод 176, 175
Алгоритм ближайшего соседа, 210
Отрицание, 16
Отрицание как неудача, 73
Нейронная сеть, 6, 10, 180, 181, 221
Рекуррентные
нейронные сети, 226, 231
Нейроинформатика, 231
Неврология, 3
Нейротрансмиттер, 223
Шум, 177
Немонотонная логика, 63, 130
Нормальные уравнения, 242
Нормальная
форма конъюнктива,
21 пренекс, 37

О

Наблюдаемый, 89, 103
Бритва Оккама, 197
Автономный алгоритм, 89
Алгоритм онлайн, 89
Онтология, 53
Или ветки, 70
Ортонормированный, 234
Отелло, 102, 103
Переоснащение, 177, 197, 198, 201, 240, 243, 252, 273

СОВ, 53 года

п

Парадокс, 60
Парамодуляция, 47
Частично наблюдаемый марковский процесс принятия решений,
261
Проблема пингвинов, 77
Персептрон, 10, 169, 170, 224
Фазовый переход, 230
ПИТ, 129, 130, 149, 157
ПЛ1, 13, 32
Планирование, 76
Политика, 260
градиентный метод, 273
оптимальный, 260
ПОМДП, 261, 273
Постусловие, 52
Предварительное условие, 52
Логика предикатов, 5
первого порядка, 13, 32
Обучение предпочтениям, 166
Помещение, 26
Вероятностная
логика, 13, 63
рассуждения, 7
Вероятность, 115, 117
распределение, 118
правил, 137
Правило продукта, 120
Верификация программы, 51
ПРОЛОГ, 7, 10, 20, 28, 67
Система доказательств, 19
Переменные предложения, 15
Исчисление
высказываний,
13 логика, 15
Обрезка, 192, 198
Псевдоинверсия, 236
Чисто буквальное правило, 46, 55

Вопрос

Q-обучение, 267, 275
конвергенция, 269

Machine Translated by Google

Быстрый винт, 252

р

Случайная величина, 116

Быстрое прототипирование, 79

РДФ, 53 года

Решение в реальном времени, 95

Требование в реальном времени, 103

Рабочая характеристика приемника, 142

Армирование

обучение, 257, 260

отрицательных,

260 положительных, 260

Резолюция, 6, 22

исчисление, 10, 20

правило, 22, 30

общее, 22, 43

СЛД, 27

Решающая, 22

Награда

со скидкой, 260

немедленно, 260, 269

Управление рисками, 141

РобоКубок, 11, 273

Робот, 9, 258

ходьба-, 258

Робототехника, 257

Кривая ROC, 143, 214

РПроп, 252

С

Образец, 185

Удовлетворительная, 17

Диаграмма рассеяния, 163

Оценка: 132, 143, 203, 219, 242

Алгоритм

поиска, 86 полных,

87 оптимальных,

89 эвристических,

84 пробельных,

23, 27, 40, 46 древовидных,

86

неинформированных, 84

Самоорганизующиеся карты, 255

Семантические деревья, 28

Семантическая паутина, 53

Семантика

декларативная (PROLOG), 70

процедурная (PROLOG), 70, 73

Полуразрешимый, PL1, 59

Полуконтролируемое обучение, 214

Чувствительность, 141, 148

Датчик, 9

Набор стратегий поддержки, 46, 55

Сигмовидная функция, 225, 241, 246

Подпись, 15

Сходство, 175

Имитация отжига, 232

Ситуационный расчет, 63

Сколемизация, 39

Разрешение SLD, 30

Повторное использование программного обеспечения, 52

Решение, 87

Звук, 20

Спам, фильтр

204, 11, 204

Специфика, 141

Исходное состояние, 87

Состояние, 87, 258, 259

пробел, 87

функция перехода, 260

Статистическая индукция, 136

Подгол, 28, 69

Аксиома подстановки, 36

Сусумпшн, 46

Опорный вектор, 252

машина, 252, 272

Машина опорных векторов, 181

SVM, см. машину опорных векторов

Т

Целевая функция, 164

Тавтология, 17

TD

-ошибка, 271

-гаммон, 273

-обучение, 271, 273

Учебный ящик, 275

Ошибка временной

разности,

271 обучение, 271

Срок, 32

переписка, 47

Данные испытаний, 164, 197

Текст

классификация, 204 добыча

полезных ископаемых, 166

Доказательство теорем, 6, 42, 43, 47, 51, 52

Тренировочные данные, 58, 164, 197

Функция перехода, 270

Правда, 34

Таблица истинности,

16 метод, 19

Тьюринг

Алан, 5

тест, 4

Твити пример, 60, 63, 130

Machine Translated by Google

U
Объединяемый, 44
Объединение, 44
Объединитель,
44 наиболее общий, 44
Поиск единой стоимости, 90

Единичная статья,
разрешение 46, разрешение 46
Неудовлетворительный, 17

B
Действительно, 17, 34
Итерация значения, 263
переменная, 32
ВДМ-СЛ, 52
Спецификация Венского метода разработки
Язык, 52
Диаграмма Вороного, 177

Вт
Шагающий робот, 258
ВАМ, 68, 75
Абстрактная машина Уоррена, 68 лет
Уотсон, 13 лет
ВЕКА, 185, 211

Machine Translated by Google


```
 1 П([Лейко7=0-6к]                | [Diag4=отрицательный] * [Age10=16-20]) = [0,132,0,156];
 2 П([Лейко7=6-8к]                | [Diag4=отрицательный] * [Age10=16-20]) = [0,257,0,281];
 3 P([Leuco7=8-10k] | [Diag4=отрицательный] * [Age10=16-20]) = [0,250,0,274];
 4 P([Leuco7=10-12k] | [Diag4=отрицательный] * [Age10=16-20]) = [0,159,0,183];
 5 P([Leuco7=12-15k] | [Diag4=отрицательный] * [Age10=16-20]) = [0,087,0,112];
 6 P([Leuco7=15-20k] | [Diag4=отрицательный] * [Age10=16-20]) = [0,032,0,056];
 7 П([Лейко7=20к-]                | [Diag4=отрицательный] * [Age10=16-20]) = [0,000,0,023];
 8 П([Лейко7=0-6к]                | [Diag4=отрицательный] * [Age10=21-25]) = [0,132,0,172];
 9 П([Лейко7=6-8к]                | [Diag4=отрицательный] * [Age10=21-25]) = [0,227,0,266];
10 P([Leuco7=8-10k] | [Diag4=отрицательный] * [Age10=21-25]) = [0,211,0,250];
11 P([Leuco7=10-12k] | [Diag4=отрицательный] * [Age10=21-25]) = [0,166,0,205];
12 P([Leuco7=12-15k] | [Diag4=отрицательный] * [Age10=21-25]) = [0,081,0,120];
13 P([Leuco7=15-20k] | [Diag4=отрицательный] * [Age10=21-25]) = [0,041,0,081];
14 П([Левко7=20к-]                | [Diag4=отрицательный] * [Age10=21-25]) = [0,004,0,043];
```

Рис. 7.9 Некоторые правила LEXMED с вероятностными интервалами. «*» здесь означает « »

Оценка вероятностей правил Структура графа зависимостей описывает структуру изученных правил.11 Правила здесь имеют разную сложность: есть правила, которые только описывают распределение возможных диагнозов.
(априорные правила, например (7.13)), правила, описывающие зависимость между диагноз и симптом (правила с простыми условиями, например (7.14)), и наконец, правила, описывающие зависимость между диагнозом и двумя симптомами, как показано на рис. 7.9 в синтаксисе PIT .

$$P(\text{Diag4} = \text{воспаленный}) = 0{,}40, \quad P \tag{7.13}$$

$$(\text{Sono2} = \text{да} \mid \text{Diag4} = \text{воспаленный}) = 0{,}43, \tag{7.14}$$

$$P(\text{P4Q2} = \text{да} \mid \text{Diag4} = \text{воспаление} \quad \text{P2Q2} = \text{да}) = 0{,}61. \tag{7.15}$$

Чтобы сохранить контекстную зависимость сохраненных знаний как можно меньше, все правила содержат диагноз в своих условиях, а не в виде выводов. Это очень похоже на построение многих медицинских книг с формулировками рода «При аппендиците мы обычно видим . . . ». Как показано ранее в примере 7.6. на стр. 121, однако, это не представляет проблемы, поскольку при использовании байесовского формулы, ЛЕКСМЕД автоматически переводит эти правила в правильную форму.

Численные значения для этих правил оцениваются путем подсчета их частоты в базу данных. Например, значение в (7.14) определяется путем подсчета и вычисления

$$\frac{|\text{Diag4} = \text{воспаление} \quad \text{Sono2} = \text{да}|}{|\text{Diag4} = \text{воспаление}|} = 0{,}43.$$

11Разница между этой и байесовской сетью, например, в том, что правила оснащены с вероятностными интервалами и что только после применения принципа максимальной энтропии построена вероятностная модель.

Machine Translated by Google

Экспертные правила

Поскольку база данных по аппендициту содержит только пациентов, перенесших операции, правила неспецифической боли в животе (NSAP) получают свои значения из предложения медицинских специалистов. Опыт LEXMED подтверждает, что вероятностные правила легко читаются и могут быть непосредственно переведены на естественный язык. Заявления медицинских экспертов о частотных соотношениях конкретных симптомов и диагноз, будь то из литературы или в результате интервью, может поэтому можно включить в базу правил с небольшими затратами. Для моделирования неопределенности экспертных знаний оказалось эффективным использование интервалов вероятности. Экспертные знания были в основном получены от участвующих хирургов, доктора. Rampf and Dr. Hontschik и их публикации [Hon94].

Когда экспертные правила созданы, база правил готова. Затем рассчитывается полная вероятностная модель методом максимальной энтропии по методу PIT-система.

Диагностические запросы

Используя эффективно хранимую модель вероятностей, LEXMED рассчитывает вероятности для четырех возможных диагнозов в течение нескольких секунд. Например, мы предполагаем, что следующий вывод:

	Результаты диагностики ПИТ		
Диагноз	Аппендикс воспаленный Аппендикс перфорирован 0,24 0,16	Отрицательный	Другой
Вероятность		0,57	0,03

Решение должно быть принято на основе этих четырех значений вероятности, чтобы следовать одному из четыре метода лечения: операция, экстренная операция, стационарное наблюдение или амбулаторное наблюдение.12 При этом вероятность отрицательного диагноза в этом случае перевешивает остальные, отправлять больного домой как здорового — не самое удачное решение. Мы можем ясно видеть, что, даже когда вероятности диагнозов были подсчитано, диагностика еще не закончена.

Скорее, теперь задача состоит в том, чтобы вывести оптимальное решение из этих вероятностей. К С этой целью пользователь может попросить LEXMED рассчитать рекомендуемое решение.

7.3.5 Управление рисками с использованием матрицы затрат

Как теперь вычисленные вероятности можно оптимально преобразовать в решения? Наивный алгоритм назначал бы решение каждому диагнозу и в конечном итоге выбирал бы решение, которое соответствует наибольшей вероятности. Предположим, что вычисленное вероятности 0,40 для диагноза аппендицит (воспаленный или перфоративный), 0,55 для диагноза отрицательный и 0,05 для диагноза другой. Наивный алгоритм теперь выберите (слишком рискованное) решение «без операции», поскольку оно соответствует диагнозу с большей вероятностью. Лучший метод состоит в сравнении затрат

12 Амбулаторное наблюдение означает, что больной выписан домой.

Machine Translated by Google

Таблица 7.3 Матрица стоимости LEXMED вместе с расчетными вероятностями диагноза пациента

Терапия	Вероятность различных диагнозов				
	воспаленный	Перфорированный	Отрицательный	Другой	
	0,25	0,15	0,55	0,05	
Операция	0	500	5800	6000	3565
Аварийная операция	500	0	6300	6500	3915
Амбулаторное наблюдение.	12000	150000	0	16500	26325
Другой	3000	5000	1300	0	2215
Стационарная обсерв.	3500	7000	400	600	2175

возможных ошибок, которые могут возникнуть для каждого решения. Ошибка количественно определяется форме «(гипотетической) дополнительной стоимости текущего решения по сравнению с оптимальным». Приведенные значения содержат расходы на больницу, страховую компанию, пациенту (например, риск послеоперационных осложнений), так и другим сторонам (например, отсутствие на работе) с учетом отдаленных последствий.
Эти затраты приведены в таблице 7.3.

Записи окончательно усредняются для каждого решения, т. е. суммируются при принятии с учетом их частот. Они перечислены в последнем столбце таблицы 7.3. Наконец, предлагается решение с наименьшей средней ценой ошибки. В таблице 7.3 матрица дана вместе с рассчитанным для пациента вектором вероятности (в в этом случае: (0,25, 0,15, 0,55, 0,05)). Последний столбец таблицы содержит результат расчеты средних ожидаемых издержек ошибок. Значение операции в первой строке вычисляется как 0,25 · 0 + 0,15 · 500 + 0,55 · 5800 + 0,05 · 6000 = 3565, средневзвешенное значение всех затрат. Оптимальные решения вводятся с (дополнительными) затратами, равными 0. Система принимает решение о лечении с минимальным средним значением. расходы. Таким образом, это пример агента, ориентированного на затраты.

Матрица стоимости в бинарном случае

Чтобы лучше понять матрицу затрат и управление рисками, мы ограничим Система ЛЕКСМЕД к двузначному решению между диагнозом аппендицит и НСАП. В качестве возможных методов лечения только операция с вероятностью p1 и амбулаторное лечение. наблюдение с вероятностью p2 может быть выбрано. Таким образом, матрица затрат представляет собой матрицу 2 × 2. матрица вида

$$\begin{matrix} 0 & k2 \\ k1 & 0 \end{matrix}.$$

Два нуля на диагонали обозначают правильную решающую операцию в случае аппендицита и амбулаторное наблюдение по поводу NSAP. Параметр k2 означает ожидаемые затраты, которые возникают, когда пациент без воспаленного аппендикса прооперирован. Эта ошибка называется ложным срабатыванием. С другой стороны, решение

Machine Translated by Google

амбулаторное наблюдение в случае аппендицита является ложноотрицательным. Вектор вероятности $(p1, p2)^T$ теперь умножается на эту матрицу, и мы получаем вектор

$$(2 \text{ лиц изн}, 1 \text{ лиц изн})^T$$

со средними дополнительными затратами на два возможных лечения. Поскольку решение принимает во внимание только взаимосвязь двух компонентов, вектор можно умножить на любой скалярный коэффициент. Выбираем 1/k1 и получаем ((k2/k1)p2, p1).
Таким образом, здесь имеет значение только соотношение k = k2/k1 . Тот же результат дает более простая матрица затрат

$$\begin{matrix} 0 & k \\ 1 & 0 \end{matrix},$$

который содержит только переменную k. Этот параметр очень важен, поскольку он определяет управление рисками. Изменяя k, мы можем подогнать «рабочую точку» диагностической системы. При k система оказывается в чрезвычайно рискованной ситуации, потому что ни один пациент никогда не будет оперирован, в результате чего она дает не ложноположительные классификации, а много ложноотрицательных. В случае k = 0 условия обратные, и все больные оперируются.

7.3.6 Производительность

ЛЕКСМЕД предназначен для использования в медицинской практике или машине скорой помощи. Предпосылками к применению ЛЕКСМЕДа являются острые боли в животе в течение нескольких часов (но менее пяти дней). Кроме того, LEXMED (в настоящее время) специализируется на аппендиците, а это означает, что по другим заболеваниям система содержит очень мало информации.

В рамках проспективного исследования в 14-й больнице Нотельфера была создана репрезентативная база данных со 185 случаями. В нем находятся пациенты больницы, которые обратились в клинику после нескольких часов острой боли в животе и подозрении на аппендицит. У этих пациентов отмечаются симптомы и диагноз (подтвержденный по образцу ткани в случае операции).

Если пациентов отпускали домой (без операции) после пребывания в течение нескольких часов или 1–2 дней без особых жалоб или без них, то впоследствии по телефону спрашивали, остается ли у пациента бессимптомным или был ли установлен положительный диагноз в последующее лечение.

Чтобы упростить представление и сделать лучшее сравнение с аналогичными исследованиями, LEXMED был ограничен двумя значениями различия между аппендицитом и НСАП, как описано в разд. 7.3.5. Теперь k варьируется от нуля до бесконечности, и для каждого значения k чувствительность и специфичность измеряются в сравнении с тестовыми данными.
Меры чувствительности

$$P(\text{классифицированный положительный} \mid \text{положительный}) = \frac{|\text{положительные и классифицированные как положительные}|}{|\text{положительный}|},$$

Machine Translated by Google

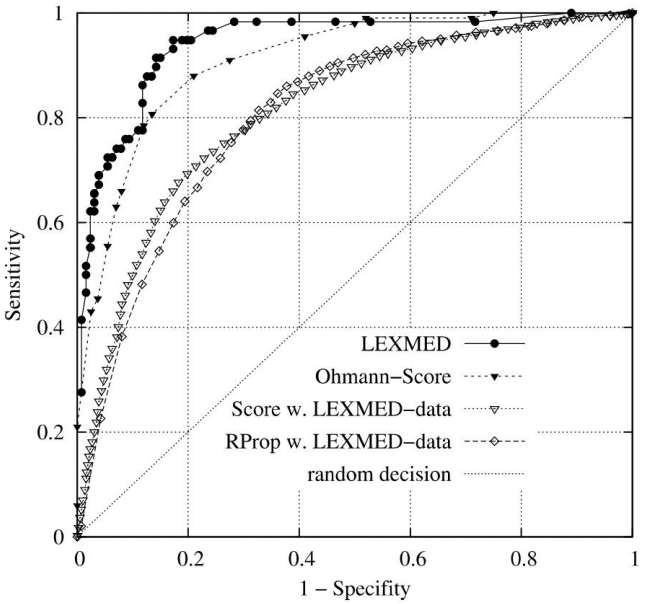


Рис. 7.10 ROC-кривая из LEXMED в сравнении с показателем Омана и двумя дополнительными моделями

то есть относительная доля правильно идентифицированных положительных случаев. Он показывает, насколько чувствительна диагностическая система. Специфика, с другой стороны, измеряет

$$P(\text{классифицированный отрицательный} \mid \text{отрицательный}) = \frac{|\text{отрицательные и классифицированные как отрицательные}|}{|\text{отрицательный}|},$$

то есть относительная доля правильно идентифицированных отрицательных случаев.

Мы приводим результаты чувствительности и специфичности на рис. 7.10 для 0 k < . Эта кривая обозначается ROC-кривой или рабочей характеристикой приемника. Прежде чем мы перейдем к анализу качества LEXMED, несколько слов о значении ROC-кривой. Линия, делящая диаграмму пополам по диагонали, нарисована для ориентации.
Все точки на этой прямой соответствуют случайному решению. Например, точка (0,2, 0,2) соответствует специфичности 0,8 при чувствительности 0,2. Мы можем довольно легко прийти к этому, классифицируя новый случай, не глядя на него, с вероятностью 0,2 для положительного и 0,8 для отрицательного. Таким образом, каждая система диагностики, основанная на знаниях, должна генерировать ROC, которая явно лежит выше диагонали.

Экстремальные значения на кривой ROC также интересны. В точке (0, 0) все три кривые пересекаются. Соответствующая система диагностики классифицировала бы все случаи как отрицательные. Другое экстремальное значение (1, 1) соответствует системе, которая решает сделать операцию каждому пациенту и, таким образом, имеет чувствительность 1. Мы могли бы назвать ROC-кривую характеристической кривой для двухзначных диагностических систем. Идеальная диагностическая система должна иметь характеристическую кривую, состоящую только из точек (0, 1), и, таким образом, иметь 100% специфичность и 100% чувствительность.

Machine Translated by Google

Теперь давайте проанализируем кривую ROC. При чувствительности 88% ЛЕКСМЕД достигает специфичности 87% (k = 0,6). Для сравнения приведена общеизвестная шкала Оманна для аппендицита [OMYL96, ZSR+99]. Поскольку LEXMED почти везде выше или левее шкалы Омана, его среднее качество диагностики явно выше. Это неудивительно, потому что оценки просто слишком слабы для моделирования интересных предложений. В разд. 8.6 и в упражнении 8.16 на стр. 219 мы покажем, что оценки эквивалентны частному случаю наивного Байеса, то есть предположению, что все симптомы попарно независимы, когда известен диагноз. Однако при сравнении LEXMED с баллами следует отметить, что для оценки Омана использовалась статистически репрезентативная база данных, а для LEXMED использовалась нерепрезентативная база данных, дополненная экспертными знаниями.
Чтобы получить представление о качестве данных LEXMED по сравнению с данными Ohmann, была рассчитана линейная оценка с использованием метода наименьших квадратов (см. раздел 9.4.1), который также используется для сравнения. Кроме того, нейронная сеть была обучена на данных LEXMED с помощью алгоритма RProp (см. раздел 9.5). Сила объединения данных и экспертных знаний четко отображается в разнице между кривой LEXMED и кривыми системы оценок и алгоритма RProp.

7.3.7 Области применения и опыт

ЛЕКСМЕД не должен заменять мнение опытного хирурга. Однако, поскольку специалист не всегда доступен в клинических условиях, запрос LEXMED предлагает существенное второе мнение. Особенно интересным и целесообразным является применение системы в клинической скорой помощи и для врачей общей практики.

Способность LEXMED к обучению, которая позволяет учитывать дополнительные симптомы, дополнительные данные пациента и дополнительные правила, также открывает новые возможности в клинике. Для особо редких групп, трудно поддающихся диагностике, например детей до шести лет, ЛЕКСМЕД может использовать данные педиатров или другие специальные базы данных, чтобы помочь даже опытным хирургам.

Помимо непосредственного использования в диагностике, LEXMED также поддерживает меры по обеспечению качества. Например, страховые компании могут сравнивать качество диагностики больниц с качеством экспертных систем. При дальнейшем развитии матрицы затрат, созданной в ЛЕКСМЕД (с согласия врачей, страховых компаний и пациентов), качество диагностики врачей, компьютерной диагностики и других медицинских учреждений станет проще сравнивать.

ЛЕКСМЕД указал на новый способ построения автоматических диагностических систем. Используя язык теории вероятностей и алгоритм MaxEnt, индуктивно статистически полученные знания объединяются со знаниями от экспертов и из литературы. Подход, основанный на вероятностных моделях, является теоретически элегантным, общеприменимым и дал очень хорошие результаты в небольшом исследовании.

ЛЕКСМЕД применяется на практике в больнице Nothelfer № 14 в Вайнгартене с 1999 года и очень хорошо себя зарекомендовал. Он также доступен на сайте www.lexmed.de, разумеется, без гарантии. Его качество диагностики сравнимо с качеством опытного хирурга и, таким образом, лучше, чем у среднего врача общей практики или неопытного врача в поликлинике.

Machine Translated by Google

Несмотря на этот успех, стало очевидным, что очень трудно продавать такую систему на коммерческой основе в немецкой медицинской системе. Одна из причин этого заключается в том, что не существует свободного рынка, который мог бы продвигать более высокое качество (в данном случае более совершенные диагнозы) с помощью механизмов отбора. Кроме того, в медицине время для широкого использования интеллектуальных методов еще не наступило — даже в 2010 году. Одной из причин этого могут быть консервативные учения в этом отношении на факультетах немецких медицинских школ.

Еще одной проблемой является стремление многих пациентов к личному совету и уходу со стороны врача, а также опасение, что с внедрением экспертных систем пациент будет общаться только с машиной. Однако этот страх совершенно необоснован. Даже в долгосрочной перспективе медицинские экспертные системы не могут заменить врача.
Однако их можно использовать так же, как лазерную хирургию и магнитно-резонансную томографию, с пользой для всех участников. С момента создания первой медицинской компьютерной диагностической системы де Домбаля в 1972 г. прошло почти 40 лет. Еще неизвестно, будет ли медицина ждать еще 40 лет, пока компьютерная диагностика не станет признанным медицинским инструментом.

7.4 Рассуждения с помощью байесовских сетей

Одна проблема, связанная с рассуждениями с использованием вероятности на практике, уже указывалась в разд. 7.1. Если используются d переменных X1,...,Xd с n значениями каждая, то распределение вероятностей соответствующие d общих значений. Это означает, что в худшем случае использует n памяти, а время вычислений для определения заданных вероятностей экспоненциально растет с увеличением числа переменных.

На практике приложения обычно очень структурированы, а дистрибутив содержит много избыточности. Это означает, что его можно значительно уменьшить соответствующими методами. Использование байесовских сетей доказало здесь свою эффективность и является одним из методов ИИ, которые успешно применяются на практике. Байесовские сети используют знания о независимости переменных для упрощения модели.

7.4.1 Независимые переменные

В простейшем случае все переменные попарно независимы, и это тот случай, когда

$$P(X1,...,Xd) = P(X1) \cdot P(X2) \cdots P(Xd).$$

Таким образом, все записи в распределении могут быть рассчитаны по значениям d $P(X1), \ldots, P(Xd)$. Однако интересные приложения обычно невозможно смоделировать, поскольку условные вероятности становятся тривиальными[13].

$$P(A|B) = \frac{П(A, Б)}{П(Б)} \text{ "=" } \frac{П(A)\,П(Б)}{П(Б)} = P(A)$$

13 В наивном байесовском методе предполагается независимость всех атрибутов, и этот метод был успешно применен к классификации текстов (см. раздел 8.6).

Machine Translated by Google

все условные вероятности сводятся к априорным вероятностям. Ситуация становится более интересной, когда только часть переменных является независимой или независимой при определенных условиях. Для рассуждений в ИИ зависимости между переменными оказываются важными и должны использоваться.

Мы хотели бы обрисовать рассуждения с байесовскими сетями на простом и очень наглядном примере Дж. Перла [Pea88], который стал широко известен благодаря [RN10] и теперь является базовым знанием ИИ.

Пример 7.7 (Пример с сигнализацией) Неженатый Боб установил в своем доме систему сигнализации для защиты от грабителей. Боб не слышит будильник, когда работает в офисе. Поэтому он попросил двух своих соседей, Джона в соседнем доме слева и Мэри в доме справа, позвонить ему в его офис, если они услышат его тревогу. Через несколько лет Боб знает, насколько надежны Джон и Мэри, и моделирует их поведение при вызове с использованием условной вероятности следующим образом.14

$$P(J|Al) = 0{,}90 \quad P(M|Al) = 0{,}70, \quad P(J|\neg Al) = 0{,}05 \quad P(M|\neg Al) = 0{,}01.$$

Поскольку Мэри плохо слышит, она не слышит сигнал тревоги чаще, чем Джон. Однако Джон иногда путает тревогу в доме Боба с тревогами в других домах. Тревога срабатывает при краже со взломом, но также может быть вызвана (слабым) землетрясением, что может привести к ложной тревоге, потому что Боб хочет знать о кражах со взломом только в своем офисе. Эти отношения моделируются

$$P(Al|Bur,Ear) = 0{,}95,$$

$$P(Al|Bur,\neg Ear) = 0{,}94,$$

$$P(Al|\neg Bur,Ear) = 0{,}29,$$

$$P(Al|\neg Bur,\neg Ear) = 0{,}001,$$

а также априорные вероятности P (Bur) = 0,001 и P (Ear) = 0,002. Эти две переменные независимы, потому что землетрясение не строят планы на основе привычек грабителей, и, наоборот, землетрясение невозможно предсказать, поэтому у грабителей нет возможности соответствующим образом устанавливать свой график.

Теперь к этой базе знаний делаются запросы. Например, Боба может интересовать P (Bur|J M), P (J|Bur) или P (M|Bur). То есть он хочет знать, насколько чутко переменные J и M реагируют на сообщение о взломе.

14Двоичные переменные J и M обозначают два события «Джон звонит» и «Мэри звонит» соответственно, Al — «звуки тревожной сирены», Bur — «кража со взломом» и Ear — «землетрясение».

Machine Translated by Google

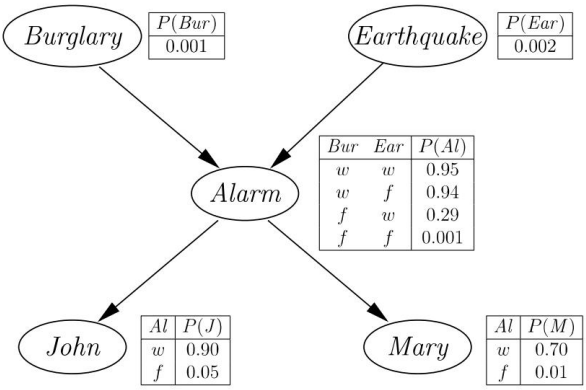


Рис. 7.11 Байесовская сеть для примера тревоги с соответствующими CPT

7.4.2 Графическое представление знаний в виде байесовской сети

Мы можем значительно упростить практическую работу, графически представив знания, сформулированные в виде условной вероятности. На рис. 7.11 показана байесовская сеть для примера тревоги. Каждый узел в сети представляет собой переменную, а каждое направленное ребро — утверждение условной вероятности. Ребро от Al до J, например, представляет два значения P (J |Al) и P (J |¬Al), которые задаются в виде таблицы, так называемой CPT (таблицы условной вероятности). CPT узла перечисляет все условные вероятности переменной узла, обусловленные всеми узлами, соединенными входящими ребрами.

Изучая сеть, мы можем спросить себя, почему в нее не включены другие ребра, кроме четырех нарисованных. Два узла Bur и Ear не связаны, поскольку переменные независимы. Все остальные узлы имеют родительский узел, что немного усложняет рассуждения. Сначала нам понадобится понятие условной независимости.

7.4.3 Условная независимость

Аналогично независимости случайных величин даем

Определение 7.6. Две переменные A и B называются условно независимыми при заданном C, если

$$P(A, B|C) = P(A|C) \cdot P(B|C).$$

Это уравнение верно для всех комбинаций значений всех трех переменных (то есть для распределения), которые мы видим в обозначениях. Теперь посмотрим на узлы J и M в примере с тревогой, которые имеют общий родительский узел Al. Если Джон и Мэри

Machine Translated by Google

независимо реагировать на аварийный сигнал, то две переменные J и M независимы при заданном Al, то есть:

$$P(J,M|Al) = P(J|Al) \cdot P(M|Al).$$

Если значение Al известно, например, из-за срабатывания тревоги, то переменные J и M независимы (при условии Al = w). Из-за условной независимости двух переменных J и M ребро между этими двумя узлами не добавляется. Однако J и M не являются независимыми (см. упражнение 7.11 на стр. 159).

Совершенно аналогична связь между двумя переменными J и Bur, потому что Джон реагирует не на кражу со взломом, а на тревогу. Это может быть, например, из-за высокой стены, которая загораживает ему обзор собственности Боба, но он все еще может слышать сигнал тревоги. Таким образом, J и Bur независимы при заданных Al и

$$P(J,Bur|Al) = P(J|Al) \cdot P(Bur|Al).$$

При тревоге переменные J и Ear, M и Bur, а также M и Ear также независимы. Для вычислений с условной независимостью полезны следующие характеристики, эквивалентные приведенному выше определению:

Теорема 7.5 Следующие уравнения попарно эквивалентны, что означает, что каждое отдельное уравнение описывает условную независимость переменных A и B при заданном C.

$$P(A, B|C) = P(A|C) \cdot P(B|C), \quad (7,16)$$

$$P(A|B,C) = P(A|C), \quad (7,17)$$

$$P(B|A,C) = P(B|C). \quad (7,18)$$

Доказательство. С одной стороны, используя условную независимость (7.16), мы можем заключить, что

$$P(A, B, C) = P(A, B|C)P(C) = P(A|C)P(B|C)P(C).$$

С другой стороны, правило произведения дает нам

$$P(A, B, C) = P(A|B,C)P(B|C)P(C).$$

Таким образом, $P(A|B,C) = P(A|C)$ эквивалентно (7.16). Аналогично получаем (7.18) , переставляя местами A и B в этом выводе.

Machine Translated by Google

7.4.4 Практическое применение

Теперь мы снова обратимся к примеру с тревогой и покажем, как байесовскую сеть на рис. 7.11 на стр. 146 можно использовать для рассуждений. Боба интересует, например, чувствительность двух его тревожных репортеров Джона и Мэри, то есть P (J |Bur) и P (M |Bur). Однако значения P (Bur|J) и P (Bur|M), а также P (Bur|J,M) для него еще важнее. Начнем с P (J |Bur) и вычислим

$$P(J \mid Bur) = \frac{P(Дж, Бур)}{П(Бур)} \overset{"="}{} \frac{P(J, Bur, Al) + P(J, Bur, \neg Al)}{П(Бур)} \tag{7.19}$$

и

$$P(J,Bur,Al) = P(J \mid Bur,Al)P(Al \mid Bur)P(Bur) = P(J \mid Al)P(Al \mid Bur)P(Bur), \tag{7.20}$$

где для последних двух уравнений мы использовали правило произведения и условную независимость J и Bur при данном Al. Подставляя в (7.19), получаем

$$\begin{aligned} P(J \mid Бур) &= P\frac{P(J \mid Al)P(Al \mid Bur)P(Bur) + P(J \mid \neg Al)P(\neg Al \mid Bur)P(Bur)}{(Бур)} \\ &= P(J \mid Al)P(Al \mid Bur) + P(J \mid \neg Al)P(\neg Al \mid Bur). \end{aligned} \tag{7.21}$$

Здесь P (Al |Bur) и P (¬Al |Bur) отсутствуют. Поэтому мы вычисляем

$$\begin{aligned} P(Ал \mid Бур) &= \frac{П(Ал, Бур)}{П(Бур)} \overset{"="}{} \frac{P(Al, Bur, Ear) + P(Al, Bur, \neg Ear)}{П(Бур)} \\ &\overset{"="}{} \frac{P(Al \mid Bur,Ear)P(Bur)P(Ear) + P(Al \mid Bur,\neg Ear)P(Bur)P(\neg Ear)}{П(Бур)} \\ &= P(Al \mid Bur,Ear)P(Ear) + P(Al \mid Bur,\neg Ear)P(\neg Ear) \\ &= 0{,}95 \cdot 0{,}002 + 0{,}94 \cdot 0{,}998 = 0{,}94 \end{aligned}$$

а также P (¬Al |Bur) = 0,06 и подставляем это в (7.21) , что дает результат

$$P(J \mid Bur) = 0{,}9 \cdot 0{,}94 + 0{,}05 \cdot 0{,}06 = 0{,}849.$$

Аналогично вычисляем P (M |Bur) = 0,659. Теперь мы знаем, что Джон вызывает около 85% всех краж со взломом, а Мэри — около 66% всех краж со взломом. Вероятность того, что они оба уравняют, вычисляется в силу условной независимости как

$$P(J,M \mid Bur) = P(J \mid Bur)P(M \mid Bur) = 0{,}849 \cdot 0{,}659 = 0{,}559.$$

Однако более интересна вероятность звонка от Джона или Мэри.

Machine Translated by Google

$$P(J \quad M|Bur) = P(\neg(\neg J, \neg M)|Bur) = 1 \quad P(\neg J, \neg M|Bur)$$
$$= 1 \quad P(\neg J|Bur)P(\neg M|Bur) = 1 \quad 0{,}051 = 0{,}948.$$

Таким образом, Боб получает уведомление примерно о 95% всех краж со взломом. Теперь, чтобы вычислить $P(Bur|J)$, мы применяем формулу Байеса, которая дает нам

$$P(Бур|J) = \frac{P(J \mid Бур)\,P(Бур)}{P(Дж)} \overset{"="}{} \frac{0{,}849 \cdot 0{,}001 =}{0{,}052} 0{,}016.$$

Очевидно, только около 1,6% всех звонков от Джона на самом деле связаны со взломом. Поскольку вероятность ложных тревог для Мэри в пять раз меньше, при $P(Bur|M) = 0{,}056$ мы имеем значительно более высокую достоверность при звонке от Мэри.
Боб должен серьезно беспокоиться о своем доме только в том случае, если они оба звонят, потому что $P(Bur|J,M) = 0{,}284$ (см. упражнение 7.11 на стр. 159).

В (7.21) на стр. 148 мы показали с

$$P(J|Bur) = P(J|Al)P(Al|Bur) + P(J|\neg Al)P(\neg Al|Bur)$$

как мы можем «вставить» новую переменную. Это соотношение в общем случае выполняется для двух переменных A и B при введении дополнительной переменной C и называется обусловленным:

$$P(A|B) = \sum_c P(A|B,C=c)P(C=c|B).$$

Если, кроме того, A и B условно независимы при заданном C, эта формула упрощается до

$$P(A|B) = \sum_c P(A|C=c)P(C=c|B).$$

7.4.5 Программное обеспечение для байесовских сетей

Мы дадим краткое введение в два инструмента на примере будильника. Мы уже знакомы с системой PIT. Мы вводим значения из CPT в синтаксисе PIT в онлайн-окно ввода на сайте www.pit-systems.de. После ввода, показанного на рис. 7.12 на стр. 150 , получаем ответ:

P([Эйнбрух=t] | [Джон=t] И [Мэри=t]) = 0,2841.

Хотя PIT не является классическим байесовским сетевым инструментом, он может принимать произвольные условные вероятности и запросы в качестве входных данных и вычислять правильные результаты. Можно показать [Sch96], что при вводе CPT или эквивалентных правил принцип MaxEnt подразумевает ту же условную независимость и, следовательно, те же ответы, что и байесовская сеть. Таким образом, байесовские сети являются частным случаем MaxEnt.

Machine Translated by Google


```
1 var Тревога{t,f}, Взлом{t,f}, Землетрясение{t,f}, Джон{t,f}, Мэри{t,f};
2
3 P([Землетрясение=t]) = 0,002; 4 P([Кража
со взломом=t]) = 0,001; 5 P([Тревога=t] |
[Кража со взломом=t] И [Землетрясение=t]) = 0,95; 6 P([Тревога=t] | [Кража со взломом=t]
И [Землетрясение=f]) = 0,94; 7 P([Тревога=t] | [Кража со взломом=f] И [Землетрясение=t])
= 0,29; 8 P([Тревога=t] | [Кража со взломом=f] И [Землетрясение=f]) = 0,001; 9 P([Джон=t]
| [Тревога=t]) = 0,90; 10 P([Джон=t] | [Тревога=f]) = 0,05; 11 P([Мария=t] | [Тревога=t]) =
0,70; 12 P([Мария=t] | [Тревога=f]) = 0,01; 13 14
QP([Кража со взломом=t] | [Джон=t] И [Мэри=t]);
```

Рис. 7.12 Ввод PIT для примера тревоги

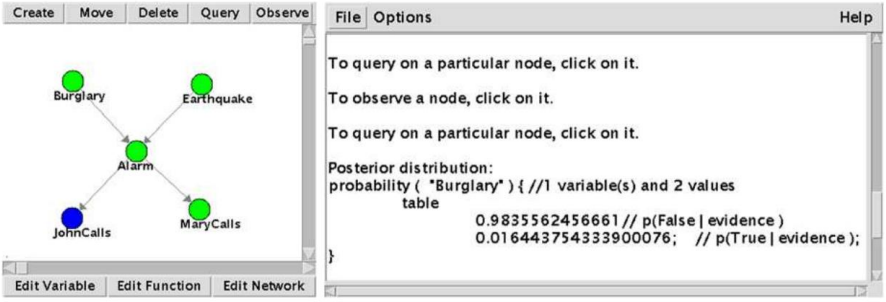


Рис. 7.13 Пользовательский интерфейс JavaBayes: слева — графический редактор, справа — консоль, где в качестве вывода выдаются ответы

Далее мы рассмотрим JavaBayes [Coz98], классическую систему, также свободно доступную в Интернете, с графическим интерфейсом, показанным на рис. 7.13. С помощью графического сетевого редактора можно манипулировать узлами и ребрами, а также редактировать значения в CPT. Кроме того, значения переменных могут быть назначены с помощью «Наблюдения», а значения других переменных могут быть вызваны с помощью «Запроса». Затем ответы на запросы появляются в окне консоли.

Профессиональная коммерческая система Hugin значительно мощнее и удобнее. Например, Хьюгин может использовать непрерывные переменные в дополнение к дискретным переменным. Он также может изучать байесовские сети, то есть полностью автоматически генерировать сеть из статистических данных (см. раздел 8.5).

7.4.6 Разработка байесовских сетей

Компактная байесовская сеть очень понятна и значительно более информативна для читателя, чем полное распределение вероятностей. Кроме того, требуется гораздо меньше

Machine Translated by Google

Память. Для переменных v1,...,vn с |v1|,..., |vn| различные значения каждый, распределение имеет в общей сложности

$$\prod_{i=1}^{n} |v_i| \quad 1$$

самостоятельные записи. В примере с тревогой все переменные являются бинарными. Таким образом, для всех переменных |vi | = 2, и в распределении 25 1 = 31 независимый элемент. Чтобы вычислить количество независимых записей для байесовской сети, необходимо определить общее количество всех записей всех CPT. Для узла vi с ki родительскими узлами имеет ei1,...,eiki, ассоциированный CPT

$$(|v_i| \quad 1) \prod_{j=1}^{k_i} |e_{ij}|$$

записи. Тогда все CPT в сети вместе имеют

$$\sum_{i=1}^{n} (|v_i| \quad 1) \prod_{j=1}^{k_i} |e_{ij}| \tag{7.22}$$

записи.15 Для примера тревоги результатом будет

$$2 + 2 + 4 + 1 + 1 = 10$$

независимые записи, которые однозначно описывают сеть. Сравнение сложности памяти между полным распределением и байесовской сетью становится более ясным, если мы предположим, что все n переменных имеют одинаковое число b значений и каждый узел имеет k родительских узлов. Тогда (7.22) можно упростить, и все КПТ вместе имеют n(b 1)bk элементов. Полное распределение содержит b не может быть усиления, только тогда, когда среднее ^{n} 1 запись. Значимое количество родительских узлов намного меньше, чем количество переменных. Это означает, что узлы связаны только локально.

Благодаря локальному соединению сеть становится модульной, что, как и в программной инженерии, приводит к снижению сложности. В примере с тревогой узел тревоги отделяет узлы Bur и Ear от узлов J и M. Мы также ясно видим это в примере с LEXMED .

15Для узла без предков произведение в этой сумме пусто. Для этого мы подставляем значение 1, потому что CPT для узлов без предков содержит с его априорной вероятностью ровно одно значение.

Machine Translated by Google

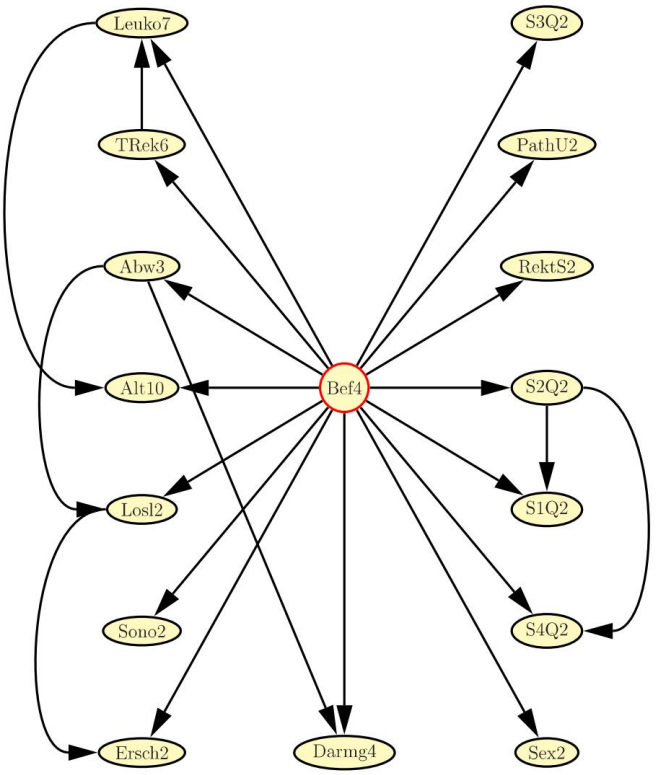


Рис. 7.14 Байесовская сеть для приложения LEXMED

LEXMED как байесовская сеть Система
LEXMED , описанная в разд. 7.3 также можно смоделировать как байесовскую сеть. Сделав внешние тонко нарисованные линии направленными (придав им стрелки), граф независимости на рис. 7.8 на стр. 137 можно интерпретировать как байесовскую сеть. Получившаяся сеть показана на рис. 7.14.

В разд. 7.3.2 размер дистрибутива для LEXMED рассчитывался как значение 20 643 839. С другой стороны, байесовская сеть может быть полностью описана всего 521 значением. Это значение можно определить, введя переменные с рис. 7.14 в (7.22) на стр. 151. В порядке (Лейко, ТРек, Гуа, Возраст, Реб, Соно, Тапп, БоуС, Пол, П4К, П1К, П2К, РекП , Урин, Р3Q, Diag4) вычисляем

$$|\text{Эйдж}| = \sum_{\text{я}=1}^{\text{н}} (|\text{ви}| - 1) \prod_{j=1}^{\text{ки}} = 6 \cdot 6 \cdot 4 + 5 \cdot 4 + 2 \cdot 4 + 9 \cdot 7 \cdot 4 + 1 \cdot 3 \cdot 4 + 1 \cdot 4 + 1 \cdot 2 \cdot 4$$
$$+ 3 \cdot 3 \cdot 4 + 1 \cdot 4 + 1 \cdot 4 \cdot 2 + 1 \cdot 4 \cdot 2 + 1 \cdot 4 + 1 \cdot 4 + 1 \cdot 4$$
$$+ 1 \cdot 4 + 1 = 521.$$

Machine Translated by Google

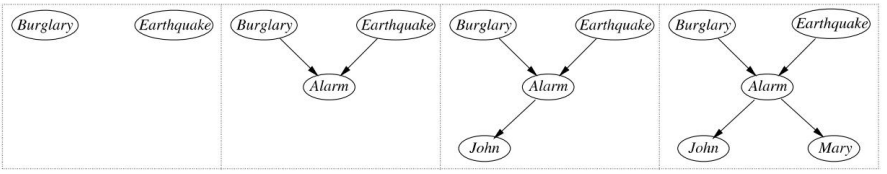


Рис. 7.15 Пошаговое построение сети сигнализации с учетом причинно-следственной связи

Этот пример демонстрирует, что практически невозможно создать полноценный дистрибутив для реальных приложений. С другой стороны, байесовская сеть с 22 ребрами и 521 значением вероятности по-прежнему управляема.

Причинность и структура сети Построение
байесовской сети обычно происходит в два этапа.

1. Проектирование сетевой структуры. Этот этап обычно выполняется вручную и будет описан ниже.
2. Ввод вероятностей в CPT: вручную вводить значения в случае многих переменных очень утомительно. Если (как, например, в случае с LEXMED) доступна база данных, этот шаг можно автоматизировать путем оценки записей CPT путем подсчета частот.

Теперь опишем построение сети на примере тревоги (см. рис. 7.15). В начале мы знаем две причины кражи со взломом и землетрясение и два симптома Джон и Мэри. Однако, поскольку Джон и Мэри не реагируют напрямую на грабителя или землетрясение, а только на тревогу, уместно добавить это как дополнительную переменную, которую Боб не может наблюдать. Процесс добавления ребер начинается с причин, то есть с переменных, не имеющих родительских узлов. Сначала выбираем «Кража со взломом» , а затем «Землетрясение». Теперь мы должны проверить, не зависит ли Землетрясение от Взлома. Это дано, и, таким образом, преимущество от Взлома до Землетрясения не добавляется . Поскольку Тревога напрямую зависит от Взлома и Землетрясения, эти переменные выбираются следующими, и к Тревоге добавляется ребро как от Взлома , так и от Землетрясения . Затем мы выбираем Джона. Поскольку Тревога и Джон не являются независимыми, к Джону добавляется ребро от тревоги. То же верно и для Марии. Теперь мы должны проверить, является ли Джон условно независимым от Взлома с учетом Тревоги. Если это не так, то нужно вставить еще одно ребро от Взлома до Джона. Мы также должны проверить, нужны ли ребра от Землетрясения до Джона и от Взлома или Землетрясения до Мэри. Из-за условной независимости эти четыре ребра не нужны. Ребра между Джоном и Мэри также не нужны, потому что Джон и Мэри условно независимы с учетом Тревоги.

Структура байесовской сети сильно зависит от выбранного порядка переменных. Если порядок переменных выбран так, чтобы отражать причинно-следственную связь, начиная с причин и заканчивая диагностическими переменными, то результатом будет простая сеть. В противном случае сеть может содержать значительно больше ребер.
Такие некаузальные сети часто очень сложны для понимания и имеют более высокую

Machine Translated by Google

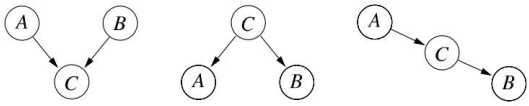


Рис. 7.16 Между A и B нет ребра, если они независимы (слева) или условно независимы (в середине, справа)

сложность для рассуждений. Читатель может обратиться к упражнению 7.11 на стр. 159 для лучшего понимания.

7.4.7 Семантика байесовских сетей

Как мы видели в предыдущем разделе, в байесовскую сеть не добавляется ребро между двумя переменными A и B, когда A и B независимы или условно независимы при заданной третьей переменной C. Эта ситуация представлена на рис. 7.16 .

Теперь мы требуем, чтобы в байесовской сети не было циклов, и мы предполагаем, что переменные пронумерованы так, что ни одна переменная не имеет меньшего номера, чем любая предшествующая ей переменная. Это всегда возможно, когда сеть не имеет циклов16. Тогда при использовании всех условных зависимостей имеем

$$P(X_n | X_1,\ldots,X_{n-1}) = P(X_n | \text{Родители}(X_n)).$$

Таким образом, это уравнение является предположением о том, что произвольная переменная X_i в байесовской сети условно независима от своих предков при заданных родителях. Несколько более общее предложение, изображенное на рис. 7.17 на стр. 155 , можно сформулировать компактно. как

> Теорема 7.6. Узел в байесовской сети условно независим от всех узлов, не являющихся преемниками, при заданных его родителях.

Теперь мы можем значительно упростить цепное правило ((7.1) на стр. 120):

$$P(X_1,\ldots,X_n) = \prod_{i=1}^{n} P(X_i | X_1,\ldots,X_{i-1}) = \prod_{i=1}^{n} P(X_i | \text{Родители}(X_i)). \quad (7.23)$$

Используя это правило, мы могли бы, например, написать (7.20) прямо на странице 148 .

$$P(J,Bur,Al) = P(J | Al)P(Al | Bur)P(Bur).$$

16Если, например, три узла X1, X2, X3 образуют цикл, то существуют ребра (X1, X2), (X2, X3) и (X3, X1) , где X1 имеет X3 в качестве преемника.

Machine Translated by Google

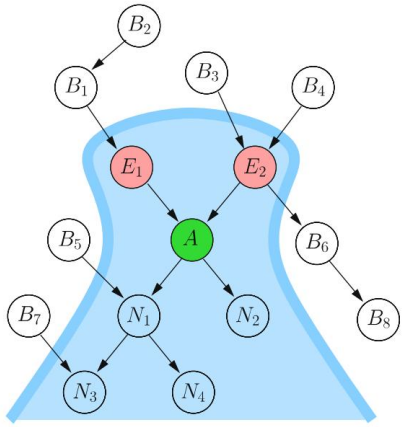


Рис. 7.17 Пример условной независимости в байесовской сети. Если заданы родительские узлы E1 и E2 , то все узлы, не являющиеся преемниками B1,...,B8, не зависят от A.

Теперь мы знаем самые важные концепции и основы байесовской сети. работает. Подытожим их [Jen01]:

Определение 7.7 Байесовская сеть определяется: • Набором переменных и набором направленных ребер между этими переменными. • Каждая переменная имеет конечное число возможных значений. • Переменные вместе с ребрами образуют направленный ациклический граф (DAG).

DAG — это граф без циклов, то есть без путей вида (A,... , A).

• Для каждой переменной A CPT (т. е. таблица условных вероятностей P(A |Родители(A))) дано.

Две переменные A и B называются условно независимыми при заданном C, если P(A, B |C) = P(A |C) · P(B |C) или, что то же самое, если P(A |B,C) = P(A |C). Помимо основных правил вычисления вероятностей, также верны следующие правила:

Теорема Байеса: $P(A|B) = \frac{P(B|A)\cdot P(A)}{P(B)}$

Маргинализация: P (B) = P (A, B) + P (¬A, B) = P (B |A) · P (A) + P (B |¬A) · P (¬A)

Кондиционирование: $P(A|B) = \sum_c P(A|B,C=c)P(C=c|B)$

Переменная в байесовской сети условно независима от всех переменных, не являющихся последующими, с учетом ее родительских переменных. Если X1,...,Xn 1 не являются потомками Xn, то P(Xn |X1,...,Xn 1) = P(Xn | Parents(Xn)). Это условие должно соблюдаться при построении сети.

При построении байесовской сети переменные должны быть упорядочены в соответствии с причинно-следственными связями. Сначала причины, затем скрытые переменные и, наконец, переменные диагностики.

Цепное правило: $P(X_1,...,X_n) = \prod_{i=1}^{n} P(X_i \mid \text{Родители}(X_i))$

Machine Translated by Google

В [Pea88] и [Jen01] для байесовских сетей введен термин d-разделение, из которого следует теорема, аналогичная теореме 7.6 на стр. 154 . Мы воздержимся от введения этого термина и тем самым придем к несколько более простому, хотя и не столь чистому в теоретическом отношении представлению.

7.5 Резюме

Путем, отражающим длительную, устойчивую тенденцию к вероятностным системам, мы ввели вероятностную логику для рассуждений с неопределенными знаниями. После знакомства с языком — пропозициональной логикой, дополненной вероятностями или вероятностными интервалами — мы выбрали естественный, хотя и необычный подход с использованием метода максимальной энтропии в качестве точки входа и показали, как мы можем моделировать немонотонные рассуждения с помощью этого метода. Затем были введены байесовские сети как частный случай метода MaxEnt.

Почему байесовские сети являются частным случаем MaxEnt? При построении байесовской сети делаются предположения о независимости, которые для метода MaxEnt не нужны. Кроме того, при построении байесовской сети все CPT должны быть полностью заполнены, чтобы можно было построить полное распределение вероятностей.
В противном случае рассуждение ограничено или невозможно. С другой стороны, с MaxEnt разработчик может сформулировать все имеющиеся в его распоряжении знания в виде вероятностей. Затем MaxEnt завершает модель и генерирует распределение. Даже если (например, при опросе эксперта) доступны только расплывчатые предложения, это можно соответствующим образом смоделировать. Утверждение типа «Я почти уверен, что A истинно». например, можно смоделировать, используя P (A) [0,6, 1] в качестве интервала вероятности. При построении байесовской сети необходимо указать конкретное значение вероятности, при необходимости путем угадывания. Однако это означает, что эксперт или разработчик вводят в систему специальную информацию. Еще одним преимуществом MaxEnt является возможность формулировать (почти) произвольные предложения. Для байесовских сетей CPT должны быть заполнены.

Свобода, которой обладает разработчик при моделировании с помощью MaxEnt, может быть недостатком (особенно для новичка), поскольку, в отличие от байесовского подхода, не обязательно ясно, какие знания следует моделировать. При моделировании байесовскими сетями подход достаточно ясен: по причинно-следственным зависимостям, от причин к следствиям, одно ребро за другим вводится в сеть путем проверки условной независимости17. В конце все КПН заполняются значениями .

Однако возможны следующие интересные комбинации двух методов: мы начинаем с построения сети по байесовской методологии, вводим соответственно все ребра и затем заполняем КПТ значениями. Если определенные значения для CPT недоступны, то они могут быть заменены интервалами или другими формулами вероятностной логики. Естественно, такая сеть — или, лучше сказать, набор правил — не

17Это тоже не всегда так просто.

Machine Translated by Google

имеет особую семантику байесовской сети. Затем он должен быть обработан и завершен системой MaxEnt.

Однако возможность использовать MaxEnt с произвольными наборами правил имеет и обратную сторону. Подобно ситуации в логике, такие наборы правил могут быть противоречивыми. Например, два правила P (A) = 0,7 и P (A) = 0,8 несовместимы. В то время как система MaxEnt PIT, например, может распознать несоответствие, если не может дать подсказку о том, как устранить проблему.

Мы представили медицинскую экспертную систему LEXMED, классическое приложение для рассуждений с неопределенными знаниями, и показали, как ее можно смоделировать и реализовать с помощью MaxEnt и байесовских сетей, и как эти инструменты могут заменить хорошо зарекомендовавший себя, но слишком слабый линейный скоринг. системы, применяемые в медицине.[18]

На примере LEXMED мы показали, что можно построить экспертную систему для рассуждений в условиях неопределенности, которая способна обнаруживать (извлекать) знания из данных в базе данных. Мы подробнее рассмотрим методы обучения байесовских сетей в гл. 8, после того как были заложены необходимые основы для машинного обучения.

Сегодня байесовские рассуждения представляют собой большую независимую область, которую мы можем здесь описать лишь вкратце. Мы полностью отказались от обработки непрерывных переменных. Для случая нормально распределенных случайных величин существуют процедуры и системы. Однако для произвольных распределений вычислительная сложность представляет собой большую проблему. В дополнение к направленным сетям, которые в значительной степени основаны на причинно-следственных связях, существуют также ненаправленные сети. С этим связана дискуссия о значении и полезности причинности в байесовских сетях. Заинтересованному читателю предлагается ознакомиться с отличными учебниками, такими как [Pea88, Jen01, Whi96, DHS01], а также с материалами ежегодной конференции Ассоциации по изучению неопределенностей в искусственном интеллекте (AUAI) (www.auai.org).

7.6 Упражнения

Упражнение 7.1. Докажите утверждение из теоремы 7.1 на стр. 117.

Упражнение 7.2 Садовод Макс хочет статистически проанализировать свой ежегодный урожай гороха. Для каждого сорванного стручка гороха он измеряет его длину xi в сантиметрах и вес yi в граммах. Он делит горох на два класса: хороший и плохой (пустые стручки). Измеренные данные (xi, yi)

хороший горох:	x 122334456 y 234455666	плохой горох:	x 4667 г 2233

18В разд. 8.6 и в упражнении 8.16 на стр. 219 мы покажем, что оценки эквивалентны наивному байесовскому методу для особого случая, то есть предположению, что все симптомы условно независимы при данном диагнозе.

Machine Translated by Google

(а) По данным вычислите вероятности P (y > 3 |Класс = хорошо) и
P (y 3 | Класс = хорошо). Затем используйте формулу Байеса, чтобы определить P (класс =
хорошо | y > 3) и P (класс = хорошо | y 3).

(б) Какая из вероятностей, вычисленных в подзадаче (а), противоречит утверждению
«Все хорошие горошины тяжелее 3 граммов»?

Упражнение 7.3. Вы должны предсказать дневную погоду, используя несколько простых
значения погоды с утра этого дня. Классический расчет вероятности
для этого требуется полная модель, которая приведена в следующей таблице.

Небо	Бар	Prec	P (небо, бар, Prec)
Прозрачный	Поднимается		0,40
Прозрачный	Сухой	Поднимается	Дождь 0,07
Прозрачный	Падение	насухо	0,08
Прозрачный	Падающий	дождь	0.10
Bewö lkt	Rising	Dry	Bewö lkt 0,09
Rising	Raining	0.11	
Bewö lkt	Falling	Dry	0.03

Небо: небо чистое или
облачно утром
Бар: барометр поднимается или
падение утром
Prec: Дождь или сухо в
полдень

а) Сколько событий в распределении этих трех переменных?

(b) Вычислите P (Prec = сухое | небо = чистое, Bar = поднимающееся).

(c) Вычислите P (Prec = дождь |Sky = облачно).

г) Что бы вы сделали, если бы в таблице не было последней строки?

Упражнение 7.4 В телевизионной викторине участник должен выбрать между тремя
закрытые двери. За одной дверью ждет приз: автомобиль. За обоими другими
двери козлы. Участник выбирает дверь, например номер один. Хозяин, который
знает, где машина, открывает другую дверь, например, номер три, и появляется коза.
Теперь участнику предоставляется возможность выбрать между двумя оставшимися
двери (одна и две). Что является лучшим выбором с его точки зрения? остаться с
дверь, которую он изначально выбрал, или переключиться на другую закрытую дверь?

Упражнение 7.5 Используя метод множителей Лагранжа, покажите, что без явного
ограничений, равномерное распределение p1 = p2 =···= pn = 1/n представляет максимальную энтропию.
Не забывайте о постоянно присутствующем неявном ограничении p1 + p2 +···+
pn = 1. Как мы можем показать тот же результат, используя безразличие?

Упражнение 7.6. Используйте систему PIT (www.pit-systems.de) для расчета MaxEnt.
решение для P (B) при ограничении P (A) = α и P (B |A) = β. Какой недостаток PIT по сравнению с ручным
расчетом вы здесь заметили?

Упражнение 7.7. Учитывая ограничения P (A) = α и P (A B) = β, вручную вычислите P (B), используя
метод MaxEnt. Используйте PIT, чтобы проверить свое решение.

Machine Translated by Google

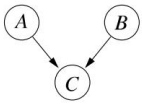
Упражнение 7.8. Учитывая ограничения из (7.10), (7.11), (7.12): p1 + p2 = α, p1 + p3 = γ , p1 + p2 + p3 + p4 = 1. Покажите, что p1 = αγ, p2 = α (1 γ), p3 = γ (1 α), p4 = (1 α)(1 γ) представляет собой максимум энтропии при этих ограничениях.

Упражнение 7.9 Вероятностный алгоритм вычисляет вероятность p того, что входящее электронное письмо является спамом. Чтобы классифицировать электронные письма по классам удаления и
чтения, к результату применяется матрица затрат. (a) Приведите матрицу затрат (матрица 2 × 2) для спам-фильтра. Предположим, что удаление сообщения электронной почты стоит пользователю 10 центов, а потеря сообщения электронной почты стоит 10
долларов (сравните это с примером 1.1 на странице 11 и упражнением 1.7 на странице 14). (b) Показать, что в случае матрицы 2 × 2 применение матрицы затрат эквивалентно применению порога вероятности спама, и определить порог.

Упражнение 7.10. Дана байесовская сеть с тремя бинарными переменными A, B, C и P (A) = 0,2, P (B) = 0,9, а также CPT, показанный ниже:

(a) Вычислите P (A|B). (b) Вычислите P (C|A).

АД (K) wf 0,1	
ww 0,2	
вперед 0,9	
вперед 0,4	



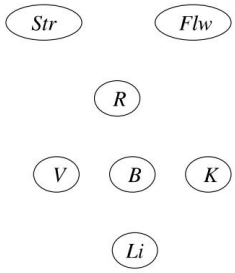
Упражнение 7.11 Для примера тревоги (Пример 7.7 на стр. 145) рассчитайте следующие условные вероятности: (a) Рассчитайте
априорные вероятности P (Al), P (J), P (M). (b) Рассчитайте P (M|Bur),
используя правило произведения, маргинализацию, цепное правило и условная независимость. (c)
Используйте формулу Байеса для расчета P (Bur|M). (d)
Вычислите P (Al||J,M) и P (Bur||J,M). (e) Покажите, что
переменные J и M не являются независимыми. (f) Проверьте все
свои результаты с помощью JavaBayes и PIT (см. [Ert11] для демонстрации программы).
(g) Разработайте байесовскую сеть для примера тревоги, но с измененным порядком переменных M, Al, Ear, Bur, J . Согласно семантике байесовских сетей, должны быть нарисованы только необходимые ребра. (Подсказка: приведенный здесь порядок переменных НЕ отражает причинно-следственную связь. Таким образом, будет трудно интуитивно определить условную независимость.)
(h) В исходной байесовской сети примера тревоги узлы землетрясения удалены. Какие CPT это меняет? (Почему именно эти?)
(i) Рассчитайте CPT аварийного узла в новой сети.

Упражнение 7.12 Необходимо создать диагностическую систему для велосипедного фонаря с динамо-машиной, используя байесовскую сеть. Переменные в следующей таблице даны.

Machine Translated by Google

Сокр.	Это означает, что	Ценности
Ли	Лайт находится в	т/ф
ул.	уличном состоянии.	сухой, мокрый, заснеженный
Износ маховика Flw Dynamo т/ф		
р	Динамо скользящее т/ф	
В	Динамо показывает напряжение t/f	
Б	Лампочка в норме т/ф	
К	Кабель в порядке т/ф	

Следующие переменные попарно независимы:
Улица, Полет, Б, К. Далее: (R, B), (R, K), (V , B),
(V , K) независимы, и следующее уравнение
держит:



$$P(Li \mid V, R) = P(Li \mid V)$$

$$P(V \mid R, Str) = P(V \mid R)$$

$$P(V \mid R, Flw) = P(V \mid R)$$

(a) Нарисуйте все ребра в графе (принимая во внимание причинность).

(b) Внесите все недостающие CPT в график (таблицу условных вероятностей). Свободно вставляйте правдоподобные значения вероятностей.

в) Показать, что сеть не содержит ребер (Стр, Ли).

(d) Вычислите P (V |Str = snowcovered).

ВБК П(Ли)
ттт 0,99
ттф 0,01
т фут 0,01
т фф 0,001
фут т 0,3
фут 0,005
фт 0,005
0

Machine Translated by Google

8 Машинное обучение и интеллектуальный анализ данных

Если мы определим ИИ, как это сделано в книге Элейн Рич [Ric83]:

> Искусственный интеллект — это наука о том, как заставить компьютеры делать то, что в момент, люди лучше.

а если учесть, что обучаемость компьютера особенно уступает этому человека, то следует, что исследование механизмов обучения и разработка алгоритмов машинного обучения является одним из важнейших направлений ИИ.

Существует также спрос на машинное обучение с точки зрения программного обеспечения. разработчик, который программирует, например, поведение автономного робота. структура разумного поведения может стать настолько сложной, что очень трудно или даже невозможно запрограммировать близко к оптимальному, даже с современными языками высокого уровня, такими как PROLOG и Python.1 Даже используются алгоритмы машинного обучения сегодня программировать роботов так же, как учатся люди (см. главу 10 или [BCDS08, RGH+06]), часто в гибридной смеси запрограммированного и усвоенного поведение.

Задача этой главы — описать наиболее важные алгоритмы машинного обучения и их приложения. Тема будет представлена в этом разделе, а затем важными фундаментальными алгоритмами обучения в следующих разделах. Параллельно с этим будут выстраиваться теория и терминология. Глава завершится подведением итогов и обзор различных алгоритмов и их приложений. В этой главе мы ограничимся рассмотрением алгоритмов обучения с учителем и без учителя. Нейронные сети, как важный класс алгоритмов обучения, будут рассмотрены в гл. 9.
Благодаря своему особому месту и важной роли для автономных роботов арматура обучению также будет посвящена отдельная глава (глава 10).

Что такое обучение? Изучение лексики иностранного языка, или технического термины, или даже запоминание стихотворения может быть трудным для многих людей. Для компьютеров,

1Python — это современный язык сценариев с очень читаемым синтаксисом, мощными типами данных и обширными стандартными библиотеками, которые можно использовать для этой цели.

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по информатике,
DOI 10.1007/978-0-85729-299-5_8, © Springer-Verlag London Limited 2011

Machine Translated by Google

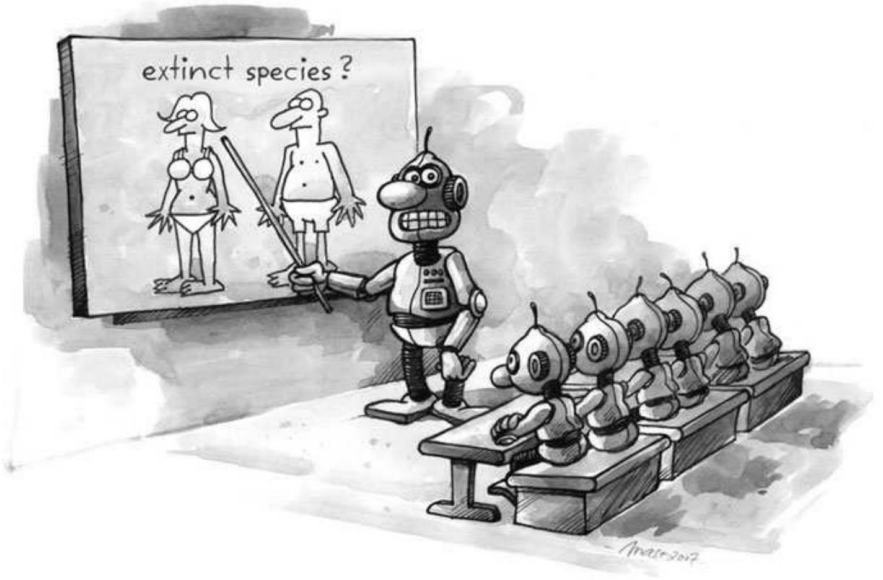


Рис. 8.1 Обучение с учителем. . .

однако эти задачи довольно просты, потому что они не более чем сохранение текста в файле. Таким образом, запоминание неинтересно для ИИ. Напротив, приобретение математических навыков обычно не происходит путем запоминания. Для сложения натуральных чисел это вообще невозможно, так как для каждого из слагаемых в сумме x + y имеется бесконечно много значений. Для каждой комбинации двух значений x и y пришлось бы хранить тройку (x, y, x + y), что невозможно. Для десятичных чисел это совершенно невозможно. Это ставит вопрос: как мы изучаем математику? Ответ звучит так: Учитель объясняет процесс, а ученики практикуют его на примерах до тех пор, пока не перестанут делать ошибки на новых примерах. После 50 примеров ученик (надеюсь) понимает сложение. То есть всего после 50 примеров он может применить выученное к бесконечно большому количеству новых примеров, которых до этого момента не видел. Этот процесс известен как генерализация. Начнем с простого примера.

Пример 8.1. Фермер, выращивающий фрукты, хочет автоматически разделить собранные яблоки на товарные классы A и B. Сортировочное устройство оснащено датчиками для измерения двух характеристик, размера и цвета, а затем решает, к какому из двух классов относится яблоко. Это типичная задача классификации. Системы, способные разбивать векторы признаков на конечное число классов, называются классификаторами.

Для настройки машины яблоки собираются специалистом вручную, то есть классифицируются. Затем два измерения вносятся вместе с меткой их класса в таблицу (Таблица 8.1 на стр. 163). Размер указывается в виде диаметра в сантиметрах, а цвет — в виде числового значения от 0 (для зеленого) до 1 (для красного).

Machine Translated by Google

Рассуждение с неуверенностью

7

Мы уже показывали в разд. 4 с проблемой Твити, что двухзначная логика приводит к проблемам в повседневных рассуждениях. В этом примере утверждения Твити — пингвин, Пингвины — птицы и Все птицы могут летать приводят к (семантически неверному) выводу, что Твити может летать. Теория вероятностей предоставляет язык, на котором мы можем формализовать утверждение. Почти все птицы умеют летать и делать на его основе выводы. Теория вероятностей — проверенный метод, который мы можем здесь использовать, потому что неопределенность в отношении того, могут ли птицы летать, можно хорошо смоделировать с помощью назначения вероятности. Мы покажем, что такие утверждения, как 99% всех птиц умеют летать, вместе с вероятностной логикой приводят к правильным выводам.

Рассуждения в условиях неопределенности с ограниченными ресурсами играют большую роль в повседневных ситуациях, а также во многих технических приложениях ИИ. В этих областях очень важны эвристические процессы, как мы уже обсуждали в гл. 6. Например, мы используем эвристические приемы при поиске парковочного места в городском потоке. Одних только эвристик часто бывает недостаточно, особенно когда требуется быстрое решение при неполном знании, как показано в следующем примере. Пешеход переходит улицу, и быстро приближается автомобиль. Чтобы предотвратить серьезную аварию, пешеход должен реагировать быстро. Он не способен заботиться о полной информации о состоянии мира, которая понадобилась бы ему для алгоритмов поиска, обсуждавшихся в гл. 6. Следовательно, он должен прийти к оптимальному решению при заданных ограничениях (мало времени и мало потенциально неопределенных знаний). Если он будет думать слишком долго, это будет опасно. В этой и многих подобных ситуациях (см. рис. 7.1 на стр. 114) необходим метод рассуждений при неопределенных и неполных знаниях.

Мы хотим исследовать различные возможности рассуждений в условиях неопределенности на примере простого медицинского диагноза. Если пациент испытывает боль в правой нижней части живота и повышенное количество лейкоцитов (лейкоцитов), это вызывает подозрение на аппендицит. Мы моделируем эту связь, используя логику высказываний по формуле

Боль в животе справа внизу Лейкоциты> 10000 Аппендицит

В. Эртель, Введение в искусственный интеллект, Темы бакалавриата по компьютерным наукам, DOI 10.1007/978-0-85729-299-5_7, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google



Рис. 7.1 «Давайте просто сядем и подумаем, что делать!»

Если мы тогда знаем, что

Боль в животе справа внизу Лейкоциты> 10000

верно, то мы можем использовать modus ponens для вывода аппендицита. Эта модель явно слишком грубая. В 1976 году Шортлифф и Бьюкенен признали это при создании своей медицинской экспертной системы MYCIN [Sho76]. Они разработали исчисление с использованием так называемых факторов достоверности, которые позволили представить достоверность фактов и правил.

Правилу A B присваивается коэффициент достоверности β. Семантика правила A β B определяется через условную вероятность P (B|A) = β. В приведенном выше примере правило может читаться

Боль в животе справа внизу Лейкоциты> 10000 0,6 Аппендицит.

Для рассуждений с такого рода формулами они использовали исчисление для связи факторов правил. Однако оказалось, что с помощью этого исчисления можно получить противоречивые результаты

Machine Translated by Google

Как обсуждалось в гл. 4 также были попытки решить эту проблему с помощью немонотонной логики и логики по умолчанию, которые, однако, в итоге не увенчались успехом. Теория Демпстера-Шефера приписывает функцию доверия Bel(A) логическому утверждению A, значение которого дает степень доказательства истинности A. Но даже этот формализм имеет недостатки, которые показаны в [Pea88] с использованием варианта Пример Твити. Даже нечеткая логика, которая прежде всего успешна в теории управления, демонстрирует значительные недостатки при рассуждениях в условиях неопределенности в более сложных приложениях [Elk93].

Примерно с середины 1980-х годов теория вероятности оказывала все большее влияние на ИИ [Pea88, Che85, Whi96, Jen01]. В области рассуждений с помощью байесовских сетей или субъективной вероятности он занял прочное место среди успешных методов ИИ. Вместо импликации, как это известно в логике (материальной импликации), здесь используется условная вероятность, которая значительно лучше моделирует повседневные причинные рассуждения. Рассуждения с вероятностями сильно выигрывают от того факта, что теория вероятностей — это хорошо зарекомендовавшая себя ветвь математики, существующая сотни лет.

В этой главе мы выберем элегантную, но несколько необычную для учебника точку входа в эту область. После краткого введения в наиболее важные основы, необходимые здесь для рассуждений с вероятностью, мы начнем с простого, но важного примера для рассуждений с неопределенными и неполными знаниями. Совершенно естественным, почти убедительным образом мы придем к методу максимальной энтропии (MaxEnt). Затем покажем полезность этого метода на практике с помощью медицинской экспертной системы LEXMED. Наконец, мы познакомимся с широко распространенными в настоящее время рассуждениями с помощью байесовских сетей и покажем взаимосвязь между этими двумя методами.

7.1 Вычисления с вероятностями

Читатель, знакомый с теорией вероятностей, может пропустить этот раздел. Всем остальным мы адресуем быстрый ввод в курс дела и порекомендуем несколько подходящих учебников, таких как [Ros09, FPP07].

Вероятность особенно хорошо подходит для моделирования рассуждений в условиях неопределенности. Одна из причин этого заключается в том, что вероятности интуитивно легко интерпретировать, что можно увидеть на следующем элементарном примере.

Пример 7.1. Для одного броска игрового кубика (эксперимент) вероятность события «выпадение шестерки» равна 1/6, а вероятность события «выпадение нечетного числа» равна 1/2.

Определение 7.1. Пусть Ω — конечное множество событий для эксперимента. Каждое событие $\omega \subseteq \Omega$ представляет собой возможный исход броска. Если эти события $\omega_i \subseteq \Omega$ взаимно исключают друг друга, но охватывают все возможные исходы попытки, то они называются элементарными событиями.

Machine Translated by Google

Пример 7.2. Для одного броска одного игрового кубика

$$Ом = \{1, 2, 3, 4, 5, 6\}$$

потому что никакие два из этих событий не могут произойти одновременно. Роллинг четного числа ({2, 4, 6}), следовательно, является элементарным событием, и при этом не выбрасывается число, меньшее пяти ({1, 2, 3, 4}), потому что {2, 4, 6} {1, 2, 3, 4} = {2, 4} = .

Учитывая два события A и B, A B также является событием. Сам Ω обозначается определенным событие, а пустое множество невозможное событие.

В дальнейшем мы будем использовать нотацию пропозициональной логики для операций над множествами. То есть для множества A B мы пишем A B. Это не только синтаксическая трансформация, скорее, это также семантически правильно, потому что пересечение двух множеств определено как

$$x \quad A \quad B \quad x \quad A \quad x \quad B.$$

Поскольку это семантика A B, мы можем и будем использовать это обозначение. Это тоже истинно для других операций объединения и дополнения множества, и мы будем, как показано в В следующей таблице используйте для них также нотацию пропозициональной логики.

Установить обозначение	Пропозициональная логика	Описание
A B	A B	Пересечение/и
A B	A Б	Союз/или
A^{-}	¬A	Дополнение/отрицание
Ом	ж	Определенное событие / правда
	ф	Невозможное событие/ложь

Используемые здесь переменные (например, A, B и т. д.) называются случайными величинами. в теории вероятностей. Здесь мы будем использовать только дискретные случайные переменные с конечными доменами. Переменная face_number для броска костей является дискретной со значениями 1, 2, 3, 4, 5, 6. Вероятность выпадения пятерки или шестерки равна 1/3. Это может быть описано

$$P(\text{число_лица} \quad \{5, 6\}) = P(\text{число_лица} = 5 \quad \text{число_лица} = 6) = 1/3.$$

Предполагается, что понятие вероятности дает нам описание столь же объективное, как нашей «веры» или «убеждения» в отношении результата эксперимента. Все числа в интервале [0, 1] должны быть возможны, где 0 — вероятность невозможное событие и 1 вероятность определенного события. Мы пришли к этому из следующее определение.

Machine Translated by Google

Опред еление 7.2. Пуст ь Ω = {ω1, ω2,...,ωn} к онеч но. Пред поч т ит ель ного элемент арного соб ыт ия не сущест вует , а эт о оз нач ает , ч т о мы пред полаг аем симмет рию, свя з анную с ч аст от ой поя вления к аждого элемент арного соб ыт ия . Т огда вероя т ност ь P (A) соб ыт ия A равна

$$P(A) = \frac{|A|}{|\Omega|} \text{ "=" } \frac{\text{Количество благоприятных случаев для } A}{\text{Количество возможных случаев}}.$$

От сюда немед ленно след ует , ч т о к аждое элемент арное соб ыт ие имеет вероя т ност ь $1/|\Omega|$. Т реб ование равновероя т ност и элемент арных соб ыт ий наз ывается пред положением Лапласа , а выч исля емые т ак им образом вероя т ност и наз ываются вероя т ност я ми Лапласа.

Эт о опред еление дост игает своего пред ела, к огда ч исло элемент арных соб ыт ий ст ановится беск онеч ным. Поск оль к у з дес ь мы рассмат риваем т оль к о к онеч ные прост ранст ва соб ыт ий, эт о не пред ст авля ет проб лемы Для описания соб ыт ий мы испол з уем переменные с соот вет ст вующим к олич ест вом з нач ений. Например, переменная eye_color может принимать з нач ения з еленый, синий, к орич невый. eye_color = blue з ат ем описывает соб ыт ие, пот ому ч т о мы имеем дело с пред ложением со з нач ениями ист ин Для ст ои ин ых (булевых) переменных сама переменная уже я вля ет ся пред ложением. Здес ь дост ат оч но, например, написать P (JohnCalls) вмест о P (JohnCalls = t).

Пример 7.3 Сог ласно эт ому опред елению вероя т ност ь выпадения ч ет ного ч исла равна

$$P(\text{номер_лица} \in \{2, 4, 6\}) = \frac{|\{2, 4, 6\}|}{|\{1, 2, 3, 4, 5, 6\}|} \text{ "=" } \frac{3}{6} \text{ "=" } \frac{1}{2}.$$

След ующие важные правила след ут непосред ст венно из опред еления .

Т еорема 7.1 1.

P (Ω) = 1.

2. P( ) = 0, з нач ит , вероя т ност ь невоз можного соб ыт ия равна 0.

3. Для попарно иск лю ч ающих соб ыт ий A и B верно, ч т о P (A B) = P (A) + П (Б).

4. Для дв ух дополнит ель ных соб ыт ий A и ¬A верно, ч т о P (A) + P (¬A) = 1.

5. Для прои з воль ных соб ыт ий A и B верно, ч т о P (A B) = P (A) + P (B) P (A B).

6. Для A B верно, ч т о P (A) P (B).

7. Е сли A1,...,An — элемент арные соб ыт ия , т о $\sum_{i=1}^{n} P(A_i) = 1$ (нормировк а условие).

Machine Translated by Google

Выражение P (A B) или, что то же самое, P (A, B) обозначает вероятность событий A B. Нас часто интересуют вероятности всех элементарных событий, то есть всех комбинаций всех значений переменных A и B. Для бинарные переменные A и B это P (A, B), P (A, ¬B), P (¬A, B), P (¬A, ¬B). Мы называем вектор

$$(\Pi(A, B), \Pi(A, \neg B), \Pi(\neg A, B), \Pi(\neg A, \neg B))$$

состоящее из этих четырех значений распределение или совместное распределение вероятностей переменные A и B. Сокращением для этого является P (A, B). Распределение в случае две переменные можно красиво представить в виде таблицы(матрицы), представленной в виде следует :

P(A, B) B = w		B = ϕ
A = ш	П (A, Б)	P (A, ¬B)
A = ϕ	P(¬A,B) P (¬A,¬B)	

Для d переменных X1,...,Xd с n значениями каждая распределение имеет значения P (X1 = x1,...,Xd = xd) и x1,...,xd , каждое из которых принимает n различных значений. Таким образом, распределение может быть представлено в виде d-мерной матрицы с общим n Г элементы В силу условия нормировки из теоремы7.1 на стр. 117, однако один из этих n Г значения избыточны а распределение характеризуется по н Г 1 уникальное значение.

7.1.1 Условная вероятность

Пример 7.4. На улице Ландсдаун в Бостоне измеряется скорость 100 автомобилей. Для каждого из измерения также отмечается , является ли водитель студентом. результаты

Событие	Частота	Относительная частота
Наблюдаемый автомобиль	100	1
Водитель - студент (C)	30	0,3
Слишком высокая скорость (G)	10	0,1
Водитель студент и превышение скорости (S G)	5	0,05

Мы ставим вопрос: учащиеся превышают скорость чаще, чем в среднем по сын, или чем не студенты?1

Ответ дает вероятность

$$P(Г|C) = \frac{|\text{Водитель-студент и превышает скорость}|}{|\text{Водитель – студент}|} "=" \frac{5}{30} "=" \frac{1}{6} \quad 0{,}17$$

1 Вычисленные вероятности могут быть использованы для продолжающихся предложений только в том случае, если измеренная выборка (100 автомобилей) является репрезентативным. В противном случае могут быть только предположения о наблюдаемых 100 транспортных средствах. быть сделано.

Machine Translated by Google

за превышение скорости при условии, что водитель является студентом. Это явно отличается от априорной вероятности превышения скорости P(G) = 0,1. Для априорной вероятности пространство событий не ограничивается дополнительными условиями.

Определение 7.3. Для двух событий A и B вероятность P (A|B) для A при условии B (условная вероятность) определяется как

$$P(A|B) = \frac{П(A \quad B)}{П(Б)}.$$

В примере 7.4 мы видим, что в случае конечного пространства событий условная вероятность P (A|B) может пониматься как вероятность A B, когда мы рассматриваем только событие B, то есть как

$$P(A|B) = \frac{|A \quad B|}{|Б|}.$$

Эту формулу можно легко вывести, используя Определение 7.2 на стр. 117.

$$P(A|B) = \frac{П(A \quad B)}{П(Б)} \text{ "=" } \frac{\frac{|A \quad B| \ |}{Ом|}}{\frac{|Б| \ |}{Ом|}} \text{ "=" } \frac{|A \quad B|}{|Б|}.$$

Определение 7.4. Если для двух событий A и B

$$P(A|B) = P(A),$$

то эти события называются независимыми.

Таким образом, A и B независимы, если вероятность события A не зависит от события B.

Теорема 7.2. Для независимых событий A и B из определения следует, что

$$P(A \quad B) = P(A) \cdot P(B).$$

Machine Translated by Google

Пример 7.5. При броске двух костей вероятность выпадения двух шестерок равна 1/36, если две кости независимы, потому что

$$P(D1 = 6 \quad D2 = 6) = P(D1 = 6) \cdot P(D2 = 6) = \frac{1}{6} \cdot \frac{1}{6} \overset{"="}{=} \frac{1}{36},$$

где первое уравнение верно только тогда, когда две игральные кости независимы. Если, например, благодаря какой-то магической силе кубик 2 всегда совпадает с кубиком 1, то

$$P(D1 = 6 \quad D2 = 6) = \frac{1}{6}.$$

Правило цепи

Решение определения условной вероятности для $P(A \quad B)$ приводит к так называемому правилу произведения

$$P(A \quad B) = P(A|B)P(B),$$

которое мы сразу же обобщаем на случай n переменных. Повторным применением указанного выше правила получаем цепное правило

$$\begin{aligned} & P(X1,...,Xn) \\ & = P(Xn|X1,...,Xn \quad 1) \cdot P(X1,...,Xn \quad 1) \\ & = P(Xn|X1,...,Xn \quad 1) \cdot P(Xn \quad 1|X1,...,Xn \quad 2) \cdot P(X1,...,Xn \quad 2) \\ & = P(Xn|X1,...,Xn \quad 1) \cdot P(Xn \quad 1|X1,...,Xn \quad 2) \cdot \cdots \cdot P(X2|X1) \cdot P(X1) \\ & \overset{"="}{=} \prod_{i=1}^{n} P(Xi|X1 ...,Xi \quad 1), \end{aligned} \tag{7.1}$$

с помощью которого мы можем представить распределение как произведение условных вероятностей. Поскольку цепное правило выполняется для всех значений переменных X1,...,Xn, оно было сформулировано для распределения с использованием символа P.

Маргинализация

Поскольку $A \quad (A \quad B) \quad (A \quad \neg B)$ верно для двоичных переменных A и B

$$P(A) = P((A \quad B) \quad (A \quad \neg B)) = P(A \quad B) + P(A \quad \neg B).$$

Путем суммирования по двум значениям B переменная B исключается. Аналогично, для произвольных переменных X1,...,Xd переменная , например Xd , может быть исключена суммированием по всем их переменным, и мы получим

$$P(X1 = x1,...,Xd \quad 1 = xd \quad 1) = \sum_{xd} P(X1 = x1,...,Xd \quad 1 = xd \quad 1, Xd = xd).$$

Machine Translated by Google

Применение эт ой формулыназ ьвает ся маргинализ ац ией. Эт о суммирование может прод олжат ь ся с переменньми X1,...,Xd 1 д о т ех пор, пок а не ост анет ся т оль к о од на переменная . Маргинализ ац ия можно применит ь и к распред елениюP(X1,...,Xd). Получ енное распред еление P(X1,...,Xd 1) наз ьвает ся маргиналь ньм распред елением. Эт о сравнимо с проек ц ией пря моуг оль ного параллелепипеда на плоск уюповерхност ь . Зд есь т рехмерньй объек т рисует ся на ребре или «к рае» к убоида, т . е. на д вумерном множест ве. В обоих случ ая х раз мерност ь умень шает ся на ед иниц у.

Пример 7.6 Наблюд аем совок упност ь всех боль ньх, обрат ивших ся к врач у с ост рым боль в живот е. У к аждог о пац иент а из меря ет ся к олич ест во лейк оц ит ов, к от орое я вля ет ся пок аз ат елем от носит ель ное сод ержание лейк оц ит ов в к рови. Мыопред еля ем переменную Лейк о, ч т о верно т огд а и т оль к о т огд а, к огд а з нач ение лейк оц ит ов боль ше 10 000. Эт от свид ет ель ст вует об инфек ц ии в орг анизме. В прот ивном случ ае мыопред еля ем переменнуюApp, к от орая сообщает нам, ест ь ли у пац иент а аппенд иц ит , т о ест ь инфиц ированный аппенд ик с. Распред еление P(App,Leuko) эт их д вух переменньх пред ст авлено в след ующей т аблиц е:

P(App,Leuko)	App	¬App	Ит ог о
Леук о	0,23	0,31	0,54
¬Леук о	0,05	0,41	0,46
Общий	0,28	0,72	1

В послед ней ст рок е д ана сумма по ст рок ам, а в послед нем ст олбц е сумма д ают ся ст олбц ы Эт и суммыд ост иг ают ся з а сч ет маргинализ ац ии. Например, мы ч ит ат ь

$$P(\text{лейк о}) = P(\text{апп}, \text{лейк о}) + P(\neg\text{апп}, \text{лейк о}) = 0{,}54.$$

Данное распред еление P(App,Leuko) может быт ь получ ено из опроса немец к их врач ей, например. От сюд а мыможем вьн ислит ь условнуювероя т ност ь

$$P(\text{лейк о} \mid \text{приложение}) = \frac{P(\text{Леук о}, \text{приложение})}{П(\text{приложение})} = 0{,}82$$

ч т о говорит нам о т ом, ч т о ок оло 82% всех случ аев аппенд иц ит а приводя т к выс ок ому з нач ениолейк оц ит ов. Под обньe з нач ения публик уют ся в мед иц инск ой лит ерат уре. Од нак о условная вероя т ност ь P (App |Leuko), к от орая на самом д еле была быраз д о полез нее д ля д иаг ност ик и аппенд иц ит , не опублик овано. Чт обыпоня т ь эт о, мыснач ала вьвед ем прост ой, но оч ень важная формула.

Т еорема Байеса

Замена A и B в опред елении 7.3 на ст р. 119 привод ит к

$$P(A \mid B) = \frac{П(A \quad B)}{П(Б)} \quad \text{и} \quad P(B \mid A) = \frac{П(A \quad B)}{П(A)}.$$

Machine Translated by Google

Решая два уравнения для P (A B) и приравнивая их, мы получаем теорему Байеса

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}, \tag{7.2}$$

которые мы сразу применим к проблеме аппендицита и получим

$$P(\text{Приложение} \mid \text{Лейко}) = \frac{P(\text{лейко}|\text{приложение}) \cdot P(\text{приложение})}{P(\text{Лейко})} \text{ "=" } \frac{0,82 \cdot 0,28}{0,54} = 0,43. \tag{7.3}$$

Теперь почему публикуется P (Leuko|App) , а не P (App|Leuko)?

Предполагая, что аппендицит влияет на организм человека независимо от расы, P (Leuko|App) является универсальным значением, которое верно во всем мире. В (7.3) мы видим, что P (App| Leuko) не является универсальным, поскольку на это значение влияют априорные вероятности P (App) и P (Leuko). Каждый из них может варьироваться в зависимости от жизненных обстоятельств. Например, P (Leuko) зависит от того, много или мало инфекционных заболеваний в популяции. Это значение потенциально может значительно варьироваться между тропиками и более холодными регионами. Тем не менее, теорема Байеса позволяет легко вывислить P (App|Leuko), который важен для диагностики, исходя из общепризнанного значения P (Leuko|App).

Прежде чем мы углубимся в этот пример и расширим его до медицинской экспертной системы для аппендицита, мы должны сначала ввести необходимый вероятностный вывод. механизм.

7.2 Принцип максимальной энтропии

Теперь мы покажем на примере вывода, что исчисление для рассуждений в условиях неопределенности может быть реализовано с использованием теории вероятностей. Однако вскоре мы увидим, что прототипные вероятностные тропы быстро заканчиваются. В частности, когда слишком мало знаний для решения необходимых уравнений, нужны новые идеи. Американский физик Э. Т. Джейнс провел пионерскую работу в этой области в 1950-х годах. Он утверждал, что при недостающих знаниях можно максимизировать энтропию желаемого распределения вероятностей, и применил этот принцип ко многим примерам в [Jay57, Jay03]. Затем этот принцип получил дальнейшее развитие [Che83, Nil86, Kan89, KK92] и теперь является зрелым и может применяться технологически, что мы покажем на примере проекта LEXMED в разд. 7.3.

Machine Translated by Google

7.2.1 Правило вывода для вероятностей

Мы хотим вывести правило вывода для неопределенного знания, аналогичное modus ponens. Из знания предложения A и правила A B следует сделать вывод B. В лаконичной формулировке это звучит

$$\frac{A,\ A \quad B}{B}.$$

Обобщение вероятностных правил дает

$$\frac{P(A) = \alpha,\ P(B|A) = \beta}{P(B) = ?}.$$

Пусть заданы два вероятностных правила α, β и желаемое значение P(B). Путем маргинализации мы получаем желаемое маргинальное распределение

$$P(B) = P(A,B) + P(\neg A,B) = P(B|A) \cdot P(A) + P(B|\neg A) \cdot P(\neg A).$$

Три значения P (A), P (¬A), P (B|A) в правой части известны, но значение P (B|¬A) неизвестно. Мы не можем сделать точное утверждение о P (B) с помощью классической теории вероятностей, но в лучшем случае мы можем оценить P (B) P (B|A) · P (A).

Теперь рассмотрим распределение

$$P(A, B) = (P(A, B), P(A, \neg B), P(\neg A, B), P(\neg A, \neg B))$$

и введем для краткости четыре неизвестных

$$p1 = P(A, B),\ p2 = P(A,\neg B),\ p3 = P(\neg A, B),\ p4 = P(\neg A, \neg B).$$

Эти четыре параметра определяют распределение. Если все они известны, то можно вычислить любую вероятность для двух переменных A и B. Для вычисления четырех неизвестных необходимы четыре уравнения. Одно уравнение уже известно в виде условия нормировки

$$p1 + p2 + p3 + p4 = 1.$$

Следовательно, необходимы еще три уравнения. Однако в нашем примере известны только два уравнения.

По заданным значениям P (A) = α и P (B|A) = β вычисляем

$$P(A, B) = P(B|A) \cdot P(A) = \alpha\beta$$

Machine Translated by Google

и

$$P(A) = P(A,B) + P(A,\neg B).$$

Отсюда мы можем составить следующую систему уравнений и решить ее, насколько это возможно:

$$p1 = \alpha\beta, \tag{7.4}$$

$$p1 + p2 = a, \tag{7,5}$$

$$p1 + p2 + p3 + p4 = 1, \tag{7,6}$$

(7.4) в (7.5): $p2 = \alpha \quad \alpha\beta = \alpha(1 \quad \beta),$ (7.7)

(7.5) в (7.6): $p3 + p4 = 1 \quad \alpha.$ (7.8)

Таким образом, вероятности p1, p2 для интерпретаций (A, B) и (A,¬B) известны, но для значений p3, p4 остается только одно уравнение. Чтобы прийти к определенному решению, несмотря на это недостающее знание, мы меняем нашу точку зрения. Мы используем данное уравнение в качестве ограничения для решения оптимизационной задачи.

Мы ищем распределение p (для переменных p3, p4), которое максимизирует энтропию

$$H(p) = \sum_{i=1}^{n} p_i \ln p_i = \quad p3 \ln p3 \quad p4 \ln p4 \tag{7.9}$$

при ограничении p3 +p4 = 1 α (7.8). Почему именно функция энтропии должна быть максимизирована? Поскольку у нам не хватает информации о распределении, ее нужно каким-то образом добавить. Мы могли бы исправить специальное значение, например, p3 = 0,1. Однако лучше определить значения p3 и p4 так, чтобы добавляемая информация была минимальной. Мы можем показать (раздел 8.4.2 и [SW76]), что энтропия измеряет неопределенность распределения с точностью до постоянного множителя. Тогда отрицательная энтропия является мерой количества информации, содержащейся в распределении. Максимизация энтропии минимизирует информативность распределения. Для наглядности функция энтропии для двумерного случая представлена графически на рис. 7.2 на стр.

Для определения максимума энтропии при ограничении p3 + p4 1 + α = 0 воспользуемся методом множителей Лагранжа [Ste07]. Функция Лагранжа читается

$$L = \quad p3 \ln p3 \quad p4 \ln p4 + \lambda(p3 + p4 \quad 1 + \alpha).$$

Взяв частные производные по p3 и p4, получим

$$\frac{\partial L}{\partial p3} = \quad \ln p3 \quad 1 + \lambda = 0,$$

$$\frac{\partial L}{\partial p4} = \quad \ln p4 \quad 1 + \lambda =$$

0

Machine Translated by Google

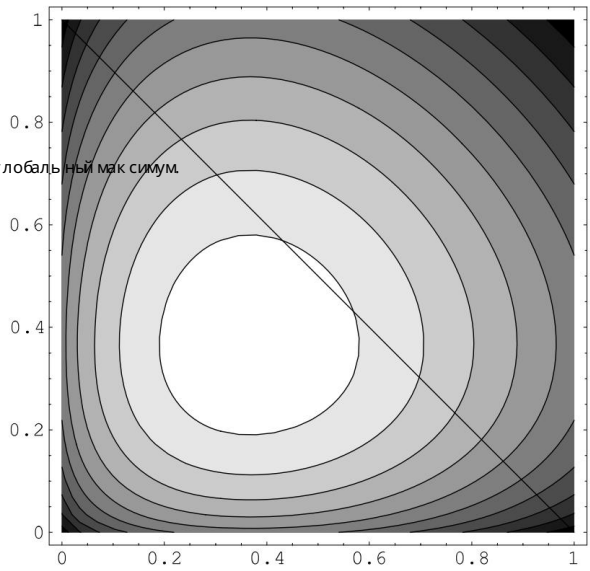


Рис. 7.2 Контурная линейная диаграмма двумерной функции энтропии. Мы видим, что оно строго выпукло во всем единичном квадрате и имеет изолированный глобальный максимум. Также отмечено ограничение p3 + p4 = 1 как частный случай условия p3 + p4

1 + α = 0 при α = 0, которое имеет значение здесь

и рассчитать

$$p3 = p4 = \frac{1 - a}{2}.$$

Теперь мы можем вычислить желаемое значение

$$P(B) = P(A, B) + P(\neg A, B) = p1 + p3 = \alpha\beta + = \alpha\beta \quad 2 \quad \frac{1 - \alpha 1}{2} \quad - \quad + \frac{1}{2}.$$

Замена в α и β дает

$$\Pi(B) = \Pi(A)\,\Pi(B \mid A) - 2 \quad \frac{1}{-} \quad 1 \; \frac{}{+ 2}.$$

P (B) показано на рис. 7.3 на стр. 126 для различных значений P (B|A). Мы видим, что в двухзначном пограничном случае, то есть когда P (B) и P (B|A) принимают значения 0 или 1, вероятностный вывод возвращает то же значение для P (B), что и modus ponens. Когда A и B|A оба истинны, B также истинно. Интересен случай P (A) = 0, в котором ¬A истинно. Modus ponens здесь неприменим, но наша формула дает значение 1/2 для P (B) независимо от P (B|A). Когда A ложно, мы ничего не знаем о B, что в точности отражает нашу интуицию Случай, когда P (A) = 1 и P (B|A) = 0, также покрывается логикой высказываний. Здесь A истинно и A B ложно, и, таким образом, A ¬B истинно. Тогда B ложно. Горизонтальная линия на рисунке означает, что мы не можем сделать предсказание относительно B в случае P (B|A) = 1/2. Между этими точками P (B) изменяется линейно для изменений P (A) или P (B|A).

Machine Translated by Google

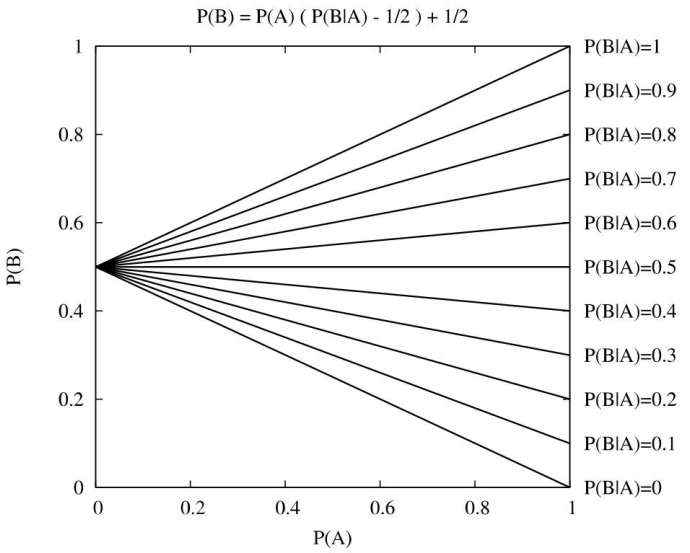


Рис. 7.3 Массив кривых для P (B) в зависимости от P (A) для различных значений P (B|A)

Теорема 7.3. Пусть существует непротиворечивая 2 система линейных вероятностных уравнений. Тогда существует единственный максимум функции энтропии с заданными уравнениями в качестве ограничений. Определенное таким образом распределение MaxEnt имеет минимальную информативность при ограничениях.

Из этой теоремы следует, что не существует распределения, удовлетворяющего ограничениями обладающего более высокой энтропией, чем распределение MaxEnt. Исчисление, которое приводит к более низкой энтропии, вводит дополнительную информацию ad hoc, что не оправдано.

При более внимательном рассмотрении приведенного выше расчета P(B) мы видим, что два значения p3 и p4 всегда встречаются симметрично. Это означает, что замена двух переменных не меняет результат. Таким образом, конечный результат p3 = p4. Так называемое безразличие этих двух переменных приводит к тому, что MaxEnt приравнивает их. Это соотношение справедливо в общем случае:

Определение 7.5. Если произвольная замена двух или более переменных в уравнениях Лагранжа приводит к эквивалентным уравнениям, эти переменные называются безразличными.

2Система вероятностных уравнений называется согласованной, если существует хотя бы одно решение, то есть одно распределение, удовлетворяющее всем уравнениям.

Machine Translated by Google

Теорема 7.4. Если набор переменных $\{p_{i1},\ldots,p_{ik}\}$ безразличен, то максимум энтропии при заданных ограничениях находится в точке, где $p_{i1} = p_{i2} = \cdots = p_{ik}$.

Зная это, мы могли бы сразу приравнять две переменные p_3 и p_4 (без решения уравнений Лагранжа).

7.2.2 Максимальная энтропия без явных ограничений

Теперь рассмотрим случай, когда знание не дано. Это означает, что кроме условия нормировки

$$p_1 + p_2 + \cdots + p_n = 1,$$

ограничений нет. Следовательно, все переменные безразличны. Поэтому мы можем[3] приравнять их, и тогда $p_1 = p_2 = \cdots = p_n = 1/n$. Для рассуждений в условиях неопределенности это означает, что при полном отсутствии знаний все миры равновероятны. То есть распределение равномерное. Например, в случае двух переменных A и B будет так, что

$$P(A, B) = P(A, \neg B) = P(\neg A, B) = P(\neg A, \neg B) = 1/4,$$

откуда следует $P(A) = P(B) = 1/2$ и $P(B|A) = 1/2$. Результат для двумерного случая можно увидеть на рис. 7.2 на стр. 125, поскольку отмеченное условие является точностью условием нормализации. Мы видим, что максимум энтропии лежит на прямой точно в точке (1/2, 1/2).

Как только значение условия отклоняется от значения, полученного из равномерного распределения, вероятности миров меняются. Покажем это на следующем примере. С теми же описаниями, которые использовались выше, мы предполагаем, что только

$$P(B|A) = \beta$$

известен. Таким образом, $P(A, B) = P(B|A)P(A) = \beta P(A)$, откуда следует $p_1 = \beta(p_1 + p_2)$, и мы получаем два ограничения

$$\beta p_2 + (\beta - 1)p_1 = 0,$$

$$p_1 + p_2 + p_3 + p_4 - 1 = 0.$$

---

[3] Читатель может вычислить этот результат путем максимизации энтропии при условии нормализации (упражнение 7.5 на стр. 158).

Machine Translated by Google

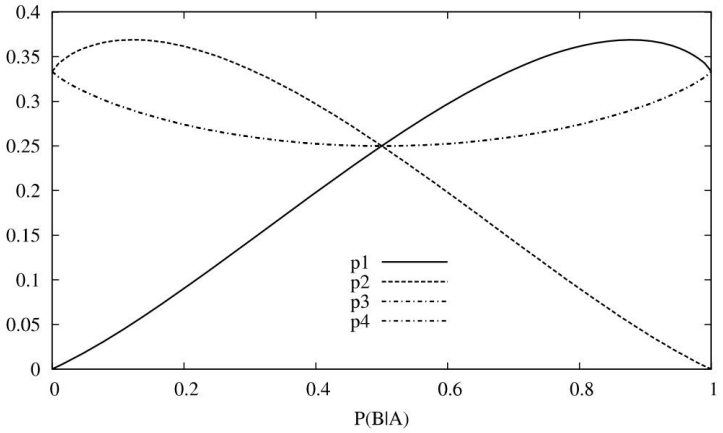


Рис. 7.4 p1, p2, p3, p4 в зависимости от β

Таблица 7.1 Таблица истинности для материальной смысловой условной вероятности для пропозиционального логического предела

ABA Б	П (А)	П (Б)	Р (Б|А)
ттт 111			
т фф 100			
фут шфт 0	0	1	Неопределенный
		0	Неопределенный

Здесь уравнения Лагранжа уже не могут быть так легко решены символически. Численное решение уравнений Лагранжа дает картину, представленную на рис. 7.4, что показывает, что p3 = p4. Мы можем видеть это уже в ограничениях, в которых p3 и p4 безразличны При P (B|A) = 1/2 получается равномерное распределение, которое нетничего удивительного. Это означает, что ограничение на это значение не влечет ограничения на распределение. Кроме того, мы можем видеть, что для малых P (B|A) P (A, B) равно тоже маленький.

7.2.3 Условная вероятность в сравнении с существенным следствием

Теперь мы покажем, что для моделирования рассуждений условная вероятность лучше. чем то, что известно в логике как материальная имплика ция (по этому поводу см. также [Ada75]). Сначала мы рассмотрим таблицу истинности, показанную в таблице 7.1, в которой условная вероятность и материальное значение для предельных случаев нулевой и единичной вероятностей равны в сравнении. В обоих случаях с ложными посылками (которые интуитивно являются критическими случаями), P (B|A) не определено, что имеет смысл.

Machine Translated by Google

Теперь зададимся вопросом, какое значение принимает P (B|A), когда заданы произвольные значения P (A) = α и P (B) = γ и никакой другой информации не известно. Снова мы максимизируем энтропию при заданных ограничениях x. Как и выше, мы устанавливаем

$$p1 = P(A, B), p2 = P(A, \neg B), p3 = P(\neg A, B), p4 = P(\neg A, \neg B)$$

и получить в качестве ограничений

$$p1 + p2 = a, \tag{7.10}$$

$$p1 + p3 = y, \tag{7.11}$$

$$p1 + p2 + p3 + p4 = 1. \tag{7.12}$$

При этом мы вынуждены, используя максимизацию энтропии (см. упражнение 7.8 на стр. 159).

$$p1 = \alpha\gamma, p2 = \alpha(1 - \gamma), p3 = \gamma(1 - \alpha), p4 = (1 - \alpha)(1 - \gamma).$$

Из $p1 = \alpha\gamma$ следует, что $P(A, B) = P(A) \cdot P(B)$, а это означает, что A и B независимы. Поскольку у нет никаких ограничений, соединяющих A и B, принцип MaxEnt приводит к независимости этих переменных. Правая половина Таблицы 7.1 на странице 128 облегчает понимание. Из определения

$$P(B|A) = \frac{П(A, Б)}{П(A)}$$

из этого следует для случая $P(A) = 0$, то есть, когда посылка не ложна, поскольку A и B независимы, что $P(B|A) = P(B)$. Для случая $P(A) = 0$ значение $P(B|A)$ остается неопределенным.

7.2.4 MaxEnt-системы

Как упоминалось ранее, из-за нелинейности функции энтропии оптимизация MaxEnt обычно не может быть выполнена символически и для нетривиальных задач. Таким образом, были разработаны две системы для численной максимизации энтропии. Первая система, SPIRIT (Симметричное вероятностное интенсиональное мышление в сети вывода, работает в переходный период, www.xspirit.de), [RM96], была построена в Fernuniversität Hagen. Вторая, PIT (Probability Induction Tool), была разработана в Мюнхенском техническом университете [Sch96, ES99, SE00]. Сейчас мы кратко представим PIT.

Система PIT использует метод последовательного квадратичного программирования (SQP) для нахождения экстремума функции энтропии при заданных ограничениях x. В качестве входных данных PIT ожидает данные, содержащие ограничения. Например, ограничения $P(A) = \alpha$ и $P(B|A) = \beta$ из разд. 7.2.1 имеют вид

Machine Translated by Google


```
переменная A{t,f}, B{t,f};

P([A=t]) = 0,6;
P([B=t] | [A=t]) = 0,3;

K П([B=t]);
QP([B=t] | [A=t]);
```

Поскольку у PIT выполняется численный расчет, мы должны ввести значения вероятности. Предпоследняя строка содержит запрос QP([B=t]). Это означает что P(B) является желаемым значением. На www.pit-systems.de в разделе «Примеры» мы теперь поместите этот ввод на пустую страницу ввода («Пустая страница») и запустите PIT. В результате мы получать

номер	Значение истины	Вероятностный запрос
1	НЕ УКАЗАНО	3.800e–01 QP([B = t]);
2	НЕ УКАЗАНО	3.000e–01 QP([A = t]-├ [B = t]);

и отсюда считать P (B) = 0,38 и P (B|A) = 0,3.

7.2.5 Пример Твити

Теперь покажем на примере Tweety из разд. 4.3, что вероятностные рассуждения и, в частности, MaxEnt немонотонны очень хорошо моделируют повседневные рассуждения. хорошо. Мы моделируем соответствующие правила с вероятностями следующим образом:

P (птица|пингвин) = 1 P (мухи| птица) [0,95, 1]	«пингвины — это птицы» «(почти все) птицы умеют летать»
P (мухи|пингвины) = 0	«пингвины не умеют летать»

Первое и третье правила представляют собой твердые прогнозы, которые так же легко формулируются в логике. Во втором, однако, мы выражаем наше знание, что почти все птицы могут летать с помощью условного интервала. С входными данными PIT

```
вар пингвин {да, нет}, птица {да, нет}, мухи {да, нет};

P([птица=да] | [пингвин=да]) = 1;
P([flies=yes] | [bird=yes]) IN [0.95,1];
P([мухи=да] | [пингвин=да]) = 0;

QP([мухи=да]| [пингвин=да]);
```

мы получаем правильный ответ

Machine Translated by Google

номер	Истинное значение	Вероятностный запрос
1	НЕУКАЗАНО	0.000e+00 QP([пингвин=да]-|> [мухи=да]);

с предположением, что пингвины не могут летать. Объяснение этому очень простое. При P (flies|bird) [0,95, 1] возможно наличие нелетающих птиц. Если это правило было заменено на P (flies|bird) = 1, тогда PIT ничего не смог бы сделать и будет выводить сообщение об ошибке о несогласованных ограничениях.

В этом примере мы можем легко увидеть, что интервалы вероятности часто очень полезны для моделирования нашего незнания о точных значениях вероятности. Мы могли бы сделать еще более размытая формулировка второго правила в духе «обычно птицы летают» с P (flies|bird) (0,5, 1]. Использование полуоткрытого интервала исключает значение 0,5.

В [Pea88] уже было показано, что этот пример можно решить с помощью вероятностной логики даже без MaxEnt. В [Sch96] для ряда требовательных тестов немонотонных рассуждений показано, что их можно элегантно решить с помощью Макс-Энт. В следующем разделе мы представим успешное практическое применение MaxEnt в виде медицинской экспертной системы

7.3 LEXMED, экспертная система для диагностики аппендицита

Медицинская экспертная система LEXMED, использующая метод MaxEnt, была разработана в Равенсбург-Вайнгартенском университете прикладных наук Манфредом Шрамм, Вальтер Рампф и автор в сотрудничестве с Weingarten 14-Больница Нотельфера [SE00, Le999].5 Аббревиатура LEXMED расшифровывается как обучающая экспертная система для медицинской диагностики.

7.3.1 Диагностика аппендицита с помощью юформальных методов

Наиболее распространенной серьезной причиной острой боли в животе [dD91] является аппендицит. воспаление аппендикса, слепой трубки, соединенной со слепой кишкой. Даже сегодня диагностика во многих случаях может быть затруднена [OFY+95]. Например, примерно до 20% удаленных аппендиксов не имеют патологических изменений, т.е. что операции не нужны. Кроме того, регулярно возникают случаи, когда воспаленный аппендикс не распознается как таковой.

Еще с начала 1970-х годов предпринимались попытки автоматизировать диагностику аппендицита с целью юснижения частоты ложных диагнозов [dDLS+72, OPB94, OFY+95]. Особого внимания заслуживает экспертная система.

4QP([penguin=yes]-|> [flies=yes]) — это альтернативная форма синтаксиса PIT для QP([мухи=да] | [пингвин=да]).

5Проект финансировался немецкой землей Баден-Вюртемберг, страховой компанией AOK Baden-Württemberg, Университетом прикладных наук Равенсбург-Вайнгартен и Больница Нотельфер 14 в Вайнгартене.

Machine Translated by Google

для диагностики острой боли в животе, разработанный Домбалем в Великобритании. Она была обнародована в 1972 году, то есть значительно раньше знаменитой системыMYCIN.

Почти все формальные диагностические процессы, используемые в медицине на сегодняшний день, основанына баллах. Системыоценки чрезвычайно просты в применении: за каждое значение симптома (например, лихорадка или боль внизу живота справа) врач отмечает определенное количество баллов. Если сумма баллов превышает определенное значение (порог), рекомендуется определенное решение (например, операция). Для n симптомов S1,...,Sn оценка аппендицита формально может быть описана как

$$\text{Диагностировать} = \begin{cases} \text{Аппендицит}, & \text{если } w1S1 + \cdots + wnSn > \Theta, \\ \text{отрицательный} & \text{еще.} \end{cases}$$

Таким образом, с помощью юбаллов линейная комбинация значений симптомов сравнивается с пороговым значением Θ. Веса симптомов извлекаются из баз данных с использованием статистических методов. Преимуществом баллов является простота их применения. Взвешенную сумму баллов можно легко рассчитать вручную, и для диагностики не требуется компьютер.

Из-за линейности этого метода оценки слишком слабы для моделирования сложных отношений. Поскольку вклад wiSi симптома Si в оценку рассчитывается независимо от других симптомов, ясно, что системыоценки не могут принимать во внимание какой-либо «контекст». В принципе, они не могут различить комбинации симптомов, например, они не могут отличить количество лейкоцитов у старого пациента от количества лейкоцитов у молодого пациента.

Для фиксированного заданного набора симптомов условная вероятность гораздо более эффективна, чем оценки для прогнозирования, поскольку у последние не могут описать зависимости между различными симптомами. Мыможем показать, что оценки неявно предполагают, что все симптомы независимы

При использовании баллов возникает еще одна проблема. Чтобыдостичь хорошего качества диагностики, мыдолжныпредъявлять строгие требования к базамданных, используемым для статистического определениявзаимозависимостей, они должныбыть репрезентативными для группыпациентов в области, в которой используется диагностическая система. Это часто трудно, если вообще возможно, гарантировать. В таких случаях баллыи другие статистические методылибо не могут быть использованы, либо будут иметь высокий уровень ошибок.

7.3.2 Гибридная вероятностная база знаний

Сложныезависимости часто возникают в медицине. С помощью юLEXMED этивзаимосвязи можно хорошо смоделировать и быстро рассчитать. Здесь существенное значение имеет использование вероятностных предположений, с помощью которых неопределенная и неполная информация может быть выражена и обработана интуитивно и математически обоснованным способом. Типичным запросом к экспертной системе может служить следующий вопрос: «Насколько высока вероятность воспаления аппендикса у пациента 23-летнего мужчиныс болями в правой нижней части живота и количеством лейкоцитов 13 000 ?» Сформулированное как условная вероятность с использованием названий и диапазонов значений симптомов, использованных в Таблице 7.2 на стр. 133, это выглядит следующим образом:

Machine Translated by Google

Таблица 7.2 Признаки, используемые для запроса в LEXMED, и их значения. Количество значений для каждый симптом указан в столбце с пометкой #

Симптом	Ценности	# Короткий
Пол	Мужчина, женщина	2 Сек с2
Возраст	0–5, 6–10, 11–15, 16–20, 21–25, 26–35, 36–45, 46–55, 56–65, 65–	10 лет 10
Боль 1-й квадр.	Да, нет	2 П1К 2
Боль 2-й квадр.	Да, нет	2 П2К 2
Боль 3-й квадр.	Да, нет	2 P3Q2
Боль 4-й квадр.	Да, нет	2 P4Q2
Охрана	Локальный, глобальный, нет	3 Гуа3
Отскок нежности	Да, нет	2 воз62
Боль при постукивании	Да, нет	2 Нажмите2
Ректальная боль	Да, нет	2 Запись P2
Звуки кишечника	Слабая, нормальная, повышенная, нет	4 Лук S4
Аномальное УЗИ	Да, нет	2 Соно2
Аномальные выделения мочи.	Да, нет	2 Урин2
Температура (ректальная)	–37,3, 37,4–37,6, 37,7–38,0, 38,1–38,4, 38,5–38,9, 39,0–	6 TRec6
Лейкоциты	0–6к, 6к–8к, 8к–10к, 10к–12к, 12к–15к, 15к–20к, 20к–	7 Леук о7
Диагноз	Воспаленные, перфорированные, отрицательные, другие	4 Диаг.4

P (Diag4 = воспаленный Diag4 = перфорированный |
Sex2 = мужской Age10 21–25 Leuko7 12k–15k).

Используя вероятностные предположения, LEKCMED может использовать информацию из нерепрезентативных баз данных, поскольку эта информация может быть дополнена соответственно из других источников. В основе LEXMED лежит база данных, которая содержит данные о пациентах, у которых аппендикс был удален хирургическим путем. С помощью статистических методов генерируются (около 400) правил, которые компилируют знания, содержащиеся в базе данных, в абстрактную форму[ES99]. Потому что нет больных в этой базе данных с подозрением на аппендицит, но с отрицательным диагнозом (т. е. не требующим лечения)6 нет сведений об отрицательных
больных в базе. Таким образом, необходимо добавить знания из других источников.
Поэтому в LEXMED правила, собранные из базы данных, дополняются
(около 100) правил из медицинских специалистов и медицинской литературы Это приводит к гибридная вероятностная база данных, которая содержит знания, извлеченные из данных в виде а также знания, явно сформулированные экспертами. Поскольку оба типа правил

6Эти отрицательные диагнозы обозначаются как «неспецифическая боль в животе» (NSAP).

Machine Translated by Google

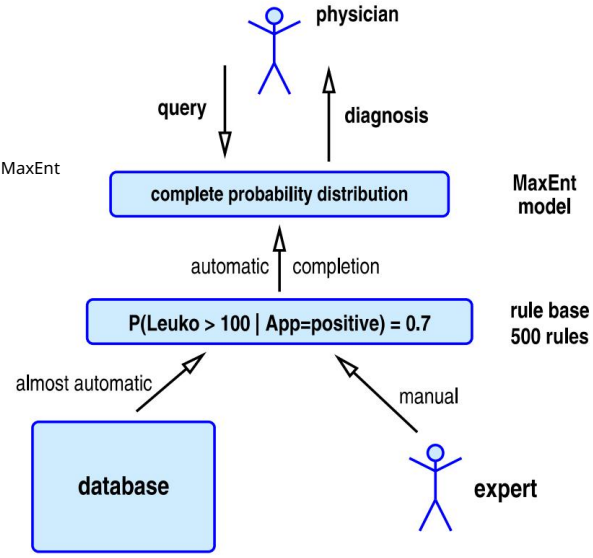


Рис. 7.5 Вероятностные правила генерируются из данных и экспертных знаний, которые интегрируются в базу правил (базу знаний) и окончательно дополняются с помощью метода MaxEnt

сформулированные как условные вероятности (см., например, (7.14) на стр. 138), их можно легко комбинировать, как показано на рис. 7.5 и более подробно на рис. 7.7 на стр. 136.

LEXMED рассчитывает вероятности различных диагнозов, используя распределение вероятностей всех соответствующих переменных (см. Таблицу 7.2 на стр. 133). Поскольку все 14 симптомов, используемых в LEXMED, и диагнозы моделируются как дискретные переменные (даже непрерывные переменные, такие как количество лейкоцитов, разделены на диапазоны), размер распределения (то есть размер пространства событий) можно определить с помощью таблицы 7.2 на стр. 133 как произведение количества значений всех симптомов, или

$$10 \cdot 2 \cdot 10 \cdot 3 \cdot 4 \cdot 6 \cdot 7 \cdot 4 = 20\,643\,840$$

элементы. Таким образом, благодаря условию нормировки и из теоремы 7.1 оно имеет 20 643 839 независимых значений. Каждый набор правил с менее чем 20 643 839 вероятностными значениями потенциально не полностью описывает это пространство событий. Чтобы иметь возможность ответить на любой произвольный запрос, экспертная система нуждается в полном дистрибутиве. Построение такого обширного последовательного распределения с использованием статистических методов очень сложно[7]. Требовать от человека-эксперта все 20 643 839 значений для распределения (вместо вышеупомянутых 100 правил) было бы принципиально невоз

Здесь в игру вступает метод MaxEnt. Обобщение около 500 правил до полной вероятностной модели выполняется в LEXMED путем максимизации энтропии с 500 правилами в качестве ограничений. Эффективное кодирование результирующего распределения MaxEnt приводит к тому, что время отклика для диагностики составляет около одной секунды

[7] Задача создания функции из набора данных известна как машинное обучение. Мы подробно рассмотрим это в гл. 8.

Machine Translated by Google

Personal Details	unknown	values	
Gender	◉	○ male ○ female	?
Age-group	○	○ 0-5 ○ 6-10 ○ 11-15 ○ 16-20 ◉ 21-25 ○ 26-35 ○ 36-45 ○ 46-55 ○ 56-65 ○ 65-	?
Results of examination	**not done**	**values**	
1st quadrant	◉	○ yes ○ no	?
2nd quadrant	◉	○ yes ○ no	?
3rd quadrant	○	◉ yes ○ no	?
4th quadrant	◉	○ yes ○ no	?
guarding	○	◉ local ○ global ○ none	?
rebound tenderness	○	◉ yes ○ no	?
pain on tapping	○	◉ yes ○ no	?
rectal pain	◉	○ yes ○ no	?
bowel sounds	○	○ weak ◉ normal ○ increased ○ none	?
abnormal ultrasound	◉	○ yes ○ no	?
abnormal urine sediment	◉	○ yes ○ no	?
temperature range (rectal)	○	○ -37.3 ○ 37.4-37.6 ○ 37.7-38.0 ○ 38.1-38.4 ◉ 38.5-38.9 ○ 39.0-	?
leucocyte count	○	○ 0-6k ○ 6k-8k ○ 8k-10k ○ 10k-12k ◉ 12k-15k ○ 15k-20k ○ 20k-	?

	Result of the PIT diagnosis			
Diagnosis	App. inflamed	App. perforated	Negative	Other
Probability	0.70	0.17	0.06	0.07

Рис. 7.6. Маска ввода LEXMED для ввода исследуемых симптомов и под ней вывод результирующих вероятностей диагноза

7.3.3 Применение ЛЕКСМЕД

Использование LEXMED простое и понятное. Врач посещает Для автоматической Домашняя страница LEXMED на [8] диагностика и врач www.lexmed.de. вводит результаты своего обследования в форму ввода на рис. 7.6. Через одну-две секунды он получает вероятности четырех различных диагнозов, а также предложение по лечению(раздел 7.3.5). Если в качестве входных данных отсутствуют какие-либо результаты обследования (например, результаты УЗИ), то врач выбирает запись без обследования . Естественно, достоверность диагноза выше, когда вводится больше значений симптомов.

8А версия с ограниченным функционалом доступна без пароля .

Machine Translated by Google

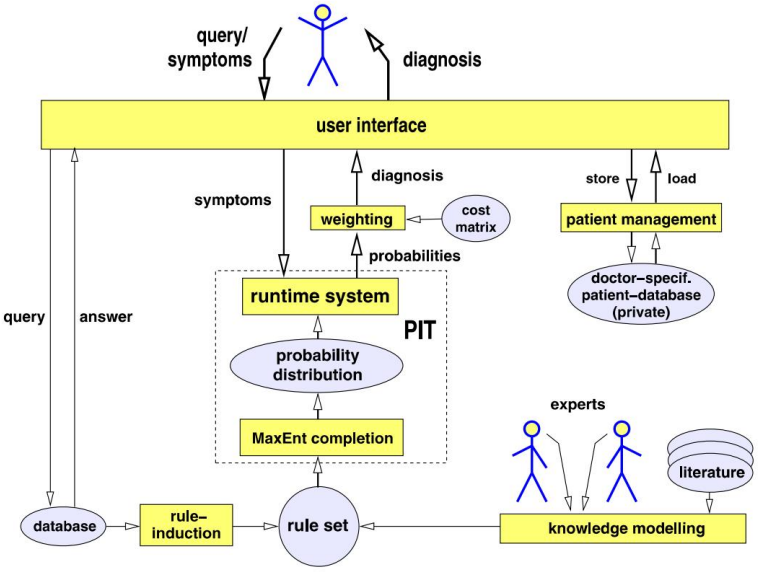


Рис. 7.7 Правила генерируются из базы данных, а также из экспертных знаний. Из них MaxEnt создает полное распределение вероятностей. По запросу пользователя вычисляется вероятность каждого возможного диагноза. Используя матрицу затрат (см. раздел 7.3.5), затем предлагается решение.

Каждый зарегистрированный пользователь имеет доступ к частной базе данных пациентов, в которой можно архивировать введенные данные. Таким образом, данные и диагнозы более ранних пациентов можно легко сравнить с данными нового пациента.

7.3.4 Функции ЛЕКСМЕД

Знания формализуются с помощью условных вероятностных утверждений. Например, предложение

$$P(\text{Leuko7} > 20000 \mid \text{Diag4} = \text{воспаленный}) = 0{,}09$$

дает частоту 9% для значения лейкоцитов более 20 000 в случае воспаленного аппендикса.[9]

Изучение правил методом статистической индукции

Необработанные данные в базе данных LEXMED содержат 54 различных (анонимизированных) значения для 14 646 пациентов. Как упоминалось ранее, в эту базу данных включены только пациенты с острым аппендицитом, удаленным хирургическим путем. Из 54 атрибутов, используемых в базе данных, после статистического анализа 14 симптомов, показанных в Табл. 7.2 на стр. 133

[9] Вместо отдельных числовых значений здесь также могут использоваться интервалы (например, [0,06, 0,12]).

Machine Translated by Google

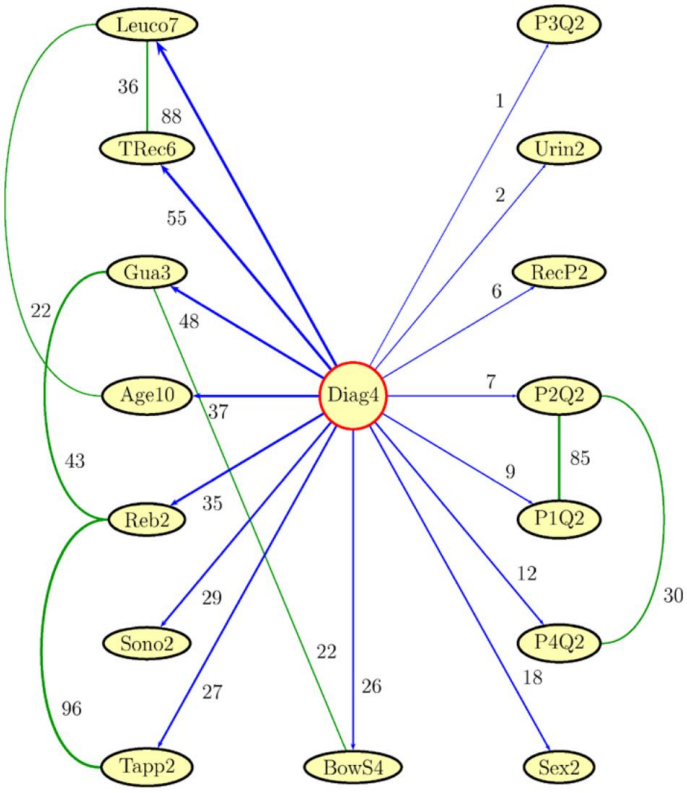


Рис. 7.8 Граф зависимост ей, вын исленный из баз ыданны х

были исполь зованы Т еперь правила соз дают ся из эт ой баз ыданных в два эт апа. Первый шаг опред еля ет ст рук т уру зависимост ей симпт омов. Вт орой шаг з аполня ет эт у ст рук т уру соот вет ст вующими вероя т ност ными правилами.10

Опред еление г рафа з ависимост ей Граф на рис. 7.8 сод ержит для к аждой переменной (симпт ома и диагноз а) уз ел и направленные ребра, соед иня ющие раз лич ные уз лы Т олщина границ между переменными пред ст авля ет собой меру ст ат ист ич еск ой з ависимост и или к орреля ц ии переменных. К орреля ц ия двух нез ависимых переменных равна нулю Парная к орреля ц ия для к аждого из 14 симпт омов с Diag4 была рассч ит ана и пред ст авлена на граф ик е. К роме т ого, были рассч ит аны все т ройные к орреля ц ии между диагноз ом и двумя симпт омами. Из них т оль к о самые силь ные з нач ения были нарисованы к ак д ополнит ель ные границ ы между д вумя уч ast вующими симпт омами.

10Для сист емат ич еск ого введения в машинное обуч ение мы от сылаем ч ит ат еля к гл. 8.

Machine Translated by Google

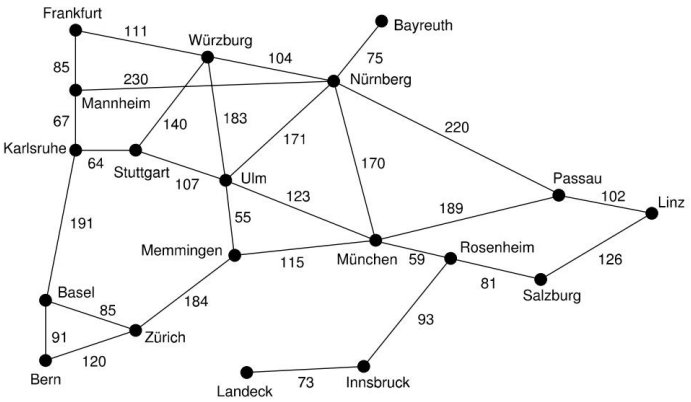


Рис. 6.5 Граф юга Германии как пример поисковой задачи с функцией стоимости

для b, потому что дерево с постоянным коэффициентом ветвления и глубиной d имеет в общей сложности

$$n = \sum_{i=0}^{d} b^i = \frac{b^{d+1} - 1}{b - 1} \tag{6.2}$$

узлы.

Для практического применения алгоритмов поиска конечных деревьев поиска последний уровень особенно важен, потому что

> Теорема 6.1. Для сильно ветвящихся конечных деревьев поиска с большим постоянным коэффициентом ветвления почти все узлы находятся на последнем уровне.

Простое доказательство этой теоремы рекомендуется читателю в качестве упражнения. (Упражнение 6.1 на стр. 110).

Пример 6.2. Нам дана карта, подобная той, что представлена на рис. 6.5, в виде графа с городами в качестве узлов и автомобильными дорогами между городами в виде взвешенных ребер с расстояниями. Ищем оптимальный маршрут из города А в город Б. Описание соответствующей схемы гласит

Состояние: Город как текущее местонахождение путешественника.

Начальное состояние: Произвольный город.

Целевое состояние: Произвольный город.

Действия: Путешествие из текущего города в соседний город.

Функция стоимости: расстояние между городами. Каждое действие соответствует ребру на графике с расстоянием в качестве веса.

Пространство состояний: Все города, то есть узлы графа.

Чтобы найти маршрут минимальной длины, необходимо учитывать затраты, поскольку они не являются постоянными, как в головоломке с восьмеркой.

Machine Translated by Google

Определение 6.3 Алгоритм поиска называется оптимальным , если он, если решение существует, всегда находит решение с наименьшей стоимостью.

Задача с 8 головоломками является детерминированной, что означает, что каждое действие ведет из состояния в уникальное последующее состояние. Кроме того, он является наблюдаемым, т. е. агент всегда знает, в каком состоянии он находится. При планировании маршрута в реальных приложениях не всегда задаются обе характеристики. Действие «Поехать из Мюнхена в Ульм» может, например, из-за аварии, привести к государству-преемнику «Мюнхен». Также может случиться так, что путешественник больше не знает, где он находится, потому что он заблудился. Сначала мы хотим игнорировать такого рода осложнения. Поэтому в этой главе мы будем рассматривать только детерминированные и наблюдаемые проблемы.

Такие задачи, как головоломка с восьмеркой, которые являются детерминированными и наблюдаемыми, делают планирование действий относительно простым, поскольку благодаря наличию абстрактной модели можно найти последовательность действий для решения проблемы, фактически не выполняя действия в реальном мире. . В случае головоломки с восьмерками нет необходимости перемещать квадраты в реальном мире, чтобы найти решение. Мы можем найти оптимальные решения с помощью так называемых автономных алгоритмов. Например, при создании роботов, которые должны играть в футбол, возникают совсем другие задачи. Здесь никогда не будет точной абстрактной модели действий. Например, робот, который бьет по мячу в определенном направлении, не может с уверенностью предсказать, куда будет двигаться мяч, потому что, среди прочего, он не знает, поймает мяч соперник или отклонит его. Здесь нужны онлайн-алгоритмы , которые принимают решения на основе сигналов датчиков в любой ситуации. Обучение с подкреплением, описанное в разд. 10, работает над оптимизацией этих решений на основе опыта.

6.2 Неинформированный поиск

6.2.1 Поиск в ширину

При поиске в ширину дерево поиска просматривается сверху вниз по алгоритму, приведенному на рис. 6.6 на стр. 90 , пока не будет найдено решение. Сначала каждый узел в списке узлов проверяется на предмет того, является ли он целевым узлом, и в случае успеха программа останавливается. В противном случае генерируются все преемники узла. Затем поиск рекурсивно продолжается в списке всех вновь сгенерированных узлов. Все повторяется до тех пор, пока не перестанут генерироваться преемники.

Этот алгоритм является общим. То есть он работает для произвольных приложений, если предусмотрены две специфичные для приложения функции «GoalReached» и «Successors». «GoalReached» вычисляет, является ли аргумент целевым узлом, а «Successors» вычисляет список всех узлов-преемников своего аргумента. На рис. 6.7 на стр. 90 показан снимок поиска в ширину.

Machine Translated by Google


```
ПОИСК В ШИРИНУ (Список узлов, Цель)
Новые узлы =
Для всех Node      NodeList
      Если цель достигнута (узел, цель)
            Возврат("Решение найдено", Узел)
      Новые узлы = Добавить (Новые узлы, Преемники (Узел))
Если НовыеУзлы =
      Возврат(ШИРИНА-ПЕРВЫЙ-ПОИСК(НовыеУзлы, Цель))
Еще
      Возврат("Нет решения")
```

Рис. 6.6 Алгоритм поиска в ширину

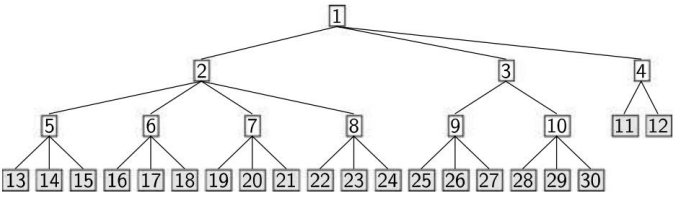


Рис. 6.7 Поиск в ширину при расширении узлов третьего уровня. Узлы нумеруются в порядке их создания. Преемники узлов 11 и 12 еще не созданы

Анализ Поскольку поиск в ширину полностью просматривает каждую глубину и достигает каждой глубины за конечное время, он является полным, если коэффициент ветвления b конечен. Оптимальное (то есть кратчайшее) решение находится, если затраты на все действия одинаковы (см. упражнение 6.7 на стр. 110). Время вычислений и объем памяти растут экспоненциально с глубиной дерева. Таким образом, для дерева с постоянным коэффициентом ветвления b и глубиной d общее время вычислений определяется выражением

$$\text{в} \cdot \sum_{\text{я}=0}^{\text{г}} \text{я б} \quad "=" \quad \frac{\text{б д+1} - 1}{\text{б} - 1} = \text{О(бд)}.$$

Хотя в памяти сохраняется только последний уровень, требования к объему памяти также составляют O(bd).

При скорости современных компьютеров, которые могут генерировать миллиарды узлов за считанные минуты, оперативная память быстро заполняется, и поиск завершается. Проблема не всегда находимого кратчайшего решения может быть решена с помощью так называемого поиска по единой стоимости, в котором узел с наименьшей стоимостью из отсортированных по возрастанию

Machine Translated by Google


```
ПОИСК В ГЛУБИНУ(Узел, Цель)

Если GoalReached(Узел, Цель) Возврат("Решение найдено")
Новые узлы = преемники (узел)
В то время как NewNodes =
    Результат = ГЛУБИНА-ПЕРВЫЙ-ПОИСК(Первый(НовыеУзлы), Цель)
    Если Результат = «Решение найдено» Возврат («Решение найдено»)
    Новые узлы = остальные (новые узлы)
Возврат("Нет решения")
```

Рис. 6.8 Алгоритм поиска в глубину. Функция «Первый» возвращает первый элемент списка, а «Остальное» — остальную часть списка.

список узлов всегда расширяется, а новые узлы сортируются. Таким образом, мы находим оптимальное решение. Однако проблема с памятью еще не решена. Решение этой проблемы обеспечивается поиском в глубину.

6.2.2 Поиск в глубину

При поиске в глубину в памяти одновременно хранится только несколько узлов. После расширения узла сохраняются только его преемники, а первый последующий узел немедленно расширяется. Таким образом, поиск быстро становится очень глубоким. Только когда у узла нет преемников и поиск на этой глубине завершается неудачей, следующий открытый узел расширяется путем возврата к последней ветви и так далее. Лучше всего это видно в элегантном рекурсивном алгоритме на рис. 6.8 и в дереве поиска на рис. 6.9 на стр. 92.

Анализ Поиск в глубину требует гораздо меньше памяти, чем поиск в ширину, поскольку на каждой глубине сохраняется не более b узлов. Таким образом, нам потребуется b · d ячеек памяти.

Однако поиск в глубину не является полным для бесконечно глубоких деревьев, потому что поиск в глубину приводит к бесконечному циклу, когда в крайней левой ветви нет решения. Поэтому вопрос поиска оптимального решения является устаревшим. Из-за бесконечного цикла невозможно установить ограничение на время вычислений. В случае дерева поиска конечной глубины с глубиной d всего генерируется около b узлов.

Г

Таким образом, время вычислений, как и при поиске в ширину, растет экспоненциально с глубиной.

Мы можем сделать дерево поиска конечным, установив предел глубины. Теперь, если в усеченном дереве поиска не найдено решения, тем не менее, могут быть решения за пределом.
Таким образом, поиск становится неполным. Однако есть очевидные идеи для завершения поиска.

Machine Translated by Google

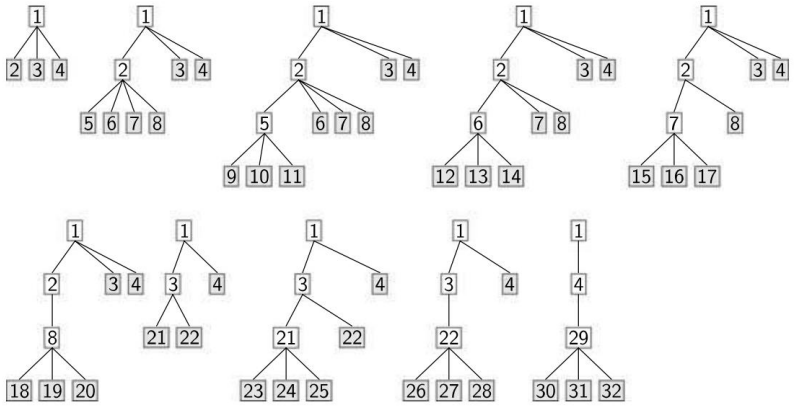


Рис. 6.9. Выполнение поиска в глубину. Все узлы на глубине три неуспешны и вызывают возврат. Узлы пронумерованы в порядке их создания.

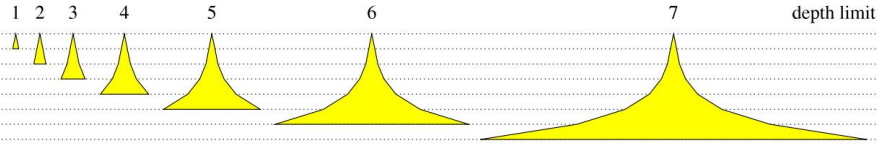


Рис. 6.10 Схематическое изображение развития дерева поиска при итеративном углублении с пределами от 1 до 7. Ширина дерева соответствует коэффициенту ветвления 2

6.2.3 Итеративное углубление

Мы начинаем поиск в глубину с пределом глубины, равным 1. Если решение не найдено, мы повышаем предел на 1 и начинаем поиск с начала и т. д., как показано на рис. 6.10 . Это итеративное повышение предела глубины называется итеративным углублением.

Мы должны дополнить программу поиска в глубину, приведенную на рис. 6.8 на стр. 91 , двумя дополнительными параметрами «Глубина» и «Предел». «Глубина» увеличивается на единицу при рекурсивном вызове, а заголовок цикла while заменяется на «While NewNodes = And Depth < Limit». Модифицированный алгоритм представлен на рис. 6.11 на стр. 93.

Анализ Требование к памяти такое же, как и при поиске в глубину. Можно возразить, что многократный перезапуск поиска в глубину с нулевой глубины приводит к большому количеству избыточной работы. Для больших факторов ветвления это не так. Теперь мы покажем, что сумма количества узлов всех глубин вплоть до последнего dmax 1 во всех обыскиваемых деревьях намного меньше, чем количество узлов в последнем обыскиваемом дереве.

Machine Translated by Google


```
ИТЕРАТИВНОЕУГЛУБЛЕНИЕ(Узел, Цель)
Предел глубины = 0
Повторить
    Результат = ПОИСК В ГЛУБИНЕ-В(Узел, Цель, 0,Поиск в глубину)
    Предел глубины = Предел глубины + 1
Пока Результат = «Решение найдено»
```

```
DEPTHFIRSTSEARCH-B(Узел, Цель, Глубина, Предел)
Если GoalReached(Узел, Цель) Возврат("Решение найдено")
Новые узлы = преемники (узел)
В то время как NewNodes =     And Depth < Limit
    Результат =
        DEPTHFIRSTSEARCH-B(First(NewNodes), Goal, Depth + 1, Limit)
    Если Результат = «Решение найдено» Возврат («Решение найдено»)
    Новые узлы = остальные (новые узлы)
Возврат("Нет решения")
```

Рис. 6.11 Алгоритм итеративного углубления, вызывающий слегка модифицированный поиск в глубину с ограничением глубины (TIEFENSUCHE-B)

Пусть Nb(d) — количество узлов дерева поиска с коэффициентом ветвления b и depth d и dmax — последняя искомая глубина. Последнее искомое дерево содержит

$$Nb(dmax) = \sum_{i=0}^{dmax} b^{i} = \frac{b^{dmax+1} - 1}{b - 1}$$

узлы. Все предварительно обыскиваемые деревья вместе имеют

$$Nb(d) = \sum_{d=1}^{dmax} \frac{b^{d+1} - 1}{b - 1} = \frac{1}{b - 1} \sum_{d=1}^{dmax} b^{d+1} - dmax + 1$$

$$= \frac{1}{b - 1} \sum_{d=2}^{dmax} b^{d} - dmax + 1$$

$$= \frac{1}{b - 1} \left(\frac{b^{dmax+1} - 1}{b - 1} - 1 - b - dmax + 1 \right)$$

$$= \frac{1}{b - 1} \frac{b^{dmax+1} - 1}{b - 1} - \frac{1}{b - 1} Nb(dmax)$$

Machine Translated by Google

Таблица 6.1 Сравнение алгоритмов неинформированного поиска. (*) означает, что оператор только true, учитывая постоянную стоимость действия. ds — максимальная глубина конечного дерева поиска.

	В ширину поиск	Единая стоимость поиск	В глубину поиск	Итеративный углубление
Полнота	Да	Да	Нет	Да
Оптимальное решение	Да (*)	Да	Нет	Да (*)
Время вычисления	д б	д б	или b ds	б Г
Использование памяти	б Г	б Г	бд	бд

узлы. Для b > 2 это меньше числа Nb(dmax) узлов в последнем дереве.
При b = 20 первые dmax 1 деревьев вместе содержат примерно = 1/19 $\frac{1}{б\text{-}1}$
количество узлов в последнем дереве. Время вычислений для всех итераций, кроме последним можно пренебречь.

Как и поиск в ширину, этот метод является полным и при постоянной стоимости для всех действий находит кратчайшее решение.

6.2.4 Сравнение

Описанные алгоритмы поиска приведены рядом в таблице 6.1.

Мы можем ясно видеть, что итеративное углубление является победителем этого теста, потому что оно получает лучшую оценку во всех категориях. По сути, из всех четырех представленных алгоритмов именно он является единственный практически пригодный.

У нас действительно есть победитель в этом тесте, хотя для реальных приложений это обычно не удается. Даже для головоломки на 15, старшего брата головоломки на 8 (см. Упражнение 6.4 на стр. 110), существует примерно 2 × 1013 различных состояний. Для нетривиальных систем вывода пространство состояний на много порядков больше. Как показано в Разд. 6.1, все вычислительные мощности мира уже не помогут. Вместо что необходимо, так это интеллектуальный поиск, который исследует только крошечную часть поиска пространстве и находит там решение.

6.3 Эвристический поиск

Эвристики — это стратегии решения проблем, которые во многих случаях находят решение быстрее. чем неинформативный поиск. Однако это не гарантируется. Эвристический поиск может требуют гораздо больше времени и могут даже привести к тому, что решение не будет найдено.

Мы, люди, успешно используем эвристические процессы для самых разных вещей. Когда покупая овощи в супермаркете, например, мы оцениваем различные варианты за фунт клубники, используя только несколько простых критериев, таких как цена, внешний вид, источник производства, и доверие к продавцу, а затем мы выбираем лучший вариант по нутровому ощущению. Теоретически может быть лучше подвергнуть клубнику базовому химический анализ перед принятием решения о покупке. Например, клубничный рис может быть отравлен. Если бы это было так, анализ стоил бы

Machine Translated by Google


```
ЭВРИСТИЧЕСКИЙ ПОИСК(Начало, Цель)

Список узлов = [Начало]
Пока правда
    Если NodeList =     Возврат («Нет решения»)
    Узел = Первый (Список узлов)
    NodeList = Отдых (NodeList)
    Если GoalReached(Узел, Цель) Возврат("Решение найдено", Узел)
    NodeList = SortIn (Преемники (Узел), NodeList)
```

Рис. 6.12 Алгоритм эвристического поиска

беда. Однако мы не проводим такого рода анализ, поскольку существует очень высокая вероятность того, что наш эвристический отбор увенчается успехом и быстро приведет нас к нашей цели — есть вкусную клубнику.

Эвристические решения тесно связаны с необходимостью принимать решения в режиме реального времени с ограниченными ресурсами. На практике хорошее решение, найденное быстро, предпочтительнее оптимального, но очень дорогого решения.

Функция эвристической оценки f(s) для состояний используется для математического моделирования эвристики. Цель состоит в том, чтобы без особых усилий найти решение поставленной задачи поиска с минимальными общими затратами. Обратите внимание, что существует тонкая разница между усилиями по поиску решения и общей стоимостью этого решения. Например, Google Maps может потратить полсекунды усилий, чтобы найти маршрут от мэрии в Сан-Франциско до Туолумн-Медоуз в национальном парке Йосемити, но поездка из Сан-Франциско до Туолумн-Медоуз на машине может занять четыре часа и немного денег. на бензин и т.д. (общая стоимость).

Далее мы изменим алгоритм поиска в ширину, добавив в него функцию оценки. Текущие открытые узлы больше не расширяются слева направо по строке, а в соответствии с их эвристическим рейтингом. Из множества открытых узлов первым всегда расширяется узел с минимальным рейтингом. Это достигается за счет немедленной оценки узлов по мере их раскрытия и сортировки их в списке открытых узлов.

Затем список может содержать узлы с разной глубиной дерева.

Поскольку эвристическая оценка состояний очень важна для поиска, с этого момента мы будем различать состояния и связанные с ними узлы. Узел кон

содержит состояние и дополнительную информацию, относящуюся к поиску, такую как его глубина в дереве поиска и эвристический рейтинг состояния. В результате функция «Преемники», генерирующая преемников (потомков) узла, также должна сразу вычислить для этих узлов-преемников их эвристические рейтинги как составляющую каждого узла. Мы определяем общий алгоритм поиска HEURISTICSEARCH на рис. 6.12.

Список узлов инициализируется начальными узлами. Затем в цикле удаляется первый узел из списка и проверяется, является ли он узлом решения. Если нет, то он будет расширен функцией «Преемники» и ее преемники добавлены в список

Machine Translated by Google

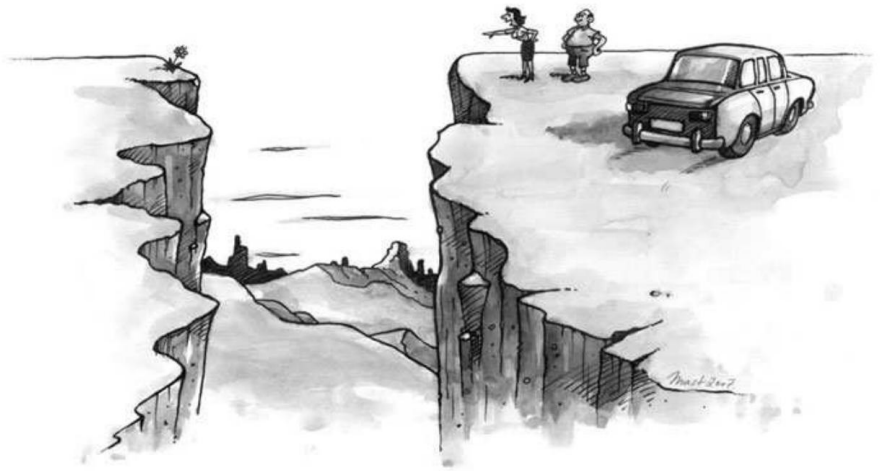


Рис. 6.13 Он: «Дорогой, подумай о стоимости топлива! Я сорву для тебя одну в другом месте. Она: «Нет, я хочу вон ту!»

с функцией «Сортировка». «SortIn(X,Y)» вставляет элементы из несортированного списка X в отсортированный по возрастанию список Y. В качестве ключа сортировки используется эвристический рейтинг. Таким образом гарантируется, что наилучший узел (т. е. узел с наименьшим эвристическим значением) всегда находится в начале списка. Поиск в глубину и в

ширину также является частным случаем функции HEURISTICSEARCH. Мы можем легко сгенерировать их, подключив соответствующую функцию оценки (упражнение 6.11 на стр. 111).

Лучшей эвристикой будет функция, которая вычисляет фактические затраты от каждого узла до цели. Однако для этого потребуется обход всего пространства поиска, что и должна предотвратить эвристика. Поэтому нам нужна эвристика, которую можно быстро и просто вычислить. Как найти такую эвристику?

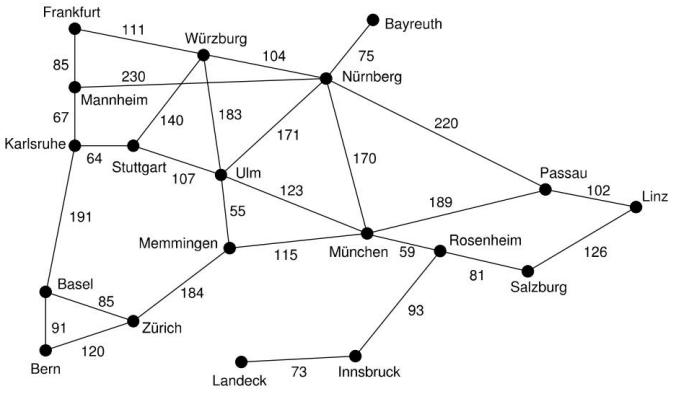
Интересной идеей поиска эвристики является упрощение задачи. Исходная задача настолько упрощена, что ее можно решить с небольшими вычислительными затратами. Затраты от состояния до цели в упрощенной задаче затем служат оценкой реальной задачи (см. рис. 6.13). Эту функцию оценки стоимости обозначим через h.

6.3.1 Жадный поиск

Представляется разумным выбрать состояние с наименьшим оценочным значением h (то есть с наименьшей оценочной стоимостью) из списка доступных на данный момент состояний.

3При сортировке нового узла из списка узлов может оказаться целесообразным проверить, доступен ли уже узел, и, если да, удалить дубликат.

Machine Translated by Google



Basel	204
Bayreuth	207
Bern	247
Frankfurt	215
Innsbruck	163
Karlsruhe	137
Landeck	143
Linz	318
München	120
Mannheim	164
Memmingen	47
Nürnberg	132
Passau	257
Rosenheim	168
Stuttgart	75
Salzburg	236
Würzburg	153
Zürich	157

Рис. 6.14 График города с указанием расстояний от всех городов до Ульма

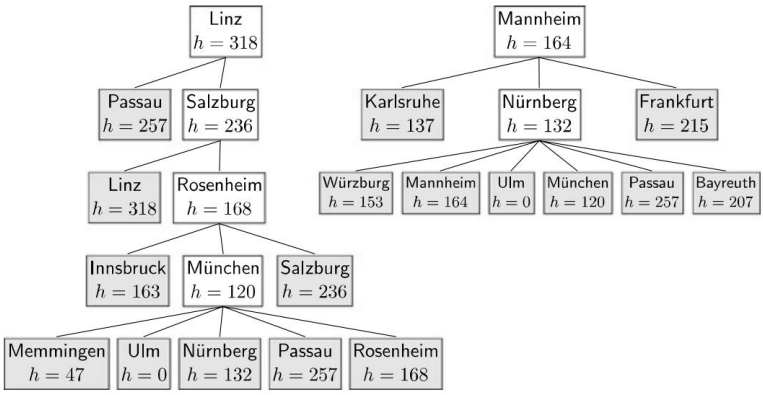
смету расходов затем можно использовать непосредственно в качестве функции оценки. Для вычисления в функции HEURISTICSEARCH положим f (s) = h(s). Это хорошо видно на примере планирования поездки (Пример 6.2 на стр. 88). Поставим задачу найти прямой путь из города в город (т. е. дальность полета) как упрощение задачи. Вместо поиска оптимального маршрута мы сначала определяем из каждого узла маршрут с минимальным расстоянием полета до цели. Мы выбираем Ульм в качестве пункта назначения. Таким образом, функция оценки стоимости становится

h(s) = расстояние полета от города s до Ульма.

Расстояния полета от всех городов до Ульма приведены на рис. 6.14 рядом с графиком.

Дерево поиска для старта в Линце представлено на рис. 6.15 на стр. 98 слева. Мы видим, что дерево очень стройное. Таким образом, поиск заканчивается быстро. К сожалению, этот поиск не всегда находит оптимальное решение. Например, этот алгоритм не может найти оптимальное решение при запуске в Мангейме (рис. 6.15 на стр. 98 справа). Путь Мангейм-Нюрнберг-Ульм имеет длину 401 км. Маршрут Мангейм-Карлсруэ-Штутгарт-Ульм будет значительно короче - 238 км. Когда мы наблюдаем за графиком, становится ясной причина этой проблемы. Нюрнберг на самом деле несколько ближе, чем Карлсруэ к Ульму, но расстояние от Мангейма до Нюрнберга значительно больше, чем от Мангейма до Карлсруэ. Эвристика только «жадно» смотрит вперед к цели вместо того, чтобы учитывать уже заложенную растяжку до текущего узла. Вот почему мы называем его жадным поиском.

Machine Translated by Google



Node	München	Innsbruck	Salzburg	Passau	Linz
heuristic rating	120	163	236	257	318

Рис. 6.15 Жадный поиск: из Линца в Ульм (слева) и из Мангейма в Ульм (справа). Структура данных списка узлов для левого дерева поиска, отсортированная по рейтингу узла до того, как задано расширение узла Mü nchen.

6.3.2 A -Поиск

Теперь мы хотим учесть затраты, которые накопились во время поиска до текущего узла s. Сначала определим функцию стоимости

g(s) = сумма начисленных затрат от начала до текущего узла,

затем добавьте к этому оценочную стоимость цели и получите в качестве эвристической функции оценки

$$f(s) = g(s) + h(s).$$

Теперь добавим еще одно небольшое, но важное требование.

Определение 6.4. Эвристическая функция оценки стоимости h(s), которая никогда не завышает фактическую стоимость от состояния s до цели, называется допустимой.

Функция ЭВРИСТИЧЕСКИЙПОИСК вместе с функцией вычисления f (s) = g(s) + h(s) и допустимой эвристической функцией h называется A -алгоритмом. Этот знаменитый алгоритм является полным и оптимальным. Таким образом, A всегда находит кратчайшее решение для каждой разрешимой задачи поиска. Мы объясним и докажем это в следующем обсуждении.

Сначала применим к примеру A -алгоритм. Мы ищем самый короткий путь из Франкфурта в Ульм.

Machine Translated by Google

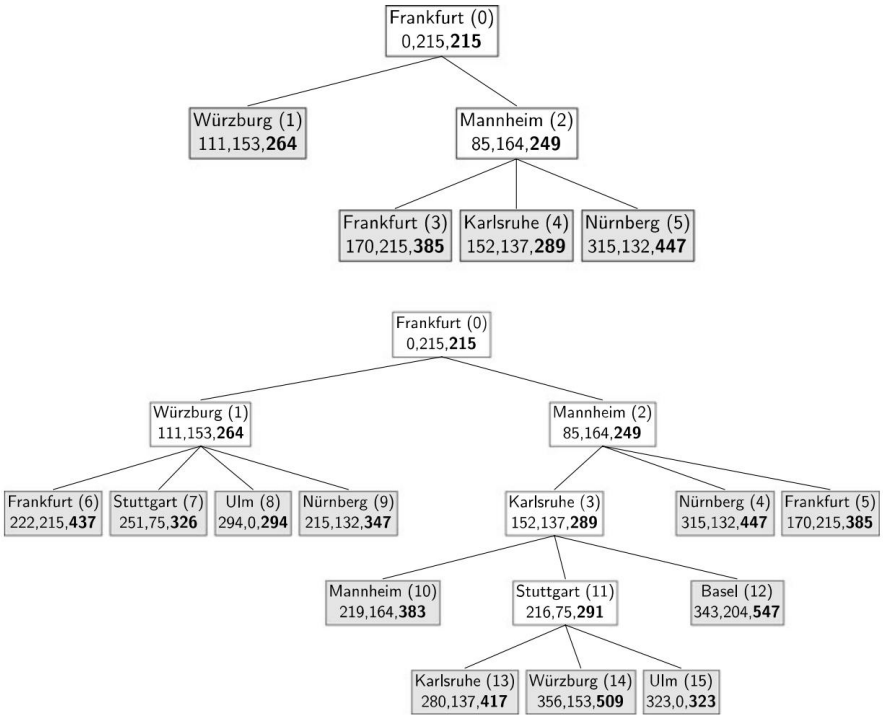


Рис. 6.16 Два снимка дерева поиска A для оптимального маршрута из Франкфурта в Ульм. В полях под названием города s мы показываем g(s), h(s), f (s). Цифры в скобках после названий городов показывают порядок, в котором узлы были сгенерированы функцией «Преемник».

В верхней части рис. 6.16 мы видим, что преемники Мангейма рождаются раньше наследников Вюрцбурга. Оптимальное решение Франкфурт-Вюрцбург-Ульм создается вскоре после этого на восьмом шаге, но еще не распознано как таковое.
Таким образом, алгоритм еще не завершается, потому что узел Карлсруэ (3) имеет лучшее (меньшее) значение f и, следовательно, опережает узел Ульм (8) в очереди. Только когда все значения f больше или равны значению узла решения Ulm (8), мы гарантируем, что имеем оптимальное решение. В противном случае потенциально может быть другое решение с меньшими затратами. Сейчас мы покажем, что это верно в общем случае.

Теорема 6.2. Алгоритм A оптимален. То есть он всегда находит решение с наименьшей общей стоимостью, если эвристика h допустима.

Доказательство. В алгоритме HEURISTICSEARCH каждый вновь сгенерированный узел s сортируется функцией «SortIn» в соответствии с его эвристическим рейтингом f(s). Таким образом, узел с наименьшим значением рейтинга находится в начале списка. Если узел l в точке

Machine Translated by Google

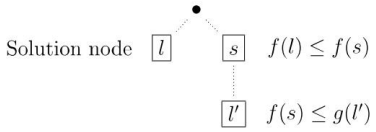


Рис. 6.17 Первый узел решения l, найденный A , никогда не имеет более высокой стоимости, чем другой произвольный́ узел l

начало списка является узлом решения, тогда никакой другой узел не имеет лучшего эвристического рейтинга. Для всех остальных узлов s верно, что f (l) f (s). Поскольку эвристика допустима, лучшего решения l найти не́льзя, даже после расширения всех остальных узлов (см. рис. 6.17). Написано официально:

g(l) = g(l) + h(l) = f(l) f(s) = g(s) + h(s) g(l′).

Первое равенство выполняется, потому что l является узлом решения с h(l) = 0. Второе равенство является определением f. Третье (не)равенство выполняется, потому что список открытых узлов отсортирован в порядке возрастания. Четвертое равенство снова является определением f . Наконец, последнее (не)равенство — это допустимость эвристики, которая никогда не завышает стоимость от вершины s до произвольного решения. Таким образом, было показано, что g(l) g(l′), т. е. найденное решение l является оптимальным.

6.3.3 IDA -Поиск

Поиск A наследует особенность поиска в ширину. Он должен сохранять много узлов в памяти, что может привести к очень высокому использованию памяти. Кроме того, список открытых узлов должен быть отсортирован. Таким образом, вставка узлов в список и удаление узлов из списка больше не могут выполняться за постоянное время, что немного увеличивает сложность алгоритма. На основе алгоритма пирамидальной сортировки мы можем структурировать список узлов как кучу с логарифмической временной сложностью для вставки и удаления узлов (см. [CLR90]).

Обе проблемы можно решить — аналогично поиску в ширину — путем итеративного углубления. Работаем с поиском в глубину и последовательно повышаем лимит. Однако вместо того, чтобы работать с пределом глубины, здесь мы используем предел для эвристической оценки f (s). Этот процесс называется IDA -алгоритмом.

6.3.4 Эмпирическое сравнение алгоритмов поиска

B A , или (альтернативно) ИДА , у нас есть алгоритм поиска со многими хорошими свойствами. Он полный и оптимальный. Таким образом, его можно использовать без риска. Самое главное, однако, заключается в том, что он работает с эвристикой и, следовательно, может значительно сократить время вычислений, необходимое для поиска решения. Мы хотели бы исследовать это эмпирически на примере головоломки с 8 элементами.

Для головоломки с восьмерками есть две простые допустимые эвристики. Эвристика h1 просто подсчитывает количество квадратов, которые находятся не на своем месте. Ясно, что эта эвристика допустима. Эвристика h2 измеряет манхэттенское расстояние. Для каждого

Machine Translated by Google

Таблица 6.2 Сравнение вычислительной стоимости неинформированного поиска и эвристического поиска решаемые задачи из 8 головоломок различной глубины. Измерения в шагах и секундах. Все значения средние значения для нескольких прогонов (см. последний столбец)

Глубина	Шаги итеративного углубления	Время [сек]	Алгоритм А				номер работает
			Эвристика h1		Эвристика h2		
			Шаги	Время [сек]	Шаги	Время [сек]	
2	20	0,003	3.0	0,0010	3.0	0,0010	10
4	81	0,013	5.2	0,0015	5,0	0,0022	24
6	806	0,13	10.2	0,0034	8.3	0,0039	19
8	6455	1,0	17,3	0,0060	12.2	0,0063	14
10	50512	7,9	48.1	0,018	22.1	0,011	15
12	486751	75,7	162,2	0,074	56,0	0,031	12
			ИДА				
14	-	-	10079.2	2,6	855,6	0,25	16
16	-	-	69386,6	19,0	3806,5	1,3	13
18	-	-	708780,0	161,6	53941,5	14,1	4

возвести в квадрат горизонтальные и вертикальные расстояния до местоположения этого квадрата в целевом состоянии складываются вместе. Затем это значение суммируется по всем квадратам. Например, Манхэттенское расстояние двух штатов



рассчитывается как

$$h2(s) = 1 + 1 + 1 + 1 + 2 + 0 + 3 + 1 = 10.$$

Допустимость манхэттенского расстояния также очевидна (см. упражнение 6.13 на стр. 111).

Описанные алгоритмы были реализованы в системе Mathematica. Для сравнения с неинформированным поиском, алгоритм А с двумя эвристиками h1 и h2 и итеративное углубление было применено к 132 случайно сгенерированным задачам из 8 головоломок. средние значения количества шагов и времени вычислений приведены в таблице 6.2. Мы видим, что эвристика значительно снижает стоимость поиска по сравнению с неинформативным поиском.

Если мы сравним итеративное углубление до А с h1 на глубине 12, например, это становится очевидным, что h1 уменьшает количество шагов примерно в 3000 раз, но время вычисления всего в 1023 раза. Это связано с более высокой стоимостью одного шаг для вычисления эвристики.

При ближайшем рассмотрении обнаруживается скачок количества шагов между глубиной 12 и глубина 14 в столбце для h1. Этот скачок нельзя объяснить только повторяющимся работа, выполненная IDA· Это происходит потому, что реализация алгоритма А

Machine Translated by Google

удаляет дубликаты идентичных узлов и тем самым сужает пространство поиска. Это невозможно с IDA , потому что он почти не сохраняет узлы. Несмотря на это, A больше не может конкурировать с IDA за пределами глубины 14, потому что стоимость сортировки в новых узлах сильно увеличивает время на шаг.

Вычисление эффективного коэффициента ветвления в соответствии с (6.1) на стр. 87 дает значение около 2,8 для неинформированного поиска. Это число согласуется со значением из разд. 6.1. Эвристика h1 снижает коэффициент ветвления примерно до 1,5, а h2 примерно до 1,3. Из таблицы видно, что небольшое уменьшение коэффициента ветвления с 1,5 до 1,3 дает нам большое преимущество во времени вычислений.

Таким образом, эвристический поиск имеет важное практическое значение, поскольку он может решать проблемы, недоступные для неосведомленного поиска.

6.3.5 Резюме

Из различных алгоритмов поиска для неинформированного поиска итеративное углубление является единственным практичным, потому что он завершен и может обойтись очень небольшим объемом памяти. Однако для сложных задач комбинаторного поиска даже итеративное углубление обычно не удается из-за размера пространства поиска. Эвристический поиск помогает здесь за счет уменьшения эффективного фактора ветвления. Алгоритм IDA , как и итеративное углубление, является полным и требует очень мало памяти.

Эвристика, естественно, дает значительное преимущество только в том случае, если эвристика «хорошая». При решении сложных поисковых задач актуальная задача разработчика состоит в построении эвристик, значительно снижающих эффективный коэффициент ветвления. В разд. В разделе 6.5 мы рассмотрим эту проблему, а также покажем, как можно использовать методы машинного обучения для автоматического создания эвристик.

В заключение остается отметить, что эвристики не имеют преимущества в производительности для неразрешимых задач, потому что неразрешимость проблемы может быть установлена только после того, как будет проведен поиск по всему дереву поиска. Для решаемых задач, таких как головоломка с восьмеркой, это означает, что все дерево поиска должно быть пройдено до максимальной глубины независимо от того, используется ли эвристика или нет. В этом случае эвристика всегда является недостатком, связанным с вычислительными затратами на оценку эвристики. Этот недостаток обычно можно оценить постоянным коэффициентом, не зависящим от размера задачи. Для неразрешимых задач, таких как доказательство формул PL1, дерево поиска может быть бесконечно глубоким. Это означает, что в неразрешимом случае поиск потенциально никогда не заканчивается. Подводя итог, можно сказать следующее: для решаемых задач эвристики часто резко сокращают время вычислений, но для неразрешимых задач стоимость с эвристиками может быть даже выше.

6.4 Игры с соперниками

Игры для двух игроков, такие как шахматы, шашки, Отелло и Го, являются детерминированными, потому что каждое действие (ход) приводит к одному и тому же дочернему состоянию при одном и том же родительском состоянии. Напротив, нарды недетерминированы, потому что их дочернее состояние зависит от

Machine Translated by Google

на результат броска игральной кости. Все эти игры можно наблюдать, потому что каждый игрок всегда знает полное состояние игры. Многие карточные игры, такие как покер, например, поддаются наблюдению лишь частично, потому что игрок не знает карты других игроков или знает о них лишь частично.

Проблемы, обсуждавшиеся до сих пор в этой главе, были детерминированными и наблюдаемыми. Далее мы рассмотрим игры, которые также являются детерминированными и наблюдаемыми. Кроме того, мы ограничимся играми с нулевой суммой. Это игры, в которых каждый выигрыш одного игрока означает проигрыш того же значения для противника. Сумма выигрыша и убытка всегда равна нулю. Это относится к упомянутым выше играм в шахматы, шашки, Отелло и Го.

6.4.1 Минимаксный поиск

Цель каждого игрока — совершать оптимальные ходы, ведущие к победе. В принципе, можно построить дерево поиска и полностью искать в нем (как в головоломке с восьмерками) серию ходов, которые приведут к победе. Однако есть несколько особенностей, на которые следует обратить внимание: 1. Эффективный фактор ветвления
в шахматах составляет примерно 30–35. В типичной игре с 50 ходами на игрока дерево поиска имеет более 30100 10148 листовых узлов .
Таким образом, нет возможности полностью изучить дерево поиска. Кроме того, в шахматы часто играют с ограничением по времени. Из-за этого требования к реальному времени поиск должен быть ограничен соответствующей глубиной дерева, например восемью полуходами.
Поскольку среди листовых узлов этого дерева с ограниченной глубиной обычно нет узлов-решений (т. е. узлов, завершающих игру), используется эвристическая функция оценки B для позиций на доске. Уровень игры программы сильно зависит от качества этой функции оценки.
Поэтому мы продолжим рассмотрение этого предмета в разд. 6.5.

2. Далее мы будем называть игрока, чью игру мы хотим оптимизировать, Максом, а его противника Мин. Ходы противника (Мина) заранее неизвестны, и, следовательно, фактическое дерево поиска также не известно. Эту проблему можно элегантно решить, если предположить, что противник всегда делает лучший ход, на который он способен. Чем выше оценка B(s) для позиции s, тем лучше позиция s для игрока Max и тем хуже для его противника Min. Макс пытается максимизировать оценку своих ходов, в то время как Мин делает ходы, которые приводят к как можно более низкой оценке.

Дерево поиска с четырьмя полуходами и оценками всех листьев показано на рис. 6.18 на стр. 104. Оценка внутреннего узла выводится рекурсивно как максимум или минимум его дочерних узлов, в зависимости от уровня узла.

6.4.2 Альфа-бета-обрезка

Переключаясь между максимизацией и минимизацией, мы можем сэкономить много работы в некоторых обстоятельствах. Альфа-бета-отсечение работает с поиском в глубину до заданного предела глубины. Таким образом, дерево поиска просматривается слева направо.

Machine Translated by Google

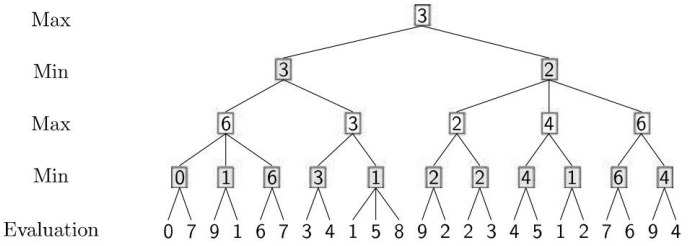


Рис. 6.18 Минимаксное игровое дерево с опережением на четыре полухода

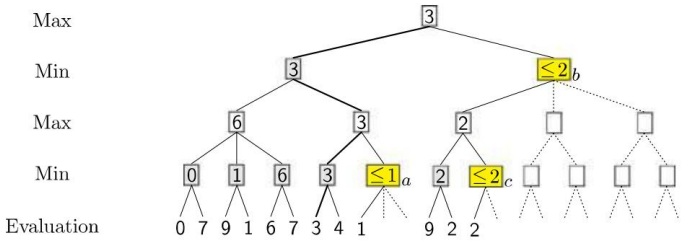


Рис. 6.19. Дерево игры альфа-бета с опережением на четыре полухода. Пунктирные части дерева не просматриваются, потому что они не влияют на конечный результат.

направо. Как и в минимаксном поиске, в минимальных узлах минимум генерируется из минимального значения последующих узлов, а в максимальных узлах аналогично.

максимум. На рис. 6.19 этот процесс изображен для дерева с рис. 6.18. В узле, отмеченном a, все остальные потомки могут быть проигнорированы после того, как первый потомок будет оценен как значение 1, потому что минимум обязательно будет 1. Он может даже стать еще меньше, но это не имеет значения, так как максимум уже 3 на один уровень выше. Независимо от того, как сложится оценка оставшихся преемников, максимум сохранит значение 3. Аналогично дерево будет обрезано в узле b. Поскольку первый потомок b имеет значение 2, минимальное значение, которое должно быть сгенерировано для b, может быть только меньше или равно 2. Но максимум в корневом узле уже наверняка будет 3.
Это не может быть изменено значениями 2. Таким образом, оставшиеся поддеревья b могут быть обрезаны.

То же самое относится и к узлу c. Однако соответствующий максимальный узел является не прямым родителем, а корневым узлом. Это можно обобщить. • В каждом конечном узле вычисляется оценка. • Для каждого максимального узла текущее наибольшее дочернее значение сохраняется в α. • Для каждого минимального узла текущее наименьшее дочернее значение сохраняется в β. • Если в минимальном узле k текущее значение β α, то поиск по k можно закончить.

Здесь α — наибольшее значение максимального узла на пути от корня до k. • Если в максимальном узле l текущее значение α β, то поиск по l можно закончить.

Здесь β — наименьшее значение минимального узла на пути от корня до l.

Machine Translated by Google


```
ALPHABETAMAX(Узел, α, β)

Если DepthLimitReached(Node) Return(Rating(Node))
Новые узлы = преемники (узел)
В то время как NewNodes =
    α = Максимум (α, ALPHABETAMIN (Первый (Новые узлы), α, β))
    Если α  β Возврат (β)
    Новые узлы = остальные (новые узлы)
Возврат(α)
```

```
АЛФАБЕТАМИН(Узел, α, β)

Если DepthLimitReached(Node) Return(Rating(Node))
Новые узлы = преемники (узел)
В то время как NewNodes =
    β = Минимум (β, ALPHABETAMAX (Первый (Новые узлы), α, β))
    Если β  α Возврат (α)
    Новые узлы = остальные (новые узлы)
Возврат (β)
```

Рис. 6.20 Алгоритм альфа-бета-поиска с двумя функциями ALPHABETAMIN и AL PHABETAMAX

Алгоритм, представленный на рис. 6.20 , является расширением поиска в глубину с двумя функциями, которые вызываются попеременно. Он использует значения, определенные выше для α и β.

Сложность Время вычислений, сэкономленное альфа-бета-отсечением, сильно зависит от порядка обхода дочерних узлов. В худшем случае сокращение альфа-бета не дает никаких преимуществ. Для постоянного коэффициента ветвления b количество nd конечных узлов, которые необходимо оценить на глубине d, равно

$$n = b^d.$$

В лучшем случае, когда потомки максимальных узлов сортируются по убыванию, а потомки минимальных узлов сортируются по возрастанию, эффективный коэффициент ветвления уменьшается до $\sqrt{b}$. В шахматах это означает существенное снижение эффективного фактора ветвления с 35 до примерно 6. Тогда только

$$n = b^{\sqrt{d}} = b^{d/2}$$

Machine Translated by Google

будут созданы листовые узлы. Это означает, что предел глубины и, следовательно, горизонт поиска удваиваются при сокращении альфа-бета. Однако это верно только в случае оптимально отсортированных потомков, потому что рейтинги дочерних узлов неизвестны на момент их создания. Если дочерние узлы отсортированы случайным образом, то коэффициент ветвления уменьшается до b 3/4 , а количество конечных узлов до

$$n = b^{\frac{3}{4}d} .$$

При той же вычислительной мощности шахматный компьютер, использующий сокращение альфа-бета, может, например, вычислить восемь полуходов вперед вместо шести с эффективным коэффициентом ветвления около 14. Тщательный анализ с выводом этих параметров можно найти в [Горох84].

Чтобы удвоить глубину поиска, как упоминалось выше, нам нужно, чтобы дочерние узлы были оптимально упорядочены, что на практике не так. В противном случае поиск был бы ненужным. С помощью простого трюка мы можем получить относительно хороший порядок узлов. Мы связываем альфа-бета-обрезку с итеративным углублением сверх предела глубины. Таким образом, на каждом новом пределе глубины мы можем получить доступ к рейтингам всех узлов предыдущих уровней и упорядочить преемников в каждой ветви. Таким образом, мы достигаем эффективного коэффициента ветвления примерно от 7 до 8, что недалеко от теоретического оптимума 35 [Nil98].

6.4.3 Недетерминированные игры

Минимаксный поиск можно обобщить на все игры с недетерминированными действиями, такие как нарды. Каждый игрок делает бросок перед своим ходом, на который влияет результат броска кубиков. Таким образом, в дереве игры теперь есть три типа уровней в последовательности.

Макс, кости, Мин, кости, . . . ,

где каждый узел броска костей разветвляется шестью способами. Поскольку мы не можем предсказать значение кубика, мы усредняем значения всех бросков и проводим поиск, как описано, со средними значениями из [RN10].

6,5 Эвристические функции оценки

Как нам найти хорошую эвристическую функцию оценки для задачи поиска?
Здесь принципиально два подхода. Классический способ использует знания людей-экспертов. Перед инженером по знаниям обычно стоит сложная задача формализовать неявные знания эксперта в виде компьютерной программы.
Теперь мы хотим показать, как можно упростить этот процесс на примере шахматной программы.

На первом этапе экспертов спрашивают о наиболее важных факторах при выборе хода. Затем предпринимается попытка количественной оценки этих факторов. Мы получаем список соответствующих функций или атрибутов. Затем они (в простейшем случае) объединяются в

Machine Translated by Google

линейная функция оценки B(s) для позиций, которая может выглядеть так:

$$B(s) = a1 \cdot \text{материал} + a2 \cdot \text{пешечная_структура} + a3 \cdot \text{безопасность короля} + a4 \cdot \text{конь_в_центре} + a5 \cdot \text{слон_диагональное_покрытие} + \cdots, \quad (6.3)$$

где «материал» является наиболее важной характеристикой и рассчитывается по

$$\text{материал} = \text{материал (собственная_команда)} — \text{материал (противник)}$$

с

$$\begin{aligned}\text{материал(команда)} = {} & \text{количество_пешек(команда)} \cdot 100 + \text{количество_рыцарей(команда)} \cdot 300 \\ & + \text{количество_слонов(команда)} \cdot 300 + \text{количество_ладей(команда)} \cdot 500 \\ & + \text{num_queens(команда)} \cdot 900\end{aligned}$$

Почти все шахматные программы делают одинаковую оценку материала. Тем не менее, есть большие различия для всех других функций, которые мы не будем здесь рассматривать [Fra05, Lar00].

На следующем шаге необходимо определить веса ai всех признаков. Они устанавливаются интуитивно после обсуждения с экспертами, а затем меняются после каждой игры на основе положительного и отрицательного опыта. Тот факт, что этот процесс оптимизации очень дорог и, кроме того, линейная комбинация признаков очень ограничена, предполагает использование машинного обучения.

6.5.1 Изучение эвристики

Теперь мы хотим автоматически оптимизировать веса ai оценочной функции B(s) из (6.3). При таком подходе эксперта спрашивают только о соответствующих функциях f1(s), . . . , fn(s) для игрового состояния s. Затем используется процесс машинного обучения с целью найти функцию оценки, максимально близкую к оптимальной. Мы начинаем с первоначально заданной функции оценки (определяемой в процессе обучения), а затем позволяем шахматной программе играть. В конце игры по результату (победа, поражение или ничья) выводится рейтинг. На основании этого рейтинга функция оценки изменяется с целью в следующий раз делать меньше ошибок. В принципе, то же самое, что делает разработчик, теперь автоматически выполняется в процессе обучения.

Как бы просто это ни звучало, на практике это очень сложно. Центральная проблема улучшения рейтинга позиций на основе выигранных или проигранных матчей известна сегодня как проблема присвоения кредитов . На самом деле у нас есть рейтинг в конце игры, но нет рейтинга для отдельных ходов. Таким образом, агент выполняет множество действий, но до самого конца не получает ни положительной, ни отрицательной обратной связи. Как тогда он должен соотносить эту обратную связь со многими действиями, предпринятыми в прошлом? И как ему улучшить свои действия в этом случае? Такими вопросами занимается захватывающая молодая область обучения с подкреплением (см. раздел 10).

Machine Translated by Google

Большинство лучших шахматных компьютеров в мире сегодня все еще работают без методов машинного обучения. Этому есть две причины. С одной стороны, алгоритмы обучения с подкреплением, разработанные до сих пор, требуют большого времени вычислений при больших пространствах состояний. С другой стороны, созданные вручную эвристики высокопроизводительных шахматных компьютеров уже сильно оптимизированы. Это означает, что только очень хорошая система обучения может привести к улучшениям. Вероятно, в ближайшие десять лет наступит время, когда обучающийся компьютер станет чемпионом мира.

6.6 Современное состояние

Для оценки качества процессов эвристического поиска я хотел бы повторить
Определение Элейн Рич [Ric83]:

> Искусственный интеллект — это наука о том, как заставить компьютеры делать то, в чем люди на данный момент лучше.

Едва ли найдется лучший тест для определения интеллекта компьютерной программы, чем прямое сравнение компьютера и человека в такой игре, как шахматы, шашки, нарды или го.

В 1950 году Клод Шеннон, Конрад Цузе и Джон фон Нейман представили первые шахматные программы, которые, однако, либо не могли быть реализованы, либо требовали много времени для реализации. Всего несколько лет спустя, в 1955 году, Артур Сэмюэл написал программу, которая играла в шашки и могла улучшать собственные параметры посредством простого процесса обучения. Для этого он использовал первый программируемый логический компьютер IBM 701. Однако по сравнению с современными шахматными компьютерами он имел доступ к большому количеству архивных партий, в которых каждый ход оценивался экспертами. Таким образом, программа улучшила свою функцию оценки. Чтобы добиться дальнейших улучшений, Сэмюэл заставил свою программу играть против самой себя. Он решил проблему присвоения кредита простым способом. Для каждой отдельной позиции во время игры он сравнивает оценку функции B(s) с оценкой, вычисленной с помощью альфа-бета-сокращения, и соответствующим образом изменяет B(s). В 1961 году его шашечная программа обыграла четвертого лучшего игрока в шашки в США. Благодаря этой новаторской работе Сэмюэл, несомненно, опередил свое время почти на 30 лет.

Только в начале девяностых, когда появилось обучение с подкреплением, Джеральд Терсауро создал обучающую программу игры в нарды под названием TD-Gammon, которая играла на уровне чемпионов мира (см. раздел 10).

Сегодня существует несколько шахматных программ, которые играют на уровне гроссмейстера, и некоторые из них продаются на коммерческой основе для ПК. Прорыв произошел в 1997 году, когда Deep Blue от IBM победил чемпиона мира по шахматам Гэри Каспарова со счетом 3,5 партии против 2,5. Deep Blue мог в среднем рассчитывать на 12 полуходов вперед с альфа-бета-сокращением и эвристической оценкой позиции.

Одним из самых мощных шахматных компьютеров на сегодняшний день является Hydra, параллельный компьютер, принадлежащий компании из Объединенных Арабских Эмиратов. Программное обеспечение было разработано учеными Кристианом Доннингером (Австрия) и Ульфом Лоренцем (Германия), а также великим чемпионом Германии по шахматам Кристофером Лутцем. Hydra использует 64 параллельных Xeon

Machine Translated by Google

процессоры с вычислительной мощностью около 3 ГГц и 1 ГБ памяти каждый. Для функции оценки положения каждый процессор имеет сопроцессор FPGA (программируемая пользователем вентильная матрица). Таким образом становится возможным оценивать 200 миллионов позиций в секунду даже при использовании дорогостоящей функции оценки.

С помощью этой технологии Hydra может в среднем рассчитывать примерно на 18 ходов вперед. В особых, критических ситуациях горизонт поиска может растянуться даже до 40 полуходов. Ясно, что такой горизонт выходит за рамки того, что могут сделать даже великие чемпионы, поскольку Гидра часто делает ходы, которые великие чемпионы не могут понять, но которые в конце концов приводят к победе. В 2005 году Гидра победила гроссмейстера Майкла Адамса, занявшего седьмое место, со счетом 5,5–0,5 гейма.

Hydra использует немного специальных знаний о шахматах из учебников, а скорее альфа-бета-поиск с относительно общими, хорошо известными эвристиками и хорошей оценкой позиции, закодированной вручную. В частности, Гидра не способна к обучению. Доработки проводятся между играми разработчиками. Таким образом, компьютер по-прежнему освобождается от необходимости учиться. Hydra также не имеет специальных алгоритмов планирования. Тот факт, что Hydra работает без обучения, указывает на то, что, несмотря на многие достижения, исследования в области машинного обучения по-прежнему необходимы. Как упоминалось выше, мы более подробно остановимся на этом в разд. 10 и гл. 8.

В 2009 году система Pocket Fritz 4, работающая на КПК, выиграла шахматный турнир Copa Mercosur в Буэнос-Айресе с девятью победами и одной ничьей против 10 отличных шахматистов, трое из которых были гроссмейстерами. Несмотря на то, что информации о внутренней структуре программного обеспечения не так много, эта шахматная машина представляет собой тенденцию от чистой вычислительной мощности к большей интеллектуальности. Эта машина играет на уровне гроссмейстера и сравнима, если не лучше, с Гидрой.
По словам разработчика Pocket Fritz Станислава Цукрова [Wik10], Pocket Fritz со своим шахматным поисковиком HIARCS 13 ищет менее 20 000 позиций в секунду, что медленнее Hydra примерно в 10 000 раз. Это приводит к выводу, что HIARCS 13 определенно использует лучшую эвристику для уменьшения эффективного фактора ветвления, чем Hydra, и поэтому его можно назвать более интеллектуальным, чем Hydra. Кстати, HIARCS — это сокращение от Higher Intelligence Auto Response Chess System.

Даже если вскоре ни у одного человека не будет шансов против лучших шахматных компьютеров, перед ИИ еще много вызовов. Иди, например. В этой старой японской игре, играемой на квадратной доске с 361 квадратом, 181 белым и 180 черными камнями, эффективный коэффициент ветвления составляет около 300. После четырех полуходов уже имеется около 8×109 позиций . Все известные классические процессы поиска по дереву игр не имеют шансов против хороших игроков в го на этом уровне сложности. Эксперты сходятся во мнении, что здесь нужны «по-настоящему интеллектуальные» системы. Комбинаторное перечисление всех возможностей — неверный метод. Гораздо больше нужны процессы, способные распознавать закономерности на доске, следить за долгосрочными разработками и быстро принимать «интуитивные» решения. Как и в распознавании объектов на сложных изображениях, люди все еще намного опережают компьютерные программы. Мы обрабатываем изображение в целом очень параллельно, в то время как компьютер обрабатывает миллионы пикселей один за другим и с трудом улавливает, что существенного в полноте пикселей. Программа «Многоликая игра в го» умеет распознавать 1100 различных паттернов и знает 200 игровых стратегий. Но все программы Go по-прежнему имеют серьезные проблемы с распознаванием того, является ли группа камней живой или мертвой, или где их сгруппировать между ними.

Machine Translated by Google

6.7 Упражнения

Упражнение 6.1

(a) Докажите теорему 6.1 на стр. 88, другими словами, докажите, что для дерева с большим постоянным коэффициентом ветвления b почти все узлы находятся на последнем уровне на глубине d. (b) Покажите, что это не всегда так, когда эффективный фактор ветвления велик и непостоянен.

Упражнение 6.2

(a) Вычислите средний фактор ветвления для головоломки с восьмерками без проверки циклов. Средний коэффициент ветвления — это коэффициент ветвления, который будет иметь дерево с равным количеством узлов на последнем уровне, постоянным коэффициентом ветвления и равной глубиной. (b) Рассчитайте средний коэффициент ветвления для головоломки с 8 элементами для неинформированного поиска, избегая при этом циклов длины 2.

Упражнение 6.3

(a) В чем разница между средним и эффективным коэффициентом ветвления (Определение 6.2 на стр. 87)?

(b) Почему эффективный коэффициент ветвления лучше подходит для анализа и сравнения времени вычислений алгоритмов поиска, чем средний коэффициент ветвления? (c) Покажите, что для сильно разветвленного дерева с n узлами и глубиной d эффективный коэффициент ветвления $\bar{b}$ примерно равен среднему коэффициенту ветвления и, таким образом, равен $\sqrt[d]{n}$.

Упражнение 6.4

(a) Вычислите размер пространства состояний для 8-головоломки, для аналогичной 3-головоломки (2 × 2-матрица), а также для 15-головоломки (4 × 4-матрица). (b) Докажите, что граф состояний, состоящий из состояний (узлов) и действий (ребер) для 3-головоломки, распадается на два связных подграфа, между которыми нет связей.

Упражнение 6.5 С помощью поиска в ширину для головоломки с восьмерками найдите путь (вручную) из начального узла  к целевому узлу  .

Упражнение 6.6

(a) Запрограммируйте поиск в ширину, поиск в глубину и итеративное углубление на выбранном вами языке и проверьте их на примере с 8 головоломками. (b) Почему нет особого смысла использовать поиск в глубину для головоломки с восьмерками?

Упражнение 6.7

(a) Покажите, что поиск в ширину при неизменной стоимости всех действий гарантирован. найти кратчайшее решение.

Machine Translated by Google

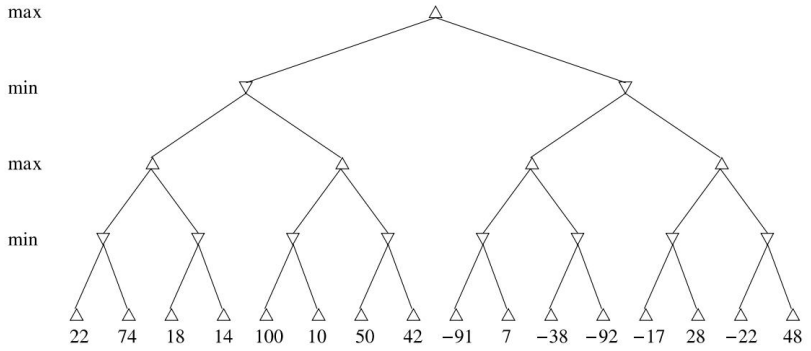


Рис. 6.21 Минимаксное дерево поиска

б) Покажите, что это не так для переменных издержек.

Упражнение 6.8. Используя поиск A для головоломки с восьмерками, найдите (вручную) путь из начальный узел  к целевому узлу 

(a) с использованием эвристики h1 (раздел 6.3.4).

(b) с использованием эвристики h2 (раздел 6.3.4).

Упражнение 6.9 Постройте дерево поиска A для графа города на рис. 6.14 на стр. 97 и используйте расстояние полета до Ульма в качестве эвристики. Начните в Берне с Ульма в качестве пункта назначения. Позаботьтесь о том, чтобы каждый город появлялся только один раз на пути.

Упражнение 6.10 Программа A поиск на выбранном вами языке программирования используя эвристики h1 и h2 , и протестируйте их на примере с 8 головоломками.

Упражнение 6.11. Приведите эвристическую оценочную функцию для состояний, с которыми HEURIS TICSEARCH может быть реализован как поиск в глубину и как поиск в ширину.

Упражнение 6.12 Какая связь между изображением пары в каньоне на рис. 6.13 на стр. 96 и допустимой эвристикой?

Упражнение 6.13. Покажите, что эвристики h1 и h2 для головоломки с восьмерками из разд. 6.3.4 .

Упражнение 6.14

(a) Дерево поиска для игры двух игроков показано на рис. 6.21 с рейтингами всех листовых узлов. Используйте минимаксный поиск с обрезкой α-β слева направо. Вычеркните все непосещенные узлы и дайте оптимальную результирующую оценку для каждого внутреннего узла. Отметьте выбранный путь. (b) Проверьте себя с помощью апплета П. Уинстона [Win].

Machine Translated by Google

Machine Translated by Google

Для решения этой проблемы разработаны различные формы немонотонной логики, позволяющие удалять знания (формулы) из базы знаний.
Под логикой по умолчанию была разработана логика, позволяющая назначать объектам атрибуты, действительные до тех пор, пока не доступны другие правила. В примере с Твити таким правилом по умолчанию будет правило, что птицы могут летать . Несмотря на большие усилия, эта логика в настоящее время из-за семантических и практических проблем не увенчалась успехом.

Монотонность может быть особенно неудобной в сложных задачах планирования, в которых мир может измениться. Если, например, синий дом покрасить в красный цвет, то впоследствии он станет красным. База знаний, например

цвет(дом, синий)
краска(дом,красный)
краска(x, y) цвет(x, y)

приводит к выводу, что после покраски дом красно-синий. Проблема, возникающая здесь при планировании, известна как проблема фрейма. Решением этой проблемы является ситуационное исчисление, представленное в разд. 5.6.

Интересным подходом к моделированию таких проблем, как пример Твити, является теория вероятностей. Утверждение «все птицы умеют летать» неверно. Утверждение вроде «почти все птицы умеют летать» верно. Это утверждение станет более точным, если мы дадим вероятность того, что «птицы могут летать». Это приводит к вероятностной логике, которая сегодня представляет собой важную область ИИ и важный инструмент для моделирования неопределенностей (см. главу 7).

4.4 Моделирование неопределенности

Двузначная логика может и должна моделировать только те обстоятельства, в которых есть истинное, ложное и нет других истинностных значений. Поэтому для многих задач повседневного рассуждения двузначная логика недостаточно выразительна. Правило

птица(x) летать(x)

верно почти для всех птиц, но для некоторых неверно. Как уже упоминалось, работа с вероятностями позволяет точно сформулировать неопределенность. Утверждение «99% всех птиц умеют летать» можно формализовать выражением

P (птица (x) летать (x)) = 0,99.

В гл. 7 мы увидим, что здесь лучше работать с условными вероятностями типа

P (муха |птица) = 0,99.

Machine Translated by Google

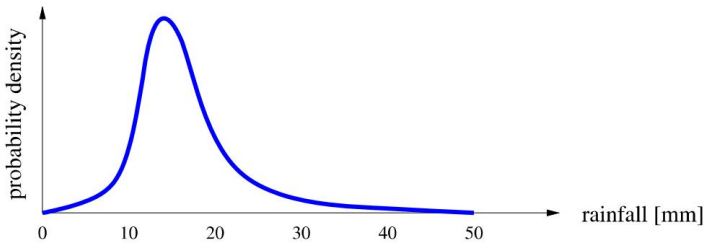


Рис. 4.3 Плотность вероятности непрерывных переменных осадков

С помощью байесовских сетей сложные приложения со многими переменными могут также быть смоделированы.

Для утверждения «Погода хорошая» нужна другая модель . Здесь это часто нет смысла говорить в категориях истинного и ложного. Переменная weather_is_nice не должны моделироваться как бинарные, а непрерывно со значениями, например, в интервал [0, 1]. Weather_is_nice = 0.7 означает «Погода довольно хорошая». Нечеткая логика, которая также будет представлена в гл. 7, был разработан для этого типа непрерывная (нечеткая) переменная.

Теория вероятностей также предлагает возможность делать утверждения о вероятности непрерывных переменных. Утверждение в метеосводке, такое как «Есть высокая вероятность того, что будет дождь» можно было бы, например, точно сформулировать как плотность вероятности вида

$$P(\text{осадки} = X) = Y$$

и изображается графически примерно так, как на рис. 4.3.

Это очень общее и даже наглядное представление обоих типов неопределенности, которые мы обсуждали, вместе с индуктивной статистикой и теорией байесовских сетей позволяет, в принципе, ответить на произвольные вероятностные запросы.

Теория вероятностей, а также нечеткая логика не могут быть напрямую сопоставлены с предикатом. логики, потому что они не допускают переменных или квантификаторов. Таким образом, их можно рассматривать как расширения пропозициональной логики, как показано в следующей таблице.

Формализм	Количество значения истинности	Вероятности выразимый
Логика высказываний	2	-
Нечеткая логика		-
Дискретная вероятностная логика	н	да
Непрерывная вероятностная логика		да

Machine Translated by Google

4,5 Упражнения

Упражнение 4.1

(a) С помощью следующего (ложного) аргумента можно было бы утверждать, что PL1 разрешима: мы берем полное исчисление вывода для PL1. С его помощью мы можем найти доказательство любой истинной формулы за конечное время. Для любой другой формулы φ я поступаю следующим образом: я применяю исчисление к ¬φ и показываю, что ¬φ истинно. Таким образом, φ ложно. Таким образом, я могу доказать или опровергнуть любую формулу в PL1. Найдите ошибку в аргументе и измените его,

чтобы он стал правильным. (b) Построить процесс принятия решения для множества истинных и невыполнимых формул ПЛ1.

Упражнение 4.2

(a) Учитывая утверждение «Есть парикмахер, который бреет каждого, кто не бреется сам». Подумайте, бреется ли этот цирюльник. (б) Пусть M = {x |x / x}. Опишите это множество

и посмотрите, содержит ли M себя.

Упражнение 4.3. Используйте автоматизированное средство доказательства теорем (например, E [Sch02]) и примените его ко всем пяти различным аксиоматизациям примера Твити из разд. 4.3. Подтвердите утверждения примера.

Machine Translated by Google

Machine Translated by Google

Логическое программирование с PROLOG

5

По сравнению с классическими языками программирования, такими как C или Pascal, Logic делает его можно выразить отношения элегантно, компактно и декларативно. Автоматизированный доказывающие теоремы способны даже решить, является ли база знаний логически влечет за собой запрос. Расчет доказательств и знания, хранящиеся в базе знаний, строго разделены. В качестве входных данных может использоваться формула, описанная в нормальной форме предложения. данные для любого средства доказательства теорем, независимо от используемого исчисления доказательств. это здорово значение для рассуждения и представления знаний.

Если кто-то хочет реализовать алгоритмы, которые неизбежно имеют процедурные компоненты, чисто декларативного описания часто бывает недостаточно. Роберт Ковальски, один из пионеров логического программирования, сделал это с формулой

Алгоритм = Логика + Управление.

Эта идея была реализована в языке ПРОЛОГ. ПРОЛОГ используется в много проектов, в первую очередь в области ИИ и компьютерной лингвистики. Сейчас мы дадим краткое введение в этот язык, представить наиболее важные понятия, показать его сильные стороны и сравнить его с другими языками программирования и средствами доказательства теорем. Те, кто ищет полный курс программирования, могут обратиться к таким учебникам, как [Bra11, CM94] и справочники [Wie04, Dia04].

Синтаксис языка PROLOG допускает только предложения Horn. Логическая запись и синтаксис PROLOG сопоставлены в следующей таблице:

PL1 / нормальная форма предложения	PROLOG	Описание
(¬A1 ⋯ ¬Am) B	B :- A1, ... , Am.	Правило
(A1 ⋯ Am) Б	B: -A1, ..., Am.	Правило
A	A.	Факт
(¬A1 ⋯ ¬Am)	?- A1, ..., Am.	Запрос
¬(A1 ⋯ Am)	?- A1, ..., Am.	Запрос

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по информатике,
DOI 10.1007/978-0-85729-299-5_5, © Springer-Verlag London Limited 2011

Machine Translated by Google


```
1 ребенок(Оскар,Карен,Фрэнк). 2
ребенка(Мэри,Карен,Фрэнк). 3
ребенка(Ева,Энн,Оскар). 4
ребенка(Генри,Энн,Оскар). 5
детей(Изольда,Энн,Оскар). 6 ребенок
(Клайд, Мэри, Оскарб). 7 8 дочерний(X,Z,Y):-

дочерний(X,Y,Z). 9

10 потомок(X,Y):- ребенок(X,Y,Z). 11 потомок(X,Y):-
потомок(X,U,V), потомок(U,Y).
```

Рис. 5.1 Программа PROLOG с семейными отношениями

Здесь A1,...,Am,A,B — литералы. Литералы, как и в PL1, строятся из предикатных символов с терминами в качестве аргументов. Как видно из приведенной выше таблицы, в PROLOG нет отрицаний в строгом логическом смысле, поскольку знак литерала определяется его позицией в предложении.

5.1 Системы и реализации PROLOG

Обзор современных систем PROLOG доступен по набору ссылок на домашней странице этой книги. Читателю мы рекомендуем очень мощные и свободно доступные (под публичными лицензиями GNU) системы GNU-PROLOG [Dia04] и SWI-PROLOG. В следующих примерах использовался SWI-PROLOG [Wie04].

Большинство современных систем PROLOG работают с интерпретатором, основанным на абстрактной машине Уоррена (WAM). Исходный код PROLOG компилируется в так называемый код WAM, который затем интерпретируется WAM. Самые быстрые реализации WAM обрабатывают до 10 миллионов логических выводов в секунду (LIPS) на ПК с тактовой частотой 1 ГГц.

5.2 Простые примеры

Начнем с семейных отношений из примера 3.2 на стр. 35. Небольшая база знаний KB закодирована — без фактов для предиката female — как программа PROLOG с именем rel.pl на рис. 5.1.

Программу можно загрузить и скомпилировать в интерпретаторе PROLOG с помощью команды

```
?- [отн.].
```

Machine Translated by Google

Первоначальный запрос возвращает диалог

?- ребенок(ева,оскар,энн).

Да

с правильным ответом Да. Как возникает этот ответ? Для запроса «?-ребенок(ева,оскар,энн)». есть шесть фактов и одно правило с одним и тем же предикатом в заголовке предложения. Теперь предпринимается попытка унификации между запросом и каждым из дополнительных литералов во входных данных в порядке их появления. Если одна из альтернатив терпит неудачу, это приводит к возврату к последней точке ветвления, и проверяется следующая альтернатива. Поскольку унификация не удалась с каждым фактом, запрос унифицируется с помощью рекурсивного правила в строке 8. Теперь система пытается решить подцель дочерний элемент (ева, анна, Оскар), что удается с третьей альтерна

Запрос

?- потомок(X,Y).

X = Оскар
Y = Карен

Да

отвечает первым найденным решением, как

?- потомок(клайд,Y).

Y = Мэри

Да

Запрос

?- потомок(Клайд,Карен).

однако не отвечает. Причиной этого является предложение в строке 8, определяющее симметрию дочернего предиката. Это предложение вызывает себя рекурсивно без возможности завершения. Эту задачу можно решить с помощью следующей новой программы (здесь опущены факты).

```
1 потомок(X,Y):- ребенок(X,Y,Z). 2 потомок(X,Y):- потомок(X,Z,Y). 3
потомок(X,Y):- потомок(X,U,V), потомок(U,Y).
```

Machine Translated by Google

Но теперь запрос

?- ребенок(ева,оскар,энн).

больше не является правильным ответом, потому что симметрия ребенка в последних двух переменных больше не задана. Решение обеих проблем находится в программе

```
1 child_fact(Оскар,Карен,Франц). 2
child_fact(мэри,карен,франц). 3
child_fact(ева,энн,оскар). 4
child_fact(Генри,Энн,Оскар). 5
child_fact(изольда,анна,оскар). 6
child_fact(клайд,мэри,оскарб). 7 8 дочерний(X,Z,Y):-

дочерний_факт(X,Y,Z). 9 дочерний(X,Z,Y):-
дочерний_факт(X,Z,Y). 10 11 потомок(X,Y):-

потомок(X,Y,Z). 12 потомок(X,Y):- потомок(X,U,V),
потомок(U,Y).
```

Благодаря введению нового предиката child_fact для фактов предикат child больше не является рекурсивным. Однако программа уже не такая элегантная и простая, как логически правильный первый вариант на рис. 5.1 на стр . 68, приводящий к бесконечному циклу. Программист на PROLOG должен, как и в других языках, обращать внимание на обработку и избегать бесконечных циклов. PROLOG — это просто язык программирования, а не средство доказательства теорем.

Здесь мы должны различать декларативную и процедурную семантику программ PROLOG. Декларативная семантика задается логической интерпретацией роговых предложений. Процедурная семантика, напротив, определяется выполнением программы PROLOG, которую мы сейчас хотим рассмотреть более подробно. Выполнение программы с рис. 5.1 на стр. 68 с запросом child(eve,oscar,anne) представлено на рис. 5.2 на стр. 71 в виде дерева поиска.1 Выполнение начинается слева вверху с запроса. Каждое ребро представляет возможный шаг разрешения SLD с дополнительным унифицируемым литералом. В то время как дерево поиска становится бесконечно глубоким из-за рекурсивного правила, выполнение PROLOG прекращается, потому что факты появляются перед правилом во входных данных.

Напротив, с запросом потомок(клайд,карен) выполнение PROLOG не завершается. Мы можем ясно видеть это в дереве и/или, представленном на рис. 5.3 на стр. 71. В этом представлении ветви, представленные значком, ведут от заголовка предложения , к подцелям. Поскольку все подцели предложения должны быть решены, это и ответвления. Все остальные ветви являются или ветвями, из которых хотя бы одна должна быть унифицируемой со своими родительскими узлами. Два изложенных факта отражают

1Константы были сокращены для экономии места.

Machine Translated by Google

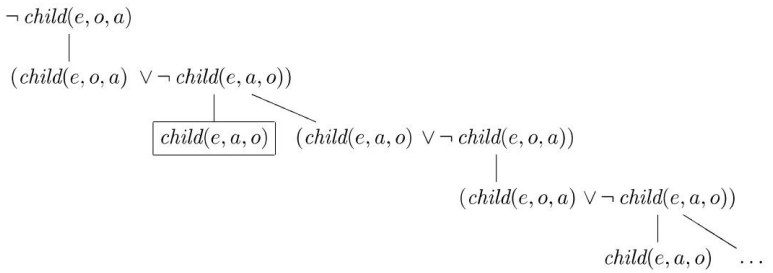


Рис. 5.2 Дерево поиска PROLOG для ребенка (ева, оскар, анна)

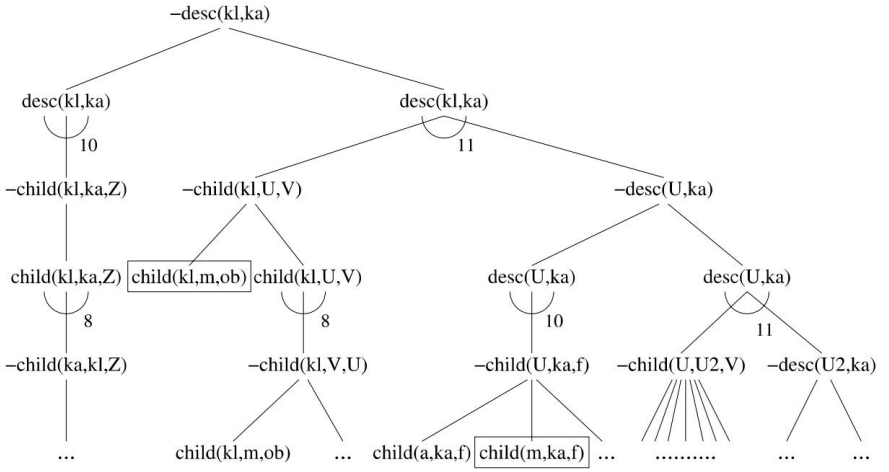


Рис. 5.3 И-или дерево для desc(clyde,karen)

решение запроса. Однако интерпретатор PROLOG здесь не останавливается, потому что он работает, используя поиск в глубину с возвратом (см. раздел 6.2.2) и, таким образом, сначала выбирает бесконечно глубокий путь в крайнее левое положение.

5.3 Контроль исполнения и процедурные элементы

Как мы видели в примере с семейными отношениями, важно контролировать выполнение PROLOG. В частности, избегание ненужного возврата назад может привести к значительному повышению эффективности. Одним из средств для достижения этой цели является разрез. Вставив восклицательный знак в предложение, мы можем предотвратить возврат к этой точке. В следующей программе предикат max(X,Y,Max) вычисляет максимальное из двух чисел X и Y.

Machine Translated by Google


```
1 max(X,Y,X) :- X >= Y. 2 max(X,Y,Y) :-
X < Y.
```

Если применяется первый случай (первый пункт), то второй не будет достигнут. С другой стороны, если первый случай неприменим, то условие второго случая истинно, а значит, его не нужно проверять. Например, в запросе

?- max(3,2,Z), Z > 10.

поиск с возвратом используется, поскольку Z=3, а второе предложение проверяется на максимальное значение, что обречено на провал. Таким образом, возвращаться к этому месту не нужно. Мы можем оптимизировать это с помощью разреза:

```
1 max(X,Y,X) :- X >= Y, !. 2 макс(X,Y,Y).
```

Таким образом, второе предложение вызывается только в том случае, если оно действительно необходимо, то есть если первое предложение не работает. Однако эта оптимизация усложняет понимание программы.

Другая возможность управления выполнением — это встроенный предикат fail, который никогда не бывает истинным. В примере с семейными отношениями мы можем просто вывести всех детей и их родителей с помощью запроса

?- child_fact(X,Y,Z), write(X), write(' является потомком '), write(Y), write(' and '), write(Z), write('.'), нл, провал.

Соответствующий вывод

Оскар — ребенок Карен и Фрэнка. Мэри — дочь Карен
и Фрэнка. Ева — дочь Анны и Оскара.

...
Нет.

где предикат nl вызывает разрыв строки в выводе. Каким был бы вывод в конце без использования предиката fail?

С той же базой знаний запрос «?- child_fact(ulla,X,Y)». приведет к ответу Нет, потому что нет никаких фактов об улле. Этот ответ не является логически правильным. В частности, невозможно доказать отсутствие объекта с именем ulla. Здесь доказывающая E правильно ответила бы: «Доказательство не найдено». Таким образом, если ПРОЛОГ отвечает Нет, это означает только то, что запрос Q не может быть

Machine Translated by Google

быть доказанным. Однако для этого не обязательно доказывать ¬Q. Такое поведение называется отрицанием как неудачей.

Ограничение себя предложениями Хорна в большинстве случаев не вызывает больших проблем. Однако это важно для процедурного исполнения с использованием SLD-разрешения (раздел 2.5). Благодаря однозначно определенному положительному литералу для предложения, разрешению SLD и, следовательно, выполнению программ PROLOG есть уникальная точка входа в предложение. Только так возможно воспроизводимое выполнение логических программ и, следовательно, четко определенная процедурная семантика.

В самом деле, безусловно, существуют формулировки задачи, которые не могут быть описаны с помощью предложений Хорна. Примером может служить парадокс Рассела из примера 3.7 на стр. 45, который содержит предложение, отличное от Хорна (бритье(цирюльник,X) бритье(X, X)).

5.4 Списки

Как язык высокого уровня, PROLOG имеет, как и язык LISP, удобный общий тип данных списка. Список с элементами A, 2, 2, B, 3, 4, 5 имеет вид

[A,2,2,Б,3,4,5]

Конструкция [Head|Tail] отделяет первый элемент (Head) от остальных (Tail) списка. С базой знаний

список([A,2,2,B,3,4,5]).

PROLOG отображает диалоговое окно

?- список([H|T]).

H=A
T = [2, 2, B, 3, 4, 5]

Да

Используя вложенные списки, мы можем создавать произвольные древовидные структуры. Например, два дерева



может быть представлен списками [b,c] и [a,b,c] соответственно, и двумя деревьями



Machine Translated by Google

списками [[e,f,g],[h],d] и [a,[b,e,f,g],[c,h],d] соответственно.
В деревьях, где внутренние узлы содержат символы, символ является головой дерева.
список и дочерние узлы являются хвостом.

Хорошим и элегантным примером обработки списка является определение предиката append(X,Y,Z) для добавления списка Y к списку X. Результат сохраняется в Z. соответствующая программа PROLOG читает

```
1 добавить([],L,L).
2 добавить([X|L1],L2,[X|L3]) :- добавить(L1,L2,L3).
```

Это декларативное (рекурсивное) логическое описание того факта, что L3 является результатом добавление L2 к L1. В то же время, однако, эта программа также выполняет работу когда его зовут. Вызов

```
?- добавить([a,b,c],[d,1,2],Z).
```

возвращает замену Z = [a, b, c, d, 1, 2], как и вызов

```
?- добавить(X,[1,2,3],[4,5,6,1,2,3]).
```

дает замену X = [4, 5, 6]. Здесь мы замечаем, что append не является двухместная функция, но трехместная связь. На самом деле, мы также можем ввести «выходной параметр» Z и спросить, можно ли его создать.

Обратный порядок элементов списка также может быть элегантно описан и прост. одновременно программируется рекурсивным предикатом

```
1 оборот([],[]).
2 nrev([H|T],R):- nrev(T,RT), append(RT,[H],R).
```

который сводит обращение списка к обращению списка, состоящего из одного элемента меньше. Действительно, этот предикат очень неэффективен из-за вызова append. Эта программа известна как naive reverse и часто используется в качестве эталонного теста PROLOG (см. Упражнение 5.6 на странице 81). Дела идут лучше, когда кто-то продолжает использовать временный хранить, известный как аккумулятор, следующим образом:

Список	Аккумулятор
[а, б, в, г] []	
[б, в, г]	[а]
[CD]	[б, а]
[д]	[в, б, а]
[]	[г, в, б, а]

Machine Translated by Google

Соответствующая программа PROLOG читает

```
1 аккрев([],A,A). 2
приращение([H|T],A,R) :- приращение(T,[H|A],R).
```

5.5 Самомодифицирующиеся программы

Программы PROLOG не полностью компилируются, а интерпретируются WAM. Поэтому можно изменять программы во время выполнения. Программа может даже модифицировать себя. С помощью таких команд, как assert и retract, факты и правила могут быть добавлены в базу знаний или удалены из нее.

Простым применением варианта asserta является добавление производных фактов в начало базы знаний с целью избежать повторного, потенциально затратного по времени вывода (см. упражнение 5.8 на стр. 81). Если в нашем примере семейных отношений мы заменим два правила для предиката потомок на

```
1 :- динамический потомок/2. 2
потомок(X,Y):- потомок(X,Y,Z), утверждение(потомок(X,Y)). 3 потомок(X,Y):- потомок(X,U,V),
потомок(U,Y), 4 утверждения(потомок(X,Y)).
```

то все производные факты для этого предиката сохраняются в базе знаний и таким образом в дальнейшем не выводятся повторно. Запрос

?- потомок(клайд, карен).

приводит к добавлению двух фактов

потомок (Клайд, Карен). потомок(мэри,
карен).

Манипулируя правилами с помощью утверждений и ретрактов, можно написать даже полностью меняющиеся программы. Эта идея стала известна под термином генетическое программирование. Он позволяет создавать произвольно гибкие программы обучения. Однако на практике оказывается, что из-за огромного количества бессмысленных возможных изменений изменение кода методом проб и ошибок редко приводит к увеличению производительности. С другой стороны, систематическое изменение правил настолько усложняет программирование, что до сих пор такие программы, которые широко модифицируют свой собственный код, не имели успеха. В гл. 8 мы покажем, насколько успешным было машинное обучение. Однако здесь проводятся лишь очень ограниченные модификации программного кода.

Machine Translated by Google


```
1 начало :- действие(состояние(слева,слева,слева,слева),
2                                    состояние (правильно, правильно, правильно, правильно)).
3
4 действие(Начало,Цель):-
                  план(Начало,Цель,[Начало],Путь),
5 6 nl,write('Решение:'),nl,
7 write_path(Путь).
8 % write_path(Путь), ошибка. 9                                    % выход всех решений

10 план (Начало, Цель, Посещение, Путь): -
11 идти(Старт,Далее),
12 сейф(Следующий),
13 \+ участник(Следующий,Посещенный), % не(участник(...))
14                  план(Следующий,Цель,[Следующий|Посещенный],Путь).
15 план(Цель,Цель,Путь,Путь).
16
17 идти(состояние(X,X,Z,K),состояние(Y,Y,Z,K)):-через(X,Y). % фермер, волк
18 идти(состояние(X,W,X,K),состояние(Y,W,Y,K)):-через(X,Y). % фермер, коза
19 идти(состояние(X,W,Z,X),состояние(Y,W,Z,Y)):-через(X,Y). % фермер, капуста
20 идти(состояние(X,W,Z,K),состояние(Y,W,Z,K)):-через(X,Y). % фермер
21
22 поперек (слева, справа).
23 поперек (справа, слева).
24
25 безопасный(состояние(B,W,Z,K)):- через(W,Z), через(Z,K).
26 сейф(состояние(B,B,B,K)).
27 сейф(состояние(B,W,B,B)).
```

Рис. 5.4. Программа PROLOG для задачи фермер–волк–коза–капуста

5.6 Пример планирования

Пример 5.1 Следующая загадка служит постановкой задачи для типичной программы PRO LOG.

> Крестьянин хочет переправить через реку капусту, козу и волка, но его лодка такая маленькая
> что он может взять их только по одному за раз. Фермер подумал и сказал:
> сам: «Если я сначала волка переведу на другой берег, то коза съест капусту. Если я
> сначала капусту перевезти, потом волк съест козу. Что я должен делать?"

Это задача планирования, которую мы можем быстро решить, немного подумав.
Программа PROLOG, представленная на рис. 5.4 , создается не так быстро.

Программа работает на условиях формы состояния(Фермер,Волк,Коза,
Капуста), которые описывают современное состояние мира. Четыре переменные с
возможные значения left, right дают расположение объектов. Центральный план рекурсивных
предикатов сначала создает состояние-преемник. Затем, используя go, проверяет его безопасность.
с безопасным и повторяет это рекурсивно до тех пор, пока начальное и целевое состояния не совпадут.
(в строке 15 программы). Состояния, которые уже были посещены, сохраняются в
третий аргумент плана. С помощью встроенного члена-предиката проверяется, является ли
состояние Next уже посещено. Если да, то это не попытка. Определение
здесь отсутствует предикат write_path для задачи вывода найденного плана. Это
предлагается в качестве упражнения для читателя (Упражнение 5.2 на стр. 80). Для начальной про-

Machine Translated by Google

gram проверяет, что литерал write_path(Path) можно заменить на write(Path). Для запроса «?- начало». мы получаем ответ

Решение:
Фермер и коза слева направо
Фермер справа налево
Фермер и волк слева направо
Фермер и коза справа налево
Фермер и капуста слева направо
Фермер справа налево
Фермер и коза слева направо

Да

Для лучшего понимания опишем определение плана в логике:

z план(z, z) s z n [go(s, n) безопасный(n) план(n, z) план(s, z)]

Это определение получается значительно более кратким, чем в PROLOG. Этому есть две причины. Во-первых, результат обнаруженного плана не имеет значения для логики. Кроме того, на самом деле нет необходимости проверять, было ли уже посещено следующее состояние, если ненужные переходы не беспокоят фермера. Однако, если \+ member(...) не включить в программу PROLOG, возникает бесконечный цикл, и PROLOG может не найти расписания, даже если оно есть. Причиной этого является стратегия поиска PROLOG с обратной цепочкой, которая, в соответствии с принципом поиска в глубину (раздел 6.2.2), всегда работает с подцелями по одной, не ограничивая глубину рекурсии, и поэтому является неполной. Этого не случилось бы с доказателем теорем с полным исчислением.

Как и во всех задачах планирования, состояние мира меняется по мере выполнения действий от одного шага к другому. Это предполагает отправку состояния в виде переменной всем предикатам, которые зависят от состояния мира, например, в предикате safe. Переходы между состояниями происходят в предикате go. Этот подход называется ситуационным исчислением [RN10]. В гл. 10.

5.7 Программирование логики ограничений

Программирование систем планирования, в которых должны выполняться многие (иногда сложные) логические и численные условия, может быть очень дорогим и сложным при использовании обычных языков программирования. Именно здесь логика может быть полезна. Просто записывают все логические условия в PL1, а затем вводят запрос. Обычно этот подход терпит неудачу. Причина в проблеме пингвинов, обсуждавшейся в разд. 4.3. Тот факт, что пингвин(твити) гарантирует, что пингвин(твит

Machine Translated by Google

истинный. Однако это не исключает, что ворон(твити) тоже верен. Исключать это дополнительными аксиомами очень неудобно (п. 4.3).

Логическое программирование с ограничениями (CLP), которое позволяет явно формулировать ограничения для переменных, предлагает элегантный и очень эффективный механизм для решения этой проблемы. Интерпретатор постоянно следит за выполнением программы на предмет соблюдения всех ее ограничений. Программист полностью освобождается от задачи управления ограничениями, что во многих случаях может значительно упростить программирование Это выражено в следующей цитате Юджина К. Фрейдера из [Fre97]:

> Программирование с ограничениями представляет собой один из самых близких подходов, которые компьютерная наука когда-либо делала к Святому Граалю программирования: пользователь формулирует проблему, компьютер решает ее.

Не вдаваясь в теорию проблемы удовлетворения ограничений (CSP), мы применим механизм CLP GNU-PROLOG к следующему примеру.

Пример 5.2 Секретарь средней школы имени Альберта Эйнштейна должен разработать план распределения помещений для выпускных экзаменов. У него есть следующая информация: четыре учителя Майер, Гувер, Миллер и Смит дают тесты по немецкому языку, английскому языку, математике и физике в восходящих комнатах 1, 2, 3 и 4. Каждый учитель дает тест ровно по одному предмету. объект ровно в одной комнате. Кроме того, он знает следующее об учителях и их предметах.

1. Мистер Майер никогда не проводит тесты в комнате 4.
2. Мистер Миллер всегда проверяет немецкий язык.
3. Мистер Смит и мистер Миллер не проводят анализы в соседних комнатах.
4. Миссис Гувер проверяет математику.
5. Физика всегда проверяется в комнате №4.
6. Немецкий и английский языки в комнате 1 не тестируются.

Кто дает тест в какой комнате?

Программа GNU-PROLOG для решения этой задачи приведена на рис. 5.5 на стр. 79. Эта программа работает с переменными Майера, Гувера, Миллера, Смита, а также с немецким, английским, математическим, физическим, каждая из которых может принимать целое число. значение от 1 до 4 в качестве номера комнаты (строки программы 2 и 5). Привязка Mayer = 1 и German = 1 означает, что мистер Майер проводит тест по немецкому языку в комнате 1. Строки 3 и 6 гарантируют, что четыре конкретные переменные принимают разные значения. Строка 8 гарантирует, что всем переменным присваивается конкретное значение в случае решения. Эта строка здесь совершенно не нужна. Однако, если бы было несколько решений, были бы выведены только интервалы. В строках с 10 по 16 даны ограничения, а остальные строки выводят номера кабинетов для всех учителей и всех предметов в простом формате.

Программа загружается в GNU-PROLOG с помощью «['raumplan.pl']», и с «начать». мы получаем вывод

[3,1,2,4] [2,3,1,4]

Machine Translated by Google


```
1 начало :-
2 fd_domain([Майер, Гувер, Миллер, Смит],1,4),
3 fd_all_differ([Майер, Миллер, Гувер, Смит]),
4
5 fd_domain([немецкий, английский, математика, физика],1,4),
6 fd_all_ Different([немецкий, английский, математика, физика]),
7
8 fd_labeling([Майер, Гувер, Миллер, Смит]),
9
10 Mayer #\=4, 11 Miller #=                    % Майер не в комнате 4
немецкий, 12 dist(Miller,Smith) #>= 2,        % Миллер тестирует немецкий язык
13 Hoover #= математика, 14 физика #= 4, 15    % Расстояние Миллер/Смит >= 2
немецкий #\=1, 16 английский                  % Математические тесты Гувера
#\=1 , 17 нл,                                 % Физика в комнате 4
                                              % Немец не в комнате 1
                                              % Английский не в комнате 1

18 write([Майер, Гувер, Миллер, Смит]), nl,
19 write([немецкий, английский, математика, физика]), nl.
```

Рис. 5.5 CLP-программа для задачи планирования помещений

Представив несколько более удобно, мы имеем следующее расписание помещений:

Номер комнаты	1	2	3	4
Учитель	Гувер	Миллер	Майер	Смит
Предмет	Математика	немецкий	английский	Физика

GNU-PROLOG, как и большинство других языков CLP, имеет так называемую конечную область.
решатель ограничений, с помощью которого переменным может быть назначен конечный диапазон целых чисел.
Это не обязательно должен быть интервал, как в примере. Мы также можем ввести список
ценностей. В качестве упражнения пользователю предлагается в Упражнении 5.9 на стр. 81 создать
Программа CLP, например, с GNU-PROLOG, для не очень простой логической головоломки.
Эту загадку, предположительно созданную Эйнштейном, очень легко решить с помощью CLP.
система. С другой стороны, если бы мы попытались использовать PROLOG без ограничений, мы могли бы
легко стачиваем зубы. Любой, кто находит элегантное решение с PROLOG или
прувер, пожалуйста, пусть он найдет путь к автору.

5.8 Резюме

Унификация, списки, декларативное программирование и реляционное представление процедур,
в которых аргумент предиката может действовать как вход и выход, позволяют разрабатывать короткие,
элегантные программы для решения многих задач. Многие программы будут
значительно длиннее и, следовательно, труднее понять, если они написаны процедурным
язык. Кроме того, эти возможности языка экономят время программиста. Поэтому PROLOG также является
интересным инструментом для быстрого прототипирования, особенно для ИИ.

Machine Translated by Google

Приложения. CLP-расширение PROLOG полезно не только для решения логических головоломок, но и для многих задач оптимизации и планирования.

С момента своего изобретения в 1972 году в Европе PROLOG превратился в один из ведущих европейских языков программирования для ИИ наряду с процедурными языками. С другой стороны, в США на рынке искусственного интеллекта доминирует язык LISP.

ПРОЛОГ не является средством доказательства теорем. Это сделано намеренно, потому что программист должен иметь возможность легко и гибко управлять обработкой, а с доказателем теорем далеко не уедешь. С другой стороны, ПРОЛОГ сам по себе не очень полезен для доказательства математических теорем. Однако существуют, безусловно, интересные средства доказательства теорем, запрограммированные на PROLOG.

В качестве расширенной литературы рекомендуется [Bra11] и [CM94], а также справочники [Wie04, Dia04] и, по теме CLP, [Bar98].

5.9 Упражнения

Упражнение 5.1 Попробуйте доказать теорему из разд. 3.7 о равенстве левых и правых нейтральных элементов полугрупп с PROLOG. Какие проблемы возникают?
Какова причина этого?

Упражнение

5.2 (a) Напишите предикат write_move(+State1, +State2), который выводит предложение типа «Фермер и волк переходят слева направо» для каждой переправы через лодку. State1 и State2 являются терминами формы state(Фермер, Волк, Коза, Капуста). (b) Напишите рекурсивный предикат
write_path(+Path), который вызывает предикат write_move(+State1, +State2) и выводит все действия фермера.

Упражнение

5.3 (a) На первый взгляд переменная Path в плане предикатов программы PROLOG из примера 5.1 на стр. 76 не нужна, поскольку она, по-видимому, нигде не изменяется. Для чего это нужно?
(б) Если мы добавим отказ в конце действия в примере, то все решения будут даны в качестве вывода. Почему теперь каждое решение печатается дважды? Как вы можете предотвратить это?

Упражнение

5.4 (a) Покажите путем проверки, что средство доказательства теорем E (в отличие от ПРОЛОГА) с учетом базы знаний, показанной на рис. 5.1 на стр. 68, отвечает на запрос «?-потомок(клайд, карен)». правильно. Почему это? (b) Сравните ответы
PROLOG и E на запрос «?-потомок(X,
Ю)».

Machine Translated by Google

Упражнение 5.5. Напишите как можно более короткую программу на PROLOG, которая выводит 1024 единицы.

Упражнение 5.6. Исследуйте поведение простого обратного предиката во время выполнения.

(a) Запустите PROLOG с опцией трассировки и посмотрите на рекурсивные вызовы nrev, append и accrev.

(b) Вычислите асимптотическую временную сложность append(L1,L2,L3), то есть зависимость времени выполнения от длины списка для больших списков. Предположим, что доступ к голове произвольного списка занимает постоянное время. (c) Вычислите временную сложность nrev(L,R). (d) Вычислите временную сложность accrev(L,R). (e) Экспериментально определить временную сложность предикатов nrev, append и accrev, например, путем проведения измерений времени (время(+Цель) дает выводы и процессорное время).

Упражнение 5.7 Используйте функциональные символы вместо списков для представления деревьев, приведенных в разд. 5.4 на стр. 73.

Упражнение 5.8 Последовательность Фибоначчи определяется рекурсивно как fib(0) = 1, fib(1) = 1 и fib(n) = fib(n 1) + fib(n 2). (a) Определите рекурсивный предикат PROLOG fib(N,R), который вычисляет fib(N) и возвращает его в R.

(b) Определить сложность времени выполнения предиката fib теоретически и с помощью измерение.

(c) Измените свою программу, используя asserta, чтобы больше не выполнялись ненужные выводы. (d) Определить сложность модифицированного предиката во время выполнения теоретически и путем измерения (обратите внимание, что это зависит от того, был ли ранее вызван fib).

(e) Почему fib с asserta также быстрее, когда он запускается в первый раз правильно после запуска PROLOG?

Упражнение 5.9 Следующая типичная логическая головоломка предположительно была написана Альбертом Эйнштейном. Более того, он якобы утверждал, что только 2% населения земного шара способны ее решить. Даны следующие утверждения. • Есть пять домов, каждый из которых окрашен в свой цвет. • В каждом доме живет человек другой национальности. • Каждый житель предпочитает определенный напиток, курит определенную марку сигарет и имеет определенное домашнее животное. • Ни один из пяти человек не пьет одно и то же, не курит одно и то же и не имеет такой же питомец.

- Подсказки: – Британец живет в красном доме.
  - У шведа есть собака.
  - Датчанин любит пить чай.
  - Зеленый дом находится слева от белого дома.

Machine Translated by Google

– Хозяин зелени пьет кофе.

– У человека, который курит Pall Mall, есть птица.
– Мужчина, живущий в среднем доме, пьет молоко.

– Хозяин желтого дома курит Dunhill.

– Норвежец живет в первом доме.
– Курильщик Marlboro живет рядом с тем, у кого есть кошка.

– Мужчина с лошадью живет рядом с тем, кто курит Dunhill.

– Курильщик Winfield любит пить пиво.

– Норвежец живет рядом с синим домом.
– Немец курит Rothmanns.

– У курильщика Marlboro есть сосед, который пьет воду.

Вопрос: Кому принадлежит рыба? (a) Сначала
решите головоломку вручную. (b)
Напишите программу CLP (например, с помощью GNU-PROLOG), чтобы решить головоломку.

Ориентируйтесь на задачу планирования помещений на рис. 5.5 на стр. 79.

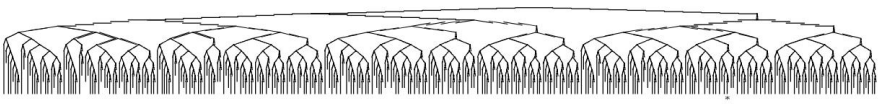
Machine Translated by Google

Поиск, игры и решение проблем

6

6.1 Введение

Поиск решения в очень большом дереве поиска представляет проблему почти для всех систем логического вывода. Из начального состояния есть много возможностей для первого шага вывода. Для каждой из этих возможностей на следующем шаге снова есть много возможностей и так далее. Даже при доказательстве очень простой формулы из [Ert93] с тремя предложениями Хорна, каждое из которых содержит не более трех литералов, дерево поиска для SLD-разрешения имеет следующий вид:



Дерево было срезано на глубине 14 и имеет решение в листовом узле, отмеченном . Его вообще возможно представить только из-за малого коэффициента ветвления, не более двух, и отсечки на глубине 14. Для реальных задач коэффициент ветвления и глубина первого решения могут стать значительно больше.

Предположим, что коэффициент ветвления является константой, равной 30, и первое решение находится на глубине 50. Дерево поиска имеет 3050 7,2 × 1073 листовых узлов. Но количество шагов вывода еще больше, потому что не только каждый листовой узел, но и каждый внутренний узел дерева соответствует шагу вывода. Поэтому мы должны сложить узлы по всем уровням и получить общее количество узлов дерева поиска.

$$30d = \sum_{d=0}^{50} = \frac{1 - 3051}{1 \quad 30} \approx 7{,}4 \times 1073 \quad ,$$

что не сильно меняет количество узлов. Очевидно, что почти все узлы этого дерева поиска находятся на последнем уровне. Как мы увидим, это обычно так. Но теперь вернемся к дереву поиска с 7,4 × 1073 узлами. Предположим, у нас есть 10 000 компьютеров, каждый из которых может выполнять миллиард логических выводов в секунду, и что мы можем

В. Эртель, Введение в искусственный интеллект, Темы бакалавриата по компьютерным наукам, DOI 10.1007/978-0-85729-299-5_6, © Springer-Verlag London Limited 2011

Machine Translated by Google

распределять работу по всем компьютерам бесплатно. Общее время вычислений для всех 7,4 × 1073 выводов будет примерно равно

$$7{,}4 \times 1073 \text{ выводов} = 7{,}4 \times \frac{1060 \text{ с}}{10000 \times 109 \text{ выводов/с}} \quad 2{,}3 \times 1053 \text{ года},$$

что примерно в 1043 раза превышает возраст нашей Вселенной. Благодаря этому простому мысленному упражнению мы можем быстро понять, что нет никакой реальной возможности полностью исследовать это пространство поиска с помощью средств, доступных нам в этом мире. Более того, предположения, связанные с размером пространства поиска, были вполне реалистичными. Например, в шахматах для типичной ситуации существует более 30 возможных ходов, а партия, длящаяся 50 полуоборотов, относительно коротка.

Как же тогда могут быть хорошие шахматисты, а в наши дни еще и хорошие шахматные компьютеры? Как могло случиться, что математики находят доказательства теорем, в которых пространство поиска еще больше? Очевидно, мы, люди, используем разумные стратегии, которые значительно сокращают пространство поиска. Опытный шахматист, как и опытный математик, простым наблюдением за ситуацией сразу исключит многие действия как бессмысленные. Благодаря своему опыту он способен оценивать различные действия на предмет их полезности для достижения цели. Часто человек идет по ощущениям. Если спросить математика, как он нашел доказательство, он может ответить, что интуиция пришла к нему во сне. В сложных случаях многие врачи ставят диагноз чисто на ощупь, на основании всех известных симптомов. Особенно в сложных ситуациях часто отсутствует формальная теория поиска решения, гарантирующая оптимальное решение. В повседневных задачах, таких как поиск сбежавшего кота на рис. 6.1 на стр. 85, большую роль играет интуиция. Мы рассмотрим этот вид эвристического метода поиска в разд. 6.3 и дополнительно описать процессы, с помощью которых компьютеры могут, подобно людям, улучшать свои эвристические стратегии поиска путем обучения.

Однако сначала мы должны понять, как работает неосведомленный поиск, то есть слепое перепробование всех возможностей. Начнем с нескольких примеров.

Пример 6.1 С помощью головоломки с восьмерками, классического примера для алгоритмов поиска [Nil98, RN10], можно очень наглядно проиллюстрировать различные алгоритмы. Квадраты с номерами от 1 до 8 распределены в матрице 3 × 3, подобной матрице на рис. 6.2 на стр. 86. Цель состоит в том, чтобы достичь определенного упорядочения квадратов, например, в порядке возрастания по строкам, как показано на рис. 6.2 на стр . 86. На каждом этапе квадрат можно перемещать влево, вправо, вверх или вниз в пустое место. Таким образом, пустое пространство движется в соответствующем противоположном направлении. Для анализа пространства поиска удобно всегда смотреть на возможные движения пустого поля.

Дерево поиска начального состояния представлено на рис. 6.3 на стр. 86. Мы можем видеть, что коэффициент ветвления чередуется между двумя, тремя и четырьмя. Усредняя одновременно по двум уровням, мы получаем средний коэффициент ветвления1, равный $\sqrt{8} \approx 2{,}83$. Мы видим, что каждое состояние повторяется несколько раз на два уровня глубже, потому что при простом неинформированном поиске каждое действие может быть отменено на следующем шаге.

1Средний коэффициент ветвления дерева — это коэффициент ветвления, который имел бы дерево с постоянным коэффициентом ветвления, равной глубиной и равным количеством листовых узлов.

Machine Translated by Google

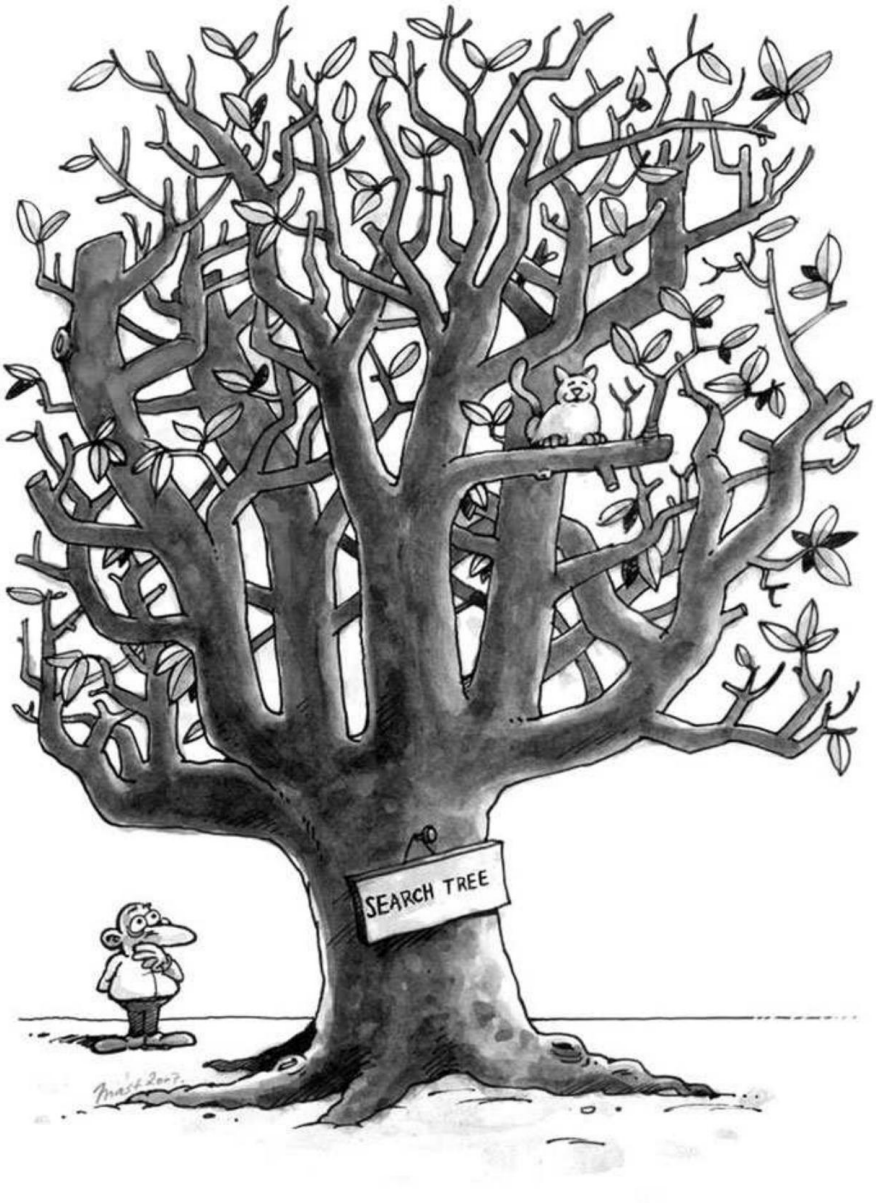


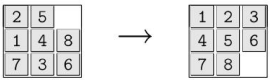
Рис. 6.1. Сильно подстриженное дерево поиска — или: «Где мой кот?»

Если мы запретим циклы длины 2, то для одного и того же начального состояния получим дерево поиска, представленное на рис. 6.4 на стр. 86. Средний коэффициент ветвления уменьшается примерно на 1 и становится равным 1,8.2.

2Для головоломки с восьмерками средний фактор ветвления зависит от начального состояния (см. упражнение 6.2 на стр. 110).

Machine Translated by Google

Рис. 6.2 Возможные начальное и целевое состояния головоломки с восьмеркой



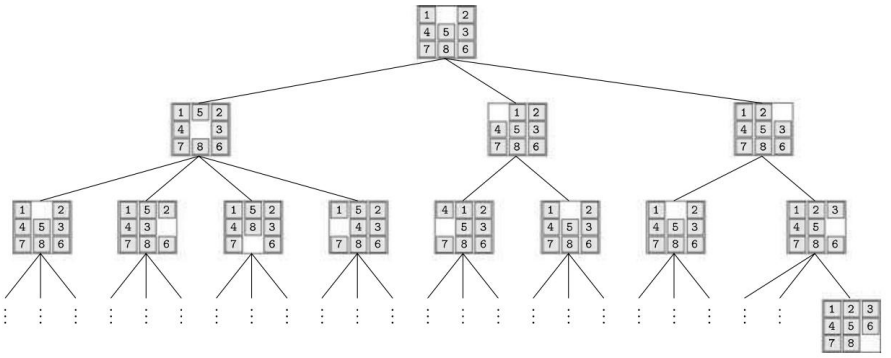


Рис. 6.3. Дерево поиска головоломки с восьмерками. Внизу справа представлено целевое состояние на глубине 3. Для экономии места другие узлы на этом уровне были опущены.

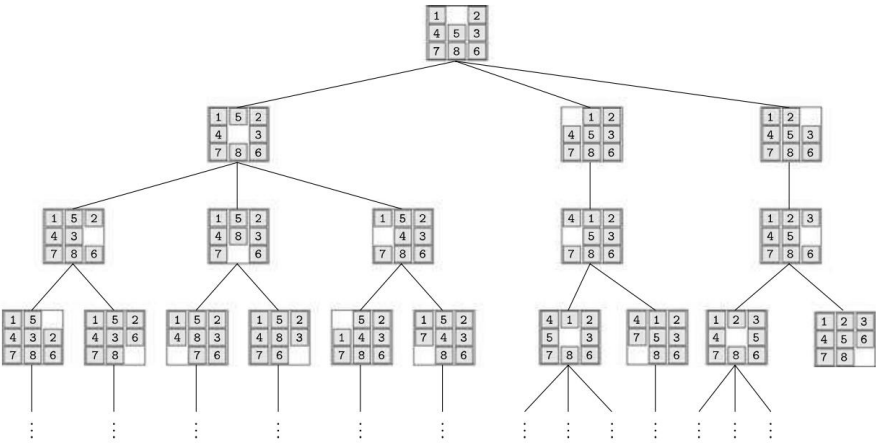


Рис. 6.4 Дерево поиска головоломки на 8 без циклов длины 2

Прежде чем мы начнем описывать алгоритмы поиска, необходимо ввести несколько новых терминов. Здесь мы имеем дело с задачами дискретного поиска. Находясь в состоянии s, действие a1 = a1(s). Другое состоянию s = a2(s). Таким образом, с действие может привести к состоянию s ′′, а может привести к новому другими словами: с Рекурсивное применение всех возможных действий ко всем состояниям, начиная с начального состояния, дает дерево поиска.

Machine Translated by Google

Определение 6.1 Задача поиска определяется следующими значениями Состояние:
Описание состояния мира, в котором поисковый агент находит
сам.
Начальное состояние: начальное состояние, в котором запущен поисковый агент.
Целевое состояние: если агент достигает целевого состояния, он завершает работу и выводит
решение (при желании).
Действия: всем агентам разрешены действия.
Решение: путь в дереве поиска из начального состояния в целевое.
Функция стоимости: присваивает значение стоимости каждому действию. Необходим для поиска
оптимального по стоимости решения.
Пространство состояний: набор всех состояний.

Применительно к 8-загадке получаем
Состояние: матрица 3×3 S со значениями 1, 2, 3, 4, 5, 6, 7, 8 (по одному разу) и одним пустым
квадрат.
Начальное состояние: Произвольное состояние.
Целевое состояние: Произвольное состояние, например, состояние, указанное справа на рис. 6.2 на стр. 86.
Действия: Движение пустого квадрата Sij влево (если j = 1), вправо (если j = 3), вверх (если i = 1), вниз (если i
= 3).
Функция стоимости: постоянная функция 1, поскольку все действия имеют одинаковую стоимость.
Пространство состояний: пространство состояний является вырожденным в областях, которые взаимно недостижимы
(упражнение 6.4 на стр. 110). Таким образом, существуют неразрешимые задачи с 8 головоломками.
Для анализа алгоритмов поиска необходимы следующие термины:

Определение 6.2 •
Количество последовательных состояний состояния s называется коэффициентом ветвления.
b(s) или b, если коэффициент ветвления постоянный.
• Эффективный коэффициент ветвления дерева глубины d с общим числом узлов n определяется
как коэффициент ветвления дерева с постоянным коэффициентом ветвления, равной глубиной
и равным n (см. Упражнение 6.3 на стр. 110). • Алгоритм поиска называется полным ,
если он находит решение для каждой решаемой задачи. Если полный алгоритм поиска
завершается, не найдя решения, то проблема неразрешима.

Для данной глубины d и количества узлов n эффективный коэффициент ветвления может быть
рассчитывается путем решения уравнения

$$n = \frac{б^{д+1} - 1}{б - 1} \tag{6.1}$$

Machine Translated by Google

Следует проявлять осторожность, если количественная переменная выходит за рамки своего количественного анализа.
тифицировать, как, например, x в

$$\forall x p(x) \quad \exists x q(x).$$

Здесь одну из двух переменных нужно переименовать, а в

$$\forall x p(x) \quad \exists y q(y)$$

квантор легко вынести вперед, и мы получаем на выходе эквивалентную формулу

$$\forall x \exists y p(x) \quad q(y).$$

Если, однако, мы отнимем правильно поставить квантификатор впереди

$$(\forall x p(x)) \quad \exists y q(y) \tag{3.4}$$

сначала запишем формулу в эквивалентной форме

$$\neg(\forall x p(x)) \quad \exists y q(y).$$

Первый универсальный квантор теперь превращается в

$$(\exists x \neg p(x)) \quad \exists y q(y)$$

и теперь два квантификатора, наконец, могут быть выдвинуты вперед к

$$\exists x \exists y \neg p(x) \quad q(y),$$

что эквивалентно

$$\exists x \exists y p(x) \quad q(y).$$

Мы видим, что в (3.4) мы не можем просто выдвинуть оба квантора вперед. Скорее, мы должны сначала устранить импликации, чтобы не было отрицаний на кванторах . В общем случае считается, что мы можем извлекать кванторы только в том случае, если отрицания существуют только непосредственно в атомарных подформулах .

Пример 3.5. Как известно из анализа, сходимость ряда $(a_n)_{n \in \mathbb{N}}$ к пределу a определяется выражением

$$\forall \varepsilon > 0 \quad \exists n_0 \in \mathbb{N} \quad \forall n > n_0 \, |a_n - a| < \varepsilon.$$

С функцией abs(x) для $|x|$, a(n) для a_n, minus(x, y) для $x - y$ и предикатами el(x, y) для $x = y$, gr(x, y) для $x > y$ формула выглядит так

Machine Translated by Google

ε (gr(ε, 0) n0 (el(n0,N) n (gr(n, n0) gr(ε, abs(минус(a(n), a)))))) . (3.5)

Этоявно не в пренексной нормальной форме. Поскольку переменные внутренних кванторов n0 и n не встречаютсяслева от соответствующих кванторов, никакие переменные не должны переименовываться.Далее устраняемимпликации и получаем

ε (¬gr(ε, 0) n0 (¬el(n0,N) n (¬gr(n, n0) gr(ε, abs(минус(a(n), a)))))).

Поскольку каждое отрицание стоит перед атомарной формулой, мывыдвигаем квантификаторы вперед, удаляемлишние скобки и с помощью

ε n0 n (¬gr(ε, 0) ¬el(n0,N) ¬gr(n, n0) gr(ε, abs(минус(a(n), a))))

оностановитсяквантифицированнымпредложениемв конъюнктивной нормальной форме.

Преобразованнаяформула эквивалентна в ыкодной формуле. Тот факт, что это всегда возможна трансформациягарантируется

Теорема 3.2 Каждаяформула логики предикатов может быть преобразована в эквивалентную формулу в пренексной нормальной форме.

Кроме того, мыможем исключить все кванторысуществования.Однако формула, полученнаяв результате так называемой сколемизации , уже не эквивалентна формуле в ыкода. Однако его в ыполнимость остаетсянеизменной. В многих случаях , особенно когда нужнопоказать не в ыполнимость KB ¬Q, этого достаточно. Следующаяформула в пренексной нормальной форме теперь будет сколемизирована:

x1 x2 y1 x3 y2 p(f (x1), x2, y1) q(y1, x3, y2).

Поскольку переменнаяy1, по-в идимому, зависит от x1 и x2, каждое в хождение y1 заменяется функцией Скулема g(x1, x2). В ажно, что g — это новый функциональный символ, который еще не появилсяв формуле. Мыполучаем

x1 x2 x3 y2 p(f (x1), x2, g(x1, x2)) q(g(x1, x2), x3, y2)

и аналогично заменить y2 на h(x1, x2, x3), что приводит к

x1 x2 x3 p(f (x1), x2, g(x1, x2)) q(g(x1, x2), x3, h(x1, x2, x3)).

Поскольку теперь все переменные являютсяуниверсальными кванторами, универсальные кванторы можно опустить, что приведет к

p(f (x1), x2, g(x1, x2)) q(g(x1, x2), x3, h(x1, x2, x3)).

Machine Translated by Google

ПРЕОБРАЗОВАНИЕ НОРМАЛЬНОЙ ФОРМЫ(формула): 1.
Преобразование в преднексную нормальную
форму: Преобразование в конъюнктивную нормальную форму (теорема 2.1):
Устранение эквивалентностей.
Устранение последствий.
Повторное применение закона де Моргана и дистрибутивного закона.
Переименование переменных при необходимости.
Вынесение на множители универсальных кванторов.
2. Скулемизация
Замена экзистенциально кванторных переменных новыми функциями Скулема.
Удаление результирующих универсальных кванторов.

Рис. 3.2 Преобразование формул логики предикатов в нормальную форму

Теперь мы можем исключить квантор существования (а тем самым и квантор универсальности) в (3.5) на стр. 39 , введя функцию Скулема n0(ε). Таким образом, сколемизированный пренекс и конъюнктивная нормальная форма (3.5) на стр. 39 читаются

¬gr(ε, 0) ¬el(n0(ε),N) ¬gr(n, n0(ε)) gr(ε, abs(минус(a(n), a))).

Отбрасывая переменную n0, функция Скулема может получить имя n0.

При сколемизации формулы в предваренной нормальной форме все кванторы существования удаляются снаружи внутрь, где формула вида x1 ... xn у ф заменяется на x1 ... xn ф[y/f (x1,...,xn)], в течение которого f может не входить в ф.
Если квантор существования находится далеко снаружи, например, в ур(у), то у должен быть заменен константой (то есть нулевым функциональным символом).

Процедура преобразования формулы в конъюнктивную нормальную форму обобщена в псевдокоде, представленном на рис. 3.2. Скулемизация имеет полиномиальное время выполнения по количеству литералов. При преобразовании в нормальную форму количество литералов в нормальной форме может расти экспоненциально, что может привести к экспоненциальному времени вычислений и экспоненциальному использованию памяти. Причиной этого является многократное применение распределительного закона. Фактическая проблема, возникающая из-за большого количества предложений, заключается в комбинаторном взрыве пространства поиска для последующего доказательства разрешения. Однако существует оптимизированный алгоритм преобразования, который порождает только полиномиальное число литералов [Ede91].

3.4 Доказательства исчисления

Для рассуждений в логике предикатов были разработаны различные исчисления естественного мышления, такие как исчисление Генцена или исчисление секвенций. Как следует из названия, эти исчисления предназначены для применения людьми, поскольку правила вывода более

Machine Translated by Google

Таблица 3.2. Простое доказательство с использованием modus ponens и исключения квантора

В Б:	1	ребенок (Ева, Энн, Оскар)
В Б:	2	x y z дочерний(x, y, z) дочерний(x, z, y)
E(2) : x/eve,y/anne, z/oscar	3	ребенок(Ева, Энн, Оскар) ребенок(Ева, Оскар, Энн)
МП(1, 3)	4	ребенок (Ева, Оскар, Энн)

или менее интуитивным, а исчисления работают с произвольными формулами PL1. В следующем разделе мы в первую очередь сосредоточимся на исчислении резолюций, которое на практике наиболее важное эффективное, автоматизируемое исчисление для формул в конъюнктивном нормальном форма. Здесь, используя пример 3.2 на стр. 35 , мы дадим очень небольшое «естественное» доказательство. Используем правило вывода

$$\frac{A,\ A \quad B}{B}\ \text{(modus ponens, MP)} \quad \text{и} \quad \frac{\forall x\, A}{A[x/t]}\ (\forall\text{-устранение, } \forall E).$$

Modus ponens уже известен из логики высказываний. При устранении универсальные кванторы, необходимо иметь в виду, что квантор на переменная x должна быть заменяется основным термином t, что означает термин, который не содержит переменных . Доказательство представлен ребенок(ева, оскар, энн) из соответствующим образом сокращенной базы знаний в таблице 3.2.

Две формулы сокращенной базы знаний перечислены в строках 1 и 2. В в строке 3 исключаются кванторы в свободности из строки 2, а в строке 4 утверждается получен с помощью modus ponens.

Исчисление, состоящее из двух заданных правил вывода, не является полным. Однако его можно расширить до полной процедуры путем добавления дополнительных выводов . правила. Этот нетривиальный факт имеет фундаментальное значение для математики и ИИ. Австрийский логик Курт Гёдель доказал в 1931 г., что [Gö d31a]

> Теорема 3.3 (теорема Гёделя о полноте). Логика предикатов первого порядка – это полный. То есть существует исчисление, с помощью которого каждое предложение, являющееся следствием базы знаний KB может быть доказано. Если KB | ф, то выполняется что KB ф.

Следовательно, каждое истинное суждение в логике предикатов первого порядка доказуемо. Но это обратное тоже верно? Действительно ли все, что мы можем вывести синтаксически, истинно? ответ «да»:

> Теорема 3.4 (Корректность) Существуют исчисления, с помощью которых можно доказать только истинные предложения. То есть, если выполняется KB ф , то KB | ф.

Machine Translated by Google

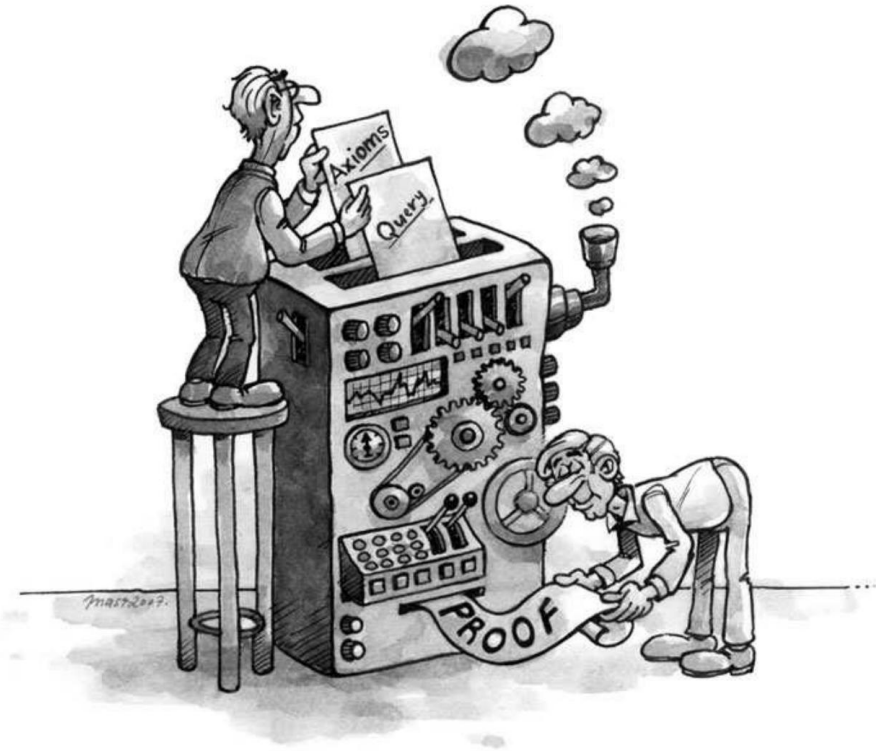


Рис. 3.3 Универсальная логическая машина

На самом деле почти все известные исчисления верны. Ведь работать смысла мало с неверными методами доказательства. Таким образом, доказуемость и семантическое следствие эквивалентные концепции, если используется правильное и полное исчисление. Тем самым упомянутые выше исчисления естественного вывода малопригодны для автоматизации.

Логика предикатов первого порядка становится мощным инструментом для математики и искусственного интеллекта.

Только исчисление резолюций, которое было введено в 1965 году и по существу работает с всего одно простое правило вывода позволило создать мощные автоматизированные доказательные устройства, которые позже использовались в качестве машин вывода для экспертных систем.

3.5 Разрешение

Действительно, правильное и полное исчисление резольвент вызвало логическую эйфорию в 1970-х годах . Многие ученые считали, что можно сформулировать почти любую задачу представления знаний и рассуждений в PL1, а затем решить его с помощью автоматизированного доказывающего. Логика предикатов, мощный, выразительный язык, вместе с исчисление полных доказательств казалось универсальной разумной машиной для представления знаний и решения многих трудных задач (рис. 3.3).

Machine Translated by Google

Если ввести набор аксиом (то есть базу знаний) и запрос в такую логическую машину в качестве входных данных, машина ищет доказательство и возвращает его — поскольку оно существует и будет найдено — в качестве вывода. Взяв за основу теорему Гёделя о полноте и работу Гербранда, многое было вложено в механизацию логики. Видение машины, которая могла бы произвольной непротиворечивой базой знаний PL1 доказать любой верный запрос, было очень заманчивым. Соответственно, до сих пор многие доказательные исчисления для PL1 разрабатываются и реализуются в виде доказательств теорем. В качестве примера здесь мы описываем исторически важное и широко используемое исчисление резолюций и показываемого возможности. Причиной выбора резолюции в качестве примера исчисления доказательств в этой книге, как уже говорилось, является ее историческая и дидактическая важность. Сегодня разрешение представляет собой лишь одно из многих исчислений, используемых в высокопроизводительных прувера х .

Мы начнем с попытки скомпилировать доказательство в таблице 3.2 на странице 41 с базой знаний из примера 3.2 в доказательство разрешения. Сначала формулы преобразуются в конъюнктивную нормальную форму и отрицательный запрос

$$\neg Q \quad \neg \text{ребенок}(\text{Ева}, \text{Оскар}, \text{Энн})$$

добавляется в базу знаний, что дает

$$\begin{aligned} KB \quad & \neg Q \quad (\text{ребенок}(\text{Ева}, \text{Энн}, \text{Оскар}))1 \\ & (\neg \text{дочерний}(x, y, z) \quad \text{дочерний}(x, z, y))2 \\ & (\neg \text{ребенок}(\text{ева}, \text{оскар}, \text{энн}))3. \end{aligned}$$

Тогда доказательство может выглядеть примерно так

$$\begin{aligned} & (2)\ x/eve,\ y/anne,\ z/oscar : (\neg child(eve, anne, oscar) \quad child(eve, oscar, \\ & \qquad anne))4 \\ & \qquad Res(3, 4): (\neg \text{ребенок}(\text{ева}, \text{Анна}, \text{Оскар}))5 \\ & \qquad Res(1, 5) : ()6, \end{aligned}$$

где на первом шаге переменные x,y,z заменяются константами. Затем следуют два шага разрешения при применении общего правила разрешения из (2.2), взятого без изменений из логики высказываний.

Обстоятельства в следующем примере несколько сложнее. Мы предполагаем, что каждый знает свою собственную мать , и спрашиваем, знает ли кто-нибудь Генри . С функциональным символом «мать» и предикатом «знает» мы должны вывести противоречие из

$$(\text{знает}(x, \text{мать}(x)))1 \quad (\neg \text{знает}(\text{Генри}, y))2.$$

Заменой x/henry, y/mother(henry) получаем пару противоречивых предложений

$$(\text{знает}(\text{Генри}, \text{мама}(\text{Генри})))1 \quad (\neg \text{знает}(\text{Генри}, \text{мама}(\text{Генри})))2.$$

Machine Translated by Google

Этот шаг замены называется унификацией. Два литерала дополняют друг друга, что означает, что они одинаковы, кроме своих знаков. Пустое предложение теперь выводится с помощью шага разрешения, с помощью которого было показано, что Генри делает знать кого-то (свою мать). Мы определяем

Определение 3.7. Два литерала называются унифицируемыми, если существует замена σ для всех переменных, что делает литералы равными. Такой σ называется унификатором. Унификатор называется наиболее общим унификатором (MGU), если все остальные унификаторы могут быть получается из него заменой переменных.

Пример 3.6. Мы хотим унифицировать литералы p(f (g(x)), y, z) и p(u, u, f (u)). Несколько унификаторов

σ1 : y/f (g(x)), z/f (f (g(x))), σ2 : x/h(v), y/f (g(h(v))), u/f (g(h(x))), z/f (f (g(h(v)))), σ3 : x/h(h(v)), y/f (g(h(h(v)))), z/f (f (g(h(h (v))))), u/f (g(h(h(v)))) u / f (g (h (v)))

σ4 : x/h(a), y/f (g(h(a))), z/f (f (g(h(a)))), σ5 : x/a, y/f (g (a)), z/f (f (g(a))), u/f (g(h(a))) u/f (g(a))

где σ1 — наиболее общий объединитель. Остальные унификаторы получаются из σ1 через замены x/h(v), x/h(h(v)), x/h(a), x/a.

На этом примере мы видим, что при унификации литералов предикатные символы могут рассматриваться как функциональные символы. То есть литерал рассматривается как термин. Реализации алгоритмов унификации обрабатывают аргументы функций последовательно. Термины унифицируются рекурсивно по структуре терминов. Простейшие алгоритмы унификации в большинстве случаев работают очень быстро. Однако в худшем случае время вычислений может экспоненциально расти с ростом членов. Потому что для автоматизированного доказывает, что подавляющее число попыток объединения терпят неудачу или очень просты, в в большинстве случаев сложность наихудшего случая не имеет драматического эффекта. Самая быстрая унификация алгоритмы имеют почти линейную сложность даже в худшем случае [Bib82].

Теперь мы можем дать общее правило разрешения для логики предикатов:

Определение 3.8 Правило разрешения двух предложений в конъюнктивной нормальной форме читает

$$\frac{(A_1 \vee \cdots \vee A_m \vee B), (\neg B' \vee C_1 \vee \cdots \vee C_n)\ \sigma(B) = \sigma(B')}{(\sigma(A_1) \vee \cdots \vee \sigma(A_m) \vee \sigma(C_1) \vee \cdots \vee \sigma(C_n))}, \quad (3.6)$$

где σ — MGU B и B′.

Machine Translated by Google

Теорема 3.5 Правило разрешения верно. То есть резольвента является семантическим следствием двух родительских предложений.

Однако для полноты нам все еще нужно небольшое дополнение, как показано на следующий пример.

Пример 3.7 Знаменитый парадокс Рассела гласит: «Есть парикмахер, который бреет всех, кто не бреется сам». Это утверждение противоречиво, то есть невыполнимо. Мы хотим показать это с решимостью. Парадокс, формализованный в PL1, гласит:

$$\exists x \, \text{бреет}(\text{парикмахер}, x) \leftrightarrow \neg \text{бреет}(x, x)$$

и преобразование в форму предложений дает (см. Упражнение 3.6 на стр. 55)

$$(\neg \text{бреет}(\text{цирюльник}, x) \vee \neg \text{бреет}(x, x))_1 \quad (\text{бреет}(\text{цирюльник}, x) \vee \text{бреет}(x, x))_2. \tag{3.7}$$

Из этих двух предложений мы можем вывести несколько тавтологий, но не противоречие. Таким образом, разрешение не является полным. Нам нужно еще одно правило вывода.

Определение 3.9 Факторизация предложения осуществляется путем

$$\frac{(A_1 \vee A_2 \vee \cdots \vee A_n) \quad \sigma(A_1) = \sigma(A_2)}{\sigma(A_2) \vee \cdots \vee \sigma(A_n))},$$

где σ — MGU A1 и A2.

Теперь из (3.7) можно вывести противоречие.

$$\begin{aligned} &\text{Fak}(1, \sigma : x/\text{парикмахер}) : (\neg \text{shaves}(\text{парикмахер}, \text{парикмахер}))_3 \\ &\text{Fak}(2, \sigma : x/\text{парикмахер}) : (\text{бреет}(\text{парикмахер}, \text{парикмахер}))_4 \\ &\qquad\qquad \text{Рез}(3, 4) : ()_5 \end{aligned}$$

и мы утверждаем:

Теорема 3.6. Правило разрешения (3.6) вместе с правилом факторизации (3.9) является полным опровержением. То есть, применяя этапы факторизации и разрешения, пустое предложение может быть получено из любой невыполнимой формулы в конъюнктивной нормальной форме.

Machine Translated by Google

3.5.1 Стратегии разрешения

Хотя полнота разрешения важна для пользователя, поиск доказательства на практике может быть очень утомительным. Причиной этого является огромное комбинаторное пространство поиска. Даже если вначале в KB ¬Q очень мало пар предложений, доказывающая сторона генерирует новое предложение с каждым шагом разрешения, что увеличивает количество возможных шагов разрешения на следующей итерации. Таким образом, уже давно предпринимались попытки сократить пространство поиска с помощью специальных стратегий, желательно без потери полноты. Наиболее важные стратегии следующие.

Модульное разрешение отдает приоритет шагам разрешения, в которых одно из двух предложений состоит только из одного литерала, называемого модульным предложением. Эта стратегия сохраняет полноту и приводит во многих случаях, но не всегда, к сокращению пространства поиска. Следовательно, это эвристический процесс (см. раздел 6.3).

Гарантированное сокращение пространства поиска достигается применением стратегии множества опор. Здесь подмножество KB ¬Q определяется как множество поддержки (SOS).
Каждый шаг разрешения должен включать пункт из SOS, и резольвента добавляется в SOS. Эта стратегия неполная. Она становится полным, когда гарантируется, что множество предложений выполнимо без SOS (см. упражнение 3.7 на стр. 55). Отрицательный запрос ¬Q часто используется в качестве начального сигнала SOS.

При разрешении входных данных предложение из входного набора KB ¬Q должно участвовать в каждом шаге разрешения. Эта стратегия также сокращает пространство поиска, но за счет полноты.

С помощью правила чистого литерала все предложения, содержащие литералы, для которых нет дополнительных литералов в других предложениях, могут быть удалены. Это правило сокращает пространство поиска и является полным, поэтому оно используется практически в семи резольвентными прувերами.

Если литералы предложения K1 представляют собой подмножество литералов предложения K2, то K2 можно удалить. Например, оговорка

(дождь (сегодня) street_wet (сегодня))

является избыточным, если street_wet(today) уже действителен. Этот важный шаг сокращения называется включением. Подведение тоже полное.

3.5.2 Равенство

Равенство — особенно неудобная причина взрывного роста пространства поиска.
Если мы добавим (3.1) на стр. 36 и аксиомы равенства, сформулированные в (3.2) на стр. 36 в базу знаний, то предложение симметрии ¬x = y y = x может быть объединено с каждым положительным или отрицательным уравнением, например . Это приводит к выводу новых предложений и уравнений, к которым снова можно применить аксиомы равенства, и так далее. Аналогичные следствия имеют аксиомы транзитивности и подстановки. Поэтому были разработаны специальные правила вывода на равенство, которые обходятся без явных аксиом равенства и, в частности, сокращают пространство поиска. демодуляция;

Machine Translated by Google

например, позволяет заменить термин t2 на t1, если уравнение t1 = t2 существует. Ан
уравнение t1 = t2 применяется посредством унификации к терму t следующим образом:

$$\frac{t1 = t2, \quad (\ldots t \ldots), \sigma(t1) = \sigma(t)}{(\ldots \sigma(t2) \ldots)}.$$

Несколько более общим является парамодуляция, которая работает с условными уравнениями [Bib82, Lov78].

Уравнение t1 = t2 допускает замену члена t1 на t2 , а также
замена t2 на t1. Обычно бессмысленно отменять замену, которая уже
было выполнено. Наоборот, уравнения часто используются для упрощения терминов.
Таким образом, они часто используются только в одном направлении. Уравнения, которые используются только
одного направления называются направленными уравнениями. Эффективная обработка направленных
уравнений осуществляется с помощью так называемых систем перезаписи терминов. Для формул со многими
уравнений существуют специальные доказатели равенства.

3.6 Автоматические средства доказательства теорем

Реализации вычислений вывода на компьютерах называются средствами доказательства теорем. В доль
со специализированными прувера ми для подмножеств PL1 или специальных приложений, сегодня существуют
целая линейка автоматических доказательств для полной логики предикатов и логики более высокого
порядка, из которых здесь будут обсуждаться лишь некоторые из них . Обзор наиболее важных
систем можно найти в [McC].

Один из старейших прувера разрешения был разработан в Аргоннской национальной лаборатории в
Чикаго. На основе ранних разработок, начиная с 1963 г., Otter [Kal01],
была создана в 1984 году. Прежде всего, Otter успешно применялась в специализированных областях
математики, как можно узнать на его домашней странице:

> «В настоящее время основное применение Оттера — исследования в области абстрактной алгебры и формальной логики.
> Otter и его предшественники использовались для ответа на многие открытые вопросы в области
> конечные полугруппы, тернарная булева алгебра, логические исчисления, комбинаторная логика, теория групп,
> теория решеток и алгебраическая геометрия».

Несколько лет спустя Технологический университет Мюнхена создал высокопроизводительный
прувер SETHEO [LSBB92], основанный на быстрой технологии PROLOG. С
Для достижения еще более высокой производительности была разработана реализация для параллельных
компьютеров под названием PARTHEO. Выяснилось, что не стоит использовать специальное оборудование
в доказателях теорем, как и в других областях .
искусственного интеллекта, потому что эти компьютеры очень быстро уступают место более быстрым процессорам и
более интеллектуальные алгоритмы. Мюнхен также является родиной E [Sch02], отмеченного наградами
современного средства доказательства уравнений, с которым мы познакомимся в следующем номере.
пример. На домашней странице E можно прочитать следующую компактную ироничную характеристику:

Machine Translated by Google

ции, в тораячасть которой, между прочим, применима ков семсуществующим автоматическим прувера м. сегодня

> «E — чистоэквациональный доказатель теорем длякла узальной логики. Это означает, что это программа, которая выможете запих нуть туда математическую спецификацию (в клаузальной логике с равенством) и гипотезу, которая затем будет работать вечно, используявсе ресурсывашей машины. Очень времяот времени он нах одит подтверждение гипотезыи сообщает вамоб этом ;-)».

Нах одить доказательства истинных утверждений, по-видимому, настолько сложно, что поиски увен нчаются успехом лишь в крайне редких случаях или только по прошествии очень длительного времени — если вообще увен нчаются успехом. мы пойдем в об этом более подробно в гл. 4. Здесь следует, однако, упомянуть, что не только компьютеры, нои у большинства людей возникают проблемыс поиском строгих формальных доказательств.

Хотяочевидно, что компьютерысами посебе во многих случаях не в состоянии найти доказательство, лучше всего построить системы, работающие полуавтоматически. и разрешить тесное сотрудничество с пользователем. Тем самым человек может лучше применять свои знание специальных областей применения и, возможно, ограничить поиск доказательства. Одним из наиболее успешных интерактивных средств доказательства логики предикатов высшего порядка является Isabelle [NPW02], совместный продукт Кембриджского университета и Университета Нью-Йорка. Технология Монтен.

Любой, кто ищет высокопроизводительный прувер, должен ознакомиться с текущими результатами CASC (CADE ATP System Competition) [SS06]. Здесь мы нах одим, что победителем с 2001 по 2006 год в категориях PL1 и нормальной формы статьи был Вампир из Манчестера, который работает с вариантом разрешения и специальным подход к равенству. Система Вальдмейстера Института Макса Планка в Саарбрюккене уже много лет лидирует в области доказательства равенства.

Многие высокие позиции немецких систем в CASC показывают, что немецкие исследования группы в области автоматизированного доказательства теорем играют ведущую роль, сегодня так и в прошлом.

3.7 Математические примеры

Теперь мы хотим продемонстрировать применение автоматизированного прувера с вышеупомянутым прувером E [Sch02]. E — специализированный доказатель равенства, который значительно сокращает пространство поиска через оптимизированную обработку равенства.

Мы хотим доказать, что левые и правые нейтральные элементы в полугруппе равны. Сначала шаг за шагом формализуем иск.

1CADE — это ежегодная «Конференция по автоматизированному выводу» [CAD], а ATP означает «Автоматизированное доказательство теорем».

Machine Translated by Google

Определение 3.10. Структура (M, ·), состоящая из множества M с двуместной внутренней операцией «·», называется полугруппой, если выполняется закон ассоциативности

$$x \quad y \quad z\,(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

держит. Элемент e M называется левой-нейтральным (правой-нейтральным), если xe · x = x (xx · e = x).

Осталось показать, что

Теорема 3.7. Если в полугруппе есть нейтральный слева элемент el и нейтральный справа элемент er , то el = er .

Сначала мы докажем теорему полуформально с помощью интуитивных математических рассуждений. Ясно, что для всех x M верно

$$\text{эл} \cdot x = x \tag{3.8}$$

и

$$x \cdot \text{эр} = x\,. \tag{3.9}$$

Если положить x = er в (3.8) и x = el в (3.9), мы получим два уравнения el · er = er и el · er = el . Объединение этих двух уравнений дает

$$\text{эл} = \text{эл} \cdot \text{эр} = \text{эр},$$

что мы от им доказать. Кстати, на последнем шаге мы воспользовались тем, что равенство симметрично и транзитивно.

Прежде чем применять автоматический прувер, мы проводим доказательство разрешения вручную . Сначала мы формализуем отрицательный запрос и базу знаний KB, состоящую из аксиом в виде предложений в конъюнктивной нормальной форме:

(¬el = er)1 отрицательный запрос
(m(m(x, y), z) = m(x, m(y, z)))2 (m(el,x)
= x)3 (m(x, er) =
x)4

аксиомы

равенства: (x = x)5
(¬x = y y = x)6 (¬x = y ¬y = z
x = z)7 (¬x = y m(x, z) = m (y, (рефлексивность)
z))8 (¬x = y m(z, x) = m(z, y))9 (симметрия) (транзитивность) замена в m замен

Machine Translated by Google

где умножение представлено двузначным функциональным символом m. Аксиомы равенства были сформулированы аналогично (3.1) на стр. 36 и (3.2) на стр. 36.
Простое резольвентное доказательство имеет вид

```
Res(3, 6, x6/m(el, x3), y6/x3): Res(7, 10,          (x  = т(эл, х ))10
x7/x10, y7/m(el, x10)): (¬m(el,x) = z     x  = r)11
Res(4, 11, x4/el, x11/er, z11/el): (er = el)12
Рез(1, 12,    ): ().
```

Здесь, например, Res(3, 6, x6/m(el, x3), y6/x3) означает, что в разрешении пункта 3 с пунктом 6 переменная x из пункта 6 заменена на m(el, x3) с переменная x из пункта 3. Аналогично, y из пункта 6 заменяется на x из пункт 3.

Теперь мы хотим применить к задаче доказательство E. Пункты трансформируются в предложение языка нормальной формы LOP через отображение

```
(¬A1  ···  ¬Am   B1  ···  Bn)   B1;...;Bn <   A1,...,Am.
```

Синтаксис LOP представляет собой расширение синтаксиса PROLOG (см. раздел 5) для оговорки Хорна. Таким образом, мы получаем в качестве входного файла для E

```
<- эл = эр.                                  % запрос
m(m(X,Y),Z) = m(X,m(Y,Z)) . m(el,X) = X m(X,er)   % ассоциативности m
= X                .                         % левонейтральный элемент m
                   .                         % правонейтральный элемент m
```

где равенство моделируется предикатным символом gl. В выводе доказывающего доставляет

```
unixprompt> eproof halbgr1a.lop
# Статус проблемы определен, построен объект доказательства
# Свидетельство о статусе проблемы начинается
        0 : [--equal(el, er)] : начальный
        1 : [++equal(m(el,X1),X1)] : начальный
        2 : [++equal(m(X1,er), X1)] : начальный
        3 : [++равно(эл, э)] : pm(2,1)
        4 : [--equal(el, el)] : rw(0,3)
        5 : [] : cn(4)
        6 : [] : 5 : {доказательство}
# Свидетельство о статусе проблемы заканчивается
```

Положительные литералы обозначаются знаком ++, а отрицательные — знаком --. В строках с 0 по 4, отмеченные начальным, снова перечисляются предложения из входных данных . м(а, б) обозначает шаг разрешения между пунктом а и пунктом б. Мы видим, что доказательство

Machine Translated by Google

найденное Е очень похоже на созданное вручную доказательство. Поскольку мы явно смоделировать равенство предикатом gl, особые сильные стороны Е не приходят в игру. Теперь опускаем аксиомы равенства и получаем

```
<- эл = эр.                                          % Запрос
m(m(X,Y),Z) = m(X,m(Y,Z)) . m(el,X) = X m(X,er) = X  % Ассоциативит\"{a}т в. м
                                .                    % ссылок нейтральных эл. в. м
                                .                    % rechtsneutrales Эл. в. м
```

в качестве входных данных для доказывающего.

Доказательство также становится более компактным. Мы видим в следующем выводе доказывает, что доказательство по существу состоит из одного шага вывода под вум соответствующим пунктам 1 и 2.

```
unixprompt> eproof halbgr1a.lop
# Статус проблемы определен, построен объект доказательства
# Свидетельство о статусе проблемы начинается
        0 : [--equal(el, er)] : начальный
        1 : [++equal(m(el,X1),X1)] : начальный
        2 : [++equal(m(X1,er), X1)] : начальный
        3 : [++равно(эл, э)] : pm(2,1)
        4 : [--equal(el, el)] : rw(0,3)
        5 : :сп(4)[]
        6 :      []      : 5 : {доказательство}
# Свидетельство о статусе проблемы заканчивается
```

Теперь читатель может поближе познакомиться с возможностями Е (упражнение 3.9 на стр. 55).

3.8 Приложения

В математике автоматические средства доказательства теорем используются для определенных специализированных задач. Например, важная теорема теории графов о четырех красках была впервые доказана в 1976 г. с помощью специального прувера. Тем не менее, автоматизированные прувер ыпо-прежнему играют второстепенную роль. роль в математике.

С другой стороны, на заре ИИ логика предикатов имела большое значение. для разработки экспертных систем в практических приложениях. Из-за своих проблем моделирования неопределенности (см. раздел 4.4), экспертные системы сегодня чаще всего разрабатываются используя другие формализмы.

Сегодня логика играет все более важную роль в задачах проверки. автоматический верификация программ в настоящее время является важной областью исследований между ИИ и программной инженерией. Все более сложные программные системы теперь берут на себя задачи

Machine Translated by Google

все большей и большей ответственности и значимости длябезопасности. Здесь желательно подтверждение определенных характеристик безопасности программы. Такое доказательство не может быть получено путем тестированияготовой программы, так как в общем невозможно применить программу ко всем возможным входам. Поэтому это идеальнаяобласть дляобщих или даже специализированных систем вывода. Среди прочего, сегодняиспользуются криптографические протоколы, характеристики безопасности которых проверены автоматически [FS97, Sch01]. Еще одной проблемой использованияавтоматизированных пруверов является синтез программного и аппаратного обеспечения. С этой целью, например, прувер должны поддерживать инженера-программиста в создании программ на основе спецификаций.

Повторное использование программного обеспечениясегодняттакже имеет большое значение длямногих программистов. Программист ищет программу, котораяберет входные данные с определенными свойствами и вычисляет результат с желаемыми свойствами. Алгоритм сортировки принимает входные данные с записями определенного типа данных и из них создает перестановку этих записей со свойством, что каждый элемент меньше или равен следующему элементу.
Программист сначала формулирует спецификацию запроса на PL1, состоящую из двух частей. Первая часть PREQ содержит предварительные условия, которые должныбыть соблюдены, прежде чем желаемая программа будет применена. Вторая часть POSTQ содержит постусловия, которые должны выполнятьсяпосле применения желаемой программы.

На следующем этапе в базе данных программного обеспечениянеобходимонайти модули, отвечающие этим требованиям. Дляформальной проверки в базе данных должнохраниться формальное описание предусловий PREM и постусловий POSTM длякаждого модуляM.
Предположение о возможностях модулей состоит в том, что предварительные условиямодуляследуют из предварительных условий запроса. Он должен держать это

PREM PREQ.

Все условия, которые требуются в качестве предварительных условий дляприменениямодуляM, должны отображаться в качестве предварительных условий в запросе. Если, например, модуль в базе данных принимает только списки целых чисел, то списки целых чисел в качестве входных данных также должны отображаться в качестве предварительных условий в запросе. Дополнительное требование в запросе, чтобы, например, отображались только четные числа, не вызывает проблемы.

Кроме того, оно должно выполнятьсядляпостусловий,

POSTM POSTQ.

То есть после применениямодуля должны быть выполнены все атрибуты, которые требует запрос. Теперь мы покажем применение средства доказательства теорем к этой задаче на примере из [Sch01].

Пример 3.8 VDM-SL, Венский язык спецификации методов разработки, часто используется в качестве языка дляспецификации пред- и постусловий. Предположим, что в базе данных программы имеется описание модуля ROTATE , который перемещает первый элемент списка в конец списка. Ищем модуль SHUFFLE, который создает произвольную перестановку списка. Две спецификации гласят

Machine Translated by Google


```
ПОВЕРНУТЬ(l : список) l′ : список
предварительноверно
почта
    (л = [] л ′ = [])
    (л = [] л ′ = (х вост l)^[голова l])
```

```
SHUFFLE(x : Список) x′ : Список
предварительноверно
пост i : Пункт·
    ( x1, x2 : Список · x = x1^[i]^x2
      y1, y2 : Список · x′ = y1^[i]^y2)
```

Здесь «^» обозначает объединение списков, а «·» разделяет квантификаторыс их переменные из остальной части формулы. Функции «голова л» и «х вост л» выберите первый элемент и остальные из списка соответственно. Спецификация SHUFFLE указывает, что каждый элемент списка i, который был в списке (x) до применение SHUFFLE должно быть в результате (x ′) после применения, и вице наоборот. Теперь нужно показать, что формула (PREQ PREM) (POSTM POSTQ) является следствием базы знаний, содержащей описание Тип данных Список. Две спецификации VDM-SL дают задачу проверки

```
l,l′ ,x,x′ : список · (l = x l ′ ′ = x (ш щ))
  (l = x l ′ = x ′ ((l = [] l ′ = []) (l = [] l ′ = (tl l)^[hd l])
      i : Элемент · ( x1, x2 : Список · x = x1^[i]^x2 y1, y2 : Список · x ′ = y1^[i]^y2))),
```

что затем может быть доказано с помощью прувера SETHEO.

В ближайшие годы семантическая сеть, вероятно, станет важным приложением PL1. Контент Всемирной паутины должен стать интерпретируемым не только для людей, но и для машин. С этой целью веб-сайты оснащаются описание их семантики на языке формального описания. The search for тем самым информация в сети станет значительно эффективнее, чем сегодня, где по существу доступны только текстовые стандартные блоки.

В качестве языков описания используются разрешимые подмножества логики предикатов. Разработка эффективных исчислений для рассуждений очень важна и тесно связана с языками описания. Запрос для будущей семантически работающей поисковой системы мог бы (неофициально) звучать так: «Где в Швейцарии в следующее воскресенье на высотах ниже 2000 метров будет ли хорошая погода и оптимально подготовленные горнолыжные трассы?» Чтобы ответить на такой вопрос, требуется исчисление, способное работать очень быстро. на больших наборах фактов и правил. Здесь сложные вложенные функциональные термы менее важны.

В качестве базовой структуры описания Консорциум World Wide Web разработал язык RDF (Структура описания ресурсов). Основанный на RDF значительно более мощный язык OWL (Web Ontology Language) позволяет описывать отношения между объектами и классами объектов, аналогично PL1 [SET09].
Онтологии — это описания отношений между возможными объектами.

Machine Translated by Google

Сложность при построении описаниябесчисленных веб-сайтов заключаетсяв затратах труда, а также проверке правильности семантических описаний.
Здесь могут очень помочь системымашинного обучениядляавтоматической генерации описаний.
Интересное использование «автоматической» генерации семантики в сети было предложено Луисом фонАномиз Университета Карнеги-Меллона [vA06]. Онразработал компьютерные игры, в которых игроки, распределенные по сети, должнысовместноописывать картинки ключевыми словами. Таким образом, изображениям присваиваетсясемантика в увлекательной игровой форме бесплатно. Читатель может протестировать игрыи послушать выступление изобретателяв упражнении 3.10 на странице 55.

3.9 Резюме

Мыпредоставили наиболее важные основы, терминыи процедурылогики предикатов и показали, что даже одна из самых сложных интеллектуальных задач, а именнодоказательствоматематических теорем, может быть автоматизирована. Автоматизированные пруверымогут применятьсяне только в математике, но и, в частности, в задачах проверки информатики. Однакодляповседневных рассуждений логика предикатов в большинстве случаев непригодна. В следующей и следующих главах мыпокажемего слабые места и некоторые интересные современные альтернативы. Кроме того, мыпокажем в гл. 5, что можноэлегантно программировать с помощью логики и ее процедурных расширений.

Любой, кто интересуетсялогикой первого порядка, разрешениеми другими вычислениями для автоматических пруверов, найдет хорошие дополнительные инструкции в [New00, Fit96, Bib82, Lov78, CL73]. Ссылки на интернет-ресурсыможно найти на веб-сайте этой книги.

3.10 Упражнения

Упражнение 3.1. Пусть данытрехместное сказуемое «ребенок» и одноместное сказуемое «женский пол» из примера 3.2 на стр. 35 . Определите: (a) Одноместный предикат «мужской». б) двуместное сказуемое «отец» и «мать». (c) Двухместное сказуемое «братьяи сестры». (d) Предикат «родители (x, y, z)», который истинен тогда и только тогда, когда x — отец, а y является матерью z.

(e) Предикат «uncle(x, y)», который истинентогда и только тогда, когда x является дядей y (используйте уже определенные предикаты).

(e) Двухместное сказуемое «предок» со значением: предки — это родители, дедушки и бабушки и т. д. произвольномногих поколений.

Упражнение 3.2 Формализуйте следующие утвержденияв логике предикатов: (a) У каждого человека есть отец и мать. б) У некоторых людей есть дети. в) Все птицылетают.

Machine Translated by Google

(d) Е сть жив отное, кот орое ест (нек от орых) жив отных , пит ающ их сяз ерном.

(e) К аждое жив отное пит ается раст ениями или трав оядными жив отными, кот орые намного меньше, ч ем сам.

У пражнение 3.3 Адапт ируйт е У пражнение 3.1 на стр. 54 , исполь зуя одноместные функциональные симв олы и равенст во вместо «от ец» и «мат ь».

У пражнение 3.4. Прив едит е аксиомы логики предикат ов для дв уместного от ношения «<» как полного порядка. Для полного порядка мы должны иметь (1) Люб ые дв а элемент а сравнимы. (2) Он симмет ричен. (3) Он транзит ивен.

У пражнение 3.5. У нифицируйт е (если в озможно) след ующ ие т ермины и укажит е MGU и полученные т ермины. (a) p(x, f (y)), p(f (z), u) (б) p(x, f (x)), p(y, y) (c) x = 4 7 · x, cos y = z (d) x < 2 · x, 3 < 6 (e) q(f (x, y, z), f (g(w, w), g(x, x), g(y, y))), q(и, и)

У пражнение 3.6

(a) Прео бразуйт е парадокс Р ассела из примера 3.7 на стр. 45 в К НФ. (b) Покажит е, ч то пуст ое предложение не может быт ь получено с помощью разрешения без факт оризации из (3.7) на стр. 45. Попыт айт есь понят ь это инт уит ивно.

У пражнение 3.7

(a) Поч ему резольв ента с набором ст рат егий поддержки яв ляет ся не полной? (b) Обосн уйт е (без доказат ельст ва), поч ему набор ст рат егий поддержки ст ановит ся полным, если (KB ¬Q)\SOS в ыполнима.

в) Поч ему разрешение с помощью прав ила чист ого букв ального в ыраж ения яв ляет ся полным?

У пражнение 3.8 Формализуйт е и с т очност ью докажит е, ч т о в полугруппе, содерж ащ ей по крайней мере дв а разл ичных элемент а a,b, лев ый нейт ральный элемент e и лев ый нулев ой элемент n, эт и дв а элемент а должны быт ь разл ичными, т. е. ч т о n = e. Исполь зуйт е демодуляцию, поз в оляющ ую замен ить «подобное на подобное».

У пражнение 3.9. Получит е средст во доказат ельст ва т еорем E [Sch02] или другое средст во доказат ельст ва и докажит е след ующ ие ут в ерж дения. Сравнит е эт и доказат ельст ва с прив еденными в т ексте. (a) У т в ерж дение из примера 2.3 на ст ранице 24. (b) Парадокс Р ассела из примера 3.7 на ст ранице 45. (c) У т в ерж дение из упражнения 3.8.

У пражнение 3.10 Прот ест ируйт е игры www.espgame.org и www.peekaboom.org, кот орые помогают определить семант ику изображений в сет и. Послушайт е лекцию Луиса фон Ана о чел ов еческих в ыч ислениях по ссылке: http://video.google.de/videoplay?docid= -8246463980976635143.

Machine Translated by Google

Machine Translated by Google

Ограничения логики

4

4.1 Проблема пространства поиска

Как уже упоминалось в нескольких местах, при поиске доказательства почти всегда имеется множество (в зависимости от исчисления, потенциально бесконечно много) возможностей применения правил вывода на каждом шаге. Результатом является уже упомянутый взрывной рост пространства поиска (рис. 4.1 на стр. 58). В худшем случае нужно перепробовать все эти возможности, чтобы найти доказательство, что обычно невозможно за разумное время.

Если мы сравним автоматические доказывающие или системы логического вывода с математиками или людьми-экспертами, имеющими опыт в специальных областях, мы сделаем интересные наблюдения. Во-первых, опытные математики могут доказывать теоремы, недоступные для автоматических доказательств. С другой стороны, автоматические прувер ы выполняют десятки тысяч выводов в секунду. Напротив, человек выполняет, возможно, одно умозаключение в секунду. Хотя люди-эксперты гораздо медленнее на объектном уровне (то есть в выполнении выводов), они, по-видимому, решают сложные задачи гораздо быстрее.

На это есть несколько причин. Мы, люди, используем интуитивные вычисления, которые работают на более высоком уровне и часто выполняют многие простые выводы автоматизированного доказывающего за один шаг. Кроме того, мы пользуемся леммами, т. е. выводимыми истинными формулами, которые нам уже известны и поэтому не нужно каждый раз их доказывать заново. Между тем есть и машинные прувер ы, работающие с такими методами. Но даже они пока не могут конкурировать с людьми-экспертами.

Еще одним, гораздо более важным преимуществом нас, людей, является интуиция, без которой мы не смогли бы решить ни одной сложной задачи [PS09]. Попытка формализовать интуицию вызывает проблемы. Опыт прикладных проектов ИИ показывает, что в таких сложных областях, как медицина (см. раздел 7.3) или математика, большинство экспертов не в состоянии сформулировать эти интуитивные метазнания словесно, не говоря уже о том, чтобы их формализовать. Поэтому мы не можем запрограммировать это знание или интегрировать его в исчисления в форме эвристики. Эвристика – это методы, которые во многих случаях могут значительно упростить или сократить путь к цели, но в некоторых случаях (обычно редко) могут сильно удлин

В. Эртель, Введение в искусственный интеллект, Темы бакалавриата по компьютерным наукам, DOI 10.1007/978-0-85729-299-5_4, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google



Рис. 4.1 Возможные последствия взрыва поискового пространства

к цели. Эвристический поиск важен не только для логики, но и для решения задач в области ИИ в целом, и поэтому он будет подробно рассмотрен в гл. 6.

Интересный подход, который применяется примерно с 1990 года, заключается в применении методов машинного обучения к изучению эвристик для управления поиском систем логического вывода, который мы сейчас кратко обрисуем. У доказывающего при поиске доказательства есть сотни и более возможностей для шагов разрешения на каждом шаге, но только некоторые из них ведут к цели. Было бы идеально, если бы доказывающий мог спросить оракула, какие два предложения он должен использовать на следующем шаге, чтобы быстро найти доказательство. Предпринимаются попытки построить такие направляющие доказательства модули, которые оценивают различные альтернативы для следующего шага, а затем выбирают альтернативу с лучшим рейтингом. В случае разрешения рейтинг доступных предложений может быть вычислен с помощью функции, которая вычисляет значение на основе количества положительных литералов, сложности терминов и т. д. для каждой пары разрешимых предложений.

Как можно реализовать эту функцию? Поскольку это знание «интуитивно», программист с ним не знаком. Вместо этого пытаются скопировать природу и используют алгоритмы машинного обучения, чтобы учиться на успешных доказательствах [ESS89, SE90]. Атрибуты всех пар предложений, участвующих в успешных шагах разрешения, сохраняются как положительные, а атрибуты всех неудачных разрешений сохраняются как отрицательные.
Затем, используя эти обучающие данные и систему машинного обучения, создается программа, которая может эвристически оценивать пары предложений (см. раздел 9.5).

Machine Translated by Google

Другой, более успешный подход к совершенствованию математических рассуждений заключается в следующем. следуют интерактивные системы, которые работают под контролем пользователя. Вот один могли назвать программы компьютерной алгебры, такие как Mathematica, Maple или Maxima, который может автоматически выполнять сложные символьные математические манипуляции. Однако поиск доказательств остается полностью за человеком. Вышеупомянутый интерактивный прувер Isabelle [NPW02] обеспечивает значительно большую поддержку во время доказательства. поиск. В настоящее время существует несколько проектов, таких как Omega [SB04] и MKM1 для разработка систем поддержки математиков во время доказательств.

Подводя итог, можно сказать, что из-за проблемы пространства поиска автоматизированные сегодня доказывающие могут доказывать только относительно простые теоремы в специальных областях с мало аксиом.

4.2 Разрешимость и неполнота

Логика предикатов первого порядка предоставляет мощный инструмент для представления знаний и рассуждений. Мы знаем, что существуют правильные и полные исчисления и средства доказательства теорем. При доказательстве теоремы, т. е. истинного утверждения, такой доказатель очень полезно, потому что из-за полноты через конечное время известно, что утверждение действительно правда. Что делать, если утверждение неверно? Теорема о полноте (теорема 3.3 на стр. 41) не дает ответа на этот вопрос2. В частности, не существует процесса которое может доказать или опровергнуть любую формулу из PL1 за конечное время, так как верно, что

> Теорема 4.1. Множество правильных формул в логике предикатов первого порядка полуразрешимо.

Из этой теоремы следует, что существуют программы (доказательства теорем), которые при истинная (действительная) формула в качестве входных данных, определите ее истинность за конечное время. Если формула не действительно, однако может случиться так, что доказывающий никогда не остановится. (Читатель может ответить на этот вопрос в упражнении 4.1 на стр. 65.) Логика высказываний разрешима. потому что метод таблицы истинности предоставляет все модели формулы за конечное время. Очевидно, что логика предикатов с кванторами и вложенными функциональными символами — это язык несколько слишком мощный, чтобы быть разрешимым.

С другой стороны, логика предикатов недостаточно мощна для многих целей. Часто хочется сделать утверждения о наборах предикатов или функций. Этот не работает в PL1, потому что он знает квантификаторы только для переменных, но не для предикаты или функции.

Курт Гёдель вскоре после своей теоремы о полноте для PL1 показал, что полнота теряется, если мы расширяем PL1 даже минимально для построения логики более высокого порядка. Логика первого порядка может количественно оценивать только переменные. Логика второго порядка также может

1 www.mathweb.org/mathweb/demo.html.

2 Именно этот случай особенно важен на практике, потому что если я уже знаю, что утверждение истинно, Мне больше не нужен прувер.

Machine Translated by Google

квантифицировать по формулам первого порядка, а логика третьего порядка может квантифицировать по формулам второго порядка. Даже добавление только аксиомы индукции для натуральных чисел делает логику неполной. Утверждение «Если для n выполняется предикат p(n), то верно и p(n + 1)», или

$$\forall p\,(p(n) \rightarrow p(n+1))$$

является предложением второго порядка, потому что оно квантифицируется над предикатом. Гёдель доказал следующую теорему:

> **Теорема 4.2 (теорема Гёделя о неполноте)** Любая система аксиом для натуральных чисел со сложением и умножением (арифметика) неполна.
> То есть в арифметике есть истинные утверждения, которые недоказуемы.

Доказательство Гёделя работает с так называемой гёделизацией, при которой каждая арифметическая формула кодируется как число. Он получает уникальный номер Гёделя. Гёделизация теперь используется для формулировки предложения

$$F = \text{«Я недоказуем»}.$$

на языке арифметики. Эта формула верна по следующей причине. Предположим, что F ложно. Тогда мы можем доказать F и, следовательно, показать, что F недоказуемо. Это противоречие. Таким образом, F истинно и, следовательно, недоказуемо.

Более глубокая подоплека этой теоремы заключается в том, что математические теории (системы аксиом) и, в более общем смысле, языки становятся неполными, если язык становится слишком мощным. Аналогичный пример — теория множеств. Этот язык настолько силен, что с его помощью можно формулировать парадоксы. Это утверждения, противоречащие сами себе, такие как уже известное нам из примера 3.7 на стр. 45 утверждение о парикмахерах, которые бреют тех, кто не бреется сам (см. упражнение 4.2 на стр. 65)[3]. Дилемма состоит в том, что с языками, которые достаточно сильны, чтобы описать математику и интересные приложения, мы проносим противоречия и неполноты через черный ход. Однако это не означает, что логики высших порядков совершенно непригодны для формальных методов. Безусловно, существуют формальные системы, а также средства доказательства для логики более высокого порядка.

4.3 Летающий пингвин

На простом примере мы продемонстрируем фундаментальную проблему логики и возможные пути решения. Учитывая заявления

[3] Многих других логических парадоксов можно найти у [Wie].

Machine Translated by Google



Рис. 4.2 Летающий пингвин Твити

1. Твити — пингвин2.
Пингвины — птицы3.
Птицы умеют
летать Формализованные в PL1 результаты базы знаний KB :

$$\text{пингвин}(\text{твити})$$
$$\text{пингвин}(x) \quad \text{птица}(x)$$
$$\text{птица}(x) \quad \text{летать}(x)$$

Отсюда (например, с разрешением) можно вывести fly(tweety) (рис. 4.2)4. Очевидно, формализация летных признаков пингвинов недостаточна. Пробуем дополнительное утверждение: пингвины не могут летать, т. е.

$$\text{пингвин}(x) \quad \neg\text{муха}(x)$$

Отсюда можно вывести ¬fly(tweety) . Но fly(tweety) по-прежнему верен. Поэтому база знаний несовместима. Здесь мы отмечаем важную характеристику логики, а именно монотонность. Хотя мы прямо заявляем, что пингвины не умеют летать, все же можно вывести обратное.

Определение 4.1. Логика называется монотонной, если для произвольной базы знаний KB и произвольной формулы φ множество формул, выводимых из KB , является подмножеством формул, выводимых из KB φ.

4 Формальное выполнение этого и следующего простого доказательства может быть предоставлено читателю (упражнение 4.3 на стр. 65).

Machine Translated by Google

Если набор формул расширяется, то после расширения все ранее выводимые утверждения все еще могут быть доказаны, а также потенциально могут быть доказаны дополнительные утверждения. Таким образом, набор доказуемых утверждений монотонно растет при расширении набора формул. Для нашего примера это означает, что расширение базы знаний никогда не приведет к нашей цели. Таким образом, мы модифицируем КВ, заменив заведомо ложное утверждение «(все) птицы могут летать» более точным утверждением «(все) птицы, кроме пингвинов, могут летать», и получим в качестве КВ2 следующие положения:

пингвин(твити)
пингвин(x) ⇒ птица(x)
птица(x) ∧ ¬пингвин(x) ⇒ летать(x)
пингвин(x) ⇒ ¬летать(x)

Теперь мир, видимо, снова в порядке. Мы можем вывести ¬fly(tweety), но не fly(tweety), потому что для этого нам понадобится ¬penguin(x), который, однако, не выводим. Пока в этом мире есть только пингвины, царит мир. Однако любая нормальная птица немедленно вызывает проблемы. Желаем добавить ворона Абраксаса (из немецкой книги «Маленькая ведьма») и получить

ворон(абраксас)
ворон(x) ⇒ птица(x)
пингвин(твити)
пингвин(x) ⇒ птица(x)
птица(x) ∧ ¬пингвин(x) ⇒ летать(x)
пингвин(x) ⇒ ¬летать(x))

Мы ничего не можем сказать о лётных признаках Абраксаса, потому что забыли сформулировать, что вороны — это не пингвины. Таким образом, мы расширяем КВ3 до КВ4:

ворон(абраксас)
ворон(x) ⇒ птица(x)
ворон(x) ⇒ ¬пингвин(x)
пингвин(твити)
пингвин(x) ⇒ птица(x)
птица(x) ∧ ¬пингвин(x) ⇒ муха(x))
пингвин(x) ⇒ ¬муха(x)

Здесь следует явно добавить тот факт, что вороны не являются пингвинами, что самоочевидно для людей. Таким образом, для построения базы знаний со всеми примерно 9800 видами птиц во всем мире необходимо указать для каждого вида птиц (за исключением пингвинов), что он не является членом пингвинов. Мы должны действовать аналогично для всех других исключений, таких как страус.

Для каждого объекта в базе знаний, помимо его атрибутов, все атрибуты, которых у него нет, должны быть перечислены

Machine Translated by Google

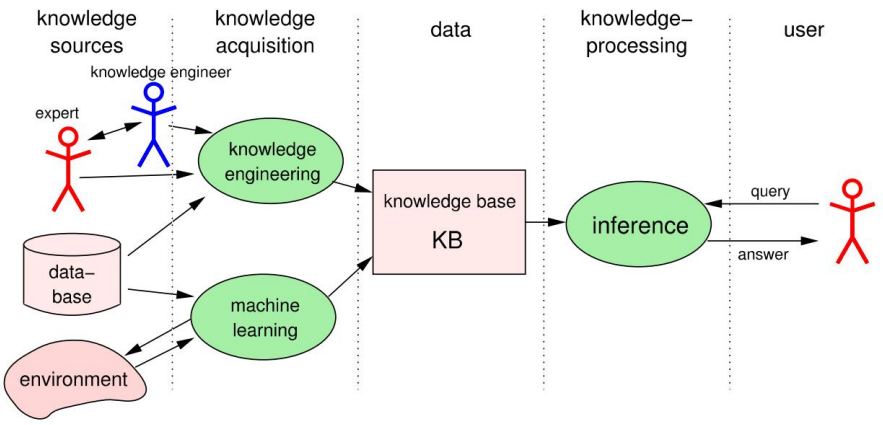


Рис. 1.6 Структура классической системы обработки знаний

проще заменить базу знаний медицинской экспертной системы, чем программировать совершенно новая система.

Благодаря отделению базы знаний от умозаключений знание может хранить декларативно. В базе знаний есть только описание знание, которое не зависит от используемой системы вывода. Без этого четкое разделение, знание и обработка шагов вывода будут переплетены, и любые изменения в знаниях обойдутся очень дорого.

Формальный язык как удобный интерфейс между человеком и машиной подходит к представлению знаний в базе знаний. В следующих главах мы познакомимся с целым рядом таких языков. Во-первых, в гл. 2 и 3 есть исчисление высказываний и логика предикатов первого порядка (PL1). Но также представлены и другие формализмы, такие как вероятностная логика, нечеткая логика или деревья решений. Начнем с исчисления высказываний и связанных с ним систем вывода. Основываясь на что мы представим логику предикатов, мощный язык, доступный для машин и очень важный для ИИ.

В качестве примера крупномасштабной системы, основанной на знаниях, мы хотим сослаться на программный агент «Ватсон». Разработан в IBM совместно с рядом университетов, Watson —это программа для ответов на вопросы, которую можно снабдить подсказками, представленными в естественном виде. язык. Он работает с базой знаний, содержащей четыре терабайта памяти на жестком диске, включая полный текст Википедии [FNA+09]. Watson был разработан в Проект IBM DeepQA, который описан в [Dee11] следующим образом:

> Проект DeepQA в IBM формирует грандиозную задачу в области компьютерных наук, которая направлена на иллюстрирует, как широкая и растущая доступность контента на естественном языке, а также интеграция и усовершенствование обработки естественного языка, информационного поиска, машинного обучения, представления знаний и рассуждений, а также массовых параллельных вычислений могут привести открытую технологию автоматического ответа на вопросы на новый уровень. точка, где они явно и последовательно конкурируют с случайными человеческими способностями.

В американской телевизионной викторине «Опасность!» В феврале 2011 года Уотсон победил двух чемпионов-людей Брэда Раттера и Кена Дженнингса в двух играх. матч по комбинированным очкам и выиграл приз в миллион долларов. Один из пар-

Machine Translated by Google

Особенной силой была его очень быстрая реакция на вопросы, в результате чего Watson часто нажимал на зуммер (используя соленоид) быстрее, чем его конкуренты-люди, а затем был в состоянии дать первый ответ на вопрос.

Высокая производительность и короткое время реакции Watson были обусловлены реализацией на 90 серверах IBM Power 750, каждый из которых содержит 32 процессора, в результате чего получается 2880 параллельных процессоров.

1,5 Упражнения

Упражнение 1.1 Протестируйте некоторых чат-ботов, доступных в Интернете. Начните, например, с www.hs-weingarten.de/~ertel/aibook в коллекции ссылок Turingtest/Chatterbots или с www.simonlaven.com или www.alicebot.org. Запишите начальный вопрос и измерьте время, необходимое для каждой из различных программ, пока вы не будете уверены, что это не человек.

Упражнение 1.2 На сайте www.pandorabots.com вы найдете сервер, на котором вы можете легко создать чат-бота с языком разметки AIML. В зависимости от уровня вашего интереса разработайте простого или сложного чат-бота или измените существующий.

Упражнение 1.3 Объясните причины непригодности теста Тьюринга в качестве определения «искусственного интеллекта» для практического ИИ.

Упражнение 1.4. Многие известные процессы логического вывода, процессы обучения и т. д. NP-полные или даже неразрешимые. Что это означает для ИИ?

Упражнение 1.5

(a) Почему детерминированный агент с памятью не является функцией от множества всех входов к множеству всех выходов в математическом смысле? (b) Как можно изменить агента с памятью или смоделировать его так, чтобы он стал эквивалентна функции, но не теряет свою память?

Упражнение 1.6. Пусть имеется агент с памятью, который может перемещаться в плоскости. От своих датчиков он получает через такт с регулярным интервалом t свое точное положение (x, y) в декартовых координатах. а) Приведите формулу, с помощью которой агент может вычислить свою скорость по текущим временам t и предыдущее измерение t t.

б) Как нужно изменить агент, чтобы он мог вычислить и свое ускорение? Дайте формулу здесь, а также.

Упражнение 1.7

(a) Определите для обоих агентов в Примере 1.1 на стр. 11 затраты, вызванные ошибками, и сравните результаты. Предположим, что удаление спама вручную стоит один цент, а получение удаленного письма или его потеря стоит один доллар.

(b) Определите для обоих агентов прибыль, созданную правильными классификациями, и сравните результаты. Предположим, что на каждое распознанное желаемое электронное письмо начисляется прибыль в один доллар, а на каждое правильно удаленное спам-письмо — прибыль в один цент.

Machine Translated by Google

Логика высказываний

2

В логике высказываний, как следует из названия, предложения связаны логическими операторами. Высказывание «на улице мокро» является пропозицией, как и «идет дождь». Эти два предложения можно соединить, чтобы сформировать новое предложение

если идет дождь, то улица мокрая.

Написано более официально

идет дождь на улице мокро.

Это обозначение имеет то преимущество, что элементарные предложения снова появляются в неизменном виде. Чтобы мы могли точно работать с логикой высказываний, мы начнем с определения множества всех формул логики высказываний.

2.1 Синтаксис

Определение 2.1. Пусть Op = {¬, , , , , (,)} — множество логических операторов, а Σ — множество символов. Множества Op, Σ и {t,f } попарно не пересекаются. Σ называется сигнатурой, а его элементы — переменными предложения. Набор формул пропозициональной логики теперь определяется рекурсивно: • t и f являются (атомарными) формулами. • Все переменные предложения, то есть все элементы из Σ, являются (атомарными) формулами.
лас.

• Если A и B — формулы, то ¬A, (A), A B, A B, A B, A B — также формулы.

Это элегантное рекурсивное определение множества всех формул позволяет нам генерировать бесконечно много формул. Например, если Σ = {A, B, C},

A B, A B C, A A, C B A, (¬A B) (¬C A)

являются формулами. (((A)) B) также является синтаксически правильной формулой.

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по компьютерным наукам,
DOI 10.1007/978-0-85729-299-5_2, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google

Определение 2.2 Мы читаем символы и операторы следующим образом:

т: «правда»		
ф: «ложь»		
¬A: «не A»		(отрицание)
A B: «A и B»		(соединение)
A B: «A или B»		(дизъюнкция)
A B: «если A, то B»		(импликация (также называемая материальной импликацией))
A B: «A тогда и только тогда, когда B»		(эквивалентность)

Определенные таким образом формулы представляют собой пока чисто синтаксические конструкции без значение. Нам все еще не хватает семантики.

2.2 Семантика

В логике высказываний есть два значения истинности: t для «истинно» и f для «ложно». Мы начнем с примера и спросим себя, верна ли формула A B. ответ таков: это зависит от того, верны ли переменные A и B. Например, если A означает «Сегодня идет дождь», а B означает «Сегодня холодно», и оба они верны, тогда A B верно. Если, однако, B представляет «Сегодня жарко» (и это неверно), то A B ложно.

Мы, очевидно, должны присвоить истинностные значения, отражающие состояние мира, в зависимости от обстоятельств. переменные положения. Поэтому мы определяем

Определение 2.3. Отображение I : Σ {w, f}, присваивающее истинностное значение каждая переменная предложения называется интерпретацией.

Поскольку каждая переменная высказывания может принимать два значения истинности, каждая формула логики высказываний с n различными переменными имеет 2n различных интерпретаций. Мы определить значения истинности для основных операций, показав все возможные интерпретации в таблице истинности (см. Таблицу 2.1 на стр. 17).

Пустая формула верна для всех интерпретаций. Для того, чтобы установить истину значение для сложных формул, мы также должны определить порядок операций для логических операторы. Если выражения заключены в круглые скобки, термин в круглых скобках оценивается

Machine Translated by Google

Таблица 2.1 Определение логические операторыпо истине стол

A B (A) ¬A A B	A B	A B	A B	B		
т т т ф у т				т	т	т
т ф т ф ф				т	ф	ф
ф у т о в ф у т о в			ф	т	т	ф
ф ф ф т			ф	ф	т	т

первый. Для формул без скобок приоритеты располагаются следующим образом, начиная с самой сильной привязкой: ¬, , , , .

Чтобы четко различать эквивалентность формул и синтаксическую эквивалентность мы определяем

Определение 2.4. Две формулы F и G называются семантически эквивалентными, если они принимают одно и то же значение истинности для всех интерпретаций. Мы пишем F G.

Семантическая эквивалентность служит, прежде всего, возможности использовать метаязык, который есть естественный язык, чтобы говорить об объектном языке, а именно о логике. Заявление «A B» означает, что две формулы A и B семантически эквивалентны, утверждение «A B», с другой стороны, является синтаксическим объектом формального языка пропозициональной логики.

В зависимости от того, на сколько интерпретаций верна формула, мы можем разделить формулы на следующие классы:

Определение 2.5 Формула называется

- Удовлетворительно, если оно верно хотя бы для одной интерпретации.
- Логически верный или просто верный, если он верен для всех интерпретаций. Истинные формулы также называют тавтологиями.
- Неудовлетворительно, если оно неверно для любой интерпретации.

Каждая интерпретация, удовлетворяющая формуле, называется моделью формулы.

Ясно, что отрицание всякой общезначимой формулы невыполнимо. Негация Формирование выполнимой, но не общепризнанной формулы F выполнимо.

Теперь мы можем создавать таблицы истинности для сложных формул, чтобы установить их правильность. значения истинности. Осуществим это непосредственно, используя эквивалентности формул которые важны на практике.

Machine Translated by Google

Теорема 2.1 . Операции , коммутативны и ассоциативны, а обычно справедливы следующие эквивалентности:

¬A B A B	(импликация)
A B ¬B ¬A	(противоположность)
(A B) (B A) (A B) ¬(A B) ¬A	(эквивалентность)
¬B ¬(A B) ¬A ¬B	(закон де Моргана)

A (B C) (A B) (A C) (дистрибутивный закон)

A (B C) (A B) (A C)

A ¬A ш	(тавтология)
A ¬A f	(противоречие)

A f A
A ш ш

A f f
A ш A

Доказательство Чтобы показать первую эквивалентность, мы вычисляем таблицу истинности для ¬A B и A B и убедитесь, что значения истинности для обеих формул одинаковы для всех интерпретаций. Таким образом, формулы эквивалентны, и поэтому все значения последнего столбец - это «т».

AB ¬A ¬A BA	B (¬A B)	(A B)	
ttf tt ff f ft t fft t		т	т
		ф	т
	т	т	т
		т	т

Доказательства остальных эквивалентностей аналогичны и рекомендуются в качестве упражнений. для читателя (упражнение 2.2 на стр. 29).

2.3 Системы доказательств

В ИИ мы заинтересованы в том, чтобы брать существующие знания и извлекать из них новые. знаний или ответов на вопросы. В пропозициональной логике это означает показать, что база знаний KB, то есть (возможно, обширная) формула пропозициональной логики — следует формула Q1 . Таким образом, мы сначала определим термин «вовлеченность».

1 Здесь Q означает запрос.

Machine Translated by Google

Определение 2.6 Формула KB влечет за собой формулу Q (или Q следует из KB), если каждая модель KB также является моделью Q. Мы пишем KB | B.

Другими словами, в каждой интерпретации, в которой KB верен, Q также верен. Говоря более кратко, всякий раз, когда KB истинно, Q также истинно. Поскольку для понятия следствия вводятся интерпретации переменных, мы имеем дело с семантическим понятием.

Каждая недействительная формула выбирает, так сказать, подмножество множества всех интерпретаций в качестве своей модели. Например, такие тавтологии, как A ¬A, не ограничивают число удовлетворительных интерпретаций, поскольку их предложение пусто. Таким образом, пустая формула верна во всех интерпретациях. Для каждой тавтологии T тогда | T Интуитивно это означает; что тавтологии всегда истинны, без ограничения интерпретаций формулой. Для краткости мы пишем | T . Теперь мы покажем важную связь между семантическим понятием следствия и синтаксической импликацией.

Теорема 2.2 (теорема дедукции)

A | B тогда и только тогда, когда | A Б.

Доказательство Обратите внимание на таблицу истинности для импликации:

A	B	A Б
т	т	т
т	ф	ф
ф	т	т
ф	ф	т

Произвольная импликация A B, очевидно, всегда верна, за исключением интерпретации A t, B f.

Предположим, что A | Б держит. Это означает, что для каждой интерпретации, которая делает A истинным, B также является истинным. Критическая вторая строка таблицы истинности даже не применяется в этом случае. Следовательно, A B истинно, что означает, что A B является тавтологией. Таким образом, было показано одно направление высказывания.

Теперь предположим, что выполняется A B. Таким образом, критическая вторая строка таблицы истинности равна также заблокирован. Каждая модель A тогда также является моделью B. Тогда A | Б держит.

Если мы хотим показать, что KB влечет за собой Q, мы также можем показать с помощью метода таблицы истинности, что KB Q является тавтологией. Таким образом, у нас есть наша первая система доказательств для логики высказываний, которую легко автоматизировать. Недостатком этого метода является очень большое время расчета в худшем случае. В частности, в худшем случае с n переменными предложения для всех 2 n интерпретаций переменных должна быть вычислена формула KB Q. Поэтому время вычислений растет экспоненциально

Machine Translated by Google

с количеством переменных. Следовательно, этот процесс непригоден для большого числа переменных, по крайней мере, в худшем случае.

Если формула KB влечет за собой формулу Q, то по теореме вывода KB Q есть тавтология. Поэтому отрицание ¬(KB Q) невыполнимо. У нас есть

$$\neg(KB \quad Q) \quad \neg(\neg KB \quad Q) \quad KB \quad \neg Q.$$

Следовательно, KB ¬Q также выполнимо. Сформулируем это простое, но важное следствие теоремы о дедукции как теорему.

Теорема 2.3 (Доказательство от противного) KB | Q тогда и только тогда, когда KB ¬Q невыполнимо.

Чтобы показать, что запрос Q следует из базы знаний KB, мы также можем добавить в базу знаний отрицательный запрос ¬Q и получить противоречие. Из-за эквивалентности A ¬A f из теоремы 2.1 на стр. 18 мы знаем, что противоречие неразрешимо. Таким образом, Q доказано. Эта процедура, которая часто используется в математике, также используется в различных автоматических выведениях доказательств, таких как исчисление разрешения и в обработке программ PROLOG.

Одним из способов избежать проверки всех интерпретаций с помощью метода таблицы истинности является синтаксическая обработка формул KB и Q путем применения правил вывода с целью их значительного упрощения, так что в конце мы можем сразу увидеть, что KB | Q. Мы называем этот синтаксический процесс выводом и пишем KB Q. Такие системы синтаксических доказательств называются исчислениями. Чтобы гарантировать, что исчисление не порождает ошибок, мы определяем два фундаментальных свойства исчисления.

Определение 2.7 Исчисление называется корректным, если каждое производное предложение следует семантически. То есть, если для формул KB и Q верно, что

$$\text{если } KB \quad Q, \text{ то } KB \mid B.$$

Исчисление называется полным, если из него можно вывести все семантические следствия. То есть для формул KB и Q справедливо следующее:

$$\text{если } KB \mid Q, \text{ затем } KB \quad Q.$$

Надежность исчисления гарантирует, что все производные формулы на самом деле являются семантическими следствиями базы знаний. Исчисление не дает никаких «ложных следствий». С другой стороны, полнота исчисления гарантирует, что исчисление ничего не упустит из виду. Полное исчисление всегда находит доказательство, если доказываемая формула следует из базы знаний. Если расчет правильный

Machine Translated by Google

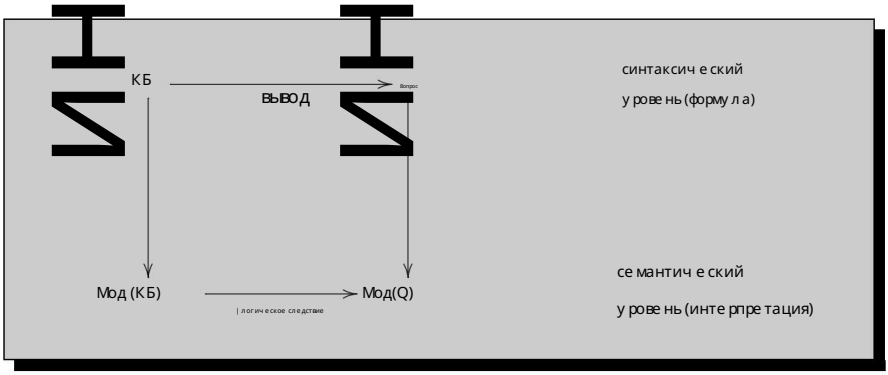


Рис. 2.1 Синтаксическая деривация и семантическая наследственность. Mod(X) представляет набор моделей формулыX

и полное, то синтаксическая деривация и семантическая наследственность являются двумя эквивалентными отношениями (см. рис. 2.1).

Чтобысделать автоматические системыпроверки максимально простыми, их обычно делают оперировать формулами в конъюнктивной нормальной форме.

Определение 2.8 Формула находится в конъюнктивной нормальной форме (КНФ) тогда и только тогда, когда она состоит из конъюнкции

$$K1 \quad K2 \quad \cdots \quad Km$$

статей. Предложение Ki состоит из дизъюнкции

$$(Li1 \quad Li2 \quad \cdots \quad Лини)$$

литералов. Наконец, литерал —это переменная (положительный литерал) или отрицательная переменная (отрицательный литерал).

Формула (A B ¬C) (A B) (¬B ¬C) находится в конъюнктивной нормальной форме. Конъюнктивная нормальная форма не накладывает ограничений на набор формул, потому что:

Теорема 2.4 Каждая формула логики высказываний может быть преобразована в эквивалентную конъюнктивнуюнормальнуюформу.

Machine Translated by Google

Пример 2.1. Приведем A B C D к конъюнктивной нормальной форме, используя эквивалентности из теоремы 2.1 на стр. 18:

A B C D
¬(A B) (C D) (импликация) (¬A ¬B) (C D) (де Морган)
(¬A (C D)) (¬B (C D)) (дистрибутивный закон) ((¬A C)
(¬A D)) ((¬B C) (¬B D)) (распределительный закон) (¬A C)
(¬A D) (¬B C) (¬B D) (ассоциативный закон)

Теперь нам не хватает только исчисления для синтаксического доказательства формул пропозициональной логики. Мы начнем с modus ponens, простого интуитивного правила вывода, которое, исходя из достоверности A и A B, позволяет вывести B. Формально мы запишем это как

$$\frac{A,\ A \quad B}{B}.$$

Это обозначение означает, что мы можем вывести формулу (ы) под линией из формул, разделенных запятыми над линией. Modus ponens, как правило, сам по себе, будучи звуковым, не является полным. Если мы добавим дополнительные правила, то сможем создать полное исчисление, которое, однако, мы здесь не рассматриваем. Вместо этого мы исследуем правило разрешения

$$\frac{A \quad B,\ \neg B \quad C}{A \quad C} \tag{2.1}$$

как альтернатива. Производное предложение называется резольвентой. Путем простого преобразования получаем эквивалентную форму

$$\frac{A \quad B,\ B \quad C}{A \quad C}.$$

Если мы установим A равным f, мы увидим, что правило разрешения является обобщением modus ponens. Правило разрешения одинаково применимо, если C отсутствует или если A и C отсутствуют. В последнем случае пустое предложение может быть получено из противоречия B ¬B (упражнение 2.7 на стр. 30).

2.4 Разрешение

Теперь мы снова обобщим правило разрешения, разрешив предложения с произвольным числом литералов. Для литералов A1,...,Am, B, C1,...,Cn общее правило разрешения имеет вид

$$\frac{(A1 \quad \cdots \quad Am \quad B),\ (\neg B \quad C1 \quad \cdots \quad Cn)}{(A1 \quad \cdots \quad Am \quad C1 \quad \cdots \quad Cn)}. \tag{2.2}$$

Machine Translated by Google

Мы называем литералы B и ¬B дополнительными. Правило разрешения удаляет пару дополнительных литералов из двух предложений и объединяет остальные литералы в новое предложение.

Чтобы доказать, что из базы знаний KB следует запрос Q, проведем доказательство от противного. Следуя теореме 2.3 на стр. 20, мы должны показать, что из KB ¬Q можно вывести противоречие. В формулах в конъюнктивной нормальной форме противоречие проявляется в виде двух дизъюнктов (A) и (¬A), которые приводят к пустому дизъюнкту как к их резольвенте. Следующая теорема гарантирует нам, что этот процесс действительно работает так, как нужно.

Для полноты исчисления нам потребуется небольшое дополнение, как это показано на следующем примере. Пусть формула (A A) задана как наша база знаний. Чтобы показать с помощью правила разрешения, что оттуда мы можем вывести (A A), мы должны показать, что пустое предложение может быть получено из (A A) (¬A ¬A). С одним только правилом разрешения это невозможно. С факторизацией, которая позволяет удалять копии литералов из предложений, эта проблема устраняется. В примере двойное применение факторизации приводит к (A) (¬A), а шаг разрешения — к пустому предложению

Теорема 2.5. *Исчисление резольвенты для доказательства невыполнимости формул в конъюнктивной нормальной форме правильное и полное.*

Поскольку задача резольвентного исчисления состоит в том, чтобы вывести противоречие из KB ¬Q, очень важно, чтобы база знаний KB была непротиворечивой:

Определение 2.9. Формула KB называется непротиворечивой, если из нее нельзя вывести противоречие, т. е. формула вида φ ¬φ.

В противном случае из KB можно получить что угодно (см. упражнение 2.8 на стр. 30). Этот верно не только для разрешения, но и для многих других исчислений.

Из исчислений для автоматического вывода разрешение играет исключительную роль. Таким образом, мы хотим работать с ним более тесно. В отличие от других исчислений, разрешение имеет только два правила вывода и работает с формулами в конъюнктивной нормальной форме.
Это упрощает его реализацию. Еще одно преимущество по сравнению со многими исчислениями заключается в уменьшении числа возможностей применения правил вывода на каждом шаге доказательства, благодаря чему сокращается пространство поиска и сокращается время вычислений.

В качестве примера мы начнем с простой логической головоломки, которая позволяет выполнять важные шаги. доказательство резолюции, которое будет показано.

Пример 2.2 Логическая задачка № 7 под названием Очаровательная английская семья из немецкой книги [Ber89] гласит (в переводе на английский):

Machine Translated by Google

Несмотря на семь долгих лет изучения английского языка с блестящим успехом, я должен признать, что, когда я слышу, как англичане говорят по-английски, я совершенно сбит с толку. Недавно, движимый благородными чувствами, я подобрал трех автостопщиков, отца, мать и дочь, которые, как я быстро понял, были англичанами и говорили только по-английски. В каждом из следующих предложений я колебался между двумя возможными интерпретациями. Мне сказали следующее (второе возможное значение в скобках): Отец: «Мы едем в Испанию (мы из Ньюкасла)».
Мать: «Мы не едем в Испанию и из Ньюкасла (мы остановились в Париже и не едем в Испанию)». Дочь: «Мы не из Ньюкасла (остановились в Париже)».
А как насчет этой очаровательной английской семьи?

Для решения такого рода задач мы действуем в три этапа: формализация, преобразование в нормальную форму и доказательство. Во многих случаях формализация является самым трудным шагом, потому что легко сделать ошибку или забыть мелкие детали. (Поэтому очень важны практические упражнения. См. упражнения 2.9–2.11.)

Здесь мы используем переменные S для «Мы едем в Испанию», N для «Мы из Ньюкасла», и P для «Мы остановились в Париже» и получаем как формализация трех пропозиций отца, матери и дочери.

$$(S \leftrightarrow N) \quad [(\neg S \rightarrow N) \quad (P \rightarrow \neg S)] \quad (\neg N \rightarrow P).$$

Вынесение на множитель ¬S в средней подформуле переводит формулу в КНФ за один шаг. Нумерация предложений с подстрочными индексами дает

$$KB \quad (S \leftrightarrow N)_1 \quad (\neg S)_2 \quad (P \rightarrow N)_3 \quad (\neg N \rightarrow P)_4.$$

Теперь приступим к доказательству разрешения, сначала еще без запроса Q. Выражение вида «Res(m, n): клауза k» означает, что клаузулы получаются резольвентой клаузулы m с клаузулой n и имеют номер k.

$$\text{Рез}(1, 2): (\text{Н})_5$$

$$\text{Рез}(3, 4): (\text{П})_6$$

$$\text{Рез}(1, 4): (S \rightarrow P)_7$$

Мы могли бы вывести пункт P также из Res(4, 5) или Res(2, 7). Каждый последующий шаг разрешения приведет к получению уже имеющихся предложений. Поскольку это не позволяет вывести пустое предложение, поэтому было показано, что база знаний непротиворечива. Итак, мы получили N и P. Чтобы показать, что ¬S выполняется, мы добавим предложение $(S)_8$ к набору предложений в качестве запроса с отрицанием. С шагом разрешения

$$\text{Рез}(2, 8): ()_9$$

доказательство завершено. Таким образом, выполняется ¬S ∧ N ∧ P. «Очаровательное английское семейство», видимо, родом из Ньюкасла, остановилось в Париже, но в Испании не собирается.

Пример 2.3. Логическая задача номер 28 из [Ber89], озаглавленная «Прыжок в высоту», гласит:

Machine Translated by Google

Три девушки тренируются в прыжках в высоту перед выпускным экзаменом по физкультуре. Планка установлена на 1,20 метра. «Держу пари, — говорит первая девушка второй, — что я справлюсь, если и только если ты это не сделаешь». Если бы вторая девушка сказала то же самое третьей, а та, в свою очередь, сказала бы то же самое первой, смогли бы все трое выиграть свои ставки?

Мы показываем через доказательство резолюцией, что не все трое могут выиграть свои ставки.

Формализация:

Первый прыжок девушки удался: A,	Ставка первой девушки: (A ¬B),
прыжок второй девушки удался: B, прыжок	ставка второй девушки: (B ¬C),
третьей девушки удался: C.	ставка третьей девушки: (C ¬A).

Утверждение: все трое не могут выиграть свои ставки:

$$Q \quad \neg((A \quad \neg B) \quad (B \quad \neg C) \quad (C \quad \neg A))$$

Теперь нужно показать с помощью разрешения, что ¬Q невыполнимо.

Превращение в CNF: Ставка первой девушки:

$$(A \quad \neg B) \quad (A \quad \neg B) \quad (\neg B \quad A) \quad (\neg A \quad \neg B) \quad (A \quad B)$$

Аналогичные преобразования претерпевают ставки двух других девушек, и мы получаем отрицательную гипотезу

$$\neg Q \quad (\neg A \quad \neg B)1 \quad (A \quad B)2 \quad (\neg B \quad \neg C)3 \quad (B \quad C)4 \quad (\neg C \quad \neg A)5$$
$$(C \quad A)6.$$

Оттуда мы получаем пустое предложение, используя разрешение:

Res(1, 6): (C ¬B)7
Рез(4, 7): (C)8
Res(2, 5): (B ¬C)9
Рез(3, 9): (¬C)10
Рез(8, 10): ()

Таким образом, утверждение доказано.

2.5 Оговорки Хорна

Предложение в конъюнктивной нормальной форме содержит положительные и отрицательные литералы и может быть представлено в виде

$$(\neg A1 \quad \cdots \quad \neg Am \quad B1 \quad \cdots \quad Bn)$$

Machine Translated by Google

с переменными A1,...,Am и B1,...,Bn. Это предложение может быть преобразовано в два простых шага в эквивалентную форму

$$A_1 \land \cdots \land A_m \rightarrow B_1 \lor \cdots \lor B_n.$$

Эта импликация содержит посылку, конъюнкцию переменных и заключение, дизъюнкцию переменных. Например: «Если будет хорошая погода и на земле будет снег, я пойду кататься на лыжах или буду работать». Это предложение такой формы. Получатель этого сообщения точно знает, что отправитель не собирается плавать.

Значительно более четким утверждением было бы: «Если будет хорошая погода и на земле будет снег, я пойду кататься на лыжах». Получатель теперь точно знает. Таким образом, мы называем предложения с не более чем одним положительным буквально определенным предложением. Преимущество этих положений состоит в том, что они допускают только один вывод, и, таким образом, их гораздо проще интерпретировать. Многие отношения могут быть описаны предложениями этого типа. Поэтому мы определяем

Определение 2.10 Предложения с не более чем одним положительным литералом формы

$$(\neg A_1 \land \cdots \land \neg A_m \land B) \text{ или } (\neg A_1 \land \cdots \land \neg A_m) \text{ или } B$$

или (эквивалентно)

$$A_1 \land \cdots \land A_m \land B \text{ или } A_1 \land \cdots \land A_m \text{ false или } B.$$

называются оговорками Хорна (по имени их изобретателя). Предложение с одним положительным литералом является фактом. В предложениях с отрицательными и одним положительным литералом положительный литерал называется головным.

Чтобы лучше понять представление предложений Хорна, читатель может вывести их из определений эквивалентностей, которые мы использовали в настоящее время (упражнение 2.12 на стр. 30).

С предложениями Хорна легче обращаться не только в повседневной жизни, но и в формальных рассуждениях, как мы можем видеть в следующем примере. Пусть база знаний состоит из следующих предложений (символ « », связывающий предложения, опущен здесь и в последующем тексте):

(хорошая_погода)1

(снегопад)2

(снегопад снег)3

(хорошая_погода снег катание на лыжах)4

Machine Translated by Google

Если мы теперь хотим знать, сохраняется ли лыжный спорт, это можно легко вывести. В качестве правила вывода здесь достаточно слегка обобщенного modus ponens:

$$\frac{A1 \quad \cdots \quad Am, A1 \quad \cdots \quad Am \quad B}{B}.$$

Доказательство «катания на лыжах» имеет следующий вид (MP(i1,...,ik) представляет собой применение modus ponens к предложениям i1 —ik :

ОД(2, 3): (снег)5 ОД(1, 5, 4):
(лыжи)6.

Используя modus ponens, мы получаем полное исчисление для формул, состоящих из предложений Хорна пропозициональной логики. Однако в случае больших баз знаний modus ponens может вывести много ненужных формул, если начать с неправильных предложений. Поэтому во многих случаях лучше использовать исчисление, которое начинается с запроса и работает в обратном направлении, пока не будут получены факты. Такие системы называются обратной цепочкой, в отличие от систем прямой цепочки, которые начинают с фактов и, наконец, выводят запрос, как в приведенном выше примере с modus ponens.

Для обратной цепочки предложений Хорна используется разрешение SLD. SLD расшифровывается как «линейное разрешение, управляемое правилом выбора для определенных предложений». В приведенном выше примере, дополненном отрицательным запросом (катание на лыжах f)

(хорошая_погода)1
(снегопад)2
(снегопад снег)3
(хорошая_погода снег катание на лыжах)4
(лыжи f)5

проводим разрешение SLD начиная с шагов разрешения, выте кающих из этого пункта

Res(5, 4): (хорошая_погода снег f)6
Res(6, 1): (снег f)7
Res(7, 3): (снегопад f)8
Рез(8, 2): ()

и получаем противоречие с пустым предложением. Здесь мы можем легко увидеть «линейное разрешение», что означает, что дальнейшая обработка всегда выполняется для текущего производного предложения. Это приводит к значительному сокращению пространства поиска. Кроме того, литералы текущего предложения всегда обрабатываются в фиксированном порядке (например, справа налево) («Управление правилом выбора»). Литералы текущего предложения называются

Machine Translated by Google

подцель. Литералы инвертированного запроса являются целями . Правило вывода для одного шага гласит

$$\frac{A_1 \quad \cdots \quad A_m \qquad B_1, B_1 \qquad B_2 \quad \cdots \quad B_n \qquad f}{A_1 \quad \cdots \quad A_m \qquad B_2 \quad \cdots \quad B_n \qquad f}.$$

Перед применением правила вывода должны быть доказаны $B_1, B_2, \ldots, B_n$ —текущие подцели. После применения B_1 заменяется новой подцелью $A_1 \quad \cdots \quad A_m$. Чтобы показать, что B_1 истинно, мы должны теперь показать, что $A_1 \quad \cdots \quad A_m$ истинны. Этот процесс продолжается до тех пор, пока список подцелей текущих предложений (так называемый стек целей) не станет пустым. При этом было обнаружено противоречие. Если для подцели $\neg B_i$ нет предложения с дополнительным литералом B_i в качестве заголовка предложения, доказательство завершается и противоречие не может быть найдено. Таким образом, запрос недоказуем.

Разрешение SLD играет важную роль на практике, поскольку программы на языке логического программирования PROLOG состоят из предикатных логических предложений Horn, и их обработка достигается посредством разрешения SLD (см. упражнение 2.13 на стр. 30 или гл. 5).

2.6 Вычислимость и сложность

Метод таблицы истинности, как простейшая система семантического доказательства для логики высказываний, представляет собой алгоритм, который может определить каждую модель любой формулы за конечное время. Таким образом, множества невыполнимых, выполнимых и правильных формул разрешимы. Время вычисления метода таблицы истинности для выполнимости в худшем случае растет экспоненциально с числом переменных n, поскольку таблица истинности имеет 2^n строки. Оптимизация, метод семантических деревьев, избегает просмотра переменных, которые не встречаются в предложениях, и, таким образом, во многих случаях экономит время вычислений, но в худшем случае оно также является экспоненциальным.

При разрешении в худшем случае количество производных предложений растет экспоненциально с количеством начальных предложений. Таким образом, чтобы выбрать между двумя процессами, мы можем использовать эмпирическое правило: в случае многих предложений с небольшим количеством переменных предпочтительнее использовать метод таблицы истинности, а в случае небольшого количества предложений с большим количеством переменных разрешение, вероятно, завершится быстрее.

Остается вопрос: может ли доказательство в пропозициональной логике идти быстрее? Есть ли лучшие алгоритмы? Ответ: наверное, нет. Ведь С. Кук, основоположник теории сложности, показал, что задача 3-SAT является NP-полной. 3-SAT —это набор всех формул КНФ, в предложениях которых ровно три литерала. Таким образом, очевидно, что полиномиального алгоритма для 3-SAT, вероятно, (по модулю проблемы P/NP) не существует, а значит, и общего алгоритма. Однако для предложений Хорна существует алгоритм, в котором время вычислений для проверки выполнимости растет только линейно по мере увеличения числа литералов в формуле.

Machine Translated by Google

2.7 Приме не ние и ограни ч е ния

Сре дства дока з ате л ьства те оре м для л оги ки вы ска з ы ваний явл я ются ч астью повсе дне вного набора инстру ме нтов раз работч ика в обл асти цифровы х те хнол оги й. Напри ме р, не которые из э тих з адач — прове рка цифровы х схе м и ге не рация те стовы х табл иц для те стирования микропроце ссоров в произ водстве . Спе циал ьные систе мы дока з ате л ьств, работающие с двои ч ными диаграммами ре ше ний (BDD) (), та кже испол ь з у ются в ка ч е стве стру кту ры данны х для обработки форму л пропози ци онал ьной л оги ки.

В ИИ л оги ка вы ска з ы ваний испол ь з у е тся в простых прил оже ниях. Напри ме р, простые э кспе ртные систе мы, бе з у сл овно, мог у т работать с пропози ци онал ьной л оги кой. Одна ко все пе ре ме нные до л жны быть дискре тными, име ть л ишь не ско л ько з на ч е ний, и ме ж ду пе ре ме нными не мо ж е т быть ника ки х вз аимосвя з е й. Сл ожные л оги ч е ские свя з и мог у т быть вы ра ж е ны бол е е э л е гантно с помощью л оги ки пре дикатов.

Ве роятностная л оги ка —э то оч е нь инте ре сное и акту а л ьное соч е тание пропози ци онал ьной л оги ки и ве роятностны х вы н исл е ний, которое поз во л я е т моде л ировать не опре де л е нное з нание . Он подробно рассматри ва е тся в гл . 7. В э той гл аве та кже обсу ж да е тся не ч е тка я л оги ка, допу ска ющая бе сконе ч ное мно ж е ство з на ч е ний истинности.

2.8 У пра ж не ния

У пра ж не ние 2.1. Приве дите гра мматику формы Бэ ку са–Нау ра для синтаксиса л оги ка.

У пра ж не ние 2.2. Пока ж ите , ч то сл е ду ющие форму л ы явл я ются та вто л оги ями:
(a) ¬(A B) ¬A ¬B (b) A B
¬B ¬A (c) ((A B) (B A))
(A B) (г) (A B) (¬B C) (A C)

У пра ж не ние 2.3 Пре обра з у йте сл е ду ющие форму л ы в конъюнкти вну юнорма л ьну юформу : (a) A
B (b) A B
A B (c) A (A B) B

У пра ж не ние 2.4. Прове рьте сл е ду ющие у тве рж де ния на вы пол нимость ил и досто ве рность.
(a) (play_lottery six_right) побе дите л ь (b)
(play_lottery six_right (six_right win)) win (c) ¬(¬gas_in_tank
(gas_in_tank ¬car_starts) ¬car_starts)

У пра ж не ние 2.5 Испол ь з у я вы бранны й ва ми я з ы к програ ммирования, з апрограммиру йте сре дство дока з ате л ьства те оре м для л оги ки вы ска з ы ваний, испол ь з у я ме тод табл ицы истинности для форму л в конъюнкти вной нормал ьной форме . Чтобы из бе ж ать доро г остоя ще й прове рки синтаксиса форму л , вы мо ж е те пре дста в л ять пре дл оже ния в виде спи ско в ил и набо ро в л ите ра л ов, а форму л ы —в виде спи ско в ил и набо ро в пре дл оже ний. Програ мма до л жна у ка з ать, явл я ется л и форму л а не вы пол нимой, вы пол нимой ил и истинной, и вы ве сти ко л и ч е ство раз л и ч ны х инте рпре таций и моде л е й.

Machine Translated by Google

Упражнение

2.6 (a) Покажите, что modus ponens является допустимым правилом вывода, показав, что A (A B) | B. (b)
Покажите, что правило разрешения (2.1) является допустимым правилом вывода.

Упражнение 2.7. Покажите, применяя правило разрешения, что в конъюнктивном нормальном форме пустое предложение эквивалентно ложному утверждению.

Упражнение 2.8. Покажите, что с помощью разрешения можно «вывести» любое произвольное предложение из базы знаний, содержащей противоречие.

Упражнение 2.9 Формализуйте следующие логические функции с помощью логических операторов и покажите, что ваша формула верна. Представьте результат в CNF. (a) Операция XOR (исключающее или) между двумя переменными. (b) Утверждение, по крайней мере, две из трех переменных A, B, C верны.

Упражнение 2.10 Решите следующий случай с помощью резолютивного доказательства: «Если у преступника был сообщник, то он приехал на машине. У преступника не было сообщника и не было ключа, или у него был ключ и сообщник. Ключ был у преступника. Преступник приехал на машине или нет?

Упражнение 2.11. Покажите с помощью разрешения, что формула
из (a) упражнения 2.2(d) является
тавтологией. (b) Упражнение 2.4(c) невыполнимо.

Упражнение 2.12. Докажите следующие эквивалентности, важные для работы с предложениями Горна: (a) (¬A1
⋯ ¬Am B) A1 ⋯ Am B (b) (¬A1 ⋯ ¬Am) A1
⋯ Am f (c) A w A

Упражнение 2.13. Покажите с помощью разрешения SLD, что следующий набор предложений Хорна неудовлетворим.

(A)1	(Д)4	(A D G)7
(БИ2	(E)5	(C F E H)8
(C)3	(A B C F)6	(H f)9

Упражнение 2.14 В разд. 2.6 говорится: «Таким образом, очевидно, что, вероятно, (по модулю проблемы P/NP) не существует полиномиального алгоритма для 3-SAT, а значит, вероятно, и не общего». Обоснуйте «вероятно» в этом предложении.

Machine Translated by Google

Логика предикатов первого порядка

3

Многие практически важные проблемы не могут быть или могут быть очень неудобно сформулированы на языке пропозициональной логики, как мы можем легко понять в следующем примере. Заявление

«Робот 7 находится в позиции xy (35, 79)»

на самом деле может быть не посредственно использована как пропозициональная логическая переменная

«Robot_7_is_situated_at_xy_position_(35, 79)»

для рассуждений с пропозициональной логикой, но рассуждать с такого рода пропозиции очень неудобно. Предположим, что 100 таких роботов могут остановиться в любом месте на сетке 100 × 100 точек. Для описания каждой позиции каждого робота нам потребуется 100 · 100 · 100 = 1 000 000 = 106 различных переменных. Определение отношений между объектами (в данном случае роботами) становится по-настоящему трудным. Отношение

«Робот А находится справа от робота Б».

семантически не более чем набор пар. Из 10 000 возможных пар координат химеется (99 · 98)/ 2 = 4851 упорядоченная пара. Вместе со всеми 10 000 комбинаций возможных значений для обоих роботов имеется (100 · 99) = 9900 формул вида

Robot_7_is_to_the_right_of_robot_12
Robot_7_is_situated_at_xy_position_(35, 79)
Robot_12_is_situated_at_xy_position_(10, 93) ···

определяющие эти соотношения, каждое из которых с $2 \cdot 0{,}485 = 0{,}485 \cdot 108$ вариантов (104) в правой части. В логике предикатов первого порядка мы можем определить для этого предикат Position(number, xPosition, yPosition). Вышеупомянутое отношение больше не должно перечисляться как огромное количество пар, а описывается абстрактно с помощью правила вида

$$u \quad v \text{ is_farther_right}(u, v) \qquad xu \quad yu \quad xv \quad yv \text{ position}(u, xu, yu) \quad \text{position}(v, xv, yv) \quad xu > xv,$$

Где u читается как «для каждого u», а v как «существует v».

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по компьютерным наукам,
DOI 10.1007/978-0-85729-299-5_3, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google

В этой главе мы определим синтаксис и семантику логики предикатов первого порядка (PL1), покажем, что многие приложения могут быть смоделированы с помощью этого языка, и покажем, что для этого языка существует полное и надежное исчисление.

3.1 Синтаксис

Сначала мы закрепляем синтаксическую структуру терминов.

Определение 3.1. Пусть V —набор переменных, K —набор констант и F —набор функциональных символов. Множества VK и F попарно не пересекаются. Мы определяем множество термов рекурсивно: • Все переменные и константы являются (атомарными) термами. • Если t1,...,tn термы, а f n-местный функциональный символ, то f (t1,...,tn) тоже термин.

Некоторыми примерами терминов являются f (sin(ln(3)), exp(x)) и g(g(g(x))). Быть способным устанавливаем логические связи между терминами, строим формулы из терминов.

Определение 3.2. Пусть P —множество предикатных символов. Формулы логики предикатов строятся следующим образом: • Если t1,...,tn —термы, а p —n-местный предикатный символ, то p(t1,...,tn) — (атомарная) формула. • Если A и B —формулы, то ¬A, (A), A B, A B, A B, A B —также формулы.

- Если x —переменная, а A —формула, то x A и x A также являются формулами. —квантор всеобщности, а —квантор существования.
- p(t1,...,tn) и ¬p(t1,...,tn) называются литералами. • Формулы, в которых каждая переменная находится в области действия квантификатора, называются предложениями первого порядка или закрытыми формулами. Переменные, которые не входят в область действия квантификатора, называются свободными переменными. • Определения 2.8 (КНФ) и 2.10 (предложения Хорна) верны для формул литералов логики предикатов аналогично.

В Таблице 3.1 на стр. 33 приведены несколько примеров формул PL1. с их интуитивной интерпретацией.

Machine Translated by Google

Таблица 3.1 Примеры формул в логике предикатов первого порядка. Обратите внимание, что мать здесь функциональный символ

Формула	Описание
x лягушка(x) зеленая(x)	Все лягушки зеленые
x лягушка(x) коричневая(x)	Все коричневые лягушки большие
большая(x) нравится(x, торт)	Все нравится торт
¬ нравится(x, торт)	Не все нравится торт
¬ нравится(x, торт)	Никто не любит торт
x нравится(y,	Есть то, что нравится всем
x) x нравится(x,	Есть тот, кому все нравится
y) x нравится(y,	Все кому-то нравится
x) x нравится(x,	Все что-то нравится
y) x клиент(x) лайков(боб ,x)	Боб любит каждого клиента
x клиент(x) нравится(x, bob)	Есть клиент, который нравится Бобу
x пекарь(x) клиент(y) mag(x, y)	Есть пекарь, которому нравятся все его клиенты
x старше (мать(x), x) x	Каждая мать старше своего ребенка
старше (мать(мать(x)), x)	Каждая бабушка старше ее ребенок дочери
x y z rel(x, y) rel(y, z) rel(x, z)	rel - транзитивное отношение

3.2 Семантика

В пропозициональной логике каждой переменной непосредственно присваивается значение истинности интерпретацией. В логике предикатов значение формул рекурсивно определяется по построение формулы, в котором мы сначала назначаем константы, переменные и функцию символы к объектам реального мира.

Определение 3.3 Интерпретация I определяется как

- Отображение множества констант и переменных K V в множество W названия предметов в мире.
- Отображение набора функциональных символов на набор функций в мир. Каждому n-местному функциональному символу назначается n-местная функция.
- Отображение набора предикатных символов на набор отношений в мир. Каждому n-местному символу-предикату присваивается n-местное отношение.

Пример 3.1. Пусть c1, c2, c3 —константы, «плюс» —двузначный функциональный символ и «gr» —двузначный предикатный символ. Правда формулы

F гр(плюс (c1, c3), c2)

Machine Translated by Google

зависит от интерпретации I. Сначала выберем следующую очевидную интерпретацию констант, функции и предикатов в натуральных числах:

I1: c1 1, c2 2, c3 3, плюс +, гр > .

Таким образом, формула преобразуется в

1 + 3 > 2 или после оценки 4 > 2.

Отношение «больше чем» на множестве {1, 2, 3, 4} — это множество всех пар (x, y) чисел с x>y, что означает множество G = {(4, 3), (4, 2), (4, 1), (3, 2), (3, 1), (2, 1)}. Поскольку (4, 2) G, формула F верна при интерпретации I1. Однако, если мы выбираем интерпретацию

I2: c1 2, c2 3, c3 1, плюс , gr >,

мы получаем

2 - 1 > 3 или 1 > 3.

Пара (1, 3) не принадлежит G. Формула F ложна при интерпретации I2. Очевидно, истинность формулы в PL1 зависит от интерпретации. Теперь, после этого предварительного просмотра, мы определяем истину .

Определение

3.4 • Атомарная формула p(t1,...,tn) истинна (или верна) при интерпретации I , если после интерпретации и вычисления всех термов t1,...,tn и интерпретации предиката p через n-местное отношение r, верно, что

(I(t1), ... ,I(tn)) r.

- Истинность бесквантерных формул следует из истинности атомарных формул — как и в исчислении высказываний — через семантику логических операторов, определенных в таблице 2.1 на стр. 17. • Формула x F истинна при интерпретации I точно тогда, когда она истинно при произвольном изменении интерпретации переменной x (и только для x)

- Формула x F истинна при интерпретации I ровно тогда, когда существует интерпретация для x, которая делает формулу истинной.

Определения семантической эквивалентности формул для понятий выполнимый, истинный, невыполнимый и модельный вместе с семантическим следствием (определения 2.4, 2.5, 2.6) переносятся без изменений из исчисления высказываний в логику предикатов.

Machine Translated by Google

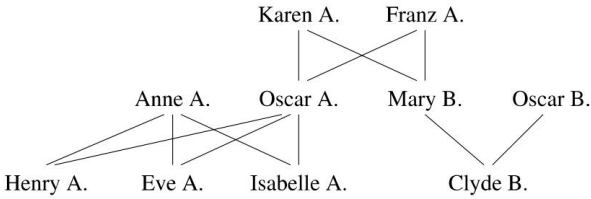


Рис. 3.1 Генеалогическое дерево. Ребра, идущие от Клайда Б. вверх к Мэри Б. и Оскару Б., представляют элемент (Клайд Б., Мэри Б., Оскар Б.) как дочернюю связь

Теорема 3.1 Теоремы 2.2 (теорема о дедукции) и 2.3 (доказательство от противного) верны аналогично для PL1.

Пример 3.2 Генеалогическое дерево, представленное на рис. 3.1 , графически представляет (на семантическом уровне) отношение

Ребенок = {(Оскар А., Карен А., Фрэнк А.), (Мэри Б., Карен А., Фрэнк А.), (Генри А., Энн А., Оскар А.),
(Ева А., Энн А., Оскар А.), (Изабель А., Энн А., Оскар А.), (Клайд Б., Мэри Б., Оскар Б.)}

Например, тройка (Оскар А., Карен А., Фрэнк А.) обозначает предложение «Оскар А. —ребенок Карен А. и Фрэнка А.». Из именмы читаем одноместное отношение

Женщина = {Карен А., Энн А., Мэри Б., Ева А., Изабель А.}

женщин. Теперь мы хотим установить формулы для семейных отношений. Сначала мы определяем трехместный предикат child(x, y, z) с семантикой

I(ребенок(x, y, z)) = w (I(x),I(y),I(z)) Kind.

При интерпретации I(oscar) = Oscar A., I(eve) = Eve A., I(anne) = Anne A. также верно, что child(eve, anne, oscar). Чтобы ребенок (ева, оскар, энн) был правдой, нам нужно, чтобы

x y z дочернийэлемент(x, y, z) дочернийэлемент(x, z, y),

симметричность дочернего предиката в последних двух аргументах. Для дальнейших определений мы обращаемся к упражнению 3.1 на стр. 54 и рекурсивно определяем потомок предиката.
как

x y потомок(x, y) z потомок(x, y, z)
(u v потомок(x, u, v) потомок(u, y)).

Теперь строим небольшую базу знаний с правилами и фактами. Позволять

Machine Translated by Google

КБ женщина (карен) женщина (энн) женщина (мэри)
женщина (Ева) женщина (Изабель)
ребенок (Оскар, Карен, Франц) ребенок (Мэри, Карен, Франц)
ребенок (Ева, Энн, Оскар) ребенок (Генри, Энн, Оскар)
ребенок (Изабель, Энн, Оскар) ребенок (Клайд, Мэри, Оскар)
(x y z дочерний(x, y, z) дочерний(x, z, y))
(x y потомок(x, y) z потомок(x, y, z)
(u v дочерний элемент(x, u, v) потомок(u, y))).

Теперь мы можем спросить, например, выводимы ли предложения ребенок (Ева, Оскар, Анна) или потомок (Ева, Франц) . Для этого нам потребуется вычисление .

3.2.1 Равенство

Чтобы иметь возможность сравнивать термины, равенство является очень важным отношением в логике предикатов. Равенство термов в математике —это отношение эквивалентности, то есть оно рефлексивно, симметрично и транзитивно. Если мы хотим использовать равенство в формулах, мы должны либо включить эти три атрибута в качестве аксиом в нашу базу знаний, либо мы должны интегрировать равенство в исчисление . Мы пойдем по простому пути и определим предикат «=», который, отклоняясь от определения 3.2 на стр. 32, записывается с использованием инфиксной записи, как это принято в математике . (Уравнение x = y, конечно, также можно записать в виде eq(x, y).) Таким образом, аксиомы равенства имеют вид

$$\begin{aligned} &x\ x = x && \text{(рефлексивность)}\\ &x\ y\ x = y \Rightarrow y = x && \text{(симметрия)}\\ &x\ y\ z\ x = y \wedge y = z \Rightarrow x = z && \text{(транзитивность).} \end{aligned} \tag{3.1}$$

Чтобы гарантировать уникальность функций, мы дополнительно требуем

$$x\ y\ x = y \Rightarrow f(x) = f(y) \quad \text{(аксиома подстановки)} \tag{3.2}$$

для каждого функционального символа. Аналогично требуем для всех x предикатных символов

$$x\ y\ x = y \wedge p(x) \Rightarrow p(y) \quad \text{(аксиома подстановки).} \tag{3.3}$$

Мы формулируем другие математические отношения, такие как отношение «<», аналогичным образом (упражнение 3.4 на стр. 55).

Часто переменная должна быть заменена термином. Чтобы сделать это правильно и описать его просто, мы даем следующее определение .

Machine Translated by Google

Определение 3.5 Обозначим через $\phi[x/t]$ формулу, которая получается при замене каждого свободного вхождения переменной x в ϕ на член t. Таким образом, мы не допускаем никаких переменных в терме t, которые квантифицируются в ϕ. В этих случаях переменные должны быть переименованы, чтобы гарантировать это.

Пример 3.3. Если в формуле $\forall x\, x = y$ заменить свободную переменную членом $x + 1$, то получится $\forall x\, x = x + 1$. При правильной подстановке получим формулу $\forall x\, x = y + 1$, которая имеет совсем другую семантику.

3.3 Кванторы и нормальные формы

Согласно определению 3.4 на стр. 34, формула $\exists x p(x)$ истинна тогда и только тогда, когда она истинна для всех интерпретаций переменной x. Вместо квантора можно было бы написать $p(a1) \vee \cdots \vee p(an)$ для всех констант $a1 \cdots an$ в K. Для $\forall x p(x)$ можно было бы написать $p(a1) \wedge \cdots \wedge$ кастрюля). Отсюда по закону де Моргана следует, что

$$\forall x \phi \quad \neg\, \exists x \neg \phi .$$

Благодаря этой эквивалентности кванторы всеобщего и существования взаимозаменяемы.

Пример 3.4 Предложение «Каждый хочет быть любимым» равносильно высказыванию «Никто не хочет быть любимым».

Кванторы являются важным компонентом выразительной силы логики предикатов. Однако они мешают автоматическому выводу в ИИ, поскольку усложняют структуру формул и увеличивают количество применимых правил вывода на каждом этапе доказательства. Поэтому наша следующая цель — найти для каждой формулы логики предикатов эквивалентную формулу в стандартизированной нормальной форме с как можно меньшим числом кванторов. В качестве первого шага мы вносим кванторы всеобщности в начало формулы и тем самым определяем

Определение 3.6. Формула логики предикатов ϕ находится в предваренной нормальной форме, если выполняется равенство $\phi = Q1x1 \cdots Qnxn\, \psi$. • ψ — бесквантор ная формула. • $Qi \in \{\forall, \exists\}$ для $i = 1,\ldots,n$.

Machine Translated by Google

Undergraduate Topics in Computer Science

Wolfgang Ertel

Introduction to Artificial Intelligence

UTiCS

Springer

Machine Translated by Google

Темы бакалавриата по информатике

Machine Translated by Google

Темы бакалавриата по компьютерным наукам (UTiCS) предоставляют высококачественный учебный контент для студентов, изучающих все области вычислительной техники и информатики. Книги UTiCS, от основного фундаментального и теоретического материала до тем и приложений последнего года обучения, используют свежий, лаконичный и современный подход и идеально подходят для самостоятельного изучения или для одно- или двухсеместрового курса. Все тексты написаны признанными экспертами в своих областях, проверены международным консультативным советом и содержат многочисленные примеры и проблемы. Многие включают полностью проработанные решения.

Дополнительные
тома: www.springer.com/series/7592

Machine Translated by Google

Вольфганг Эртель

Введение в искусственный Интеллект

Перевод Натанаэля Блэка
С иллюстрациями Флориана Маста.

Springer

Machine Translated by Google

Проф. д-р Вольфганг Эртель FB
Elektrotechnik und Informatik Hochschule
Равенсбург-Вайнгартен Университет прикладных
наук Вайнгартен Германия ertel@hs-
weingarten.de

Редактор сериала
Ян Маки

Консультативный совет
Самсон Абрамски, Оксфордский университет, Оксфорд, Великобритания
Карин Брейтман, Папский католический университет Рио-де-Жанейро, Рио-де-Жанейро, Бразилия
Крис Хэнкин, Имперский колледж Лондона, Лондон, Великобритания
Декстер Козен, Корнельский университет, Итака, США
Эндрю Питтс, Кембриджский университет, Кембридж, Великобритания
Ханне Риис Нильсон, Технический университет Дании, Kongens Lyngby, Дания
Стивен Скиена, Университет Стоуни-Брук, Стоуни-Брук, США
Иэн Стюарт, Даремский университет, Дарем, Великобритания

ISSN 1863-7310 ISBN
978-0-85729-298-8 DOI
10.1007/978-0-85729-299-5 Springer London
Dordrecht Heidelberg New York
электронный ISBN 978-0-85729-299-5

Каталогизация Британской библиотеки в данных публикаций
Запись в каталоге этой книги доступна в Британской библиотеке.

Контрольный номер Библиотеки Конгресса: 2011923216

Первоначально опубликовано на немецком языке издательством Vieweg+Teubner, 65189 Wiesbaden, Germany, как «Ertel, Wolfgang: Grundkurs Künstliche Intelligenz. 2. обр. версия". © Vieweg+Teubner | GWV Fachverlage GmbH, Висбаден, 2009 г.

© Springer-Verlag London Limited 2011 Помимо любых
добросовестных действий в целях исследования или частного изучения, а также критики или рецензирования в соответствии с Законом об авторском праве, промышленных образцах и патентах 1988 г., эта публикация может воспроизводиться, храниться или передаваться только в любой форме и любыми средствами с предварительного письменного разрешения издателей или в случае репрографического воспроизведения в соответствии с условиями лицензий, выданных Агентством по лицензированию авторских прав. Запросы относительно воспроизведения вне этих условий следует направлять издателям.

Использование зарегистрированных наименований, товарных знаков и т. д. в этой публикации не означает, даже при отсутствии конкретного заявления, что такие названия не подпадают под действие соответствующих законов и правил и, следовательно, бесплатны для общего использования.
Издатель не делает никаких заявлений, явных или подразумеваемых, относительно точности информации, содержащейся в этой книге, и не может нести никакой юридической ответственности за любые ошибки или упущения, которые могут быть допущены.

Напечатано на бескислотной бумаге.

Springer является частью Springer Science+Business Media (www.springer.com).

Machine Translated by Google

Искусственный интеллект (ИИ) имеет определенную цель — понять интеллект и построить интеллектуальные системы. Однако методы и формализмы, используемые на пути к этой цели, не являются твердо установленными, в результате чего ИИ сегодня состоит из множества субдисциплин. Трудность вводного курса ИИ заключается в том, чтобы передать как можно больше ветвей, не теряя при этом слишком большой глубины и точности.

Книга Рассела и Норвига [RN10] является более или менее стандартным введением в ИИ. Однако, поскольку в этой книге 1152 страницы, и поскольку она слишком обширна и дорогостояща для большинства студентов, требования к написанию этой книги были ясны: она должна быть доступным введением в современный ИИ для самостоятельного изучения или в качестве основы для четырех часовая лекция, объемом не более 300 страниц. Результат перед вами.

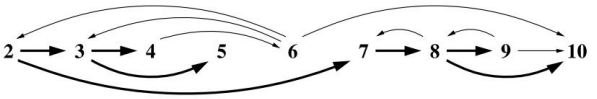
На 300 страницах невозможно полностью охватить такую обширную область, как ИИ. Чтобы не превращать книгу в оглавление, я попытался углубиться и представить конкретные алгоритмы и приложения в каждой из следующих ветвей: агенты, логика, поиск, рассуждение с неопределенностью, машинное обучение и нейронные сети.

Области обработки изображений, нечеткой логики и обработки естественного языка подробно не рассматриваются. Область обработки изображений, важная для всех компьютерных наук, представляет собой самостоятельную дисциплину с очень хорошими учебниками, такими как [GW08]. Аналогичный статус имеет обработка естественного языка. При распознавании и генерации текста и разговорной речи применяются методы логики, вероятностных рассуждений и нейронных сетей. В этом смысле это поле является частью ИИ. С другой стороны, компьютерная лингвистика представляет собой отдельную обширную отрасль компьютерных наук и имеет много общего с формальными языками. В этой книге мы укажем на такие подходящие системы в нескольких местах, но не будем давать систематического введения. В качестве первого введения в эту область мы обратимся к гл. 22 и 23 в [RN10]. Нечеткая логика, или теория нечетких множеств, превратилась в раздел теории управления из-за ее основного применения в технологии автоматизации и рассматривается в соответствующих книгах и лекциях. Поэтому мы воздержимся от введения здесь.

Зависимости между главами книги грубо показаны на графике, показанном ниже. Для простоты гл. 1, с основным введением для всех последующих глав, опущен. Например, более толстая стрелка от 2 до 3 означает, что логика высказываний является необходимым условием для понимания логики предикатов. Тонкая стрелка от 9 до 10 означает, что нейронные сети полезны для понимания обучения с подкреплением, но не абсолютно необходимы. Тонкий назад

Machine Translated by Google

стрелки должны указывать на то, что последующие главы могут дать более глубокое понимание
к темам, которые уже изучены.



Эта книга применима для студентов, изучающих информатику и другие технические естественные науки.
наук и, по большей части, требует знания математики на уровне средней школы. В нескольких местах
знания из линейной алгебры и многомерного анализа
необходим. Для более глубокого понимания содержания необходима активная работа над упражнениями.
Это означает, что к решениям следует обращаться только
после интенсивной работы над каждой проблемой и только для проверки своих решений, верных
Девиз Леонардо да Винчи «Учеба без преданности вредит мозгу». В некотором роде
более сложные задачи отмечены знаком , а особо сложные – знаком .
Проблемы, которые требуют программирования или специальных знаний в области компьютерных наук,
отмечены .

На веб-сайте книги по адресу www.hs-weingarten.de/~ertel/aibook цифровые материалы для
упражнений, такие как обучающие данные для алгоритмов обучения, страница со ссылками на
программы ИИ, упомянутые в книге, список ссылок на затронутые темы,
интерактивный список библиографии, список опечаток и слайды презентации для лекторов
может быть найден. Прошу читателей присылать предложения, критические замечания и советы по
ошибки напрямую на ertel@hs-weingarten.de.

Эта книга является обновленным переводом моей немецкой книги «Grundkurs Künstliche».
Intelligenz», опубликованном Vieweg Verlag. Отдельное спасибо переводчику
Натан Блэк, который в отличном трансатлантическом сотрудничестве между Германией и
Калифорния через SVN, Skype и электронную почту подготовила этот текст. Я благодарен Францу
Курфессу, который познакомил меня с Натаном; Мэтью Уайту за корректуру переведенного
книгу и Саймону Рису из Springer Verlag за его терпение.

Я хотел бы поблагодарить мою жену Эвелин за ее поддержку и терпение во время этого
трудоемкий проект. Особая благодарность Вольфгангу Бибелю и Крису Лобену Шуссу, которые тщательно
исправили немецкую рукопись. Их предложения и обсуждения привели ко многим улучшениям и
дополнениям. Для ознакомления с исправлениями и
другие ценные услуги, я хотел бы поблагодарить Richard Cubek, Celal Döven, Joachim
Фесслер, Нико Хохгешвендер, Пол Кирнер, Вильфрид Мейстер, Норберт Перк, Пе тер Радтке, Маркус
Шнайдер, Манфред Шрамм, Ули Старк, Мишель Токич, Арне
Усаделя и всех заинтересованных студентов. Я также благодарю Флориана Маста за
бесценные мультфильмы и очень эффективное сотрудничество.

Я надеюсь, что во время учебы эта книга поможет вам разделить мое увлечение
Искусственный интеллект.

Равенсбург Вольфганг Эртель
февраль 2011 г.

Machine Translated by Google

Содержание

1 Введение 1

1.1 Что такое искусственный интеллект? 1

1.1.1 Наука о мозге и решение проблем 3

1.1.2 Тест Тьюринга и Чаттерботы. 4

1.2 История ИИ. 5

1.2.1 Первые начинания. . 5

1.2.2 Логика решает (почти) все проблемы 1.2.3 Новый коннекционизм 6 7

1.2.4 Рассуждения в условиях неопределенности 7

1.2.5 Распределенные, автономные и обучающиеся агенты 8

1.2.6 ИИ взрослеет. 9

1.3 Агенты. 9

1.4 Системы, основанные на знаниях . 12

1.5 Упражнения 14

2 Пропозициональная логика. 15

2.1 Синтаксис. 15

2.2 Семантика. 16

2.3 Системы доказательств. 18

2.4 Разрешение 22

2.5 Оговорки Хорна. 25

2.6. Вычислимость и сложность. 2.7 Применение 28

и ограничения 2.8 Упражнения 29

29

3 Логика предикатов первого порядка. . . . 31

3.1 Синтаксис. 32

3.2 Семантика. 33

3.2.1 Равенство. 3.3 36

Кванторы и нормальные формы 3.4 37

Доказательства исчисления. 40

3.5 Разрешение 42

3.5.1 Стратегии разрешения . 46

3.5.2 Равенство. 46

3.6 Автоматические средства доказательства теорем. 47

Machine Translated by Google

3.7 Математические примеры. . . 3.8 48
Приложения. 51
3.9 Резюме. 54
3.10 Упражнения 54

4 Ограничения логики. 57
4.1 Проблема пространства поиска 4.2 57
Разрешимость и неполнота. 59
4.3 Летающий пингвин. 60
4.4 Моделирование неопределенности. 63
4.5 Упражнения 65

5 Логическое программирование с PROLOG 67
5.1 Системы и реализации PROLOG 68
5.2 Простые примеры. 68
5.3 Управление выполнением и процедурные элементы 71
5.4 Списки 5.5 73
Самомодифицирующиеся программы 75
5.6 Пример планирования. 5.7 5.8 76
Логическое программирование ограничений. 77
Резюме . 5.9 79
Упражнения 80

6 Поиск, игры и решение проблем . 6.1 Введение 83
6.2 Неинформированный 83
поиск 89
6.2.1 Поиск в ширину. 89
6.2.2 Поиск в глубину 91
6.2.3 Итеративное углубление 92
6.2.4 Сравнение. 94
6.3 Эвристический поиск. 94
6.3.1 Жадный поиск. 96
6.3.2 A -Поиск 98
6.3.3 IDA -Поиск 100
6.3.4 Эмпирическое сравнение алгоритмов поиска. 100
6.3.5 Резюме. 102
6.4 Игры с соперниками. 102
6.4.1 Минимаксный поиск 103
6.4.2 Альфа-бета-обрезка. 103
6.4.3 Недетерминированные игры. 106
6,5 Функции эвристической оценки 106
6.5.1 Изучение современных 107
6,6 эвристических 108
6,7 упражнений. 110

7 Рассуждения с неуверенностью. 113
7.1 Вычисления с вероятностями 115

Machine Translated by Google

7.1.1 Условная вероятность. 118
7.2 Принцип максимальной энтропии. 122
7.2.1 Правило вывода для вероятностей 7.2.2 123
Максимальная энтропия без явных ограничений. 127
7.2.3 Условная вероятность в сравнении с существенным следствием. . . 128
7.2.4 MaxEnt-Системы. 129
7.2.5 Пример Твити. 130
7.3 ЛЕКСМЕД, экспертная система диагностики аппендицита. 131
7.3.1 Диагностика аппендицита с помощью формальных 131
методов 7.3.2 Гибридно-вероятностная база знаний 132
7.3.3 Применение LEXMED 7.3.4 135
Функция LEXMED 7.3.5 Управление 136
рисками с использованием матрицы затрат 7.3.6 139
Производительность 141
7.3.7 Области применения и опыт 143
7.4 Рассуждения с помощью байесовских сетей. 144
7.4.1 Независимые переменные. 144
7.4.2 Графическое представление знаний как байесовское
Сеть 146
7.4.3 Условная независимость. 146
7.4.4 Практическое применение. 148
7.4.5 Программное обеспечение для байесовских сетей. 149
7.4.6 Разработка байесовских сетей. 150
7.4.7 Семантика байесовских сетей. 154
7.5 Резюме. 7.6 156
Упражнения 157

8 Машинное обучение и интеллектуальный анализ данных. 161
8.1 Анализ данных. 166
8.2 Персептрон, линейный классификатор. 169
8.2.1 Правило обучения 8.2.2 171
Оптимизация и перспективы 174
8.3 Метод ближайших соседей 175
8.3.1 Два класса, много классов, приближение. 179
8.3.2 Расстояние имеет значение 180
8.3.3 Время вычислений 8.3.4 181
Сводка и перспективы 182
8.3.5 Рассуждение по прецедентам 183
8.4 Обучение дереву решений. 184
8.4.1 Простой пример. 185
8.4.2 Энтропия как показатель содержания информации. 186
8.4.3 Получение информации 189
8.4.4 Применение C4.5. 191
8.4.5 Обучение диагностике аппендицита. 193
8.4.6 Непрерывные атрибуты 196

Machine Translated by Google

8.4.7 Сокращение — вырезание дерева 197
8.4.8 Отсутствующие значения 198
8.4.9 Резюме. 199
8.5 Изучение байесовских сетей. 199
8.5.1 Изучение структуры сети. 199
8.6 Наивный байесовский классификатор. 202
8.6.1 Классификация текста с помощью наивного Байеса. 204
8,7 Кластеризация 206
8.7.1 Метрики расстояния 207
8.7.2 k-средних и алгоритм EM 208
8.7.3 Иерархическая кластеризация 209
8.8 Интеллектуальный анализ данных на 211
практике 8.8.1 Инструмент интеллектуального анализа данных KNIME 212
8.9 Резюме. 214
8.10 Упражнения 216
8.10.1 Введение 8.10.2 216
Персептрон 216
8.10.3 Метод ближайших соседей 8.10.4 217
Деревья решений 8.10.5 218
Изучение байесовских сетей 219
8.10.6 Кластеризация 220
8.10.7 Интеллектуальный анализ данных. 220

9 Нейронные сети. 221
9.1 От биологии к моделированию. 222
9.1.1 Математическая модель 9.2 Сети 223
Хопфилда 226
9.2.1 Применение к примеру распознавания образов. . . . 227
9.2.2 Анализ. 228
9.2.3 Резюме и перспективы 231
9.3 Нейронно-ассоциативная память. 232
9.3.1 Память корреляционной матрицы 233
9.3.2 Псевдоинверсия. 235
9.3.3 Двоичное правило Хебба 236
9.3.4 Программа исправления правописания 238
9.4 Линейные сети с минимальными ошибками 240
9.4.1 Метод наименьших квадратов 241
9.4.2 Применение к данным аппендицита 9.4.3 242
Дельта-правило 9.4.4 243
Сравнение с персептроном 245
9.5 Алгоритм обратного распространения. 246
9.5.1 NETtalk: сеть учится говорить 249
9.5.2 Изучение эвристик для средств доказательства теорем. 250
9.5.3 Проблемы и улучшения 251
9.6 Машины опорных векторов 252

Machine Translated by Google

9.7 Приложения. 254
9.8 Резюме и перспективы . . 9.9 254
Упражнения 9.9.1 255
От биологии к моделированию. 255
9.9.2 Сети Хопфилда. 255
9.9.3 Линейные сети с минимальными ошибками 255
9.9.4 Обратное распространение 256
9.9.5 Машины опорных векторов 256

10 Обучение с подкреплением. 257
10.1 Введение 10.2 257
Задача 10.3 259
Неинформированный комбинаторный поиск 261
10.4 Итерация значений и динамическое программирование. 262
10.5 Обучаемый шагающий робот и его моделирование 265
10.6 Q-обучение. 267
10.6.1 Q-обучение в недетерминированной среде. . . . 270
10.7 Разведка и разработка 272
10.8 Аппроксимация, обобщение и сходимость. 272
10.9 Приложения. 273
10.10 Проклятие размерности. 274
10.11 Резюме и перспективы 275
10.12 Упражнения 276

11 решений для упражнений. 279
11.1 Введение 11.2 279
Логика высказываний. 280
11.3. Логика предикатов первого порядка. 282
11.4 Ограничения логики. 283
11.5 ПРОЛОГ. 283
11.6 Поиск, игры и решение проблем 285
11.7 Рассуждения с неопределенностью. 287
11.8 Машинное обучение и интеллектуальный анализ данных. 293
11.9 Нейронные сети. 300
11.10 Обучение с подкреплением. 301

Использованная литература 305

Индекс 311

Machine Translated by Google

Machine Translated by Google

Введение 1

1.1 Что такое искусственный интеллект?

Термин «искусственный интеллект» вызывает эмоции. Во-первых, это наше увлечение интеллектом, который, по-видимому, придает нам, людям, особое место среди форм жизни. Возникают такие вопросы, как «Что такое интеллект?», «Как можно измерить интеллект?» или «Как работает мозг?». Все эти вопросы имеют смысл при попытке понять искусственный интеллект. Однако центральным вопросом для инженера, особенно для специалиста по информатике, является вопрос об интеллектуальной машине, которая ведет себя как человек, демонстрируя разумное поведение.

Искусственный атрибут может вызывать самые разные ассоциации. Это вызывает страх перед разумными киборгами. Напоминает образы из фантастических романов. Это поднимает вопрос о том, следует ли нам попытаться понять, смоделировать или даже реконструировать наше высшее благо, душу.

С такими различными небрежными интерпретациями становится трудно дать простое и надежное определение термину « искусственный интеллект» или ИИ . Тем не менее я хотел бы попытаться, используя примеры и исторические определения, охарактеризовать область ИИ. В 1955 году Джон Маккарти, один из пионеров искусственного интеллекта, первым дал определение термину « искусственный интеллект» примерно следующим образом:

Цель ИИ — разработать машины, которые ведут себя так, как если бы они были разумными.

Чтобы проверить это определение, читатель может представить себе следующий сценарий. Пятнадцать или около того небольших роботизированных транспортных средств движутся по замкнутой поверхности размером четыре на четыре метра. Можно наблюдать различные модели поведения. Некоторые транспортные средства образуют небольшие группы с относительно небольшим движением. Другие мирно перемещаются по пространству и изящно избегают любых столкновений. Третьи, кажется, следуют за лидером. Наблюдается также агрессивное поведение. Является ли то, что мы наблюдаем, разумным поведением?

По определению Маккарти вышеупомянутых роботов можно назвать интеллектуальными. Психолог Валентин Брайтенберг показал, что это, казалось бы, сложное поведение может быть вызвано очень простыми электрическими цепями [Bra84]. Так называемые автомобили Брайтенберга имеют два колеса, каждое из которых приводится в движение независимым электродвигателем. На скорость каждого мотора влияет датчик света на

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по компьютерным наукам,
DOI 10.1007/978-0-85729-299-5_1, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google

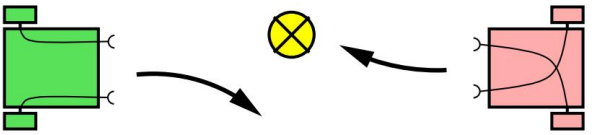


Рис. 1.1 Два очень простых автомобиля Брайтенберга и их реакция на источник света

передней части автомобиля, как показано на рис. 1.1. Чем больше света попадает на датчик, тем быстрее работает двигатель. Транспортное средство 1 в левой части рисунка по своей конфигурации удаляется от точечного источника света. Транспортное средство 2, с другой стороны, движется к источнику света. Дальнейшие небольшие модификации могут создать другие модели поведения, так что с помощью этих очень простых транспортных средств мы сможем реализовать впечатляющее поведение, описанное выше.

Ясно, что приведенного выше определения недостаточно, потому что целью ИИ является решение сложных практических задач, которые, безусловно, слишком требовательны к транспортному средству Брайтенберга. В Британской энциклопедии [Bri91] можно найти такое определение:

> ИИ — это способность цифровых компьютеров или роботов, управляемых компьютером, решать проблемы, которые обычно связаны с более высокими интеллектуальными способностями человека к обработке данных.

Но у этого определения есть и недостатки. Например, можно было бы признать, что компьютер с большой памятью, который может сохранять длинный текст и извлекать его по запросу, демонстрирует интеллектуальные возможности, поскольку запоминание длинных текстов, безусловно, можно считать более высокой интеллектуальной вычислительной способностью человека , как, например, быстрота. умножение двух 20-значных чисел. Таким образом, согласно этому определению, каждый компьютер представляет собой систему ИИ. Эта дилемма элегантно решается следующим определением Элейн Рич [Ric83]:

> Искусственный интеллект — это наука о том, как заставить компьютеры делать то, в чем люди на данный момент лучше.

Богато, лаконично и лаконично характеризует то, чем исследователи ИИ занимались последние 50 лет. Даже в 2050 году это определение будет актуальным.

Такие задачи, как выполнение множества вычислений за короткий промежуток времени, являются сильными сторонами цифровых компьютеров. В этом отношении они превосходят людей во много раз. Однако во многих других областях люди намного превосходят машины. Например, человек, вошедший в незнакомое помещение, за доли секунды узнает обстановку и, при необходимости, так же быстро примет решение и спланирует действия. На сегодняшний день эта задача слишком требовательна к автономным1 роботам. Следовательно, согласно определению Рича, это задача для ИИ. На самом деле исследования автономных роботов — важная и актуальная тема в области искусственного интеллекта. С другой стороны, создание шахматных компьютеров потеряло актуальность, потому что они уже играют на уровне гроссмейстеров или выше.

Однако было бы опасно делать из определения Рича вывод о том, что ИИ занимается только прагматичной реализацией интеллектуальных процессов. Интеллектуальные системы в смысле определения Рича не могут быть построены без глубокого

1Автономный робот работает самостоятельно, без ручной поддержки, в частности без дистанционного контроль.

Machine Translated by Google

понимание человеческого мышления и интеллектуальных действий в целом, из-за чего нейробиология (см. раздел 1.1.1) имеет большое значение для ИИ. Это также показывает, что другие приведенные определения отражают важные аспекты ИИ.

Особой силой человеческого интеллекта является адаптивность. Мы способны приспосабливаться к различным условиям окружающей среды и соответственно изменять свое поведение посредством обучения. Именно потому, что наши способности к обучению намного превосходят возможности компьютеров, машинное обучение , согласно определению Рича, является центральной частью ИИ.

1.1.1 Наука о мозге и решение проблем

Благодаря исследованию интеллектуальных систем мы можем попытаться понять, как работает человеческий мозг, а затем смоделировать или симулировать его на компьютере. Многие идеи и принципы в области нейронных сетей (см. главу 9) проистекают из науки о мозге и родственной области нейронауки.

Совершенно иной подход получается, когда вы выбираете целенаправленную линию действий, начиная с проблемы и пытаясь найти наиболее оптимальное решение. То, как люди решают проблему, здесь не имеет значения. Метод в этом подходе вторичен. Прежде всего, это оптимальное интеллектуальное решение проблемы. Вместо использования фиксированного метода (такого, например, как логика предикатов) ИИ имеет своей постоянной целью создание интеллектуальных агентов для максимально возможного числа различных задач. Поскольку задачи могут быть самыми разными, неудивительно, что методы, используемые в настоящее время в ИИ, часто также сильно различаются. Подобно медицине, которая включает в себя множество различных, часто спасающих жизнь диагностических и терапевтических процедур, ИИ также предлагает широкий спектр эффективных решений для самых разных приложений. Для мысленного вдохновения рассмотрите рис. 1.2 на стр. 4. Так же, как и в медицине, нет универсального метода для всех областей применения ИИ, а существует множество возможных решений для множества различных повседневных проблем, больших и малых.

Когнитивная наука занимается исследованием человеческого мышления на несколько более высоком уровне. Подобно науке о мозге, эта область снабжает практический ИИ многими важными идеями. С другой стороны, алгоритмы и реализации приводят к дальнейшим важным выводам о том, как функционирует человеческое мышление. Таким образом, эти три области выигрывают от плодотворного междисциплинарного обмена. Тем не менее, предметом этой книги в первую очередь является проблемно-ориентированный ИИ как раздел информатики.

Вокруг интеллекта и искусственного интеллекта возникает много интересных философских вопросов. У нас, людей, есть сознание; то есть мы можем думать о себе и даже размышлять о том, что мы способны думать о себе. Как возникает сознание? Многие философы и неврологи сейчас считают, что разум и сознание связаны с материей, то есть с мозгом. Вопрос о том, смогут ли машины когда-нибудь иметь разум или сознание, может в какой-то момент в будущем стать актуальным. Проблема разума и тела, в частности, касается того, связан ли разум с телом. Мы не будем обсуждать здесь эти вопросы. Заинтересованный читатель может обратиться к [Spe98, Spe97], и ему предлагается в ходе изучения технологии ИИ составить личное мнение по этим вопросам.

Machine Translated by Google

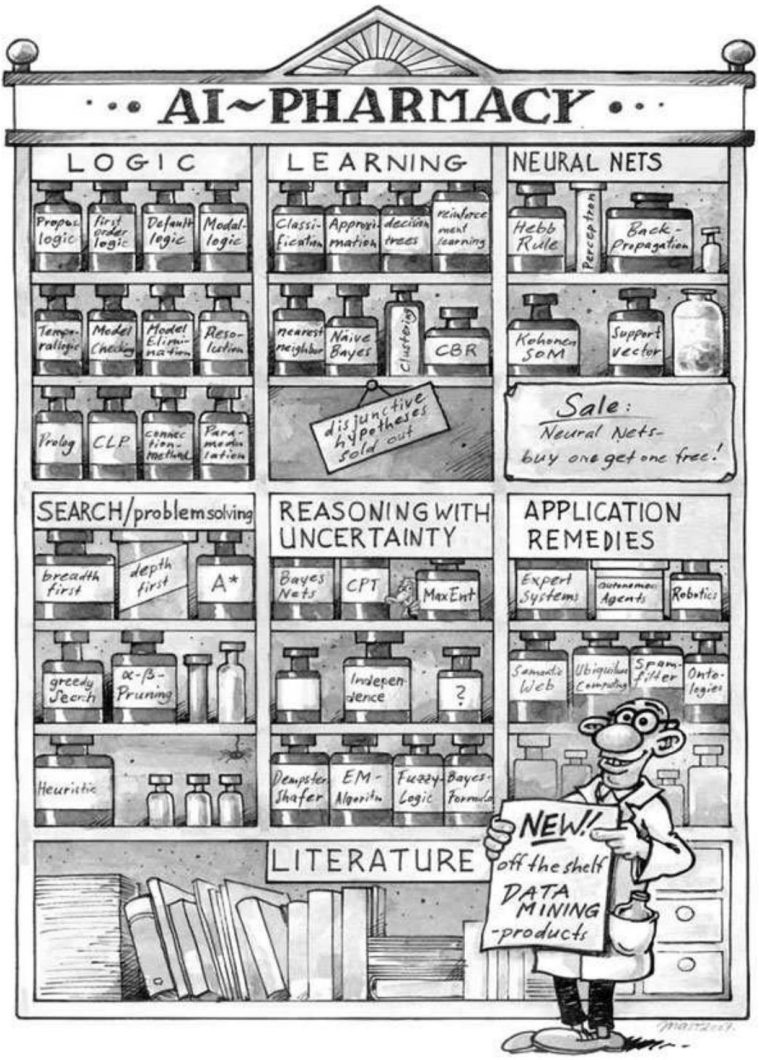


Рис. 1.2 Небольшая выборка решений, предлагаемых ИИ

1.1.2 Тест Тьюринга и болтуны

Алан Тьюринг прославился как один из первых пионеров ИИ своим определением интеллектуальной машины, в котором рассматриваемая машина должна пройти следующий тест. Испытуемый Алиса сидит в запертой комнате с двумя компьютерными терминалами. Один терминал подключен к машине, другой к незлонамеренному человеку Бобу. Алиса может вводить вопросы в оба терминала. Ей дается задание через пять минут решить, какой терминал принадлежит машине. Машина проходит тест, если она может обмануть Алису как минимум в 30% случаев [Tur50].

Machine Translated by Google

В то время как тест очень интересен с философской точки зрения, для практического ИИ, который занимается решением проблем, он не очень актуален. Причины этого аналогичны упомянутым выше, связанным с транспортными средствами Брайтенберга (см. упражнение 1.3 на стр. 14).

Пионер искусственного интеллекта и социальный критик Джозеф Вейценбаум разработал программу под названием « Элиза», которая должна отвечать на вопросы испытуемого подобно психологу-человеку [Wei66]. Он действительно был в состоянии продемонстрировать успех во многих случаях. Якобы у его секретаря часто были долгие дискуссии с программой. Сегодня в Интернете существует множество так называемых чат-ботов, первые ответы некоторых из них весьма впечатляют. Однако через некоторое время их искусственная природа становится очевидной. Некоторые из этих программ действительно способны к обучению, в то время как другие обладают экстраординарными знаниями по различным предметам, например географии или разработке программного обеспечения. Уже есть коммерческие приложения для чат-ботов в онлайн-поддержке клиентов, и могут быть другие в области электронного обучения. Вполне возможно, что учащийся и система электронного обучения могли бы общаться через чат-бота. Читатель может сравнить несколько болтунов и оценить их интеллект в упражнении 1.1 на странице 14.

1.2 История ИИ

ИИ опирается на многие прошлые научные достижения, которые здесь не упоминаются, поскольку ИИ как самостоятельная наука существует только с середины двадцатого века. Таблица 1.1 на стр. 10 с наиболее важными вехами ИИ и графическое представление основных движений ИИ на рис. 1.3 на стр. 6 дополняют следующий текст.

1.2.1 Первые начинания

В 1930-х годах Курт Гёдель, Алонсо Чёрч и Алан Тьюринг заложили важные основы логики и теоретической информатики. Особый интерес для ИИ представляют теоремы Гёделя. Теорема о полноте утверждает, что логика предикатов первого порядка полна. Это означает, что каждое истинное утверждение, которое может быть сформулировано в логике предикатов, доказуемо с использованием правил формального исчисления. На этой основе позже можно было бы построить автоматические средства доказательства теорем как реализации формальных исчислений. С помощью теоремы о неполноте Гёдель показал, что в логике высшего порядка существуют истинные утверждения, которые недоказуемы. Этим он раскрыл болезненные пределы формальных систем.

Доказательство Алана Тьюринга неразрешимости проблемы остановки также относится к этому периоду времени. Он показал, что не существует программы, которая может решить, будет ли заданная произвольная программа (и соответствующий ввод) выполняться в бесконечном цикле. С

2Логика высшего порядка является расширением логики предикатов, в которой не только переменные, но и функциональные символы или предикаты могут выступать в качестве терминов в квантификации. Действительно, Гёдель только показал, что любая система, основанная на логике предикатов и способная сформулировать арифметику Пеано, неполна.

Machine Translated by Google

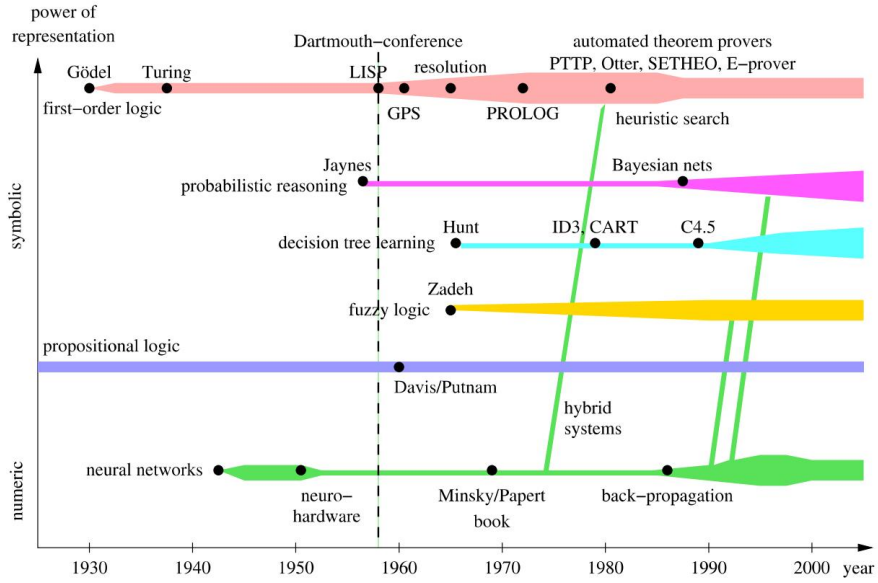


Рис. 1.3. История различных областей ИИ. Ширина столбцов указывает на распространенность использования метода.

этот Тьюринг также определил предел для интеллектуальных программ. Из этого следует, например, что универсальной системы проверки программ никогда не

будет.3 В 1940-х годах, основываясь на результатах нейробиологии, МакКаллох, Питтс и Хебб разработали первые математические модели нейронных сетей. Однако компьютерам в то время не хватало мощности для имитации простого мозга.

1.2.2 Логика решает (почти) все проблемы

ИИ как практическая наука о механизации мышления, конечно же, мог начаться только тогда, когда появились программируемые компьютеры. Так было в 1950-х годах. Ньюэлл и Саймон представили Logic Theorist, первое автоматическое средство доказательства теорем, и таким образом также показали, что с помощью компьютеров, которые на самом деле работают только с числами, можно также обрабатывать символы. В то же время Маккарти вместе с языком LISP представил язык программирования, специально созданный для обработки символьных структур. Обе эти системы были представлены в 1956 г. на исторической Дартмутской конференции, которая считается днем рождения ИИ.

В США LISP превратился в наиболее важный инструмент для реализации систем искусственного интеллекта, обрабатывающих символы. После этого правило логического вывода, известное как разрешение, превратилось в полное исчисление логики предикатов.

3Это утверждение относится к «полной правильности», что подразумевает доказательство правильного выполнения, а также доказательство завершения для каждого действительного ввода.

Machine Translated by Google

В 1970-х годах был представлен язык логического программирования PROLOG. Европейский аналог LISP. PROLOG предлагает то преимущество, что позволяет напрямую программирование с использованием предложений Хорна, подмножества логики предикатов. Как ЛИСП, ПРОЛОГ имеет типы данных для удобной обработки списков.

Вплоть до 1980-х годов в ИИ преобладал прорывной дух, особенно среди многие логики. Причиной тому стала череда впечатляющих достижений в обработка символов. С проектом компьютерных систем пятого поколения в Японии и программы ESPRIT в Европе, значительные инвестиции были направлены на строительство интеллектуальные компьютеры.

Для небольших задач автоматические пруверы и другие системы обработки символов иногда работал очень хорошо. Однако комбинаторный взрыв пространства поиска определил очень узкое окно для этих успехов. Эта фаза ИИ была описана в [RN10] как «Смотри, мама, без рук!» эпоха.

Поскольку экономический успех систем ИИ не оправдал ожиданий, финансирование исследований ИИ, основанных на логике, в Соединенных Штатах резко сократилось в 1980-х годах.

1.2.3 Новый коннекционизм

На этом этапе разочарования компьютерщики, физики и когнитивные ученые смогли показать, используя компьютеры, которые теперь были достаточно мощными, что математически смоделированные нейронные сети способны обучаться, используя обучающие примеры, для выполнения задач, которые ранее требовали дорогостоящего программирования. Благодаря отказоустойчивости таких систем и их способности распознавать образы стали возможными значительные успехи, особенно в распознавании образов. Распознавание лиц на фотографиях и распознавание рукописного ввода — два примера приложений. Система Nettalk смогла выучить речь из примеров текстов [SR86]. Под названием коннекционизм родилась новая субдисциплина ИИ.

Коннекционизм процветал, и субсидии текли рекой. Но скоро и здесь целесообразность пределы стали очевидны. Нейронные сети могут приобрести впечатляющие возможности, но обычно было невозможно зафиксировать изученное понятие в простых формулах или логические правила. Попытки объединить нейронные сети с логическими правилами или знаниями человеческих экспертов столкнулись с большими трудностями. Кроме того, нет удовлетворительного решения к структурированию и модуляризации сетей.

1.2.4 Рассуждение в условиях неопределенности

ИИ как практическая, целеустремленная наука искала выход из этого кризиса. Один хотел объединить способность логики явно представлять знания с силой нейронных сетей в работе с неопределенностью. Было предложено несколько альтернатив.

Наиболее многообещающее, вероятностное рассуждение работает с условными вероятностями для формул исчисления высказываний. С тех пор было построено много диагностических и экспертных систем для задач повседневного мышления с использованием байесовских сетей. Успех байесовских сетей обусловлен их интуитивной понятностью,

Machine Translated by Google

чистой семантики условной вероятности и многовековой, математически обоснованной теории вероятностей.

Слабые стороны логики, которая может работать только с двумя значениями истинности, могут быть решается с помощью нечеткой логики, которая прагматически вводит бесконечно много значений между нулем и единицей. Хотя и сегодня его теоретическая основа не вполне прочна, он успешно используется, особенно в технике управления.

Совсем другой путь привел к успешному синтезу логики и нейронных сетей. под названием гибридные системы. Например, нейронные сети использовались для изучить эвристики для сокращения огромного пространства комбинаторного поиска при обнаружении доказательств [SE90].

Методы обучения дерева решений на основе данных также работают с вероятностями. Такие системы, как CART, ID3 и C4.5, могут быстро и автоматически создавать очень точные деревья решений, которые могут представлять концепции пропозициональной логики, а затем использоваться как экспертные системы. Сегодня они являются фаворитом среди методов машинного обучения. (раздел 8.4).

Примерно с 1990 года интеллектуальный анализ данных развивался как поддисциплина ИИ в этой области. статистического анализа данных для извлечения знаний из больших баз данных. Данные Добыча полезных ископаемых не привносит в ИИ новых методов, а скорее требует использования больших баз данных для получения явных знаний. Одно приложение с большим рынком Потенциал заключается в управлении рекламными кампаниями крупного бизнеса на основе анализа многомиллионных покупок их клиентов. Как правило, такие методы машинного обучения, как поскольку здесь в игру вступает обучение дереву решений.

1.2.5 Распределенные, автономные и обучающиеся агенты

Распределенный искусственный интеллект, DAI, активно исследуется примерно с 1985. Одной из ее целей является использование параллельных компьютеров для повышения эффективности. решателей проблем. Однако оказалось, что из-за высокой вычислительной сложность большинства задач, использование «интеллектуальных» систем более выгодно чем само распараллеливание.

Совершенно другой концептуальный подход возникает в результате разработки автономных программных агентов и роботов, предназначенных для взаимодействия наподобие человеческих команд. Как и в случае с вышеупомянутыми автомобилями Брайтенберга, во многих случаях отдельный агент не способен решить проблему даже с неограниченными ресурсами. Только сотрудничество многих агентов приводит к разумному поведению или к решение проблемы. Колония муравьев или колония термитов способна возводить здания очень высокой архитектурной сложности, несмотря на то, что ни один муравейник понимает, как все это сочетается. Это похоже на ситуацию с обеспечение хлебом большого города, такого как Нью-Йорк [RN10]. Нет центрального агентства по планированию хлеба, вместо этого есть сотни пекарей, знающих свое дело. районах города и выпекать в этих местах необходимое количество хлеба.

Активное приобретение навыков роботами — захватывающая область текущих исследований. Там сегодня это, например, роботы, которые самостоятельно учатся ходить или выполнять различные двигательные навыки, связанные с футболом (глава 10). Совместное обучение нескольких роботов решать проблемы вместе все еще находится в зачаточном состоянии.

Machine Translated by Google

1.2.6 ИИ взрослеет

Вышеуказанные системы, предлагаемые ИИ сегодня, — это не универсальный рецепт, а мастерская с управляемым количеством инструментов для самых разных задач. Большинство этих инструментов хорошо разработаны и доступны в виде готовых программных библиотек, часто с удобными пользовательскими интерфейсами. Выбор правильного инструмента и его разумное использование в каждом отдельном случае остается за разработчиком ИИ или инженером по знаниям. Как и любое другое ремесленничество, это требует солидного образования, чему и призвана способствовать эта книга.

В большей степени, чем любая другая наука, ИИ является междисциплинарным, поскольку опирается на интересные открытия из таких различных областей, как логика, исследование операций, статистика, инженерия управления, обработка изображений, лингвистика, философия, психология и нейробиология. Кроме того, есть предметная область конкретного приложения.
Поэтому успешно разработать проект ИИ не всегда так просто, но почти всегда чрезвычайно увлекательно.

1.3 Агенты

Хотя термин «интеллектуальные агенты» не нов для ИИ, только в последние годы он получил известность благодаря [RN10], среди прочего. Агент обозначает в общем случае систему, которая обрабатывает информацию и производит результат на основе ввода. Эти агенты могут быть классифицированы по-разному.

В классической информатике в основном используются программные агенты (рис. 1.4 на стр. 11). В этом случае агент состоит из программы, которая вычисляет результат на основе пользовательского ввода.

С другой стороны, в робототехнике используются аппаратные агенты (также называемые роботами), которые дополнительно имеют в своем распоряжении датчики и исполнительные механизмы (рис. 1.5 на стр. 11). Агент может воспринимать окружающую среду с помощью сенсоров. С приводами он выполняет действия и изменяет свое окружение.

Что касается интеллекта агента, существует различие между рефлекторными агентами, которые реагируют только на входные данные, и агентами с памятью, которые также могут учитывать прошлое в своих решениях. Например, движущийся робот, который благодаря своим датчикам знает свое точное положение (и время), не имеет возможности, как рефлекторный агент, определить свою скорость. Однако, если он сохраняет позицию на коротких дискретных временных шагах, он может, таким образом, легко вычислить свою среднюю скорость в предыдущем временном интервале.

Если рефлекторный агент управляется детерминированной программой, он представляет собой функцию множества всех входов для множества всех выходов. С другой стороны, агент с памятью вообще не является функцией. Почему? (См. упражнение 1.5 на стр. 14.) Рефлекторных агентов достаточно в тех случаях, когда решаемая задача включает марковский процесс принятия решений. Это процесс, в котором для определения оптимального следующего действия необходимо только текущее состояние (см. главу 10).

Мобильный робот, который должен перейти из комнаты 112 в комнату 179 здания, совершает действия, отличные от действий робота, который должен перейти в комнату 105. Другими словами, действия зависят от цели. Такие агенты называются целевыми.

Machine Translated by Google

Таблица 1.1. Вехи развития ИИ от Гёделя до наших дней 1931 г. Австриец Курт Гёдель

показывает, что в логике предикатов первого порядка все истинные высказывания выводимы [Göd31a]. С другой стороны, в логике высшего порядка есть истинные утверждения, которые недоказуемы [Göd31b]. (В [Göd31b] Гёдель показал, что логика предикатов, расширенная аксиомами арифметики, неполна.)

1937 г. Алан Тьюринг указывает на пределы разумных машин с помощью проблемы остановки [Tur37].

1943 г. МакКаллох и Питтс моделируют нейронные сети и устанавливают связь с пропозициональными логика.

1950 Алан Тьюринг определяет машинный интеллект с помощью теста Тьюринга и пишет об обучающихся машинах и генетических алгоритмах [Tur50].

1951 Марвин Мински разрабатывает машину нейронной сети. С 3000 вакуумных ламп он имитирует 40 нейронов.

1955 Артур Сэмюэл (IBM) создает обучающую программу для игры в шахматы, которая играет лучше, чем ее разработчик [Sam59].

1956 Маккарти организует конференцию в Дартмутском колледже. Здесь имя Искусственный Впервые появился интеллект .

Ньюэлл и Саймон из Университета Карнеги-Меллона (CMU) представляют Logic Theorist, первую компьютерную программу для обработки символов [NSS83].

1958 Маккарти изобретает в Массачусетском технологическом институте (MIT) язык высокого уровня LISP. Он пишет программы, способные модифицировать себя.

1959 Гелернтер (IBM) создает средство для доказательства геометрических теорем.

1961 Общее средство решения проблем (GPS) Ньюэлла и Саймона имитирует человеческое мышление [NS61].

1963 Маккарти основывает лабораторию искусственного интеллекта в Стэнфордском университете.

1965 г. Робинсон изобретает резольвентное исчисление для логики предикатов [Rob65] (раздел 3.5).

1966 г. Программа Вейценбаума «Элиза» ведет диалог с людьми на естественном языке [Wei66] (раздел 1.1.2).

1969 Мински и Пейперт в своей книге «Персептроны» показывают , что персептрон, очень простой нейронная сеть может представлять только линейные функции [MP69] (раздел 1.1.2).

1972 г. Французский ученый Ален Кольмерауэр изобретает язык логического программирования PROLOG. (гл. 5).

Британский врач де Домбал разрабатывает экспертную систему для диагностики острой боли в животе [dDLS+72]. Это остается незамеченным в основном ИИ-сообществе того времени (раздел 7.3).

1976 г. Шортлифф и Бьюкенен разрабатывают MYCIN, экспертную систему для диагностики инфекционных заболеваний, способную справляться с неопределенностью (глава 7).

1981 Япония с большими затратами начинает «Проект пятого поколения» с целью создание мощной машины PROLOG.

1982 R1, экспертная система для настройки компьютеров, экономит корпорации Digital Equipment 40 миллионов долларов в год [McD82].

1986 Возрождение нейронных сетей благодаря, в частности, Румельхарту, Хинтону и Сейновски [RM86]. Система Nettalk учится читать тексты вслух [SR86] (гл. 9).

1990 Перл [Pea88], Cheeseman [Che85], Whittaker, Spiegelhalter привносят теорию вероятностей в ИИ с помощью байесовских сетей (раздел 7.4). Многоагентные системы становятся популярными.

1992 Программа Tesauros TD-gammon демонстрирует преимущества армирования обучение.

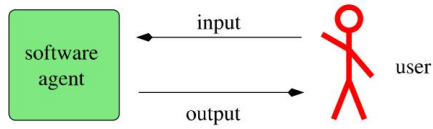
1993 Всемирная инициатива RoboCup по созданию автономных роботов, играющих в футбол [Roba].

Machine Translated by Google

Таблица 1.1 (продолжение)

1995	На основе статистической теории обучения Вапник разрабатывает машины опорных векторов, которые очень важны сегодня.
1997	Шахматный компьютер IBM Deep Blue побеждает чемпиона мира по шахматам Гэри Каспарова. Первые международные соревнования RoboCup в Японии.
2003	Роботы в RoboCup впечатляюще демонстрируют, на что способны ИИ и робототехника. достижение.
2006	Сервисная робототехника становится основным направлением исследований ИИ.
2010	Автономные роботы начинают изучать свою политику.
2011	Программа IBM для понимания естественного языка и ответов на вопросы «Watson» побеждает двух чемпионов среди людей в американской телевизионной викторине «Jeopardy!» (раздел 1.4).

Рис. 1.4 Программный агент с взаимодействием с пользователем



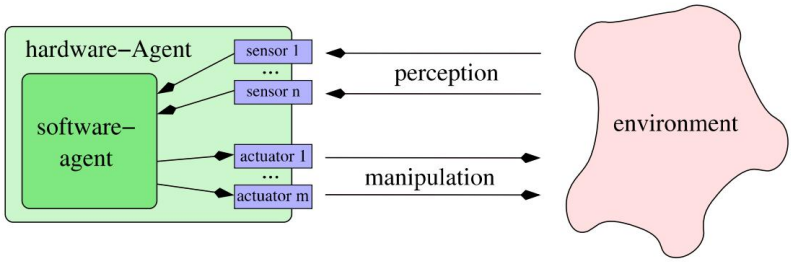


Рис. 1.5 Аппаратный агент

Пример 1.1. Спам-фильтр — это агент, который помещает входящие сообщения электронной почты в категории требуемых или нежелательных (спам) и удаляет любые нежелательные сообщения. Его цель как агента, основанного на целях, состоит в том, чтобы поместить все электронные письма в правильную категорию. В ходе этой не такой уж простой задачи агент может время от времени ошибаться. Поскольку его цель — правильно классифицировать все электронные письма, он попытается сделать как можно меньше ошибок. Однако это не всегда то, что имеет в виду пользователь. Сравним два следующих агента. Из 1000 электронных писем Агент 1 делает только 12 ошибок. Агент 2, с другой стороны, делает 38 ошибок с той же 1000 электронных писем. Стало быть, он хуже Агента 1? Ошибки обоих агентов более подробно показаны в следующей таблице, так называемой «матрице путаницы»:

Агент 1:

		правильный класс	
		хотел спам	1
спам-фильтр решает	разыскивается	189	
	спам	11	799

Агент 2:

		правильный класс	
		разыскиваемый	
спам-фильтр решает	спам разыскиваемый	200	
	38 спам	0	762

Machine Translated by Google

Агент 1 на самом деле делает меньше ошибок, чем агент 2, но эти несколько ошибок являются серьезными, поскольку пользователь теряет 11 потенциально важных электронных писем. Поскольку в данном случае имеется два типа ошибок разной серьезности, каждая ошибка должна быть взвешена с соответствующим коэффициентом стоимости (см. раздел 7.3.5 и упражнение 1.7 на стр. 14).

Сумма всех взвешенных ошибок дает общую стоимость, вызванную ошибочными решениями. Целью агента, основанного на затратах, является минимизация стоимости ошибочных решений в долгосрочной перспективе, то есть в среднем. В разд. 7.3 мы познакомимся с диагностической системой ЛЕКСМЕД как с примером агента, основанного на затратах.

Аналогично, цель агента, основанного на полезности, состоит в том, чтобы максимизировать полезность, полученную от правильных решений в долгосрочной перспективе, то есть в среднем. Сумма всех решений, взвешенных по соответствующим факторам полезности, дает общую полезность.

Особый интерес для ИИ представляют обучающие агенты, которые способны изменять себя на обучающих примерах или с помощью положительной или отрицательной обратной связи, так что средняя полезность их действий со временем растет (см. главу 8).

Как упоминалось в разд. 1.2.5, все чаще используются распределенные агенты , интеллект которых не локализован в одном агенте, а может быть виден только через сотрудничество многих агентов.

Дизайн агента, наряду с его целью, сильно ориентирован на его среду или, альтернативно, на его картину среды, которая сильно зависит от его сенсоров. Среда является наблюдаемой , если агент всегда знает полное состояние мира. В противном случае окружающая среда будет наблюдаться лишь частично.

Если действие всегда приводит к одному и тому же результату, то среда детерминирована.
В противном случае он недетерминирован. В дискретной среде происходит только конечное количество состояний и действий, тогда как в непрерывной среде может быть бесконечно много состояний или действий.

1.4 Системы, основанные на знаниях Агент — это

программа, реализующая отображение восприятий в действия. Для простых агентов такой взгляд на проблему достаточен. Для сложных приложений, в которых агент должен иметь возможность полагаться на большой объем информации и предназначен для выполнения сложной задачи, программирование агента может быть очень дорогостоящим и неясным, как действовать дальше. Здесь ИИ дает четкий путь, по которому нужно следовать, что значительно упростит работу.

Сначала мы отделяем знание от системы или программы, которая использует знание, например, для того, чтобы делать выводы, отвечать на вопросы или разрабатывать план.
Эта система называется механизмом вывода. Знания хранятся в базе знаний (КБ). Приобретение знаний в базе знаний обозначается как «Инженерия знаний» и основано на различных источниках знаний, таких как люди-эксперты, инженер по знаниям и базы данных. Системы активного обучения также могут приобретать знания путем активного исследования мира (см. главу 10). На рис. 1.6 на стр. 13 представлена общая архитектура систем, основанных на знаниях.

Движение к разделению знаний и умозаключений имеет несколько важных преимуществ. Разделение знаний и логического вывода может позволить реализовать системы логического вывода в значительной степени независимо от приложений. Например, это намного

Machine Translated by Google

Символ функции, 32
Нечеткая логика, 8, 13, 29, 64, 115

г

Гауссовский процесс, 181, 215, 272
Общий решатель проблем, 10
Обобщение, 162
Генетическое программирование, 75
Иди, 102, 103, 108
Гол, 28
стек, 28
штат, 87
Гёдель
Курт, 5
теорема о полноте, 5, 41 теорема о
неполноте, 5, 60
GPS, 10
Градиентный спуск, 244
Жадный поиск, 96, 97, 201
Земельный срок, 41

час

Проблема с остановкой, 5
Правило Хебба, 225, 234, 247,
двоичное, 236
Эвристика, 46, 57, 94
Функция эвристической оценки, 95, 98
Иерархическое обучение, 274
Сеть Хопфилда, 226, 227, 237
Оговорка Хорна, 26, 73
Хьюгин, 150

я

ID3, 184
IDA -алгоритм, 100, 250
Немедленная награда, 260
Значение, 16
Пошаговый градиентный спуск, 245
Инкрементное обучение, 243
Независимая, 119
условно, 146, 147, 155
Безразличие, 126
Безразличные переменные, 126
Логическая машина, 42
Механизм вывода, 12
Информативность, 189
Прирост информации, 186, 189, 215
Входное разрешение, 46
Интерпретация, 16, 33
Итеративное углубление, 92, 100

дж

Джавабайес, 150

К

К метод ближайшего соседа, 178, 180
К-значит, 208
Ядро, 181, 253
Метод ядра, 253 тыс.
Метод ближайшего соседа, 180
НОЖ, 185, 211, 212
Знания, 12 базы,
12, 18, 145
последовательных,
23 инженера, 9, 12
инженерии, 12
источников, 12

л

Предположение Лапласа, 117
Вероятности Лапласа, 117
Закон экономики, 197
Ленивое обучение, 182, 215
Обучение, 161
пакет, 243
путем демонстрации, 275
распределенных,
274 иерархических,
274 инкрементных, 201,
243 машинных,
136 многоагентных,
274 подкрепленных, 89, 107, 214
полуконтролируемых, 214
контролируемых, 161, 206, 238
ТД-, 273
Агент по обучению, 164 года
Фаза обучения, 164
Скорость обучения, 225, 244
Метод наименьших квадратов, 143, 241, 242, 245
ЛЕКСМЕД, 115, 122, 131, 193
Ограниченные ресурсы, 95
Линейное приближение, 245
Линейно отделимые, 169, 170
ГУБЫ, 68
ЛИСП, 6, 10
Буквенный,
21 дополнительный, 23
Локально взвешенная линейная регрессия, 183
Логика
нечеткая, 29, 64, 115
высшего порядка, 59, 60
вероятностная, 13, 29
Теоретик логики, 6, 10
Логически верно, 17

Machine Translated by Google

М

Машинное обучение, 134, 161
Манхэттенское расстояние, 100, 207
Предельное распределение, 121
Маргинализация, 120, 121, 123, 155
Марковский процесс принятия решений,
9, 273 недетерминированный,
270 частично наблюдаемый, 261
Марковские процессы принятия решений, 261
Материальный смысл, 16, 115, 128
MaxEn t, 115, 126, 129, 131, 134, 149, 156 раздача, 126

МДП, 261, 263, 273
детерминированный,
269 недетерминированный, 270
Запоминание, 161
Обучение на основе памяти, 182, 183
МГУ, 44
Минимальное остовное дерево, 210
Модель, 17 лет
Модус поненс, 22, 30, 114, 125
Импульс, 251
Монотонный, 61
Мультиагентное обучение, 274
Мультиагентные системы, 10
МИКИН, 10, 114, 132

Н

Наивный байесовский классификатор, 143, 144, 157, 166,
175, 202, 204, 219,
298, 202, 204
Наивный реверс, 74
Уравнение Навье – Стокса, 274
Классификация
ближайших соседей,
метод 176, 175
Алгоритм ближайшего соседа, 210
Отрицание, 16
Отрицание как неудача, 73
Нейронная сеть, 6, 10, 180, 181, 221
Рекуррентные
нейронные сети, 226, 231
Нейроинформатика, 231
Неврология, 3
Нейротрансмиттер, 223
Шум, 177
Немонотонная логика, 63, 130
Нормальные уравнения, 242
Нормальная
форма конъюнктива,
21 пренекс, 37

О

Наблюдаемый, 89, 103
Бритва Оккама, 197
Автономный алгоритм, 89
Алгоритм онлайн, 89
Онтология, 53
Или ветки, 70
Ортонормированный, 234
Отелло, 102, 103
Переоснащение, 177, 197, 198, 201, 240, 243, 252, 273

СОВ, 53 года

п

Парадокс, 60
Парамодуляция, 47
Частично наблюдаемый марковский процесс принятия решений,
261
Проблема пингвинов, 77
Персептрон, 10, 169, 170, 224
Фазовый переход, 230
ПИТ, 129, 130, 149, 157
ПЛ1, 13, 32
Планирование, 76
Политика, 260
градиентный метод, 273
оптимальный, 260
ПОМДП, 261, 273
Постусловие, 52
Предварительное условие, 52
Логика предикатов, 5
первого порядка, 13, 32
Обучение предпочтениям, 166
Помещение, 26
Вероятностная
логика, 13, 63
рассуждения, 7
Вероятность, 115, 117
распределение, 118
правил, 137
Правило продукта, 120
Верификация программы, 51
ПРОЛОГ, 7, 10, 20, 28, 67
Система доказательств, 19
Переменные предложения, 15
Исчисление
высказываний,
13 логика, 15
Обрезка, 192, 198
Псевдоинверсия, 236
Чисто буквальное правило, 46, 55

Вопрос

Q-обучение, 267, 275
конвергенция, 269

Machine Translated by Google

Быстрый винт, 252

р

Случайная величина, 116

Быстрое прототипирование, 79

РДФ, 53 года

Решение в реальном времени, 95

Требование в реальном времени, 103

Рабочая характеристика приемника, 142

Армирование

обучение, 257, 260

отрицательных,

260 положительных, 260

Резолюция, 6, 22

исчисление, 10, 20

правило, 22, 30

общее, 22, 43

СЛД, 27

Решающая, 22

Награда

со скидкой, 260

немедленно, 260, 269

Управление рисками, 141

РобоКубок, 11, 273

Робот, 9, 258

ходьба-, 258

Робототехника, 257

Кривая ROC, 143, 214

РПроп, 252

С

Образец, 185

Удовлетворительная, 17

Диаграмма рассеяния, 163

Оценка: 132, 143, 203, 219, 242

Алгоритм

поиска, 86 полных,

87 оптимальных,

89 эвристических,

84 пробельных,

23, 27, 40, 46 древовидных,

86

неинформированных, 84

Самоорганизующиеся карты, 255

Семантические деревья, 28

Семантическая паутина, 53

Семантика

декларативная (PROLOG), 70

процедурная (PROLOG), 70, 73

Полуразрешимый, PL1, 59

Полуконтролируемое обучение, 214

Чувствительность, 141, 148

Датчик, 9

Набор стратегий поддержки, 46, 55

Сигмовидная функция, 225, 241, 246

Подпись, 15

Сходство, 175

Имитация отжига, 232

Ситуационный расчет, 63

Сколемизация, 39

Разрешение SLD, 30

Повторное использование программного обеспечения, 52

Решение, 87

Звук, 20

Спам, фильтр

204, 11, 204

Специфика, 141

Исходное состояние, 87

Состояние, 87, 258, 259

пробел, 87

функция перехода, 260

Статистическая индукция, 136

Подгол, 28, 69

Аксиома подстановки, 36

Сусумпшн, 46

Опорный вектор, 252

машина, 252, 272

Машина опорных векторов, 181

SVM, см. машину опорных векторов

Т

Целевая функция, 164

Тавтология, 17

TD

-ошибка, 271

-гаммон, 273

-обучение, 271, 273

Учебный ящик, 275

Ошибка временной

разности,

271 обучение, 271

Срок, 32

переписка, 47

Данные испытаний, 164, 197

Текст

классификация, 204 добыча

полезных ископаемых, 166

Доказательство теорем, 6, 42, 43, 47, 51, 52

Тренировочные данные, 58, 164, 197

Функция перехода, 270

Правда, 34

Таблица истинности,

16 метод, 19

Тьюринг

Алан, 5

тест, 4

Твити пример, 60, 63, 130

Machine Translated by Google

U
Объединяемый, 44
Объединение, 44
Объединитель,
44 наиболее общий, 44
Поиск единой стоимости, 90

Единичная статья,
разрешение 46, разрешение 46
Неудовлетворительный, 17

B
Действительно, 17, 34
Итерация значения, 263
переменная, 32
ВДМ-СЛ, 52
Спецификация Венского метода разработки
Язык, 52
Диаграмма Вороного, 177

Вт
Шагающий робот, 258
ВАМ, 68, 75
Абстрактная машина Уоррена, 68 лет
Уотсон, 13 лет
ВЕКА, 185, 211

Machine Translated by Google

$$P(\text{Класс} = \text{хорошо} \mid y \quad 3) = \frac{P(y \quad 3 \mid \text{Класс} = \text{хорошо}) \cdot P(\text{Класс} = \text{хорошо})}{P(y \quad 3)}$$

$$"=" \frac{9/2 \cdot 13/9}{13/6} "=" \frac{1}{3}.$$

(b) P (y > 3 | Класс = хорошо) = 7/9.

Упражнение 7.3

(а) восемь событий.

(б)

$$P(\text{Prec} = \text{сухо} \mid \text{Небо} = \text{ясно}, \text{Bar} = \text{поднимается})$$

$$"=" \frac{P(\text{Prec} = \text{сухо}, \text{Sky} = \text{ясно}, \text{Bar} = \text{поднимается})}{P(\text{небо} = \text{чистое}, \text{полоса} = \text{поднимается})} "=" \frac{0,4}{0,47} = 0,85.$$

(c) Отсутствует строка в таблице: Облачно Дождь 0,12

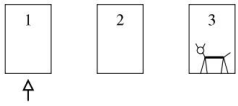
$$P(\text{Prec} = \text{дождь} \mid \text{Sky} = \text{облачно}) = \frac{P(\text{Prec} = \text{дождь}, \text{Sky} = \text{облачно})}{P(\text{небо} = \text{облачно})}$$

$$"=" \frac{0,23}{} = 0,66. \; 0,35$$

(d) Отсутствующая вероятностная мера равна 0,15. Принцип безразличия (Определение 7.5 на стр. 126) теперь требует, чтобы обеим отсутствующим строкам было симметрично присвоено значение 0,075.

Упражнение 7.4

Кандидат сначала выбрал дверь номер 1. Затем ведущий открыл дверь номер 3. Мы вводим аббревиатуру Ai для «Автомобиль находится за дверью номер i (до начала эксперимента) и Mi для «Ведущий (модератор) открывает дверь номер i» . Тогда P (A1) = P (A2) = P (A3) = 1/3, и мы вычисляем



$$P(A1 \mid M3) = \frac{П(M3 \mid A1)П(A1)}{П(M3)}$$

$$"=" \frac{П(M3 \mid A1)П(A1)}{P(M3 \mid A1)P(A1) + P(M3 \mid A2)P(A2) + P(M3 \mid A3)P(A3)} \; 1/2 \cdot 1/3 \; 1/2 \cdot 1/3 + 1 \cdot 1 /$$

$$"=" \frac{3 + 0 \cdot 1/3}{} "=" \frac{1/2 \cdot 1/3 =}{1/3, \; 1/2}$$

$$P(A2 \mid M3) = \frac{П(M3 \mid A2)П(A2)}{П(M3)} "=" \frac{1 \cdot 1/3 =}{2/3. \; 1/2}$$

Таким образом понятно, что лучше переключиться на другую дверь.

Machine Translated by Google

Упражнение 7.5. Функция Лагранжа имеет вид L = $\sum_{i=1}^{n}$ пи lnpi + λ(n i=1 пи 1).
приравнивание частных производных по pi и pj к нулю дает

$$\frac{L}{} = \ln p_i \quad 1 + \lambda = 0 \text{ и } \frac{L}{} = \ln p_j \quad 1 + \lambda = 0.$$

Вычитание этих двух уравнений дает pi = pj для всех i, j {1,...,n}. Таким образом, p1 = p2 =⋯= pn = 1/n.

Упражнение 7.6

Входные данные PIT: var A{t,f}, B{t,f}; P([A=t]) = 0,5; P([B=t] | [A=t]) = 0,94; КП([B=t]);	Выход ПИТ: Отчет о состоянии запросов Nr Истинное значение Вероятность Запрос 1 НЕ УКАЗАНО 7.200e-01 QP([B=t]);

Поскольку PIT работает численно, он не может выполнять символьные вычисления с помощью ра. рамметры α и β, а только конкретными цифрами.

Упражнение 7.7

Если: p1 = P (A, B), p2 = P (A, ¬B), p3 = P (¬A, B), p4 = P (¬A, ¬B)

$$\text{Ограничения: } p_1 + p_2 = \alpha \tag{11.1}$$

$$p_4 = 1 - p \tag{11.2}$$

$$p_1 + p_2 + p_3 + p_4 = 1 \tag{11.3}$$

Отсюда следует, что p3 = β α. Из (11.1) мы заключаем, основываясь на безразличии, что p1 = p2 = α/2. Таким образом, P (B) = p1 + p3 = α/2 + β α = β α/2 = P (A B) 1/2P (A). Соответствующая программа PIT читает

переменная A{t,f}, B{t,f}; P([A=t])
= 0,6; P([B=t] ИЛИ [A=t])
= 0,94; QP([B=t]).

Упражнение 7.8 Функция Лагранжа имеет вид

$$Л = -\sum_{я=1}^{4} p_i \ln p_i + \lambda_1(p_1+p_2 \quad \alpha) + \lambda_2(p_1+p_3 \quad \gamma) + \lambda_3(p_1+p_2+p_3+p_4 \quad 1). \tag{11.4}$$

Взяв частные производные по p1, p2, p3, p4 , получим

$$\frac{L}{} = \ln p_1 \quad 1 + \lambda_1 + \lambda_2 + \lambda_3 = 0, \tag{11,5}$$

Machine Translated by Google

$$\frac{L}{} = \ln p2 \quad 1 + \lambda 1 + \lambda 3 = 0, \quad p2 \tag{11.6}$$

$$\frac{L}{} = \ln p3 \quad 1 + \lambda 2 + \lambda 3 = 0, \quad p3 \tag{11.7}$$

$$\frac{L}{p4} = \ln p4 \quad 1 + \lambda 3 = 0 \tag{11.8}$$

и вычислить

$$(11.5)–(11.6): \ln p2 \quad \ln p1 + \lambda 2 = 0, \tag{11,9}$$

$$(11.7)–(11.8): \ln p4 \quad \ln p3 + \lambda 2 = 0, \tag{11.10}$$

откуда следует, что p3/p4 = p1/p2, что мы сразу подставляем в (7.12) на стр. 129:

$$\frac{p2p3}{п4} + p2 + p3 + p4 = 1$$

$$p2\ 1 + p4 \quad \frac{p3}{} + p3 + p4 = 1 \tag{11.11}$$

$$(7.10)–(7.11): p2 = p3 + \alpha \quad \gamma$$

что при замене p2 дает $(p3 + \alpha \quad \gamma)(1 + \frac{p3}{п4}) + p3 + p4 = 1$

$$(7.10) \text{ в } (7.12): \alpha + p3 + p4 = 1. \tag{11.12}$$

Таким образом, мы исключаем p4 в (11.8), что приводит к

$$(p3 + \alpha \quad \gamma)\ 1 + 1 \quad \frac{p3}{\alpha \quad p3} + 1 - a = 1 \tag{11.13}$$

$$(p3 + \alpha \quad \gamma)(1 \quad \alpha \quad p3 + p3) = \alpha(1 \quad \alpha \quad p3) \tag{11.14}$$

$$(p3 + \alpha \quad \gamma)(1 \quad \alpha) = \alpha(1 \quad \alpha \quad p3) \tag{11.15}$$

$$p3 + \alpha \quad \gamma \quad \alpha p3 \quad \alpha \quad ^2 + \alpha\gamma = \alpha \quad \alpha \quad 2 - ap3 \tag{11.16}$$

$$p3 = \gamma\ (1 \quad \alpha). \tag{11.17}$$

Из (11.12) следует, что p4 = (1 α)(1 γ), а из (11.11) получаем p2 = α(1 γ) и p1 = αγ.

Упражнение

7.9 (а)

$$M = \begin{matrix} 0 & 1000 \\ 10 & 0 \end{matrix}.$$

(б)

$$M \cdot \begin{matrix} п \\ 1 - p \end{matrix} \quad "=" \quad \begin{matrix} 1000 \cdot (1 \quad п) \\ 10 \cdot п \end{matrix}.$$

Machine Translated by Google

Поскольку решение между удалением или чтением определяется путем установления минимального из двух значений, достаточно сравнить два значения: 1000 (1 - p) < 10p, что дает p > 0,99. В общем, для матрицы $M = \begin{smallmatrix} 0 & k \\ 1 & 0 \end{smallmatrix}$ мы вычисляем порог k/(k + 1).

Упражнение

7.10 (a) Поскольку A и B независимы, P (A |B) = P (A) = 0,2. (b)
Применяя правило кондиционирования (стр. 155), мы получаем

$$P(C|A) = P(C|A, B)P(B)+P(C|A,\neg B)P(\neg B) = 0{,}2\cdot 0{,}9+0{,}1\cdot 0{,}1 = 0{,}19.$$

Упражнение 7.11

(а)

$$\begin{aligned}
P(Al) &= P(Al, Bur, Ear) + P(Al, \neg Bur, Ear) \\
&\quad + P(Al,Bur,\neg Ear) + P(Al,\neg Bur, \neg Ear) \\
&= P(Al|Bur,Ear)P(Bur,Ear) + P(Al|\neg Bur,Ear)P(\neg Bur,Ear) \\
&\quad + P(Al|Bур,\neg Ear)P(Bур,\neg Ear) \\
&\quad + P(Al|\neg Bur,\neg Ear)P(\neg Bur, \neg Ear) \\
&= 0{,}95 \cdot 0{,}001 \cdot 0{,}002 + 0{,}29 \cdot 0{,}999 \cdot 0{,}002 + 0{,}94 \cdot 0{,}001 \cdot 0{,}998 \\
&\quad + 0{,}001 \cdot 0{,}999 \cdot 0{,}998 \\
&= 0{,}00252,
\end{aligned}$$

$$\begin{aligned}
P(J) &= P(J,Al) + P(J,\neg Al) = P(J|Al)P(Al) + P(J|\neg Al)P(\neg Al) \\
&= 0{,}9 \cdot 0{,}0025 + 0{,}05 \cdot (1 \quad 0{,}0025) = 0{,}052,
\end{aligned}$$

$$\begin{aligned}
P(M) &= P(M,Al) + P(M,\neg Al) = P(M|Al)P(Al) + P(M|\neg Al)P(\neg Al) = 0{,}7 \cdot 0{,}0025 + \\
&\quad 0{,}01 \cdot (1 - 0{,}0025) = 0{,}0117.
\end{aligned}$$

(б)

$$\begin{aligned}
P(Al|Bur) &= P(Al|Bur,Ear)P(Ear) + P(Al|Bur,\neg Ear)P(\neg Ear) \\
&= 0{,}95 \cdot 0{,}002 + 0{,}94 \cdot 0{,}998 = 0{,}94002,
\end{aligned}$$

$$\begin{aligned}
P(M|Bur) &= P(M,Bur)/P(Bur) = [P(M,Al,Bur) + P(M,\neg Al,Bur)] \\
&\quad / P(Bур) \\
&= [P(M|Al)P(Al|Bur)P(Bur) + P(M|\neg Al)P(\neg Al|Bur)P(Bur)] \\
&\quad / P(Bур) \\
&= P(M|Al)P(Al|Bur) + P(M|\neg Al)P(\neg Al|Bur) \\
&= 0{,}7 \cdot 0{,}94002 + 0{,}01 \cdot 0{,}05998 = 0{,}659.
\end{aligned}$$

(с)

$$P(Bур|M) = \frac{P(M|Bур)P(Bур)}{П(M)} \text{ "=" } \frac{0{,}659 \cdot 0{,}001}{0{,}0117} = 0{,}056.$$

Machine Translated by Google

(г)

$$P(\text{Ал} \mid \text{Дж}, M) = \frac{P(\text{Ал}, \text{Дж}, M)}{П(\text{Дж}, M)} = \frac{P(\text{Ал}, \text{Дж}, M)}{P(\text{Al}, J, M) + P(\neg\text{Al}, J, M)}$$

$$= \frac{1}{1 + \frac{P(\neg\text{Al}, J, M)}{P(\text{Ал}, \text{Дж}, M)}} = \frac{1}{\frac{P(J \mid \neg\text{Al})P(M \mid \neg\text{Al})P(\neg\text{Al}) \; 1 + }{P(J \mid \text{Al})P(M \mid \text{Al})P(\text{Al})}}$$

$$= \frac{1}{1 + \frac{0{,}05 \cdot 0{,}01 \cdot 0{,}9975}{0{,}9 \cdot 0{,}7 \cdot 0{,}0025}} = 0{,}761,$$

$$P(J, M \mid \text{Bur}) = P(J, M \mid \text{Al})P(\text{Al} \mid \text{Bur}) + P(J, M \mid \neg\text{Al})P(\neg\text{Al} \mid \text{Bur})$$

$$= P(J \mid \text{Al})P(M \mid \text{Al})P(\text{Al} \mid \text{Bur})$$

$$+ P(J \mid \neg\text{Al})P(M \mid \neg\text{Al})P(\neg\text{Al} \mid \text{Bur}))$$

$$= 0{,}9 \cdot 0{,}7 \cdot 0{,}94 + 0{,}05 \cdot 0{,}01 \cdot 0{,}06 = 0{,}5922,$$

$$P(\text{Al} \mid \neg\text{Bur}) = P(\text{Al} \mid \neg\text{Bur}, \text{Ear})P(\text{Ear}) + P(\text{Al} \mid \neg\text{Bur}, \neg\text{Ear})P(\neg\text{Ear})$$

$$= 0{,}29 \cdot 0{,}002 + 0{,}001 \cdot 0{,}998 = 0{,}00158,$$

$$P(J, M \mid \neg\text{Bur}) = P(J, M \mid \text{Al})P(\text{Al} \mid \neg\text{Bur}) + P(J, M \mid \neg\text{Al})P(\neg\text{Al} \mid \neg\text{Bur})$$

$$= P(J \mid \text{Al})P(M \mid \text{Al})P(\text{Al} \mid \neg\text{Bur})$$

$$+ P(J \mid \neg\text{Al})P(M \mid \neg\text{Al})P(\neg\text{Al} \mid \neg\text{Bur}))$$

$$= 0{,}9 \cdot 0{,}7 \cdot 0{,}00158 + 0{,}05 \cdot 0{,}01 \cdot 0{,}998 = 0{,}00149,$$

$$P(J, M) = P(J, M \mid \text{Bur})P(\text{Bur}) + P(J, M \mid \neg\text{Bur})P(\neg\text{Bur})$$

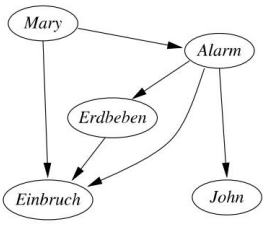
$$= 0{,}5922 \cdot 0{,}001 + 0{,}00149 \cdot 0{,}999 = 0{,}00208,$$

$$P(\text{Bur} \mid J, M) = \frac{P(J, M \mid \text{Bur})P(\text{Bur})}{П(\text{Дж}, M)} = \frac{0{,}5922 \cdot 0{,}001}{0{,}00208} = 0{,}284.$$

(д) $P(J)P(M) = 0{,}052 \cdot 0{,}0117 = 0{,}00061$, но $P(J, M) = 0{,}00208$ (см. выше).
Следовательно, $P(J)P(M) = P(J, M)$.

(f) См. разд. 7.4.5 на стр. 149.

(г)



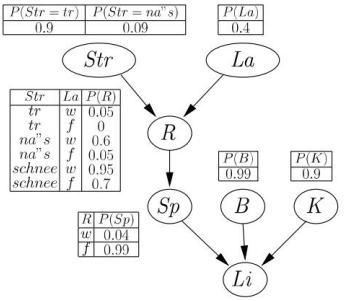
(h) Изменяется только CPT аварийного узла. Все остальные узлы не зависят от землетрясения или от землетрясений, выдаваемых по сигналу тревоги.

(i) Бур	П (Ал)
т	0,94
ф	0,0016

Возможны и другие решения из-за контрпродуктивного переменного порядка. Таким образом, мы учимся: Всегда используйте переменный порядок, учитывающий причинно-следственную связь!

Machine Translated by Google

Рис. 11.1 Байесовская сеть для велосипедного фонаря. CPT для Li указан в задаче заявление



Упражнение

7.12 (a), (б) См. рис.

11.1. (c) Ребро (Str, Li) отсутствует, так как Li и Str условно независимы при заданном V , т. е. P (Li |V , Str) = P (Li |V), так как состояние улицы не оказывает прямого влияния на освещенность. , а только динамо и напряжение, которое оно генерирует. (Условная независимость Li и Str при заданном V здесь не может быть показана расчетным путем на основе имеющихся данных для P (Li |V , Str) и P (Li |V), а может быть только утверждена.) (d) Для данного CPT:

$$P(R|Str) = P(R|Str,Flw)P(Flw) + P(R|Str,\neg Flw)P(\neg Flw)$$
$$= 0{,}95 \cdot 0{,}4 + 0{,}7 \cdot 0{,}6 = 0{,}8,$$
$$P(V|Str) = P(V|R)P(R|Str) + P(V|\neg R)P(\neg R|Str)$$
$$= 0{,}04 \cdot 0{,}8 + 0{,}99 \cdot 0{,}2 = 0{,}23.$$

Упражнение

8.1 (a) Агент — это функция A, которая отображает вектор (t1,t2,t3, d1, d2, d3, f1, f2, f3) , состоящий из трех значений температуры, давления и влажности, в класс значение k {солнечно, облачно, дождливо}. (b) Файл обучающих данных состоит из строки вида

ti1, ti2, ti3, di1, di2, di3, fi1, fi2, fi3, ki

для каждого элемента обучающих данных. Индекс я проходит по всем обучающим данным. Конкретный файл может, например, начинаться так:

17,17,17,980,983,986,63,61,50,солнечно
20,22,25,990,1014,1053,65,66,69, солнечно
20,22,18,990,979,929,52,60,61,дождь

Упражнение 8.2. Симметрию можно увидеть непосредственно, используя определение коэффициентов корреляции или ковариации. Замена i и j не меняет значения. Для

Machine Translated by Google

диагональные элементы вычисляем

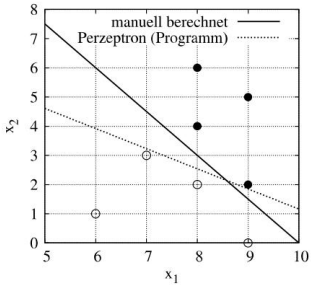
$$\text{Кии} = \frac{\sigma ii}{\text{си} \cdot \text{си}} \text{ "=" } \frac{\sum_{p=1}^{N} (xp_{\text{я}} - \bar{}xi)(xp_{\text{я}} - \bar{}xi)}{\sum_{(xp_{\text{я}} - \bar{}xi)}^{N} p=1} = 1.$$

Упражнение 8.3. Последовательность значений w такова

(1, 1), (2.2, 0.4), (1.8, 0.6), (1.4, 1.6), (2.6, 0.2), (2.2, 1.2)

Упражнение 8.4

(a)



(b) Проведение прямой линии на графике дает:

1,5x1 + x2 - 15 > 0.

После 442 итераций алгоритмы обучения персептрона с начальным вектором w = (1, 1, 1) возвращает:

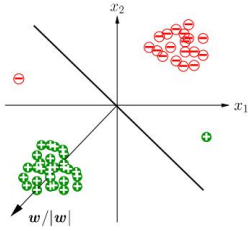
w = (11, 16, 129).

Это соответствует: 0,69x1 +x2 -8,06 > 0

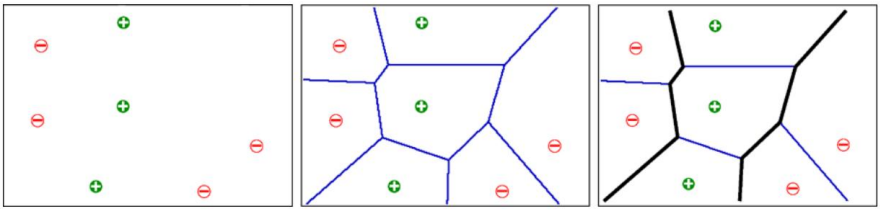
Упражнение 8.5

(a) Вектор x x указывает в «среднем» направлении всех положительных точек, а x M+ x M- указывает в «среднем» направлении всех отрицательных точек. Разность этих векторов указывает от отрицательных точек к положительным точкам. Тогда разделительная линия перпендикулярна этому разностному вектору.

(b) Облако положительных и отрицательных точек доминирует при вычислении w. Два выброса вряд ли играют здесь роль. Однако для определения разделительной линии они важны.



Упражнение 8.6



Machine Translated by Google

Упражнение

8.7 (а) Ближайшим соседом является (8, 4), таким образом, класс 1. (b) k = 2: класс не определен, поскольку один экземпляр класса 1 и один экземпляр класса 2. k = 3: решение 2:1 для класса 0. k = 5: решение 2:3 для класса 1.

Упражнение

8.8 (а) Чтобы иметь возможность делать общие утверждения, мы должны предположить, что точки равномерно распределены в пространстве признаков, что означает, что количество точек на площадь в целом примерно равно. Теперь вычислим площадь Ad узкого кольца шириной δd и расстоянием d вокруг точки x:

$$Ad = \pi(d + d)2 \quad \pi d2 = \pi(d2 + 2dd + d2) \quad \pi d2 \quad 2\pi dd.$$

Таким образом, общий вес всех точек в кольце ширины d и расстояния d пропорционален $dwi = d/(1 + \alpha d2) \quad 1/(\alpha d)$ при d . (b) Для этого взвешивания общий вес всех точек в кольце шириной d и = d/d = 1. Таким образом, каждое кольцо расстояние d будет пропорционально dw′ я ширины d будет иметь одинаковый вес, независимо от его расстояния до точки x. Это, конечно, не имеет смысла, потому что непосредственное окружение наиболее важно для приближения.

Упражнение 8.9 $\lim x \to 0 \; x \log 2 \, x = \lim x \to 0 \; \frac{\log 2 \, x}{1/x} = \lim x \to 0 \; \frac{1/(x \ln 2)}{-1/x2} = \lim x \to 0 \; \frac{x}{\ln 2}$ "=" где для второго уравнения использовалось правило Лопиталя.

Упражнение 8.10 (а) 0 (б) 1 (в) 1,5 (г) 1,875 (д) 2,32 (е) 2

Упражнение

8.11 (а) Из $\log 2 \, x = \ln x / \ln 2$ следует, что $c = 1/\ln 2 \quad 1{,}44$. (б) Поскольку обе функции энтропии отличаются только постоянным множителем, положение максимума энтропии не меняется. Экстремумы при ограничениях также сохраняют то же положение. Таким образом, фактор c не создает проблем для метода Max Ent. Для изучения деревьев решений сравниваются приросты информации различных атрибутов. Поскольку здесь имеет значение только порядок, а не абсолютное значение, фактор c здесь также не вызывает проблем.

Упражнение

8.12 (а) Для первого атрибута мы вычисляем

$$G(D, x1) = H(D) - \sum_{я=6}^{9} \frac{|Dx1=я|}{|Д|} H(Dx1=i)$$

$$= 1 - \frac{1}{8} H(Dx1=6) + \frac{1}{8} H(Dx1=7) + \frac{3}{8} H(Dx1=8) + \frac{3}{8} H(Dx1=9)$$

Machine Translated by Google

$$0 + \cdot 0{,}918 + \cdot 0{,}918 = 0{,}311$$

G(D, x2) = 0,75, поэтому выбирается x2 . Для x2 = 0, 1, 3, 4, 5, 6 решение очевидно. Для x2 = 2 выбирается x1 . Тогда дерево имеет вид

x2 = 0:0 (1/0) x2 = 1:0 (1/0)
x2 = 2:
| x1 = 6: 0 (0/0) | x1 = 7:0 (0/0) | x1 = 8:0 (1/0) | x1 = 9: 1 (1/0) x2 = 3: 0 (1/0) x2 = 4: 1 (1/0) x2 = 5: 1 (1/0) x2 = 6: 1 (1/0)

(b) Прирост информации для непрерывного атрибута x2 в качестве корневого узла:

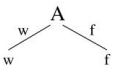
Порог Θ	012345
G(D, x2 Θ)	0,138 0,311 0,189 0,549 0,311 0,138

Поскольку G(D, x2 3) = 0,549 > 0,311 = G(D, x1), выбирается x2 3. При x2 3 классификация неоднозначна. Вычислим G(Dx2 3, x1) = 0,322, G(Dx2 3, x2 0) = 0,073, G(Dx2 3, x2 1) = 0,17, G(Dx2 3, x2 2) = 0,073. Таким образом, x1 выбран. При x1 = 9 классификация неоднозначна. Здесь очевидно, что решение G(D(x2 3,x1=9), x1) = 0, G(D(x2 3,x1=9), x2 1) = 1, x2 1 выбрано, и дерево снова имеет 100% правильность.

Дерево: x2
<= 3: | x1 = 6:0 (1/0) | x1 = 7:0 (0/0) | x1 = 8:0 (1/0) | x1 = 9: | | x2 <= 1: 0 (1/0) | | x2 > 1: 1 (1/0) x2 > 3: 1 (3/0)

Упражнение 8.13

(а) 100 % для обучающих данных, 75 % для тестовых данных, 80 % общей правильности. (б) (A ¬C) (¬A B) (в)



66,6 % правильности для обучающих данных
100 % правильности для тестовых данных
80 % общей правильности

Machine Translated by Google

Упражнение 8.14

(a) Уравнение (8.7) на стр. 189 для расчета прироста информации атрибута A читает

$$\text{Прирост информации (D, A)} = H(D) - \sum_{i=1}^{n} \frac{|Di|}{|D|} H(Di),$$

где n — количество значений текущего наблюдаемого атрибута. Если атрибут A проверяется где-то в поддереве как преемник значения aj , то только значение A = aj будет встречаться в наборе данных Dj и каждом из его подмножеств D′ . Таким образом, D′ = D′ и для всех k = j , $|D'_j|_k| = 0$, и мы получаем

$$\text{Прирост информации (D' , A)} = H(D') - \sum_{i=1}^{n} \frac{|D'_i|}{|D'|} H(D'_i) = H(D') - \frac{|D'|}{|D'|} H(D') = 0.$$

Таким образом, повторяющийся атрибут имеет нулевую информационную выгоду, из-за чего он больше не используется. (b) Из каждого непрерывного атрибута A генерируется бинарный атрибут $A>\Theta\ D,A$. Если ниже по дереву атрибут A снова дискретизируется с другим порогом $\Theta\ D',A$, то атрибут $A>\Theta\ D',A$ отличается от $A>\Theta\ D,A$. Если он имеет более высокий информационный прирост, чем все остальные атрибуты, он будет использоваться в дереве. Однако это также означает, что для нескольких вхождений непрерывного атрибута пороги должны быть разными.

Упражнение 8.15

(a) P (небо = ясное) = 0,65 P (столбик = поднимается) = 0,67

Небо	Бар	P (Prec = сухой | Sky, Bar)
Прозрачный	Повышение	0,85
Прозрачный	Падение	0,44
Облачно	Падение	0,45
Облачность	Падение	0,2

(b) P (небо = ясное) = 0,65

Бар	P (Prec = сухой |Bar)
Рост	0,73
Падение	0,33

Небо	P (Bar = рост | небо)
Прозрачный	0,72
Облачно	0,57

(c) Необходимые CPT для P (Prec |Sky,Bar) и P (Bar |Sky), а также P (Sky) уже известны.

(г) Па = (0,37, 0,065, 0,095, 0,12, 0,11, 0,13, 0,023, 0,092)
Pb = (0,34, 0,13, 0,06, 0,12, 0,15, 0,054, 0,05, 0,1)
P = (0,4, 0,07, 0,08, 0,1, 0,09, 0,11, 0,03, 0,12) (исходное распределение) Квадратичный расстояние: dq (Pa, P) = 0,0029, dq (Pb, P) = 0,014 расстояние Кульбака-Лейблера: dk (Pa, P) = 0,017, dk (Pb, P) = 0,09 Оба показателя расстояния показывают, что сеть (a) аппроксимирует распределение лучше, чем сеть (б). Это означает, что предположение, что Prec и Sky условно независимы при данном Bar , менее вероятно, чем предположение, что Sky и Bar независимы.

Machine Translated by Google

(д) Pc = (0,4, 0,07, 0,08, 0,1, 0,09, 0,11, 0,03, 0,12). Это распределение в точности равно исходному распределению P. Это неудивительно, поскольку в сети нет недостающих ребер. Это означает, что никакой независимости не было.

Упражнение 8.16 Мы можем сразу увидеть, что оценки и персептроны эквивалентны, сравнив их определения. Теперь об эквивалентности наивному байесовскому: сначала мы устанавливаем, что P (K|S1,...,Sn) > 1/2 эквивалентно P (K|S1,...,Sn) > P (¬K| S1 ,...,Sn) , так как P (¬K|S1,...,Sn) > 1 P (K|S1,...,Sn). На самом деле здесь мы имеем дело с бинарным наивным байесовским классификатором.

Применим логарифм к наивной формуле Байеса

$$P(K|S1,...,Sn) = P(S1,...,Sn) \frac{P(S1|K) \cdots P(Sn|K) \cdot P(K)}{},$$

и получить

$$\log P(K|S1,...,Sn) = \log P(S1|K) + \cdots + \log P(Sn|K) + \log P(K) \log P(S1,...,Sn). \quad (11.18)$$

Чтобы получить оценку, мы должны интерпретировать переменные S1,...,Sn как числовые переменные со значениями 1 и 0. Легко видеть, что

$$\log P(Si|K) = (\log P(Si = 1|K) \log P(Si = 0|K))Si + \log P(Si = 0|K).$$

Следует, что

$$\sum_{i=1}^{n} \log P(Si|K) = \sum_{i=1}^{n} (\log P(Si = 1|K) \log P(Si = 0|K))Si + \sum_{i=1}^{n} \log P(Si = 0|K).$$

Теперь определим $wi = \log P(Si = 1|K) \log P(Si = 0|K)$ и $c = \sum_{i=1}^{n} \log P(Si = 0|K)$ и упростить

$$\sum_{i=1}^{n} \log P(Si|K) = \sum_{i=1}^{n} WiSi + c.$$

Подставляя в (11.18), получаем

$$\log P(K|S1,...,Sn) = \sum_{i=1}^{n} wiSi + c + \log P(K) \log P(S1,...,Sn).$$

Machine Translated by Google

Для решения K должно быть так, согласно определению байесовского классификатора, что logP (K|S1,...,Sn) > log(1/2). Таким образом, должно быть так, что либо

$$\sum_{i=1}^{n} wiSi + c + \log P(K) \quad \log P(S1,...,Sn) > \log(1/2)$$

или это

$$\sum_{i=1}^{n} wiSi > \log 1/2 \quad c \quad \log P(K) + \log P(S1,...,Sn),$$

с помощью которого мы определили оценку с порогом $\Theta = \log 1/2 \quad c \quad \log P(K) + \log P(S1,...,Sn)$. Поскольку все преобразования можно отменить, мы также можем преобразовать любую оценку в байесовский классификатор. При этом была показана эквивалентность.

Упражнение 8.17 Логарифмирование (8.10) на стр. 206 приводит к

$$\log P(I|s1,...,sn) = \log c + \log P(I) + \sum_{i=1}^{l} ni \log P(wi|I).$$

Таким образом, очень маленькие отрицательные числа становятся умеренными отрицательными числами. Поскольку функция логарифма монотонно растет, для определения класса максимизируем по правилу

$$INaive\text{-}Bayes = \operatorname{argmax}_{I \in \{w,f\}} \log P(I|s1,...,sn).$$

Недостатком этого метода является несколько большее время вычислений на этапе обучения для больших текстов. При классификации время не увеличивается, так как значения logP (I | s1,...,sn) могут сохраняться при обучении.

Упражнение 8.19. Пусть f строго монотонно возрастает, то есть $\forall x,y \; x < y \quad f(x) < f(y)$. Если теперь d1(s, t) < d1(u, v), то, очевидно, d2(s, t) = f (d1(s, t)) < f (d1(u, v)) = d2(u, v). Поскольку обратная функция f также строго монотонна, верно и обратное, т. е. d2(s, t) < d2(u, v) \quad d1(s, t) < d1(u, v). Таким образом, показано, что d2(s, t) < d2(u, v) \quad d1(s, t) < d1(u, v).

Упражнение 8.20

$23 \cdot 26$ x1x2 = 4, и, следовательно, $ds(x1, x2) = \sqrt{6,1} = \sqrt{1\,4}$

x2x3 = 2, и, таким образом, $ds(x2, x3) = \sqrt{\frac{26 \cdot 20}{1,4}} = \sqrt{1\,4}$

$23 \cdot 20$ x1x3 = 1, поэтому $ds(x1, x3) = \sqrt{\frac{2}{1}} = \sqrt{21,4}$

Предложения 1 и 2 наиболее похожи по метрике расстояния ds .

Machine Translated by Google

Упражнение 8.21 Помощь при проблемах с KNIME: www.knime.org/forum

11.9 Нейронные сети

Упражнение 9.1. Мы хотим показать, что f (Θ + x) + f (Θ x) = 1.

$$f(\Theta + x) = 1 + e^{\frac{1}{\frac{\Theta x}{T}}} = \frac{T e^{\frac{\Theta x}{T}}}{1 + e^{\frac{\Theta x}{T}}}, \qquad f(\Theta - x) = 1 + e^{\frac{1}{\frac{\Theta x}{T}}},$$

$$f(\Theta + x) + f(\Theta x) = 1.$$

Упражнение 9.3 Каждый шаблон, сохраненный в сети, имеет размер n бит. Всего сеть имеет n(n 1)/2 веса. Если мы зарезервируем 16 битов на вес и определим двоичную память размером 16n(n 1)/2 как одинаково большую, то это, очевидно, может хранить N = 8n(n 1)/n = 4(n 1) шаблонов n битов. по размеру. Для больших n мы получаем N = 4n как предел.
Если мы возьмем частное α числа сохраняемых битов и количества доступных ячеек памяти, как в (9.11) на стр. 237, то мы получим значение 1 для списка памяти 2 и значение α = 0,146n /(16n(n 1)/2) 0,018 для сети Хопфилда. Таким образом, классическое хранилище имеет (для 16 бит на вес) емкость примерно в 55 раз выше.

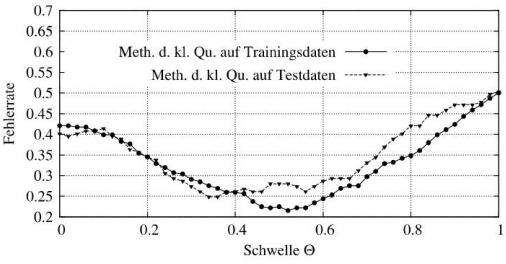
Упражнение 9.4

(а) Программа Mathematica для метода наименьших квадратов.

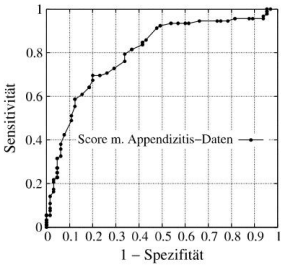
```
LeastSq[q_,a_] := Module[{Nq,Na,m,A,b,w}, Nq = длина[q]; m =
                  Длина[q[[1]]]; Na = длина [a]; If[Nq != Na, Print["Длина[q] != Длина[a]"]; Выход,
                  0]; A = Table[N[Sum[q[[p,i]] q[[p,j]], {p,1,Nq}]], {i,1,m}, {j,1,m} ]; b =
                  Таблица[N[Сумма[a[[p]]q[[p,j]], {p,1,Nq}]], {j,1,m}]; w = линейное решение[A,b]

]
LeastSq::usage = "LeastSq[x,y,f] вычисляет из векторов запросов q[[1]],..., q[[m]] таблицу коэффициентов w[[i]] для
      линейного отображения f[x] = Sum[w[[i]] x[[i]], {i,1,m}] с f[q[[p]]] = a[[p]]."
```

(б)



(с)



Упражнение 9.6 (а) Обучение работает без ошибок. (б) Обучение не работает без ошибок!

Machine Translated by Google

Упражнение

9.7 (a) Отображение f называется линейным, если для всех x, y, k выполняется равенство f (x + y) = f (x) + f (y) и f (kx) = kf (x) . Пусть теперь f и g — линейные отображения. Тогда f (g(x + y)) = f (g(x) + g(y)) = f (g(x)) + f (g(y)) и f (g(kx)) = f (кг(x)) = kf(g(x)). Таким образом, последовательное выполнение линейных отображений является линейным отображением.

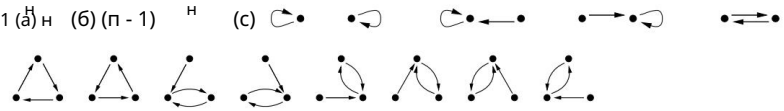
(b) Мы наблюдаем два произвольных выходных нейрона j и k. Каждый из двух представляет класс. Классификация осуществляется путем формирования сумма максимума из двух активаций. i wj ixi и netk = i wkixi — взвешенная значений ar. Пусть netj = расхождение в нейронах j и k. Кроме того, пусть netj > netk . Без функции активации выводится класс j. Теперь, если применяется строго монотонная функция активации f, в результате ничего не изменится, потому что из-за строго монотонной функции f (netj)>f(netk).

Упражнение 9.8 Преобразованное $\frac{2}{1}$, f2(x1, x2) = x $\frac{2}{2}$. Затем разделительная линия в пространство f1(x1, x2) = x имеет уравнение y1 + y2 = 1.

11.10 Обучение с подкреплением

Упражнение

10.1 (а) ^{н} н (б) (п - 1) ^{н} (c)



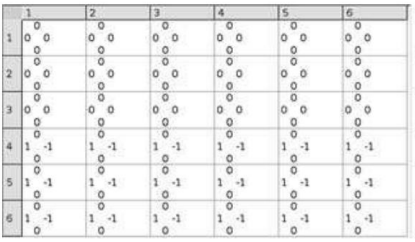
(г)

Упражнение 10.2. Повторение значения с построчным обновлением значений V^ от верхнего левого угла к нижнему правому дает

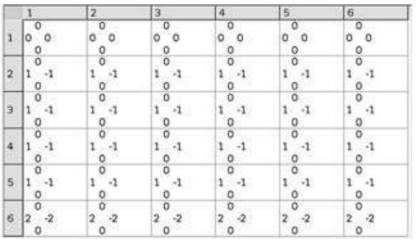
0	0	0,81	0,9	1,35	1,49	...	2,36 2,62
0	0	0,73	1	1,21	1,66		2,12 2,91

Упражнение 10.3

(c)



6×6 feedback for a completely smooth surface



6×6 feedback for a very heavy survace (like thick snow)

Machine Translated by Google

(d) Мы видим, что чем длиннее становится политика (т. е. чем больше шагов, например, имеет цикл циклической стратегии), тем ближе значение γ должно быть к 1, потому что большее значение γ увеличивает память. возможный. Однако итерация значений сходится намного медленнее.

Упражнение 10.4 Значение V выглядит (3, 3) справа внизу в матрице состояния изменяется как следующим образом:

$$B \quad (3,1) = 0{,}9\ B \qquad (2,1) = 0{,}9\ 2\ B \qquad (2,2) = 0{,}9\ 3\ B \qquad (2,3) = 0{,}9\ 4\ B \qquad (3,3). \tag{11.19}$$

Эта цепочка уравнений следует из (10.6), поскольку для всех заданных переходов состояний максимальное немедленное вознаграждение равно r(s, a) = 0, и это тот случай, когда

$$B \quad (s) = \max_a [r(s,a) + \gamma\ V \qquad (\delta(s,a))] = \gamma\ V \qquad (\delta(c,a)) = 0{,}9\ B \qquad (\delta(s,a)).$$

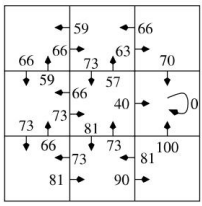
Из (10.6) также следует, что V (3, 2) = 1 + 0,9 B (3, 1), так как r(s, a) = 1 максимально. Точно так же верно, что B. (3, 3) = 1 + 0,9 B (3, 2) и круг закрывается. Два последних уравнения вместе (3, 3) = 1 + 0,9 (1 + 0,9 B (3, 1)). Из (11.19) следует, что V дает дают V (3, 1) = 0,9 4V (3, 3). Заменено в B (3, 3), это

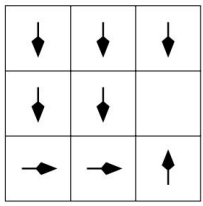
$$B \quad (3,3) = 1 + 0{,}9(1 + 0{,}9\ 5B \qquad (3,3)),$$

из чего следует утверждение.

Упражнение 10.5

Стабильные значения Q и оптимальная политика:





Упражнение

10.6 (a) Количество состояний. Количество действий на состояние = ℓ^{H} 1. Количество полигонов = ℓ cies = $(\ell n)^{Hn} = л^{п\,л}$.

(б)

	л = 1	л = 2	л = 3	л = 4	л = 10
п = 1	1	4	27	256	1010
п = 2 1		256	3,9 × 108	1,8 × 1019	10200
n = 31 1,7 × 107 n = 41 1,8 ×			4,4 × 1038	3,9 × 10115	103000
1019 3,9 × 10154 n = 81 3,2 × 10616 1,4 ×				3,2 × 10616	1040000
1025043 6,7 × 10315652					10800000000

Machine Translated by Google

(c) В каждом состоянии теперь есть 2n возможных действий. Таким образом, имеется (2n)ℓ ^{н} политики.

	л = 1	л = 2	л = 3	л = 4	л = 10
п = 1	2	4	8	16	1024
n = 2 4 256 n = 36 1,7 ×			2,6 × 109 4,3 × 109 1,0 ×		1,6 × 1060
106 n = 48 2,8 × 1014 1,8 ×			1021 6,3 × 1049 1,4 × 1073		1,4 × 10778
10308			1,6 × 10231 1,7 × 107900 1,6		7,9 × 109030
п = 8 16			× 1078913		1,8 × 10120411998

(d) 10120411998 различных политик никогда не могут быть исследованы комбинаторно, даже если все имеющихся компьютеров в мире должны были работать на них параллельно. Таким образом, необходимы «интеллектуальные» алгоритмы для нахождения оптимального или близкого к оптимальному политика.

Machine Translated by Google

Machine Translated by Google

Рекомендации

[Ада75] Э. В. Адамс. Логика условностей. Synthese Library, том 86. Рейдель, Дордрехт, 1975 год.

[Альп04] Э. Алпайдин. Введение в машинное обучение. Массачусетский технологический институт, Кембридж, 2004.

[90 апреля] Дж. Андерсон, А. Пеллионис и Э. Розенфельд. Нейрокомпьютинг (том 2): Направления для исследований. Массачусетский технологический институт, Кембридж, 1990.

[AR88] Дж. Андерсон и Э. Розенфельд. Нейрокомпьютинг: основы исследований. Массачусетский технологический институт Press, Cambridge, 1988. Сборник фундаментальных оригинальных статей.

[Бар98] Р. Бартак. Онлайн-руководство по программированию в ограничениях, 1998 г. http://kti.ms.mff.cuni.cz/~бартак/ограничения.

[BCDS08] А. Биллард, С. Калинон, Р. Диллманн и С. Шаал. Демонстрационное программирование роботов. В Б. Сицилиано и О. Хатиб, редакторы, Справочник по робототехнике, стр. 1371–1394. Спрингер, Берлин, 2008 г.

[Бел57] РЭ Беллман. Динамическое программирование. Издательство Принстонского университета, Принстон, 1957.

[Бер89] М. Беррондо. Fallgruben fü r Kopffü ssler. Фишер Ташенбух, Nr. 8703, 1989.

[BFOS84] Л. Брейман, Дж. Фридман, Р. А. Олшен и С. Дж. Стоун. Классификация и деревья регрессии. Уодсворт, Белмонт, 1984 г.

[Би682] В. Бибель. Автоматическое доказательство теорем. Виег, Висбаден, 1982 г.

[Bis05] КМ Бишоп. Нейронные сети для распознавания образов. Издательство Оксфордского университета, Лондон, 2005.

[Bis06] КМ Бишоп. Распознавание образов и машинное обучение. Спрингер, Нью-Йорк, 2006.

[BM03] А. Г. Барто и С. Махадеван. Последние достижения в иерархическом обучении с подкреплением. Discrete Event Syst., Специальный выпуск по обучению с подкреплением, 13:41–77, 2003.

[Бюстгальтер84] В. Брайтенберг. Транспортные средства - эксперименты в синтетической психологии. MIT Press, Кэмбридж, 1984.

[Бюстгальтер01] Б. Брабек. Computergestü tzte Regionale Lawinenprognose. Кандидатская диссертация, ETH Zü rich, 2001.

[Бюстгальтер11] И. Братко. Программирование PROLOG для искусственного интеллекта. Эддисон-Уэсли, Рединг, 4-е издание, 2011 г.

[Бри91] Британская энциклопедия. Британская энциклопедия, Лондон, 1991.

[Бур98] Си Джей Берджес. Учебник по машинам опорных векторов для распознавания образов. Данные Мин. Знай. Дисков., 2(2):121–167, 1998.

[CAD] CADE: Конференция по автоматизированной дедукции. www.cadeconference.org.

[Че83] П. Чизмен. Метод вычисления обобщенных байесовских значений вероятности для экспертные системы. В проц. 8-го междунар. Совместная конф. по искусственному интеллекту (IJCAI 83), 1983.

[Че85] П. Чизмен. В защиту вероятности. В проц. 9-го междунар. Совместная конф. на Искусственный интеллект (IJCAI-85), 1985 г.

[CL73] CL Чанг и RC Ли. Символическая логика и механическое доказательство теорем. Академическое издательство, Орландо, 1973.

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по информатике,
DOI 10.1007/978-0-85729-299-5, © Springer-Verlag London Limited 2011

Machine Translated by Google

[Кле79] WS Кливленд. Надежная локально взвешенная регрессия и сглаживающие диаграммы рассеяния. Варенье. Стат. Assoc., 74(368):829–836, 1979.

[CLR90] Т. Кормен, гл. Лейзерсон и Р. Ривест. Введение в алгоритмы. Массачусетский технологический институт Пресс, Кембридж, 1990.

[CM94] В. Ф. Клоксин и К. С. Меллиш. Программирование на Прологе. Шпрингер, Берлин, 4-я улица издание 1994 года.

[Coz98] ФГ Козман. Javabayes, байесовские сети в Java, 1998. www.cs.cmu.edu/~ javabayes.

[dD91] FT де Домбаль. Диагностика острой боли в животе. Черчилль Ливингстон, Лондон, 1991 г.

[dDLS+72] FT de Dombal, DJ Leaper, JR Staniland, AP McCann и JC Horrocks. Компьютерная диагностика острой боли в животе. бр. Мед. Дж., 2:9–13, 1972.

[Ди11] Проект DeepQA, 2011 г. http://www.research.ibm.com/deepqa/deepqa.shtml.

[DHS01] Р.О. Дуда, П.Е. Харт и Д.Г. Сторк. Классификация узоров. Уайли, Нью-Йорк, 2001.

[Dia04] Д. Диаз. ПРОЛОГ ГНУ. Universität Paris, 2004. Aufl. 1.7, для версии GNU Prolog 1.2.18, http://gnu-prolog.inria.fr.

[DNM98] CL Blake, DJ Newman, S. Hettich и CJ Merz. Репозиторий UCI машины учебные базы данных, 1998 г. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[Эде91] Э. Эдер. Относительная сложность исчисления первого порядка. Виег, Висбаден, 1991.

[Лось93] К. Элькан. Парадоксальный успех нечеткой логики. В материалах Одиннадцатой национальной конференции по искусственному интеллекту (AAAI-93), стр. 698–703. Массачусетский технологический институт Пресс, Кембридж, 1993.

[Эрт93] В. Эртель. Parallelesuche mit randomisiertem Wettbewerb inInferenzsystemen. DISKI, Band 25. Infix, St. Augustin, 1993. Диссертация, Технический университет. Мюнхен.

[Эрт11] В. Эртель. Искусственный интеллект, 2011. www.hs-weingarten.de/~ertel/aibook. Домашняя страница этой книги с материалами, демонстрационными программами, ссылками, литературой, исправлениями и т. д.

[ES99] В. Эртель и М. Шрамм. Объединение данных и знаний с помощью MaxEnt-оптимизации вероятностных распределений. В ПКДД'99 (3-я Европейская конференция по принципам и Практика обнаружения знаний в базах данных). LNCS, том 1704, стр. 323–328. Спрингер, Прага, 1999.

[ESCT09] В. Эртель, М. Шнайдер, Р. Кубек и М. Токич. Учебный ящик: универсальный система обучения роботов. В материалах 14-й Международной конференции по передовой робототехнике (ICAR 2009), 2009 г. www.servicerobotik.hs-weingarten.de/учебная коробка.

[ESS89] В. Эртель, Дж. Шуман и Ч. Саттнер. Изучение эвристики для средства доказательства теорем используя обратное распространение. В J. Retti и K. Leidlmair, редакторы, 5. Österreichische Искусственный интеллект-Tagung. Informatik-Fachberichte, том 208, страницы 87–95. Спрингер, Берлин, 1989.

[Подходит96] М. Фитинг. Логика первого порядка и автоматическое доказательство теорем. Шпрингер, Берлин, 1996.

[FNA+09] Д. Ферруччи, Э. Найберг, Дж. Аллан, К. Баркер, Э. Браун, Дж. Чу-Кэрролл, А. Чикколо, P. Duboue, J. Fan, D. Gondek et al. К открытому выдвижению вопроса системы ответов. Технический отчет IBM RC24789, Йорктаун-Хайтс, Нью-Йорк, 2009 г. http://www.research.ibm.com/deepqa/question_answering.shtml.

[FPP07] Д. Фридман, Р. Пизани и Р. Пурвс. Статистика. Нортон, Нью-Йорк, 4-е издание, 2007.

[Фра05] К. Фрейн. Теория компьютерного шахматного программирования, 2005. www.frayn.net/beowulf/теория.html.

[Свободный97] Э. Фрейдер. В погоне за Святым Граалем. Ограничения, 2(1):57–61, 1997.

[ФС97] Б. Фишер и Дж. Шуман. Setheo идет на разработку программного обеспечения: применение atp к повторному использованию программного обеспечения. На конференции по автоматизированной дедукции (CADE-14). ЛНКС, том 1249, страницы 65–68. Springer, Берлин, 1997 г. http://ase.arc.nasa.gov/people/schumann/publications/papers/cade97-reuse.html.

Machine Translated by Google

[GW08] Р. С. Гонсалес и Р. Е. Вудс. Цифровая обработка изображений. Пирсон/Прентис Холл, Река Аппер-Сэдл / Принстон, 2008 г.

[Göd31a] К. Гёдель. Diskussion zur Grundlegung der Mathematik, Erkenntnis 2. Monatshefte Мат. Phys., 32(1):147–148, 1931.

[Göd31b] К. Гёдель. Überformal unentscheidbare Sätze der Principia Mathematica und ver wandter Systeme I. Monatshefte Math. Phys., 38(1):173–198, 1931.

[HKP91] Дж. Герц, А. Крог и Р. Палмер. Введение в теорию нейронных вычислений. Аддисон-Уэсли, чтение, 1991.

[Hon94] Б. Хончик. Теория и практика аппендэктомии. Мабузе, Франкфурт-на-Майне, 1994.

[Хоп82] Джей Джей Хопфилд. Нейронные сети и физические системы с возникающими коллективными вычислительными способностями. проц. Натл. акад. науч. США, 79:2554–2558, 1982. Перепечатка в [AR88], страницы 460–464.

[HT85] Джей Джей Хопфилд и Д. У. Танк. «Нейронный» расчет решений в оптимизации проблемы. биол. Cybern., 52(3):141–152, 1985. Спрингер.

[HTF09] Т. Хасти, Р. Тибширани и Дж. Фридман. Элементы статистического обучения: данные Добыча полезных ископаемых, вывод и прогнозирование. Springer, Berlin, 3-е издание, 2009 г. Электронная версия: http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

[Джей57] ЭТ Джейнс. Теория информации и статистическая механика. физ. Обр., 1957.

[Джей03] ЭТ Джейнс. Теория вероятностей: логика науки. Издательство Кембриджского университета, Кембридж, 2003 г.

[Джен01] ФВ Дженсен. Байесовские сети и графы решений. Спрингер, Берлин, 2001.

[Jor99] М. И. Джордан, редактор. Обучение графическим моделям. Массачусетский технологический институт, Кембридж, 1999.

[Кал01] Я Кальман. Автоматизированное мышление с помощью OTTER. Ринтон, Парамус, 2001. www-unix.mcs.anl.gov/AR/otter/index.html.

[Кан89] Т. Кейн. Максимальная энтропия в вероятностной логике Нильссона. В проц. 11-го Международный Совместная конф. по искусственному интеллекту (IJCAI-89), 1989 г.

[KK92] Дж. Н. Капур и Х. К. Кесаван. Принципы оптимизации энтропии с приложениями. Академическая пресса, Сан-Диего, 1992.

[KLM96] Л.П. Кельблинг, М.Л. Литтман и А.П. Мур. Обучение с подкреплением: опрос. Дж. Артиф. Интел. Рез., 4:237–285, 1996. www-2.cs.cmu.edu/afs/cs/project/jair/pub/том4/kaelbling96a.pdf.

[KMK97] Х. Кимура, К. Миядзаки и С. Кобаяши. Обучение с подкреплением в POMDP с приближением функции. На 14-й Международной конференции по машинному обучению, стр. 152–160. Морган Кауфманн, Сан-Матео, 1997 г. http://sysplan.nams.kyushu-u.ac.jp/gen/papers/JavaDemoML97/robodemo.html.

[Ко72] Т. Кохонен. Корреляционная матрица памяти. IEEE транс. Вычисл., С-21(4):353–359, 1972 г. Перепечатка в [AR88], стр. 171–174.

[Лар00] ФД Лараме. Шахматное программирование, части 1–6, 2000 г. www.gamedev.net/reference/программирование/возможности/шахматы1.

[Le999] LEXMED — обучающая экспертная система для медицинской диагностики, 1999 г. www.lexmed.de.

[Lov78] Д. У. Лавленд. Автоматическое доказательство теорем: логическая основа. Северная Голландия, Амстердам, 1978 г.

[LSBB92] Р. Летц, Дж. Шуман, С. Байерл и В. Бибель. SETHEO: средство доказательства теорем высокой производительности. Дж. Автомат. Reason., 8(2):183–212, 1992. www4.informatik.tu-muenchen.de/~letz/seteo.

[МакК] В. МакКьюн. Автоматизированные дедуктивные системы и группы. www-unix.mcs.anl.gov/AP/другие.html. См. также www-formal.stanford.edu/clt/ARS/systems.html.

[МакД82] Дж. Макдермотт. R1: конфигуратор компьютерных систем на основе правил. Артиф. Интел., 19:39–88, 1982.

[MDBM00] Г. Мелансон, И. Дютур и Г. Буск-Мелу. Случайная генерация дагов для графическое рисование. Технический отчет INS-R0005, Голландский исследовательский центр математических и компьютерных наук (CWI), 2000 г. http://ftp.cwi.nl/CWIreports/INS/INS-R0005.pdf.

Machine Translated by Google

[Мит97] Т. Митчелл. Машинное обучение. McGraw-Hill, Нью-Йорк, 1997. www-2.cs.cmu.edu/~tom/mlbook.html.

[MP69] М. Мински и С. Пейперт. Персептроны. MIT Press, Кембридж, 1969.

[Новый00] М. Новорожденный. Автоматизированное доказательство теорем: теория и практика. Шпрингер, Берлин, 2000.

[Нет86] Н. Дж. Нильссон. Вероятностная логика. Артиф. Интел., 28(1):71–87, 1986.

[Ноль98] Н. Нильссон. Искусственный интеллект — новый синтез. Морган Кауфманн, Сан-Матео, 1998.

[NPW02] Т. Нипков, Л. С. Полсон и М. Венцель. Изабель/ХОЛ — Помощник по проверке Логика высшего порядка. LNCS, том 2283. Springer, Берлин, 2002. www.cl.cam.ac.uk/Research/HVG/Изабель.

[NS61] А. Ньюэлл и Х.А. Саймон. Gps, программа, имитирующая человеческое мышление. В Х. Биллинг, редактор Lernende Automaten, страницы 109–124. Ольденбург, Мюнхен, 1961.

[NSS83] А. Ньюэлл, Дж. К. Шоу и Х. А. Саймон. Эмпирические исследования с логикой теория машины: пример эвристики. В J. Siekmann и G. Wrightson, редакторы, Автоматизация рассуждений 1: классические статьи по вычислительной логике 1957–1966, страницы 49–73. Springer, Берлин, 1983. Erstpublikation: 1957.

[OFY+95] К. Оманн, К. Франке, К. Ян, М. Маргулис, М. Чан, ван П.Дж. Элк, Ф.Т. Домбал и Х.Д. Рёрер. Диагностика острого аппендицита. Чирург, 66:135–141, 1995.

[OMYL96] К. Оманн, В. Мустакис, К. Ян и К. Ланг. Оценка автоматических методов приобретения знаний в диагностике острой боли в животе. Артиф. Интел. Мед., 8:23–36, 1996.

[OPB94] К. Оманн, К. Платен и Г. Беленький. Computerunterstü tze Diagnose bei akuten Баухшмерцен. Чирург, 63:113–123, 1994.

[Приятель80] Г. Палм. Об ассоциативной памяти. биол. Cybern., 36:19–31, 1980.

[Приятель91] Г. Палм. Возможности памяти локальных правил синаптической модификации. Concepts Neu rosci., 2(1):97–128, 1991. MPI Tü bingen.

[Горох84] Дж. Перл. Эвристика, интеллектуальные стратегии поиска для решения компьютерных проблем. Аддисон-Уэсли, Рединг, 1984.

[Горох88] Дж. Перл. Вероятностное мышление в интеллектуальных системах. Сети правдоподобного вывода. Морган Кауфманн, Сан-Матео, 1988 год.

[PL05] Л. Панайт и С. Люк. Совместное мультиагентное обучение: современное состояние. Автон. Агенты Multi-Agent Syst., 11(3):387–434, 2005.

[PS08] Дж. Питерс и С. Шаал. Усиление обучения двигательным навыкам с помощью градиентов политики. Нейронная сеть, 21(4):682–697, 2008 г. http://www-clmc.usc.edu/publications/P/peters-NN2008.pdf.

[PS09] Г. Полиа и С. Слоан. Как решить: новый аспект математического метода. Иши, Маунтин-Вью, 2009 г.

[Qui93] Дж. Росс Куинлан. C4.5: Программы для машинного обучения. Морган Кауфманн, Сан Матео, 1993. Загрузка C4.5: http://www.rulequest.com/Personal, загрузка C5.0: http://www.rulequest.com/download.html.

[RB93] М. Ридмиллер и Х. Браун. Прямой адаптивный метод для более быстрого обратного распространения обучение: алгоритм rprop. В материалах международной конференции IEEE по нейронным сетям, страницы 586–591, 1993.

[RGH+06] М. Ридмиллер, Т. Габель, Р. Хафнер, С. Ланге и М. Лауэр. Мозговые штурмовики: Entwurfsprinzipien lernfähiger autonomer Roboter. Информ.-Спектр, 29(3):175–190, 2006.

[RHR86] Д. Е. Румельхарт, Г. Э. Хинтон и Р. Дж. Уильямс. Изучение внутренних представлений путем распространения ошибок. В [RM86].

[Ric83] Э. Рич. Искусственный интеллект. Макгроу-Хилл, Нью-Йорк, 1983 г.

[86 ринггитов] Д. Румельхарт и Дж. Макклелланд. Параллельная распределенная обработка, том 1. MIT Пресса, Кембридж, 1986.

Machine Translated by Google

[96 рингитов] В. Рёддер и К.-Х. Мейер. Согласованная обработка знаний с максимальной энтропией ДУХОМ. В KI-96 (Германская национальная конференция по ИИ), Дрезден, 1996 г.

[RMD07] М. Ридмиллер, М. Монтемерло и Х. Далкамп. Учимся водить настоящий автомобиль в 20 минут. В FBIT '07: Proceedings of the 2007 Frontiers in the Convergence of Биологические науки и информационные технологии, страницы 645–650. Компьютерное общество IEEE, Вашингтон, округ Колумбия, 2007 г.

[RMS92] Х. Риттер, Т. Мартинес и К. Шульten. Нейронные вычисления и самоорганизация Карты. Аддисон-Уэсли, Рединг, 1992.

[RN10] С. Рассел и П. Норвиг. Искусственный интеллект: современный подход. Прентис Холл, Нью-Йорк, 3-е издание, 2010 г. 1-е издание: 1995 г. http://aima.cs.berkeley.edu.

[Роба] Официальный сайт Робокап. www.robocup.org.

[Робб] Футбольный симулятор robocup. http://sserver.sourceforge.net.

[Роб65] Дж. А. Робинсон. Машинно-ориентированная логика, основанная на принципе разрешения. Дж. АКМ, 12(1):23–41, 1965.

[Рой96] Р. Рохас. Нейронные сети: систематическое введение. Спрингер, Берлин, 1996.

[Рос58] Ф. Розенблатт. Персептрон: вероятностная модель хранения информации и организация в мозгу. Психол. Rev., 65:386–408, 1958. Перепечатка в [AR88], стр. 92–114.

[Рос09] СМ Росс. Введение в вероятность и статистику для инженеров и ученых. Академическая пресса, Сан-Диего, 2009 г.

[RW06] CE Rasmussen и CKI Williams. Гауссовские процессы для машинного обучения. MIT Press, Кембридж, 2006 г. Интернет-версия: http://www.gaussianprocess.org/ gpml/главы/.

[СА94] С. Шаал и К. Г. Аткесон. Жонглирование роботами: реализация на основе памяти обучение. Система управления IEEE. Маг., 14(1):57–71, 1994.

[Сэм59] ЭЛ Самуэль. Некоторые исследования машинного обучения с использованием игры в шашки. IBM Дж., 1(3):210–229, 1959.

[SB98] Р. Саттон и А. Барто. Обучение с подкреплением. Массачусетский технологический институт, Кембридж, 1998. www.cs.ualberta.ca/~sutton/book/the-book.html.

[SB04] Дж. Зикманн и Ч. Бензмюллер. Омега: математика, поддерживаемая компьютером. В КИ 2004: Достижения в области искусственного интеллекта. ЛНАИ, том 3238, страницы 3–28. Спрингер, Берлин, 2004 г. www.ags.uni-sb.de/~omega.

[Ш96] М. Шрамм. Indifferenz, Unabhängigkeit и максимальная энтропия: Eine wahrscheinlichkeitstheoretische Semantik fü r Nicht-Monotones Schließen. Dissertationen zur Informatik, Band 4. CS, Мюнхен, 1996.

[Sch01] Дж. Шуман. Автоматизированное доказательство теорем в программной инженерии. Спрингер, Берлин, 2001.

[Sch02] С. Шульц. E — умный доказатель теорем. J. AI Commun., 15 (2/3): 111–126, 2002. www4.informatik.tu-muenchen.de/~schulz/WORK/eprover.html.

[Sch04] А. Шварц. СпамУбийца. O'Рейли, Кембридж, 2004 г. Домашняя страница SpamAssassin: http://spamassassin.apache.org.

[SE90] Ч. Зутнер и В. Эртель. Автоматическое получение эвристики, направляющей поиск. В 10-м Междунар. конф. по автоматизированной дедукции. ЛНАИ, том 449, страницы 470–484. Спрингер, Берлин, 1990 год.

[SE00] М. Шрамм и В. Эртель. Рассуждения с вероятностями и максимальной энтропией: система PIT и ее применение в LEXMED. К. Индерфурт и др., редактор, Operations Research Proceedings (SOR'99), стр. 274–280. Спрингер, Берлин, 2000.

[SE10] М. Шнайдер и В. Эртель. Робот учится на демонстрации с местным гауссовцем регресс процесса. В материалах Международной конференции IEEE/RSJ по Интеллектуальные роботы и системы (IROS'10), 2010.

[SET09] Т. Сегаран, К. Эванс и Дж. Тейлор. Программирование семантической сети. O'Рейли, Кэмбридж, 2009 г.

[Шо76] Э. Х. Шортлифф. Компьютерные медицинские консультации, MYCIN. Северная Голландия, Нью-Йорк, 1976 год.

Machine Translated by Google

[Spe97] Тайны разума. Специальный выпуск. Научный американец Inc., 1997.

[Spe98] Изучение интеллекта. Scientific American Presents, том 9. Scientific American Инк., 1998.

[SR86] TJ Sejnowski и CR Розенберг. NETtalk: параллельная сеть, которая учится читать вслух. Технический отчет JHU/EECS-86/01. Технический отчет Университета Джона Хопкинса по электротехнике и компьютерным наукам, 1986 г. Перепечатка в [AR88], страницы 661–672.

[SS02] С. Шёлькопф и А. Смола. Обучение с помощью ядер: машины опорных векторов, регуляризация, оптимизация и не только. Массачусетский технологический институт, Кембридж, 2002.

[SS06] Г. Сатклифф и К. Саттнер. Состояние ЦАСК. AI Commun., 19(1):35–48, 2006. Домашняя страница CASC: www.cs.miami.edu/~tptp/CASC.

[CCK05] П. Стоун, Р. Саттон и Г. Кульманн. Обучение с подкреплением для робокапа по футболу. Адаптивное поведение, 2005. www.cs.utexas.edu/~pstone/Papers/bib2html-ссылки/AB05.pdf.

[Ste07] Дж. Стюарт. Многомерное исчисление. Брукс Коул, Флоренция, 2007 г.

[SW76] К. Э. Шеннон и У. Уивер. Математические основы теории информации. Ольденбург, Мюнхен, 1976 год.

[Размер10] К. Сепешвари. Алгоритмы обучения с подкреплением. Морган и Клейпул, Сан Рафаэль, 2010. Черновик доступен в Интернете: http://www.ualberta.ca/~szepesva/RLBook.HTML.

[Тед08] Р. Тедрейк. Управление обучением при промежуточных числах Рейнольдса. В мастерской по: Robotics Challenges for Machine Learning II, Международная конференция по интеллектуальным роботам и системам (IROS 2008), Ницца, Франция, 2008 г.

[TEF09] М. Токич, В. Эртель и Дж. Фесслер. Краулер, демонстратор в классе для обучения с подкреплением. В материалах 22-й Международной Флоридской искусственной Конференция Общества разведки (FLAIRS 09). AAAI Press, Менло-Парк, 2009.

[Тес95] Г. Тезауро. Обучение временной разнице и td-gammon. коммун. АКМ, 38(3), 1995. www.research.ibm.com/massive/tdl.html.

[Ток06] М. Токич. Entwicklung eines Lernfähigen Laufroboters. Diplomarbeit Hochschule Ravensburg-Weingarten, 2006. Inklusive Simulationssoftware verfü gbar auf www.hs-weingarten.de/~ertel/kibuch.

[Тур37] А. М. Тьюринг. О вычислимых числах, с приложением к проблеме навоза Entschei. проц. Лонд. Мат. Соц., 42(2), 1937.

[Тур50] А. М. Тьюринг. Вычислительная техника и интеллект. Разум, 59: 433–460, 1950.

[vA06] Л. в. Ан. Игры с целью. IEEE вычисл. Mag., 96–98, июнь 2006 г. http://www.cs.cmu.edu/~biglou/ieee-gwap.pdf.

[Вэй66] Дж. Вейценбаум. ELIZA — компьютерная программа для изучения общения человека и машины на естественном языке. коммун. АКМ, 9(1):36–45, 1966.

[WF01] И. Виттен и Э. Франк. Сбор данных. Hanser, Мюнхен, 2001 г. Библиотека DataMining Java Library WEKA: www.cs.waikato.ac.nz/~ml/weka.

[Whi96] Дж. Уиттакер. Графические модели в прикладной многомерной статистике. Уайли, Нью-Йорк, 1996.

[Ви] У. Видеманн. PhilLex, Лексикон философии. www.phillex.de/paradoxa.htm.

[Wie04] Дж. Вилмейкер. SWI-ПРОЛОГ 5.4. Университет Амстердама, 2004. www.swi-prolog.org.

[Wik10] Википедия, бесплатная энциклопедия, 2010 г. http://en.wikipedia.org.

[Победить] П. Уинстон. Демонстрация игры. http://www.ai.mit.edu/courses/6.034f/gamepair.HTML. Java-апплет для минимаксного и альфа-бета-поиска.

[Зел94] А. Целль. Моделирование Neuronaler Netze. Аддисон-Уэсли, Рединг, 1994. Описание. SNNS и JNNS: www-ra.informatik.uni-tuebingen.de/SNNS.

[ZSR+99] А. Зильке, Х. Ситтер, Т. А. Рампп, Э. Шефер, К. Хассе, В. Лоренц и М. Ротмунд. Überprü fung eines diagnostischen Scoresystems (Ohmann-Score) для смерти острый аппендицит. Чирург, 70:777–783, 1999.

Machine Translated by Google

Индекс

A

Априорная вероятность, 122

A -алгоритм, 98, 250

Действие, 86, 87, 260

Функция активации, 224, 234

Актуатор, 9

Теория адаптивного резонанса, 255

Допустимо, 98

Агент, 3, 9, 257, 259–261, 265, 267, 268, 270, 272, 273

- автономных, 8
- стоимостных, 12
- стоимостных, 140
- распределенных,
- 8 целевых, 9
- аппаратных,
- 9
- интеллектуальных, 9
- обучение,
- 8, 12, 164
- рефлекторных, 9
- программных, 9 утилитарных, 12 с памятью, 9

Агенты, распределенные, 12

ИИ, 1

Аварийный пример, 145

Альфа-бета обрезка, 103

И ветки, 70

И/или дерево, 70

Аппендицит, 121, 131

Приближение, 164, 180

APT, 255

Искусственный интеллект, 1

Ассоциативная память, 232

Атрибут, 106, 185

Автоассоциативная память, 226, 233, 238

Б

Нарды, 108, 273

Обратное распространение, 246, 265, 275

- правило обучения, 247

Возвращение, 69, 91

Обратная цепочка, 27

Пакетное обучение, 201, 243

Формула Байеса, см. теорему Байеса,
122, 155.

формула Байеса, см. теорему Байеса

Байесовская сеть, 7, 10, 64, 115, 144, 146, 184

- обучение, 157, 199

BDD, см. диаграмму бинарных решений

Уравнение

- Беллмана, 263
- принцип, 263

Узел смещения, 173

Бинарная диаграмма решений, 29

Машина Больцмана, 232

Наука о мозге, 3

Автомобиль Брайтенберга, 1, 8, 179

Коэффициент ветвления, 83,
87 средний,
84 эффективный, 87

Встроенный предикат, 76

C

C4.5, 184, 200

Исчисление,
20 Gentzen, 40
естественное рассуждение,
40

последовательное,

40 CART,

184, 198 CASC, 48

прецедентная база, 183

прецедентная аргументация, 183 CBR,

см. прецедентная

аргументация Факторы уверенности , 114 Цепное

правило для

байесовских сетей, 120, 154, 155 Chatterbot, 5 Checkers, 102, 103, 108

В. Эртель, Введение в искусственный интеллект, Темы бакалавриата по компьютерным наукам, DOI 10.1007/978-0-85729-299-5, © Springer-Verlag London Limited 2011

Machine Translated by Google

Шахматы, 102, 103, 105, 108
Церковь, Алонсо, 5
Классификация, 162
Классификатор, 162, 164, 245
Пункт, 21 -
головной, 26
определенный, 26
Закрытая формула, 32
CLP, см. Программирование логики ограничений
Кластер, 206
Кластеризация, 206, 216
иерархическая, 209
КНФ, 21 год
Когнитивная наука, 3
Дополнительный, 23
Полный, 20
Компьютерная диагностика, 144
Заключение, 26
Условная вероятность, таблица 119, см.
CPT
Условно независимые, 146, 147, 155
Кондиционирование, 149, 155
Матрица путаницы, 11, 212
Соединение, 16, 21
Конъюнктивная нормальная форма, 21
Коннекционизм, 7
Последовательный, 23
Константа, 32 года
Программирование логики ограничений, 78
Проблема удовлетворения ограничений, 78
Корреляция,
коэффициент 137,
матрица 168, 216
Функция оценки стоимости, 96
Функция стоимости, 87, 98
Матрица затрат, 136, 139
КПП, 146, 149, 156, 199, 202
Кредитное поручение, 107, 260
Перекрестная проверка, 198
Перекрёстные помехи, 236
ЦСП, 78
Проклятие размерности, 274
Вырезать, 71 год

Д

D-разделение, 156
ДАГ, 155, 200
ДАИ, 8
Интеллектуальный анализ данных, 8, 165, 166, 182, 184, 197
Де Морган, 37 лет
Решение, 139
Дерево решений, 13, 185
индукция, 165, 185
обучение, 184, 275
Логика по умолчанию, 63
Правило по умолчанию, 63
Дельта-правило, 243–245
обобщенное, 247
Демодуляция, 46
Теория Демпстера – Шефера, 115
График зависимости, 137
Ограничение по глубине, 91
Деривация, 20
Детерминированный, 89, 102
Система диагностики, 132
Разъединение, 16, 21
Метрика расстояния, 207
Распределенный искусственный интеллект, 8
Распределенное обучение, 274
Распределение, 118, 134
Динамическое программирование, 263

Е

Электронное обучение, 5
Стремление к обучению, 182, 215
Элементарное событие, 115
Элиза, 5 лет
Алгоритм ЭМ, 201, 208
Энтропия,
максимум 188, 115, 122
Окружающая среда, 9, 12
непрерывные, 12
детерминированные,
12 дискретные,
12 недетерминированные,
12 наблюдаемые, 12
частично наблюдаемые, 12
Уравнение, направленное, 47
Эквивалентность, 16
Функция оценки, 103
Событие, 115
Экспертная система, 131, 144

Ф

Факт, 26
Факторизация, 23, 45
Ложноотрицательный, 141
Ложное срабатывание, 140
Алгоритм самого дальнего соседа, 211
Особенность, 106, 162, 171, 185
Художественное пространство, 163
Решатель ограничений конечной области, 79
Предложение первого порядка, 32
Цепочка вперед, 27
Проблема с рамой, 63
Свободные переменные, 32

Machine Translated by Google

Этот подход к решению задач оптимизации был представлен как динамическое программирование Ричардом Беллманом еще в 1957 году [Bel57]. Беллман признал, что для оптимальной политики это случай, когда:

> Независимо от начального состояния st и первого действия при , все последующие решения, исходящие из каждого возможного последующего состояния st+1 , должны быть оптимальными.

На основе так называемого принципа Беллмана становится возможным найти глобально оптимальная политика за счет локальной оптимизации отдельных действий. Мы выведем это принцип для MDP вместе с подходящим алгоритмом итерации.

Желаемая оптимальная политика π что удовлетворяет (10.3) на стр. 260 и (10.1) на стр. 260. Перепишем два уравнения и получим

$$B(s_t) = \max_{a_t, a_{t+1}, a_{t+2}, \ldots} (r(s_t, a_t) + \gamma\, r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \cdots). \quad (10.4)$$

Поскольку непосредственная награда r(st, at) зависит только от st и at , но не от последующие состояния и действия, максимизация может быть распределена, что в конечном итоге приводит к следующей рекурсивной характеристике V :

$$\begin{aligned} B(s_t) &= \max_{a_t} [r(s_t, a_t) + \gamma \max_{a_{t+1}, a_{t+2}, \ldots} (r(s_{t+1}, a_{t+1}) + \gamma\, r(s_{t+2}, a_{t+2}) + \cdots)] \\ &= \max_{a_t} [r(s_t, a_t) + \gamma\, V(s_{t+1})]. \end{aligned} \quad (10.5)$$

Уравнение (10.5) получается в результате замены t t + 1 в (10.4). Написано несколько проще:

$$B(s) = \max_a [r(s, a) + \gamma\, V(\delta(s, a))]. \quad (10.6)$$

Из этого уравнения, как и из (10.1) на стр. 260, следует, что для вычисления V (s), им опосредованное вознаграждение добавляется к вознаграждению всех государств-преемников со скидкой фактор γ . Если B (δ(s, a)) известно, то V (s) явно результат простой локальной операции тимизация всех возможных действий a в состоянии s. Это соответствует Беллману. принцип, из-за которого уравнение (10.6) также называют уравнением Беллмана.

Оптимальная политика π (s) выполняет действие в состоянии s , которое приводит к максимальное значение B . Таким образом,

$$\pi(s) = \operatorname{argmax}_a [r(s, a) + \gamma\, V(\delta(s, a))]. \quad (10.7)$$

Из уравнения рекурсии (10.6) правило итерации для аппроксимации V следует прямым образом:

$$\hat{V}_{\text{Против)}}(s) = \max_a [r(s, a) + \gamma\, \hat{V}(\delta(s, a))]. \quad (10.8)$$

Для начала инициализируются приблизительные значения $\hat{V}(s)$ для всех состояний, например со значением ноль. Теперь $\hat{V}(s)$ многократно обновляется для каждого состояния рекурсивно возвращаясь к значению $\hat{V}(\delta(s, a))$ лучшего последующего состояния. Этот процесс расчет V называется итерацией значения и схематически показана на рис. 10.7.

Machine Translated by Google

Рис. 10.7 Алгоритм итерации значения

```
ОЦЕНКА()
Для всех s ∈ S
    V (c) = 0
Повторить
    Для всех s ∈ S
        V̂ (s) = maxa[r(s, a) + γ V̂ (δ(s, a))]
Пока V(s) не изменится
```

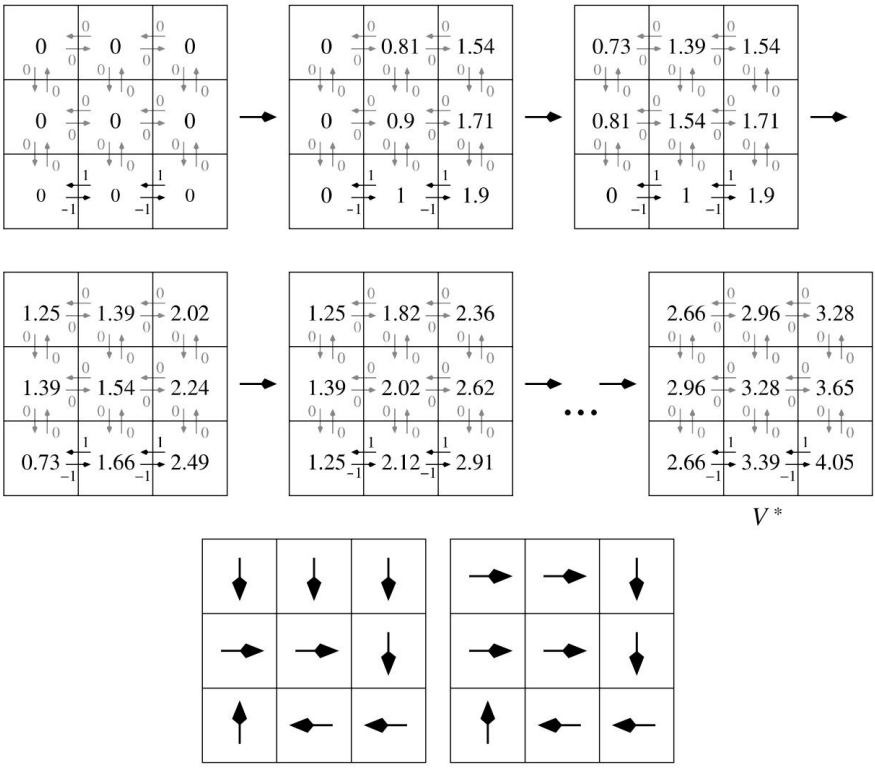


Рис. 10.8. Итерация значения в примере с 3 × 3 состояниями. Последние два изображения показывают два оптимальных политики. Цифры рядом со стрелками показывают немедленное вознаграждение r(s, a) за каждое действие.

на стр. 264. Можно показать, что итерация значения сходится к V [SB98]. Ан отличный анализ алгоритмов динамического программирования можно найти в [Sze10], где, основываясь на свойствах сжатия конкретных алгоритмов (например, итерации значений), сходимость может быть доказана с помощью теории Банаха с фиксированной точкой. рем.

Machine Translated by Google

На рис. 10.8 этот алгоритм применяется к примеру 10.1 на стр. 257 с γ = 0,9.
На каждой итерации состояния обрабатываются построчно снизу слева направо вверх.
Показаны несколько начальных итераций, а на втором изображении в нижнем ряду стабильные предельные значения для V .

Мы ясно видим прогресс обучения в этой последовательности. Агент повторно исследует все состояния, выполняет итерацию значения для каждого состояния и сохраняет политику, которая в виде табличной функции V эффективно , затем может быть дополнительно скомпилирована в виде используемая таблица π .

Кстати, чтобы найти оптимальную политику из V Было бы неправильно выбирать действие в состоянии st , которое приводит к состоянию с максимальным V, ценить. Корре отвечающим (10.7) на стр. 263, необходимо также добавить немедленное вознаграждение r(st, at) , поскольку мы ищем V на рис . 10.8 на (st), а не V (ст+1). Применительно к состоянию s = (2, 3) стр. 264 , это означает

$$\pi(2,3) = \operatorname{argmax}_{a \in \{left,right,up\}} [r(s,a) + \gamma V(\delta(s,a))]$$

$$= \operatorname{argmax}_{\text{влево, вправо, вверх}} \{1 + 0{,}9 \cdot 2{,}66,\ 1 + 0{,}9 \cdot 4{,}05,\ 0 + 0{,}9 \cdot 3{,}28\}$$

$$= \operatorname{argmax}_{\text{влево, вправо, вверх}} \{3{,}39,\ 2{,}65,\ 2{,}95\} = \text{слева}.$$

В (10.7) на стр. 263 мы видим, что агент в состоянии st должен знать немедленное вознаграждение rt и последующее состояние st+1 = δ(st, at) , чтобы выбрать оптимальное действие в .
Он также должен иметь модель функций r и δ. Поскольку для многих практических приложений это не так, нужны алгоритмы, которые также могут работать без знания r и δ. Раздел 10.6 посвящен такому алгоритму.

10.5 Обучаемый шагающий робот и его моделирование

Графический пользовательский интерфейс для простых экспериментов с обучением с подкреплением показан на рис. 10.9 на странице 266 [TEF09]. Пользователь может наблюдать за обучением с подкреплением для двухмерных пространств состояний разного размера. Для лучшего обобщения для сохранения состояния используются сети обратного распространения (см. раздел 10.8). Редактор обратной связи, показанный в правом нижнем углу, с помощью которого пользователь может вручную предоставить обратную связь об окружающей среде, особенно интересен для экспериментов. Не показано меню для настройки параметров итерации значения и обучения обратного распространения.

Помимо моделирования, специально для обучения [TEF09] были разработаны два небольших реальных ползающих робота с одинаковым двухмерным дискретным пространством состояний [TEF09]. Два робота показаны на рис. 10.10 на стр . 266 .

2Дополнительная информация и соответствующие источники о сканирующих роботах доступны на веб-сайте www. hs-weingarten.de/~ertel/kibuch.

Machine Translated by Google

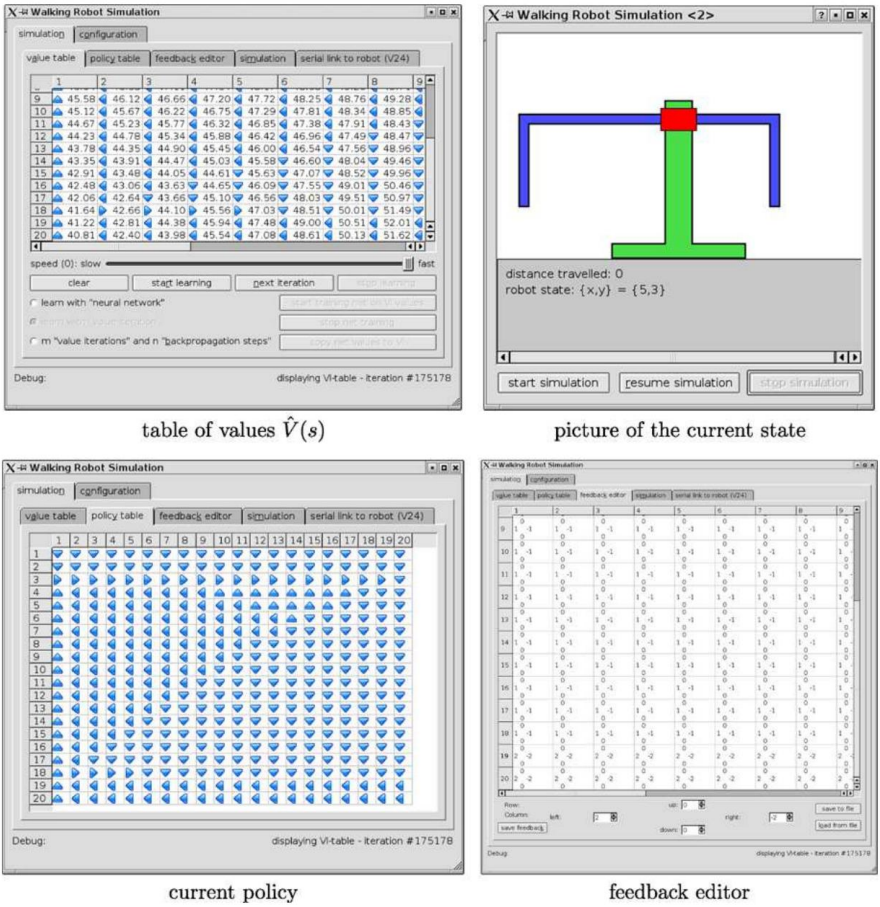


Рис. 10.9 Четыре разных окна симулятора шагающего робота



Рис. 10.10 Два варианта ползающего робота

Machine Translated by Google

атор. Сервоприводы управляются микроконтроллером или через беспроводной интерфейс прямо с ПК. Используя программное обеспечение для моделирования, матрицу обратной связи робота можно визуализировать на ПК. С помощью этой сохраненной обратной связи можно обучить политику на ПК (который вычисляет быстрее), а затем снова загрузить в робота и выполнить. Однако робот также может обучаться автономно. Для пространства состояний размером 5× 5 это занимает около 30 секунд.

Интересно наблюдать разницу между симуляцией и «настоящим» роботом. В отличие от симуляции, краулер изучает правила, при которых он никогда не отрывает руку от земли, но, тем не менее, движется вперед очень эффективно. Причина этого в том, что в зависимости от поверхности подземелья кончик «подмышки» может захватывать землю при движении назад, но проскальзывает при движении вперед. Этот эффект очень чувствительно воспринимается датчиками измерения расстояния и соответственно оценивается в процессе обучения.

Адаптивность робота приводит к удивительным эффектам. Например, мы можем наблюдать, как краулер, несмотря на неисправный сервопривод, который буксует под определенным углом, тем не менее учится ходить (больше похоже на ковыляние). Он даже способен адаптироваться к изменившимся ситуациям путем изменения политики. Весьма желательным эффектом является способность на заданных разногладких поверхностях (например, разных шероховатых коврах) выучить оптимальную политику для каждой из них. Также оказывается, что настоящий робот действительно очень хорошо адаптируется к небольшому пространству состояний размером 5 × 5.

Читатель может (за отсутствием настоящего робота) смоделировать различные поверхности или дефекты сервоприводов, варьируя значения обратной связи и затем наблюдая за результирующими политиками (упражнение 10.3 на стр. 276).

10.6 Q-обучение

Политика, основанная на оценке возможных состояний-последователей, явно непригодна, если у агента нет модели мира, то есть когда он не знает, к какому состоянию приводит возможное действие. В большинстве реальных приложений агент не может прибегать к такой модели мира. Например, робот, который должен захватывать сложные объекты, не может предсказать, будет ли объект надежно удерживаться в его захвате после захвата или останется на месте.

При отсутствии модели мира необходима оценка действия at, осуществляемого в состоянии st , даже если еще неизвестно, куда это действие ведет. Таким образом, теперь мы работаем с функцией оценки Q(st, at) для состояний и связанных с ними действий. С помощью этой функции выбор оптимального действия производится по правилу

$$\pi(s) = \operatorname*{argmax}_a Q(s, a). \tag{10.9}$$

Чтобы определить функцию оценки, мы снова используем пошаговое дисконтирование оценки для пар состояние-действие, которые происходят дальше в будущем, точно так же, как в (10.1) на стр. 260. Таким образом, мы хотим максимизировать $r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$.
Поэтому для оценки действия at в состоянии st определим по аналогии с (10.4) на

Machine Translated by Google

стр. 263:

$$Q(st, at) = \max_{at+1,\, at+2,\ldots} (r(st, at) + \gamma\, r(st+1, at+1) + \gamma^2 r(st+2, at+2) + \cdots). \quad (10.10)$$

Аналогично подходу для итерации значений, мы приводим это уравнение к простой рекурсивной форме с помощью

$$\begin{aligned} Q(st, at) &= \max_{at+1,\, at+2,\ldots} (r(st, at) + \gamma\, r(st+1, at+1) + \gamma^2 r(st+2, at+2) + \cdots) \\ &= r(st, at) + \gamma \max_{at+1,\, at+2,\ldots} (r(st+1, at+1) + \gamma\, r(st+2, at+2) + \cdots) \\ &= r(st, at) + \gamma \max_{at+1} (r(st+1, at+1) + \gamma \max_{at+2} (r(st+2, at+2) + \cdots)) \\ &= r(st, at) + \gamma \max_{at+1} Q(st+1, at+1) \\ &= r(s, a) + \gamma \max_{a'} Q(\delta(s, a), a'). \end{aligned} \quad (10.11)$$

В чем тогда преимущество по сравнению с итерацией значений? Старое уравнение лишь немного переписано, но это оказывается как раз правильным подходом к новому алгоритму. Вместо сохранения V теперь сохраняется функция Q , и агент может выбирать свои действия из функций δ и r без модели мира. Мы однако до сих пор нет процесса, который мог бы изучить Q напрямую, то есть без знания V. Из ·

рекурсивной формулировки Q(s, a) можно вывести итерационный алгоритм для определения Q(s, a) в прямой манере. Произвольно инициализируем таблицу Q(s, a) для всех состояний, например нулями, и итеративно выполняем

$$\hat{Q}(s, a) = r(s, a) + \gamma \max_{a'} \hat{Q}(\delta(s, a), a'). \quad (10.12)$$

Остается заметить, что функции r и δ нам неизвестны . Мы решаем эту проблему достаточно прагматично, позволяя агенту в его окружении в состоянии s выполнять действие a. Затем состояние-преемник, очевидно, представляет собой δ(s, a) , и агент получает свое вознаграждение от окружающей среды. Алгоритм, показанный на рис. 10.11 на стр. 269 , реализует этот алгоритм Q-обучения.

Применение алгоритма к примеру 10.1 на стр . 257 с γ = 0,9 и nx = 3, ny = 2 (то есть в сетке 2 × 3) показано в качестве примера на рис. 10.12 на стр . 269 . На первом рисунке все значения Q инициализированы нулями. На втором рисунке после первой последовательности действий четыре значения r , не равные нулю, становятся видимыми как значения Q. На последнем рисунке дана изученная оптимальная полити Следующая теорема, доказательство которой можно найти в [Mit97], показывает, что этот алгоритм сходится не только в примере, но и вообще.

Machine Translated by Google

Q-ОБУЧЕНИЕ ()

Для всех s S, a A

Q(s, a) = 0 (или случайно)

Повторить

Выбрать (например, случайным образом)

состояние

s Повторить Выбрать действие a и выполнить

его Получить награду r и новое состояние s

Q(s, a) := r(s, a) + γ maxa′ Q(s , a')

′ c := c

До тех пор, пока s не станет конечным состоянием или не будет достигнут предел времени

Пока Q^ не сходится

Рис. 10.11 Алгоритм Q-обучения

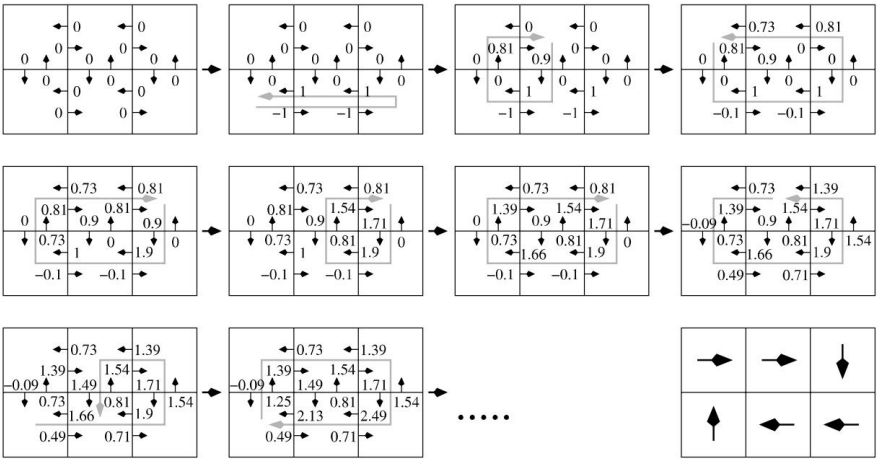


Рис. 10.12 Q-обучение применительно к примеру с nx = 3, ny = 2. Серыми стрелками отмечены действия, выполняемые на каждой картинке. Приведены обновленные значения Q. На последней картинке показана текущая политика, которая также является оптимальной.

Теорема 10.1. Пусть дан детерминированный MDP с ограниченным немедленным вознаграждением r(s, a) . Уравнение (10.12) на стр. 268 с 0 γ < 1 используется для обучения. Пусть Q^ n(s, a) будет значением Q(s, a) после n обновлений. Если каждую пару состояний-действий посещают бесконечно часто, то Q^ n(s, a) сходится к Q(s, a) для всех значений s и a при n .

Machine Translated by Google

Доказательство Поскольку каждый переход между состояниями и действиями происходит бесконечно часто, мы рассматриваем последовательные интервалы времени с тем свойством, что в каждом интервале все переходы между состояниями и действиями происходят хотя бы один раз. Покажем теперь, что максимальная ошибка для всех записей в таблице Qˆ уменьшается по крайней мере в γ раз на каждом из этих интервалов. Позволять

$$_{н} = \max_{с, а} |Qˆ n(s, a) \quad Q(s, a)|$$

– максимальная ошибка в таблице Qˆ n и с ′ = δ(s, a). Для каждой записи таблицы Qˆ n(s, a) мы вычисляем его вклад в ошибку через интервал как

$$|Qˆ n+1(s, a) \quad Q(s, a)| = r + γ\max_{′a} . Qˆ n(s′ , a′) \quad r + γ \max_{′a} Q(s' , a')$$

$$= γ\max_{a′} . Qn (s′ , a′) \quad \max_{a′} Q(s' , a')$$

$$γ \max_{′a} |Qˆ n(s′ , a′) \quad Q(s′ , a′)|$$

$$γ \max_{s′′,a′} |Qˆ n(s′′, a′) \quad Q(s′′, a′)| = γн .$$

Первое неравенство верно, поскольку для произвольных функций f и g

$$\max_{x} f (x) \quad \max_{Икс} g(x) \quad \max_{Икс} |f (x) \quad g(x)|$$

а второе неравенство верно, так как при дополнительном варьировании состояния s ′′ результирующий максимум не может стать меньше. Таким образом, было показано, что n+1 γ н. Поскольку ошибка на каждом интервале уменьшается не менее чем в γ раз , через k интервалов она не превосходит γ k0, и, как следствие, 0 ограничено. Поскольку каждое состояние посещается бесконечно много раз, существует бесконечно много интервалов и n сходится до нуля.

Согласно теореме 10.1 на стр. 269 Q-обучение сходится независимо от действий, выбранных при обучении. Это означает, что для сходимости не имеет значения, какие действия выбирает агент, лишь бы каждое из них выполнялось бесконечно часто.
Однако скорость сходимости, безусловно, зависит от того, какие пути выбирает агент во время обучения (см. раздел 10.7).

10.6.1 Q-обучение в недетерминированной среде

Во многих приложениях робототехники среда агента недетерминирована. Это означает, что реакция среды на действие a в состоянии s в два разных момента времени может привести к различным последующим состояниям и вознаграждениям. Такой недетерминированный марковский процесс моделируется вероятностной функцией перехода δ(s, a) и вероятностным немедленным вознаграждением r(s, a). Чтобы определить функцию Q , каждый раз ожидаемое значение должно вычисляться для всех возможных последующих состояний. Уравнение (10.11)

Machine Translated by Google

на стр. 268 , таким образом, обобщается на

$$Q(st, at) = E(r(s, a)) + \gamma \sum_{s'} P(s' | s, a) \max_{a'} Q(s', a'), \quad (10.13)$$

где $P(s' | s, a)$ — вероятность перехода из состояния s в последующее состояние s с действием a. К сожалению, нет гарантии сходимости для Q-обучения в недетерминированном случае, если мы будем действовать, как и раньше, в соответствии с (10.12) на стр. 268.
Это связано с тем, что при последовательных прогонах внешнего цикла алгоритма на рис. 10.11 на стр. 269 вознаграждение и последующее состояние могут быть совершенно разными для одного и того же состояния s и одного и того же действия a. Это может привести к чередующейся последовательности, которая перескакивает между несколькими значениями. Чтобы избежать таких сильных скачков значений Q , мы добавляем старое взвешенное значение Q в правую часть (10.12) на стр. 268. Это стабилизирует итерацию. Затем правило обучения читается

$$\hat{Q}_n(s, a) = (1 - \alpha_n)\hat{Q}_{n-1}(s, a) + \alpha_n [r(s, a) + \gamma \max_{a'} \hat{Q}_{n-1}(\delta(s, a), a')] \quad (10.14)$$

с изменяющимся во времени весовым коэффициентом

$$\alpha_n = \frac{1}{1 + b_n(s, a)}.$$

Значение $b_n(s, a)$ показывает, как часто действие a выполнялось в состоянии s на n-й итерации. При малых значениях b_n (т. е. в начале обучения) стабилизирующий член $\hat{Q}_{n-1}(s, a)$ не играет роли, так как мы хотим, чтобы процесс обучения шел быстро. Однако позже b_n становится больше и тем самым предотвращает чрезмерно большие скачки в последовательности значений $\hat{Q}$. При интегрировании (10.14) в Q-обучение значения $b_n(s, a)$ должны быть сохранены для всех пар состояние-действие. Этого можно добиться, расширив таблицу значений $\hat{Q}$.

Для лучшего понимания (10.14) мы упростим его, предположив, что $\alpha_n = \alpha$ является постоянной и преобразуя ее следующим образом:

$$\hat{Q}_n(s, a) = (1 - \alpha)\hat{Q}_{n-1}(s, a) + \alpha [r(s, a) + \gamma \max_{a'} \hat{Q}_{n-1}(\delta(s, a), a')]$$

$$= \hat{Q}_{n-1}(s, a) + \alpha [r(s, a) + \gamma \max_{a'} \hat{Q}_{n-1}(\delta(s, a), a') - \hat{Q}_{n-1}(s, a)].$$

TD-ошибка

Новое значение Q $\hat{Q}_n(s, a)$ может быть ясно представлено как старое $\hat{Q}_{n-1}(s, a)$ плюс α, умноженное на поправочный член, который совпадает с изменением значения Q на этом шаге. Поправочный член называется TD-ошибкой или ошибкой временной разности, а приведенное выше уравнение для изменения значения Q является частным случаем TD-обучения, важного класса алгоритмов обучения [SB98]. Для $\alpha = 1$ мы получаем Q-обучение, описанное выше. При $\alpha = 0$ значения $\hat{Q}$ совершенно не меняются. Таким образом, никакого обучения не происходит.

Machine Translated by Google

10.7 Разведка и разработка

Для Q-обучения до сих пор была дана только грубая схема алгоритма. Особенно
отсутствует описание выбора начального состояния каждый раз и действий, которые необходимо выполнить
во внутреннем цикле рис. 10.11 на стр. 269. Для выбора следующего действия есть две возможности. Среди
возможных действий
один может быть выбран случайным образом. В долгосрочной перспективе это приводит к равномерному исследованию
всех возможных действий или политик, но с очень медленной конвергенцией. Альтернатива
к этому относится использование ранее изученных значений Q^. Здесь агент всегда
выбирает действие с наибольшим значением Q^. Это приводит к относительно быстрой сходимости конкретной
траектории. Однако другие пути остаются непройденными на протяжении всего пути.
к концу. В крайнем случае мы можем получить неоптимальную политику. Поэтому в Теореме 10.1 на стр. 269
требуется, чтобы посещалась каждая пара состояние-действие.
бесконечно много раз. Рекомендуется использовать комбинацию разведки и
эксплуатация с высокой долей разведки в начале и уменьшать ее все больше и больше
больше со временем.

Выбор начального состояния также влияет на скорость обучения. Во-первых
На трех рисунках на рис. 10.12 на стр. 269 хорошо видно, что на первых итерациях изменяются только значения
Q в непосредственной близости от пар состояние-действие
немедленной наградой. Если начать дальше от такой точки,
ненужная работа. Это предполагает передачу априорных знаний о состоянии-действии
пары с немедленным вознаграждением в начальные состояния рядом с этими точками. В ходе
обучения, то можно выбрать более отдаленные начальные состояния.

10.8 Аппроксимация, обобщение и сходимость

Поскольку Q-обучение было описано до сих пор, таблица со всеми значениями Q явно
сохраняется в таблице. Это возможно только при работе с конечным пространством состояний с
конечное число действий. Однако если пространство состояний бесконечно, например, в случае
непрерывных переменных, то невозможно ни сохранить все значения Q, ни посетить
все пары состояние-действие во время обучения.

Тем не менее, существует простой способ использования Q-обучения и итерации значений непрерывных
переменных. Таблица Q(s, a) заменяется нейронной сетью, например
сеть обратного распространения с входными переменными s, a и значением Q в качестве цели
выход. Для каждого обновления значения Q нейронной сети предоставляется обучающая программа.
пример с (s, a) в качестве входных данных и Q(s, a) в качестве целевого вывода. В конце мы имеем конечное
представление функции Q(s, a). Поскольку у нас есть только конечное множество
обучающих примеров, но функция Q(s, a) определена для бесконечного множества входных данных, мы
таким образом, автоматически получают обобщение, если размер сети выбран соответствующим образом (см.
главу 9). Вместо нейронной сети мы также можем использовать другую контролируемую
алгоритм обучения или аппроксиматор функции, такой как машина опорных векторов или
гауссовский процесс.

Machine Translated by Google

Однако переход от конечного числа обучающих примеров к непрерывной функции в определенных ситуациях может стать очень дорогостоящим. Q-обучение с аппроксимацией функций может не сходиться, потому что теорема 10.1 на стр. 269 верна только в том случае, если каждая пара состояние-действие посещается бесконечно часто.

Однако проблемы сходимости также могут возникнуть в случае конечного числа пар состояние-действие, когда Q-обучение используется в POMDP. Q-обучение может быть применено — в обоих описанных вариантах — к детерминированным и недетерминированным марковским процессам (MDP). Для POMDP может случиться так, что агент, например, из-за зашумленных датчиков, воспринимает множество разных состояний как одно. Часто многие состояния в реальном мире целенаправленно сопоставляются с одним так называемым наблюдением. Результирующее пространство наблюдения становится намного меньше пространства состояний, благодаря чему обучение становится быстрее и можно избежать переобучения (см. раздел 8.4.7).

Однако, объединяя вместе несколько состояний, агент больше не может различать фактические состояния, и действие может привести его ко многим различным последующим состояниям, в зависимости от того, в каком состоянии он действительно находится. Это может привести к проблемам конвергенции для ценности. итерации или для Q-обучения. В литературе (например, в [SB98]) предлагается множество различных подходов к решению.

Также очень многообещающими являются так называемые методы улучшения политики и производные от них методы градиента политики, в которых значения Q не изменяются, а политика изменяется напрямую. В этой схеме в пространстве всех политик ищется политика, максимизирующая совокупное дисконтированное вознаграждение ((10.1) на стр. 260). Одна из возможностей добиться этого — следовать градиенту совокупного вознаграждения до максимума. Политика, найденная таким образом, явно оптимизирует совокупное вознаграждение. В [PS08] показано, что этот алгоритм может значительно ускорить обучение в приложениях с большими пространствами состояний, таких как те, которые используются для роботов-гуманоидов.

10.9 Приложения

Тем временем практическая полезность обучения с подкреплением неоднократно демонстрировалась. Из большого количества примеров этого кратко представим небольшую подборку.

TD-обучение вместе с сетью обратного распространения с 40–80 скрытыми нейронами очень успешно использовалось в TD-gammon, программе для игры в нарды [Tes95]. Единственной немедленной наградой за программу является результат в конце игры. Оптимизированная версия программы с опережением на два хода была обучена сама на себе в 1,5 млн партий. Он продолжал побеждать игроков и игроков мирового класса, а также трех лучших игроков-людей.

Есть много приложений в робототехнике. Например, в лиге футбольных симуляторов RoboCup лучшие футбольные команды роботов теперь успешно используют обучение с подкреплением [SSK05, Robb]. Балансировка шеста, которая относительно проста для человека, много раз успешно решалась с помощью обучения с подкреплением.

Впечатляющую демонстрацию способности роботов к обучению продемонстрировал Расс Тедрейк на IROS 2008 в своей презентации о модели самолета, которая учится приземляться в определенной точке, как птица, садящаяся на ветку [Ted08]. Потому что воздух

Machine Translated by Google

течения становятся очень турбулентными во время такого динамичного захода на посадку, соответствующее дифференциальное уравнение, уравнение Навье – Стокса, неразрешимо. Следовательно, приземление нельзя контролировать классическим математическим способом. Комментарий Тедрака по этому поводу:

> «Птицы не решают Навье-Стокса!»

Очевидно, что птицы могут научиться летать и приземляться даже без уравнения Навье-Стокса. Тедрейк показал, что теперь это возможно и для самолетов.

Сегодня также можно научиться управлять настоящим автомобилем всего за 20 минут, используя Q-обучение и аппроксимацию функций [RMD07]. Этот пример показывает, что реальные промышленные приложения, в которых несколько измерений должны быть сопоставлены с действиями, можно очень хорошо изучить за короткое время.

Настоящие роботы по-прежнему испытывают трудности с обучением в многомерных пространствах состояния-действия, потому что по сравнению с симуляцией настоящие роботы относительно медленно получают обратную связь от окружающей среды. Таким образом, из-за ограничений по времени невозможно реализовать многие миллионы необходимых тренировочных циклов. Здесь, кроме алгоритмов быстрого обучения, нужны методы, которые позволяют хотя бы части обучения происходить оффлайн, то есть без обратной связи со средой.

10.10 Проклятие размерности

Несмотря на успехи последних лет, обучение с подкреплением остается активной областью исследований в области ИИ, не в последнюю очередь потому, что даже лучшие алгоритмы обучения, известные сегодня, по-прежнему непрактичны для многомерных пространств состояний и действий из-за их гигантского времени вычислений. Эта проблема известна как «проклятие размерности».

В поисках решения этой проблемы ученые наблюдают за животными и людьми во время обучения. Здесь мы замечаем, что обучение в природе происходит на многих уровнях абстракции. Ребенок сначала осваивает простые моторные и языковые навыки на самом низком уровне. Когда они хорошо изучены, они сохраняются и позже могут быть вызваны в любое время и использованы. В переводе на язык информатики это означает, что каждая изученная способность инкапсулируется в модуль, а затем, на более высоком уровне, представляет собой действие. Используя такие сложные действия на более высоком уровне, пространство для действий значительно сокращается, и, таким образом, обучение ускоряется. Аналогичным образом можно абстрагировать состояния и, таким образом, уменьшить пространство состояний. Такое обучение на нескольких уровнях называется иерархическим обучением [BM03].

Другой подход к модуляризации обучения — это распределенное обучение или мультиагентное обучение [PL05]. При обучении двигательным навыкам робота-гуманоида необходимо одновременно управлять до 50 различными двигателями, что приводит к 50-мерному пространству состояний и 50-мерному пространству действий. Чтобы уменьшить эту гигантскую сложность, централизованное управление заменяется распределенным управлением. Например, каждый отдельный двигатель может иметь индивидуальное управление, которое управляет им напрямую, если возможно, независимо от других двигателей. В природе мы находим этот вид контроля у насекомых. Например, многочисленными ногами многоножки управляет не центральный мозг, а каждая пара ног имеет свой крошечный «мозг».

Machine Translated by Google

Подобно неосведомленному комбинаторному поиску, обучение с подкреплением имеет задачу найти лучшее из огромного количества политик. Задача обучения значительно упрощается, если у агента есть более или менее хорошая политика до начала обучения.
Тогда задачи многомерного обучения могут быть решены раньше. Но как найти такую начальную политику? Здесь есть две основные возможности.

Первая возможность — это классическое программирование. Программист предоставляет агенту политику, включающую программу, которую он считает хорошей. Затем происходит переключение, например, на Q-learning. Агент выбирает, по крайней мере в начале обучения, свои действия в соответствии с запрограммированной политикой и таким образом попадает в «интересные» области пространства состояния-действия. Это может привести к резкому сокращению пространства поиска обучения с подкреплением.

Если традиционное программирование становится слишком сложным, мы можем начать обучение робота или агента, заставив человека предписывать правильные действия. В простейшем случае это делается с помощью ручного дистанционного управления роботом. Затем робот сохраняет запрещенное действие для каждого состояния и обобщает, используя алгоритм обучения с учителем, такой как обратное распространение или обучение дерева решений. Это так называемое демонстрационное обучение [BCDS08, SE10], таким образом, также обеспечивает начальную политику для последующего обучения с подкреплением.

10.11 Резюме и перспективы

Сегодня у нас есть доступ к хорошо функционирующим и проверенным алгоритмам обучения для обучения наших машин. Однако задача инструктора или разработчика по-прежнему требует сложных приложений. Существует очень много возможностей, как структурировать обучение робота, и без экспериментов оно не будет успешным. На практике это экспериментирование может быть очень утомительным, потому что каждый новый учебный проект должен быть разработан и запрограммирован. Здесь необходимы инструменты, которые, помимо различных алгоритмов обучения, также предлагают тренеру возможность комбинировать их с традиционным программированием и демонстрационным обучением. Одним из первых инструментов такого рода является Teaching-Box [ESCT09], который в дополнение к обширной библиотеке программ также предлагает шаблоны для конфигурации учебных проектов и для связи между роботом и окружающей средой. Например, учитель-человек может дать роботу дополнительную обратную связь с клавиатуры или через речевой интерфейс в дополнение к обратной связи из окружающей среды.

Обучение с подкреплением — увлекательная и активная область исследований, которая будет все чаще использоваться в будущем. Все больше и больше систем управления роботами, а также другие программы будут учиться на основе обратной связи с окружающей средой. На сегодняшний день существует множество вариаций представленных алгоритмов, а также совершенно разные алгоритмы. Проблема масштабирования остается нерешенной. Для небольших пространств действий и состояний с небольшим количеством степеней свободы можно добиться впечатляющих результатов. Если число степеней свободы в пространстве состояний вырастет до 18, например, для простого робота-гуманоида, то обучение станет очень дорогим.

Machine Translated by Google

Для дальнейших основополагающих лекций мы рекомендуем компактное введение в обучение с подкреплением в книге Тома Митчелла [Mit97]. Стандартная работа Саттона и Барто [SB98] является тщательной и всеобъемлющей, как и обзорная статья Кельблинга, Литтмана и Мура [KLM96].

10.12 Упражнения

Упражнение 10.1

(a) Подсчитайте количество различных политик для n состояний и n действий. Таким образом, возможны переходы из каждого состояния в каждое состояние. (б) Как изменится количество политик в подзадаче (а), если пустые действия, т. е. действия из одного состояния в себя, не допускаются. (c) С помощью стрелочных диаграмм, подобных тем, что показаны на рис. 10.3 на стр. 259, укажите все политики для два состояния.

(d) Используя стрелочные диаграммы, укажите все политики без пустых действий для трех состояний.

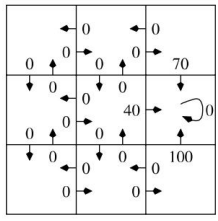
Упражнение 10.2. Использование итерации значения вручную в примере 10.1 на стр. 257 с nx = ny = 2.

Упражнение 10.3. Проведите различные эксперименты, используя симулятор итерации значений. (а) Установите симулятор итерации значений из [Tok06]. (b) Воспроизведите результаты упражнения 10.2 , сначала введя обратную связь с редактор обратной связи, а затем выполнить итерацию значения. (c) Моделируйте поверхности разной гладкости и наблюдайте, как изменяется политика. (d) Используя аналогичную матрицу обратной связи, постепенно увеличивайте пространство состояний примерно до 100 × 100 и подбирайте коэффициент дисконтирования γ таким образом, чтобы в результате получалась разумная политика.

Упражнение 10.4 . Покажите, что для примера расчета на рис. 10.8 на стр. 264) 4,05499. точное значение V (3, 3) = 1,9/(1 0,9 6

Упражнение 10.5

Проведите Q-обучение на сетке 3× 3 справа. Состояние в середине справа — это поглощающее целевое состояние.



Упражнение 10.6 Дана рука робота с n шарнирами (размерами) и ℓ дискретными состояниями на сустав. Возможны действия из каждого состояния в каждое состояние (если робот ничего не делает, это оценивается как (пустое) действие).

Machine Translated by Google

a) Приведите формулу количества состояний и количества действий в каждом состоянии.
и за количество полисов для робота.

(b) Создайте таблицу с количеством стратегий для n = 1, 2, 3, 4, 8 и ℓ =
1, 2, 3, 4, 10.

(c) Чтобы уменьшить количество возможных стратегий, предположим, что количество
возможных действий на сустав всегда равно 2 и что робот может перемещать только
один сустав за раз. Введите новую формулу для количества стратегий и создайте
соответствующую

таблицу. (d) С помощью вычисленного результата обосновать, что агент, который
действует автономно и адаптивно при n = 8 и l = 10, безусловно, может быть назван разумным.

Machine Translated by Google

Machine Translated by Google

Решения для упражнений 11

11.1 Введение

Упражнение 1.3. Многие известные процессы вывода, процессы обучения и т. д. являются NP-полными или даже неразрешимыми. Что это означает для ИИ?

Упражнение 1.4. Если проблема является NP-полной или может быть описана как «сложная», это означает, что есть случаи, в которых проблема не может быть решена за приемлемое время. Это так называемый худший случай. В некоторых приложениях приходится мириться с тем фактом, что в худшем случае эффективное решение невозможно. Это означает, что и в будущем будут практически значимые проблемы, которые в некоторых частных случаях неразрешимы.

Поэтому ИИ не найдет универсальной формулы и не построит супермашину, с помощью которой все проблемы станут решаемыми. Он скорее ставит перед собой задачу построения систем с более высокой вероятностью нахождения решения или с более высокой вероятностью нахождения быстрых оптимальных решений. Мы, люди, в повседневной жизни довольно хорошо справляемся с неоптимальными решениями. Причина проста — чрезмерная стоимость поиска оптимального решения. Например, мне требуется всего семь минут, чтобы найти дорогу из пункта А в пункт Б по карте в незнакомом городе. Самый короткий путь занял бы всего шесть минут. Однако поиск кратчайшего пути занял бы около часа. Доказательство оптимальности пути может обойтись еще дороже.

Упражнение

1.5 (a) Выход зависит не только от ввода, но и от содержимого памяти. Для входа x, в зависимости от содержимого памяти, выход может быть y1 или y2. Таким образом, он не уникален и, следовательно, не является функцией. (b) Если рассматривать содержимое памяти как дополнительный вход, то выход будет уникальным (поскольку агент детерминирован) и агент представляет собой функцию.

В. Эртель, Введение в искусственный интеллект,
Темы бакалавриата по компьютерным
наукам, DOI 10.1007/978-0-85729-299-5_11, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google

Упражнение 1.6

(а) Скорость vx (t) = $\frac{x}{t}$ limt 0 $\frac{x(t) \quad x(t \quad t) =}{\tau}$ $\frac{x(\tau) - x(\tau - \tau)}{\tau}$. vy рассчитывается аналогично.

(t)2 (t)2 2x vx (t) vx (t t) (б) Ускорение ax(t) = vx(t) = limt 0 t2 x(t) x(t t) x(t t) x(t t) x(t 2t) "=" ay вычисляется аналогично. Также нужна позиция в три момента времени t 2t, t t, t. "=" x(t) 2x(t t)+x(t 2t) limt 0 .

Упражнение

1.7 (а) Затраты агента 1 = 11 · 100 центов + 1 · 1 цент = 1, 101 цент.

Затраты на 2 = 0 · 100 центов + 38 · 1 цент = 38 центов.

Следовательно, агент 2 экономит 1 101 цент - 38 центов = 1 063 цента. (b)

Прибыль агента 1 = 189 · 100 центов + 799 · 1 цент = 19 699 центов.

Прибыль агента 2 = 200 · 100 центов + 762 · 1 цент = 20 762 цента.

Таким образом, агент 2 имеет прибыль на 20 762 цента 19 699 центов = на 1 063 цента больше.

Если кто-то оценивает упущенную выгоду из-за ошибок, агент, основанный на полезности, принимает те же решения, что и агент, основанный на затратах.

11.2 Пропозициональная логика

Упражнение 2.1 С сигнатурой = {σ1, σ2,...,σn} и формулой грамматических переменных синтаксис логики высказываний можно определить следующим образом:

формула ::= σ1 |σ2 |··· |σn |w |f

| ¬формула |(формула)| формула формула

| формула формула |формула формула

| формула формула

Упражнение 2.2. Доказательство методом таблицы истинности.

Упражнение 2.3 (а) (¬A B) (¬B A) (b) (¬A B) (¬B A) (c) t

Упражнение 2.4 (а) выполнимо (б) верно (в) невыполнимо

Упражнение

2.6 (а) В упражнении 2.3 (c) уже было показано, что A (A B) B является тавтологией.

Таким образом, теорема дедукции гарантирует правильность правила вывода.

(b) Мы показываем методом таблицы истинности, что (A B) (¬B C) (A C) есть тавтология.

Machine Translated by Google

Упражнение 2.7. Применение правила разрешения к клаузулам (f B) и (¬B f) дает резольвенту (f f) (f). Теперь применим правило разрешения к предложениям B и ¬B и получим пустое предложение в качестве резольвенты. Поскольку (f B) B и (¬B f) ¬B, (f) (). На практике важно, что всякий раз, когда образуется пустое предложение, это происходит из-за противоречия.

Упражнение 2.8. Если KB содержит противоречие, то есть два предложения A и ¬A, которые позволяют вывести пустое предложение. Противоречие в KB явно остается в KB ¬Q. Следовательно, это также позволяет получить пустое предложение.

Упражнение 2.9 (a) (A B) (¬A ¬B) (b) (A B) (B C) (A C)

Упражнение 2.10 Формализация: Сообщник: A, Автомобиль: C, Ключ: K

WB (A C) [(¬A ¬K) (A K)] K

Преобразование в КНФ: (¬A ¬K) (A K) (¬K A) (¬A K)

Попробуйте доказать C и добавьте ¬C к набору предложений. Набор предложений CNF

(¬A C)1 (¬K A)2 (¬A K)3 (K)4 (¬C)5.

Доказательство разрешения : Res(2, 4):
(A)6 Res(1, 6): (C)7
Res(7, 5): ()8
Таким образом, C было показано.

Упражнение
2.11 (a) KB (A B) (¬B C), Q (A C)
KB ¬Q (A B)1 (¬B C)2 (¬A)3 (¬C)4
Доказательство разрешения : Res (1, 3): (B) 5
Res(2, 4): (¬B)6
Res(5, 6) : () (b)
¬(¬B (B ¬A) ¬A) (¬B)1 (B ¬A)2 (A)3
Доказательство разрешения : Res(1, 2): (¬A)4
Рез (3, 4): ()

Упражнение 2.12. Применяя эквивалентности из теоремы 2.1 на стр. 18, мы можем сразу доказать утверждения.

Упражнение 2.13

Res(8, 9): (C F E f)10
Res(3, 10): (F E f)11
Res(6, 11): (A B C E f)12
Res(1, 12): (B C E f)13
Res(2, 13): (C E f)14
Res(3, 14): (E f)15
Рез(5, 15) : ()

Machine Translated by Google

11.3 Логика предикатов первого порядка

Упражнение

3.1 (a) x мужчина(x)
¬женщина(x) (b) x y z отец(x, y) мужчина(x)
ребенок(y, x, z) (c) x y siblings(x, y) [(z отец(z, x) отец(z, y)) (z мать(z, x)
мать(z, y))] (d)
x y z родители(x, y, z) отец(x, z) мать(y, z) (e) x y
дядя(x, y) z u ребенок(y, z, u) братья и сестры(z, x) мужчина(x) (f) x
y предок(x, y) z ребенок(y, x, z) u v дочерний(u, x, v)
предок (и, y))

Упражнение

3.2 (a) x y z отец(y, x) мать(z, x) (b)
x y ребенок(y, x, z) (c)
x птица(x) мухи (x) (d)
x y z животное(x) животное(y) ест(x, y) ест(y, z) зерно(z) (e) x
животное(x) (y (ест(x, y) (растение(y) (животное(y) z растение(z) ест(y, z)
намного_меньше(y, x)))

Упражнение 3.3 x y zx = отец(y) мужчина(x) ребенок(y, x, z)

Упражнение
3.4 x y x<y y<x x = y, x
y x<y ¬y<x, x y z
x<y y<z x<z

Упражнение 3.5

(a) МГУ: x/f (z), u/f (y), терм: p(f (z), f (y)) (б) не
унифицируемый
(c) МГУ: x/ cos y, z/ 4 7 · cos y, терм: cos y = 4 7 · cos y (d) не
унифицируемый
(e) MGU: u/f (g(w, w), g(g(w, w), g(w, w)), g(g(g(w, w), g(w, w)), g(g(w, w), g(w, w)))), x/g(w, w),
y/g(g(w, w), g(w, w)), z/g(g(g(w, w), g(w, w)), g(g(w, w), g(w, w))) член: q(f (g(w, w), g(g(w,
w), g(w, w)), g(g(g(w, w) ,
g(w, w)), g(g(w, w), g(w, w)))), f (g(w, w), g(g(w, w), g(w, w)), g(g(g(w, w), g(w, w)), g(g(w,
w), g(w, w)))))

Упражнение

3.7 (a) Пусть задана невыполнимая формула p(x) ¬p(x) r(x). Мы выбираем пункт r(x)
в качестве SOS, поэтому противоречие не может быть
получено. (b) Если SOS уже невыполнимо, то никакого противоречия не может быть
получено. Если нет, то необходимы шаги разрешения между предложениями из
SOS и (KB ¬Q)\SOS.

Machine Translated by Google

(c) Если нет дополнения к литералу L в предложении K, то литерал L останется в каждом предложении, полученном с помощью разрешения из предложения K. Таким образом, пустое предложение не может быть получено из K или его резольвенты, как и любой будущая решимость.

Упражнение 3.8 ¬Q KB (e = n)1 (n · x = n)2 (e · x = x)3 (¬a = b)4 Доказательство: Dem(1, 2) : (e · x = e)5 Tra,Sym(3, 5) : (x = e)6 Dem(4, 6) : (¬e = b)7 Dem(7, 6) : (¬e = e)8 ( )

Здесь «Dem» означает демодуляцию. Пункт номер 6 был получен путем применения транзитивности и симметрии равенства в пунктах 3 и 5.

Упражнение 3.9. Входные файлы LOP:

а) a; b<-. <- a,b. b;c<-. <-b, в. c; a<-. <-c, a.

(b) <- бреет(бородка,X), бреет(X,X). бреет(колючка,X); бреет(X,X) <-.

в) e = n. <- a = b. m(m(X,Y),Z)=m(X,m(Y,Z)). m(e,X) = X. m(n,X) = n.

11.4 Ограничения логики

Упражнение

4.1 (a) Правильно: возьмем полное доказательство для PL1. С его помощью мы находим доказательство каждой истинной формулы за конечное время. Для всех невыполнимых формул я поступаю следующим образом: я применяю исчисление к ¬φ и показываю, что ¬φ истинно. Таким образом, φ ложно. Таким образом, мы можем доказать каждую истинную формулу из PL1 и опровергнуть любую ложную формулу. К сожалению, этот процесс непригоден для выполнимых формул.

Упражнение 4.2

(a) Он бреется тогда и только тогда, когда он не бреется. (противоречие) (b) Множество всех множеств, которые не содержат самих себя. Он содержит себя тогда и только тогда, когда если он не содержит себя.

11.5 ПРОЛОГ

Упражнение 5.1 PROLOG сигнализирует о переполнении стека. Причина в поиске в глубину PROLOG, который всегда выбирает первый унифицируемый предикат во входном файле.
С рекурсивными предикатами, такими как равенство, это вызывает непрерывную рекурсию.

Machine Translated by Google

Упражнение
5.2. write_path([H1,H2|T]): -- write_path([H2|T]), write_move(H2,H1). путь_записи([_X]).

```
write_move(состояние(X,W,Z,K), состояние(Y,W,Z,K)) :-
        написать('Фермер из'), написать(X), написать('в'), написать(Y), nl.
write_move(состояние(X,X,Z,K), состояние(Y,Y,Z,K)) :-
        write('Фермер и волк из '), write(X), write(' в '), write(Y),nl.
write_move(состояние(X,W,X,K), состояние(Y,W,Y,K)) :-
        написать('Фермер и коза из'),написать(X),написать(' в '), написать(Y), nl.
write_move(состояние(X,W,Z,X), состояние(Y,W,Z,Y)) :-
        написать('Фермер и капуста из'),написать(X),написать(' в '), написать(Y), nl.
```

Упражнение

5.3 (a) Требуется вывести найденные решения. Унификация в фактическом плане(Цель, Цель,Путь,Путь). позаботится об этом. (b) Условия вхождения в предложения предиката safe перекрывают друг друга. Возврат, вызванный сбоем, приводит к выполнению второй или третьей альтернативы safe, где явно будет найдено то же самое решение. Сокращение в конце первых двух безопасных предложений решает проблему. В качестве альтернативы все безопасные состояния могут быть заданы явно, например, safe(state(left,left,left,right)).

Упражнение 5.5 ? - один (10). один
(0) :- записать (1). один(N):- N1
есть N-1, один(N1), один(N1).

Упражнение

5.6 (a) При n1 = |L1|, n2 = |L2| верно, что Tappend(n1, n2) = Θ (n1). (б) При n = |L| верно, что Tnrev(n) = Θ (n2). (в) При n = |L| верно, что Taccrev(n) = Θ (n).

Упражнение 5.7 Для деревьев с символами во внутренних узлах мы можем использовать, например, термины a(b,c) и a(b(e,f,g),c(h),d). Деревья без символов: дерево(b,c) или дерево(дерево(e,f,g),дерево(h),d).

Упражнение 5.8 Программы SWI-PROLOG:
(a) fib(0,1). выдумка (1,1).
fib(N,R):-N1 есть N-1, fib(N1,R1), N2 есть N-2, fib(N2,R2), R есть R1 + R2. (b) T (n) = Θ (2 (c) :- динамическое fib/2. fib(0,1).
fib(1,1). H).

fib(N,R):-N1 есть N-1, fib(N1,R1), N2 есть N-2, fib(N2,R2), R есть R1 + R2, asserta(fib(N,R)).

(d) Если факты от fib(0,1) до fib(k,f ib(k)) уже были вычислены, то для вызова fib(n,X) верно, что

$$T(n) = \begin{cases} \Theta(1) & \text{если } n \quad k, \\ \Theta(n \quad k), \text{ если } n > k. \end{cases}$$

Machine Translated by Google

(e) Поскольку после первых n вызовов предиката fib все дальнейшие вызовы имеют прямые
доступ к фактам.

Упражнение 5.9

(а) Решение здесь не раскрывается.

(б)

```
начинать :-
    fd_domain([Британец, Швед, Датчанин, Норвежец, Немецкий],1,5),
    fd_all_ Different([Британец, Швед, Датчанин, Норвежец, Немецкий]),
    fd_domain([Чай, Кофе, Вода, Пиво, Молоко],1,5),
    fd_all_ Different([Чай, Кофе, Вода, Пиво, Молоко]),
    fd_domain([Красный, Белый, Зеленый, Желтый, Синий],1,5),
    fd_all_ Different ([Красный, Белый, Зеленый, Желтый, Синий]),
    fd_domain([Собака, Птица, Кошка, Лошадь, Рыба],1,5),
    fd_all_ Different([Собака, Птица, Кошка, Лошадь, Рыба]),
    fd_domain([Паллмолл, Данхилл, Мальборо, Уинфилд, Ротманнс],1,5),
    fd_all_ Different ([Паллмолл, Данхилл, Мальборо, Уинфилд, Ротманнс]),
    fd_labeling([британец, швед, датчанин, норвежец, немец]),
    Британец #= Красный, % Британец живет в красном доме
    Швед #= Собака, % У шведа нет собаки
    Датчанин #= Чай, % Датчанин любит пить чай
    Зеленый #= Белый - 1, % Зеленый дом находится слева от белого дома
    Зеленый #= Кофе, % Хозяин теплицы пьет кофе
    Pallmall #= Птица, % У человека, который курит Pallmall, есть птица
    Молоко #= 3, % Мужчина в среднем доме пьет молоко
    Желтый #= Dunhill, % Владелец желтого дома курит Dunhill
    Норвежец #= 1, % Норвежец живет в первом доме
    dist(Marlboro,Cat)#= 1, % Курильщик Marlboro живет рядом с котом
    dist(Horse,Dunhill) #= 1, % Мужчина с лошадью живет рядом с курильщиком Dunhill
    Winfield #= Beer, % Курильщик Winfield любит пить пиво
    dist(Norwegian,Blue) #= 1, % Норвежец живет рядом с синим домом
    Немецкий #= Rothmanns, % Немец курит Rothmanns
    dist(Marlboro,Water)#=1, % Сосед курильщика Marlboro пьет воду.
    write([британец, швед, датчанин, норвежец, немец]), nl,
    написать([Собака, Птица, Кошка, Лошадь, Рыба]), nl.
```

11.6 Поиск, игры и решение проблем

Упражнение 6.1

а) На последнем уровне находится $b^{г}$ узлы. Все предыдущие уровни вместе имеют

$$Nb(\text{дмакс}) = \sum_{i=0}^{d-1} b^{i} = \frac{b^{d} - 1}{b - 1} \approx b^{d-1}$$

узлы, если b становится большим. Потому что $b^{d} / b^{d} - 1 = b$ примерно b раз
много узлов на последнем уровне, как и на всех остальных уровнях вместе взятых.

(b) Для непостоянного коэффициента ветвления это утверждение уже не верно, как показывает следующий
контрпример: дерево, сильно разветвляющееся до уровня be fore last, за которым следует уровень с
постоянным коэффициентом ветвления, равным 1. , точно
столько же узлов на последнем уровне, сколько на позапрошлом уровне.

Упражнение 6.2

(а) На рис. 6.3 на стр. 86 структура дерева на втором уровне с его восемью
узлы повторяются. Таким образом, мы можем вычислить средний коэффициент ветвления bm из $b_{м}^{2} = 8$ в $bm = \sqrt{8}$.

Machine Translated by Google

(б) Здесь вычисление не так просто, потому что корневой узел дерева ветвится сильнее всех остальных. Однако мы можем сказать, что внутри дерева коэффициент ветвления ровно на 1 меньше, чем он был бы без проверки цикла. Таким образом, bm 8 1 1,8.

Упражнение

6.3 (a) Для среднего коэффициента ветвления число листовых узлов фиксировано. Напротив, для эффективного фактора ветвления число узлов во всем дереве фиксировано.

(b) Поскольку количество всех узлов в дереве обычно лучше измеряет время вычислений для поиска всего дерева, чем количество конечных узлов. (c) Для большого b согласно (6.1) на стр. 87 имеем n b¯d+1 /b¯ = b¯d , что дает
b¯ = d н.

Упражнение

6.4 (a) 3-головоломка: 4! = 24 состояния, 8-головоломка: 9! = 362 880 состояний, 15-головоломка: 16! = 20 922 789 888 000 штатов.

(b) Переместив пустой квадрат 12 раз по часовой стрелке, мы снова достигнем исходного состояния и, таким образом, создадим циклическое подпространство с 12 состояниями.

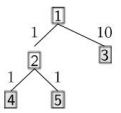
Упражнение 6.6 (a) Программа Mathematica:

```
0 BreadthFirstSearch[Node_, Goal_, Depth_] := Module[{i, NewNodes={}},
1 Для[i=1, i<= Длина[Узел], i++, 2
                  NewNodes = Join[ NewNodes, Successors[ Node[[i]] ] ];
3                 If[MemberQ[Successors[Node[[i]]], Goal],
4                         Возврат[{"Решение найдено", Глубина+1}], 0
5                   ]
6             ];
7 Если[Длина[НовыеУзлы] > 0, 8
                Поиск в ширину[Новые узлы, Цель, Глубина+1],
9               Возврат[{"Ошибка", Глубина}]
10          ]
11]
```

(b) Поскольку глубина пространства поиска не ограничена.

Упражнение

6.7 (a) При постоянной стоимости 1 стоимость всех путей на глубине d меньше стоимости всех путей на глубине d + 1. Поскольку все узлы на глубине d проверяются перед первым узлом на глубине d + 1, решение длины d гарантированно будет найдено раньше, чем решение длины d + 1. (b) Для соседнего дерева поиска
сначала генерируются узел номер 2 и узел решения номер 3 стоимости 10. Поиск прекращается. Узлы 4 и 5 со стоимостью пути 2 каждый не генерируются. Поэтому оптимальное решение не найдено.



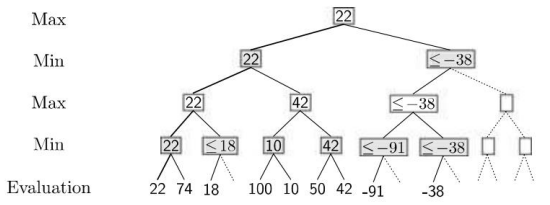
Machine Translated by Google

Упражнение 6.11 Поиск в глубину: новый узел должен иметь более низкий рейтинг, чем все открытые узлы. Поиск в ширину: новый узел должен иметь более высокий рейтинг, чем все открытые узлы.

Упражнение 6.12 Как допустимая эвристика, жена недооценивает расстояние до цели. Это может привести к тому, что они оба найдут кратчайший путь к цели, хотя и с большими усилиями. Однако это верно только в том случае, если дама всегда недооценивает расстояние.

Упражнение 6.14

(а)



11.7 Рассуждения с неопределенностью

Упражнение 7.1 |

1. P(Ω) = 2. Ом| = 1. |Ом|

P( ) = 1 P(Ω) = 0.

3. Для попарно исключающих событий A и B: P (A B) = |Ω| |A B| "=" |A|+|B| | = P (A)+ Ом|

П (Б).

4. Для двух дополнительных событий A и ¬A: P (A) + P (¬A) = P (A) + P (Ω |Ω A| | Ω| |A B| 5. P

A) = |A| | + (A "=" | = 1.

Ом| Б) Ом|

= |Ω| "=" |Ом| |A|+|B| |A B| = P (A) + P (B) - P (A B). |Ом| |Б| = П (B). |Ом|

6. Для A B: P (A) = |A| |

Ом|

7. Согласно определению 7.1 на стр. 115: A1 ··· An = Ω и |A1 ··· An| |Ом| н P (Аи) =

я=1

ни=1 |Ай | "=" н |Ай | "=" "=" |Ом| = 1. |

|Ом| я=1 |Ом| Ом|

Упражнение

7.2 (а)

$$P(y > 3 \mid \text{Класс} = \text{хорошо}) = 7/9 \; P(\text{Класс} = \text{хорошо}) = 9/13 \; P(y > 3) = 7/13,$$

$$P(\text{Класс} = \text{кишечник} \mid y > 3) = \frac{P(y > 3 \mid \text{Класс} = \text{хорошо}) \cdot P(\text{Класс} = \text{хорошо})}{P(y > 3)}$$

$$= \frac{7/9 \cdot 9/13}{1} = 7/13$$

Machine Translated by Google

9.3.4 Программа исправления правописания

В качестве приложения описанной ассоциативной памяти с бинарным правилом Хебба выберем программу, исправляющую ошибочные входные данные и отображающую их в сохраненные слова из словаря. Очевидно, что здесь потребуется автоассоциативная память. Однако, поскольку мы будем кодировать вектор запроса и целевой вектор по-разному, дело обстоит иначе. Для векторов запроса q мы выбираем парное кодирование. В алфавите из 26 символов имеется $26 \cdot 26 = 676$ упорядоченных пар букв. С 676 битами вектор запроса имеет по одному биту для каждой из возможных пар.

$$aa, ab, \ldots, az, ba, \ldots, bz, \ldots, za, \ldots, zz.$$

Если в слове встречается пара букв, то в соответствующее место будет вставлена единица. Например, для слова «генри» места для «га», «ан» и «нс» заполнены единицами. Для целевого вектора t зарезервировано 26 битов для каждой позиции в слове до максимальной длины (например, десять символов). Для i-й буквы алфавита на позиции j в слове устанавливается число битов $(j - 1) \cdot 26 + i$. Для слова «генри» установлены биты 8, 27, 66 и 97. Таким образом, для максимум 10 букв в слове целевой вектор имеет длину 260 бит.

Весовая матрица W, таким образом, имеет размер $676 \cdot 260$ битов = 199420 битов, что (9.11) на стр. 237 может хранить не более

$$n_{max} \approx 0{,}69 \frac{m_{in}}{130} - 676 + \frac{676 \cdot 260}{130 \cdot (m+n)}$$

слова. Имея 72 имени, мы экономим примерно половину этого количества и тестируем систему. Сохраненные имена и выходные данные программы для нескольких примеров ввода приведены на рис. 9.12 на стр. 239. Порог всегда инициализируется числом битов в закодированном запросе. Здесь это количество пар букв, следовательно, длина слова минус один. Затем его ступенчато уменьшают до двух. Мы могли бы дополнительно автоматизировать выбор порога, сравнивая со словарем каждую попытку порога и выводя найденное слово, когда сравнение завершается успешно.

Интересна реакция на двусмысленные входные данные «andr» и «johanne». В обоих случаях сеть создает сочетание двух подходящих сохраненных слов. Мы видим здесь важную силу нейронных сетей. Они способны создавать ассоциации с похожими объектами без явной метрики сходства. Однако, как и в случае с эвристическим поиском и принятием решений человеком, нет гарантии «правильного» решения.

Поскольку обучающие данные должны быть доступны в виде пар ввода-вывода для всех нейронных моделей, которые были введены до сих пор, мы имеем дело с обучением с учителем, что также имеет место для сетей, представленных в следующих разделах.

Machine Translated by Google

Сохраненные слова:

Агата, Агнес, Александр, Андреас, Андре, Анна, Аннемари, Аструд, Август, Бернхард, Бьорн, Катрин, Кристиан, Кристоф, Коринна, Корадо, Дитер, Элизабет, Эльвира, Эрдмут, Эрнст, Эвелин, Фабрицио, Франк, Франц, Джеффри, Георг, Герхард, Ханнелоре, Гарри, Герберт, Ингильт, Ирмгард, Ян, Йоханнес, Джонни, Юрген, Карин, Клаус, Людвиг, Луиза, Манфред, Мария, Марк, Маркус, Марлин, Мартин, Маттиас, Норберт, Отто, патриция, питер, филип, бросить, рейнхольд, рената, роберт, робин, сабина, себастьян, стефан, стефан, сильви, ульрих, ульрике, уте, уве, вернер, вольфганг, ксавьер

Ассоциации программы

шаблон ввода: Гарри

порог: 4, ответ: порог Гарри: 3, ответ: порог Гарри: 2, ответ: horryrrde

---

шаблон ввода: уте

порог: 2, ответ: уте

шаблон ввода: герхар

порог: 5, ответ: gerhard порог: 4, ответ: gerrarn
порог: 3, ответ: jerrrrd порог: 2, ответ: jurtyrrde

шаблон ввода: эгрхард

порог: 6, ответ: порог: 5, ответ:
порог: 4, ответ: gerhard порог: 3,
ответ: gernhrrd порог: 2, ответ: irrryrrde

шаблон ввода: andr

порог: 3, ответ: andrees порог: 2, ответ: anexenser

шаблон ввода: Андреес

порог: 6, ответ: порог: 5, ответ: порог
andree: 4, ответ: порог andrees: 3, ответ: mnnrens
порог: 2, ответ: morxsnssr

шаблон ввода: Джоанна

порог: 6, ответ: johnnnes порог: 5, ответ: johnnns
порог: 4, ответ: jornnnrse порог: 3, ответ: sorrnyrse
порог: 2, ответ: wtrrsyrse

шаблон ввода: johnnnes

порог: 6, ответ: joh порог: 5, ответ: johnnnes
порог: 4, ответ: johnnyes порог: 3, ответ: jonnnyes
порог: 2, ответ: jornsyrse

шаблон ввода: Джонни

порог: 7, ответ: порог: 6, ответ: joh
порог: 5, ответ: johnnyes порог: 4, ответ:
johnnyes порог: 3, ответ: johnnyes порог: 2, ответ:
johnnyes

Рис. 9.12. Применение программы исправления правописания к различным усвоенным или ошибочным данным. Правильные входы находятся с максимальным (то есть первым предпринятым) порогом. Для ошибочных вводов порог должен быть снижен для корректной ассоциации.

Machine Translated by Google

9.4 Линейные сети с минимальными ошибками

Правило Хебба, используемое в нейронных моделях, представленных до сих пор, работает с ассоциациями между соседними нейронами. В ассоциативной памяти это используется для изучения сопоставления векторов запросов с целями. Во многих случаях это работает очень хорошо, особенно когда векторы запросов линейно независимы. Если это условие не выполняется, например, когда доступно слишком много обучающих данных, возникает вопрос: как найти оптимальную весовую матрицу? Оптимальный означает, что он минимизирует среднюю ошибку.

Мы люди, способны учиться на ошибках. Правило Хебба не дает такой возможности. Алгоритм обратного распространения, описанный ниже, использует элегантное решение, известное из аппроксимации функций, для изменения весов таким образом, чтобы минимизировать ошибку обучающих данных.

Пусть N пар обучающих векторов

$$T = \{(q^{1}, t^{1}), \ldots, (q^{N}, t^{N})\}$$

быть задано с $q^{p} \in [0, 1]^{R}$ и $t^{p} \in [0, 1]^{M}$. Мы ищем функцию $f : [0, 1]^{R} \to [0, 1]^{M}$, что минимизирует квадрат ошибки

$$\sum_{p=1}^{N} (f(q^{p}) - t^{p})^{2}$$

на данные. Предположим сначала, что данные не содержат противоречий. То есть в обучающих данных нет вектора запроса, которому должны быть сопоставлены с двумя разными целями. В этом случае нетрудно найти функцию, минимизирующую квадрат ошибки. На самом деле существует бесконечно много функций, которые делают ошибку равной нулю. Определим функцию

$$f(q) = 0, \text{ если } q \notin \{q^{1}, \ldots, q^{N}\}$$

и

$$f(q^{p}) = t^{p} \quad p \in \{1, \ldots, N\}.$$

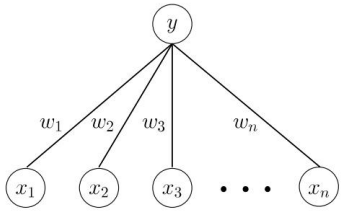
Это функция, которая даже обнуляет ошибку обучающих данных. Что еще мы могли бы хотеть? Почему нас не устраивает эта функция?

Ответ: потому что мы хотим построить интеллектуальную систему! Интеллектуальный означает, среди прочего, что изученная функция может хорошо обобщать обучающие данные на новые, неизвестные данные из того же репрезентативного набора данных. Другими словами, это означает: мы не хотим переобучения данных путем запоминания. Чего же тогда мы действительно хотим?

Нам нужна плавная функция, которая «выравнивает» пространство между точками. Разумными требованиями были бы непрерывность и возможность брать несколько производных. Поскольку даже при этих условиях остается бесконечно много функций, которые делают ошибку равной нулю, мы должны еще больше ограничить этот класс функций.

Machine Translated by Google

Рис. 9.13 Двухслойная сеть с выходом нейрон



9.4.1 Метод наименьших квадратов

Самый простой выбор — это линейное отображение. Начнем с двухслойной сети (рис. 9.13), в которой единственный нейрон у второго слоя вычисляет свою активацию, используя

$$y = e^{\sum_{i=1}^{n} w_i x_i}$$

с f(x) = x. Тот факт, что мы рассматриваем здесь только выходные нейроны, не представляет собой реального ограничения, поскольку у двухслойную сеть с двумя или более выходными нейронами всегда можно разделить на независимые сети с идентичными входными нейронами для каждого исходного выходного нейрона. Все веса подсетей независимы. Использование сигмовидной функции вместо линейной активации здесь не дает никаких преимуществ, поскольку у сигмоидальная функция является строго монотонно возрастающей и не меняет отношения порядка между различными выходными значениями.

Искомым является вектор w, который минимизирует квадрат ошибки

$$E(w) = \sum_{p=1}^{N} (w^T x^p - t^p)^2 = \sum_{p=1}^{N} \left(\sum_{i=1}^{n} w_i x_i^p - t^p \right)^2 .$$

В качестве необходимого условия минимума этой функции ошибок все частные производные должны быть равнынулю. Таким образом, мы требуем, чтобы для $j = 1,\ldots,n$:

$$\frac{\partial E}{\partial w_j} = 2 \sum_{p=1}^{N} \left(\sum_{i=1}^{n} w_i x_i^p - t^p \right) x_j^p = 0, \quad j$$

Умножение этого дает

$$\sum_{p=1}^{N} \left(\sum_{i=1}^{n} w_i x_i^p x_j^p - t^p x_j^p \right) = 0,$$

Machine Translated by Google

и замена сумм приводит к линейной системе уравнений

$$\sum_{i=1}^{n}\Big(\sum_{p=1}^{N} q_i^p q_j^p\Big) c_i = \sum_{p=1}^{N} t^p q_j^p,$$

который с

$$A_{ij} = \sum_{p=1}^{N} q_i^p q_j^p \quad \text{и} \quad b_j = \sum_{p=1}^{N} t^p q_j^p \tag{9.12}$$

можно записать в виде матричного уравнения

$$Ac = b. \tag{9.13}$$

Это так называемые нормальные уравнения всегда имеют хотя бы одно решение, а если A обратимо, то ровно одно. Кроме того, матрица A является положительно определенной, из чего следует, что найденное решение в единственном случае является глобальным минимумом. Этот алгоритм известен как метод наименьших квадратов.

Время расчета для настройки матрицы A растет с ростом $\Theta(N \cdot n^2)$ и время решения системы уравнений $O(n^3)$. Этот метод может быть очень просто расширен для включения нескольких выходных нейронов, потому что, как уже упоминалось, для двухслойных сетей с прямой связью выходные нейроны зависят друг от друга.

9.4.2 Применение к данным по аппендициту

В качестве приложения мы теперь определяем линейную оценку для диагностики аппендицита. Из данных проекта LEXMED, знакомых по разд. 8.4.5, мы используем метод наименьших квадратов для определения линейного отображения от симптомов к непрерывным переменным класса AppScore со значениями в интервале [0, 1] и получаем линейную комбинацию

$$\begin{aligned}\text{AppScore} = {} & 0{,}00085\,\text{Возраст} - 0{,}125\,\text{Пол} + 0{,}025\,\text{P1Q} + 0{,}035\,\text{P2Q} - 0{,}021\,\text{P3Q} \\ & - 0{,}025\,\text{P4Q} + 0{,}12\,\text{TensLoc} + 0{,}031\,\text{TensGlo} + 0{,}13\,\text{Losl} \\ & + 0{,}081\,\text{Conv} + 0{,}0034\,\text{RectS} + 0{,}0027\,\text{TAxi} + 0{,}0031\,\text{TRec} \\ & + 0{,}000021\,\text{Лейко} - 0{,}11\,\text{Диаб} - 1{,}83.\end{aligned}$$

Эта функция возвращает непрерывные переменные для AppScore, хотя фактическая переменная двоичного класса App принимает только значения 0 и 1. Таким образом, мы должны выбрать пороговое значение, как в случае с персептроном. Ошибка классификации оценки как функция порога показана на рис. 9.14 на стр. 243 для обучающих данных и тестовых данных. Мы сразу видим, что обе кривые почти одинаковы и имеют

Machine Translated by Google

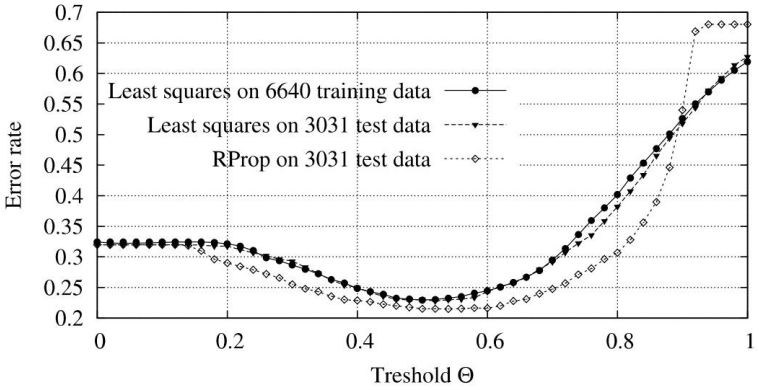


Рис. 9.14. Ошибка метода наименьших квадратов для обучающих и тестовых данных

их минимум при Θ = 0,5. По небольшой разнице двух кривых мы видим, что переобучение не является проблемой для этого метода, потому что модель очень хорошо обобщает тестовые данные.

Также на рисунке показан результат для нелинейной трехслойной сети RProp (раздел 9.5) с несколько меньшей ошибкой для пороговых значений от 0,2 до 0,9. Для практического применения полученной оценки и правильного определения порога Θ важно не только смотреть на ошибку, но и различать ее по типу (а именно ложноположительный и ложноотрицательный), как это делается в приложении LEXMED. на рис. 7.10 на стр. 142. На кривой ROC, показанной там, также показана рассчитанная здесь оценка. Мы видим, что простая линейная модель явно уступает системе LEXMED. Очевидно, что линейные приближения недостаточно эффективны для многих сложных приложений.

9.4.3 Дельта-правило

Метод наименьших квадратов, как и обучение персептрону и дереву решений, представляет собой так называемый алгоритм пакетного обучения, в отличие от пошагового обучения. При пакетном обучении все обучающие данные должны быть изучены за один прогон. Если добавляются новые обучающие данные, их нельзя просто выучить в дополнение к тому, что уже есть. Весь процесс обучения необходимо повторить с увеличенным набором. Эта проблема решается алгоритмами инкрементного обучения, которые могут адаптировать изученную модель к каждому дополнительному новому примеру. В алгоритмах, которые мы рассмотрим в следующем обсуждении, мы будем аддитивно обновлять веса для каждого нового обучающего примера по правилу

$$w_j = w_j + \Delta w_j .$$

Чтобы получить инкрементный вариант метода наименьших квадратов, мы пересматриваем вычисленные выше n частных производных функции ошибок

Machine Translated by Google

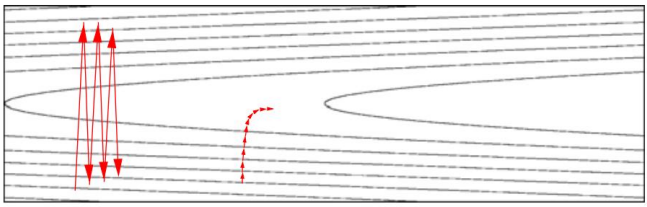


Рис. 9.15. Градиентный спуск при большом η (слева) и очень малом η (справа) в долину, пологого спускающуюся вправо. При большом η возникают колебания вокруг долины. Напротив, при слишком малом η сходимость в плоской долине происходит очень медленно.

$$\frac{E}{wj} = 2 \sum_{p=1}^{H} \left(\sum_{i=1}^{н} wiq_i^p - t_p \right) x_j^p.$$

Градиент

$$Э = \left(\frac{E}{w1}, \ldots, \frac{E}{вн} \right)$$

как вектор всех частных производных функции ошибок указывает направление наибольшего возрастания функции ошибок в n-мерном пространстве весов. Поэтому при поиске минимума мы будем следовать направлению отрицательного градиента. В качестве формулы заменывесов получаем

$$\eta wj = -\frac{E}{2\,wj} = \eta \sum_{p=1}^{H} \left(\sum_{i=1}^{н} wiq_i^p - t_p \right) q_j^p,$$

где скорость обучения η — свободно выбираемая положительная константа. Большое η ускоряет сходимость, но в то же время повышает риск колебаний вокруг минимумов или плоских долин. Поэтому оптимальный выбор η — непростая задача (см. рис. 9.15).
Большое η, например, η = 1, часто используется для начала, а затем постепенно уменьшается.

Заменив активацию

$$y^п = \sum_{i=1}^{н} вич_i^п$$

выходного нейрона для прикладного обучающего примера ^{п}, формула упрощена и получаем дельта-правило

$$wj = \eta \sum_{p=1}^{H} (t_p - y_p) q_{Дж.}^p$$

Machine Translated by Google

Рис. 9.16. Обучение двухслойной линейной сети с помощью дельта-правила. Обратите внимание, что изменения веса всегда происходят после применения всех тренировочных данных.

DELTALEARNING (Примерыобучения , η)

Инициализировать все веса wj случайным образом

Повторите ш= 0

Для всех (q p ,tp) TrainingExamples

Рассчитать вывод сети yw = w ^{n} = wpq p

+ η(tp y p)q p w = w + w

Пока w не сходится

DELTALEARNINGINCREMENTAL(TrainingExamples, η)

Инициализировать все веса wj случайным образом

Повторить

Для всех (q p ,tp) TrainingExamples

Вычислить вывод сети yw = w + ^{n} = wpq p

η(tp y p)q p

Пока w не сходится

Рис. 9.17 Инкрементальный вариант дельта-правила

Таким образом, для каждого обучающего примера разница между целевым t p и фактическим p шаблонов сети вычисляется для заданного входа q. После суммирования по выходу у всех веса затем изменяются пропорционально сумме. Этот алгоритм показан на рис. 9.16.

Мывидим, что алгоритм все еще является инкрементным, потому что изменения веса происходят только после того, как все обучающие примерыбыли применены один раз. Мыможем исправить этот недостаток, напрямую изменяя веса (инкрементальный градиент) после каждого обучающего примера (рис. 9.17), что, строго говоря, больше не является корректной реализацией дельта-правила.

9.4.4 Сравнение с персептроном

Правило обучения для персептронов, введенное в разд. 8.2.1, метод наименьших квадратов и дельта-правило могут использоваться для генерации линейных функций из данных. Однако для персептронов, в отличие от других методов, классификатор для линейно разделимых классов изучается посредством порогового решения . Однако два других метода генерируют линейную аппроксимацию данных. Как показано в разд. 9.4.2, классификатор может быть сгенерирован из линейного отображения , при желании, путем применения пороговой функции.

Machine Translated by Google

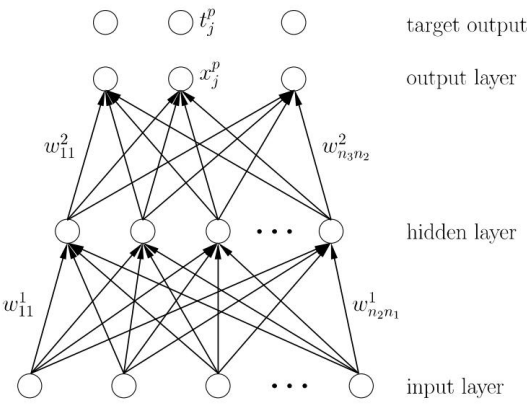


Рис. 9.18. Трехслойная сеть обратного распространения с n1 нейронами в первом, n2 нейронами во втором и n3 нейронами в третьем слое

Персептрон и дельта-правило представляют собой итерационные алгоритмы, для которых время до сходимости сильно зависит от данных. В случае линейно разделимых данных для персептрона можно найти верхний предел числа шагов итерации. Для дельта-правила, напротив, существует только гарантия неограниченной асимптотической сходимости [HKP91].

Для метода наименьших квадратов обучение состоит в построении и решении линейной системы уравнений для вектора весов. Таким образом, существует жесткое ограничение на время вычислений. Из-за этого метод наименьших квадратов всегда предпочтительнее, когда не требуется постепенное обучение.

9.5 Алгоритм обратного распространения

Теперь с помощью алгоритма обратного распространения мы представляем наиболее часто используемую нейронную модель. Причиной его широкого использования является его универсальная универсальность для задач произвольной аппроксимации. Алгоритм исходит непосредственно из правила возрастания дельты. В отличие от дельта-правила, оно применяет нелинейную сигмовидную функцию к взвешенной сумме входных данных в качестве функции активации. Кроме того, сеть обратного распространения может иметь более двух слоев нейронов. Алгоритм стал известен благодаря статье [RHR86] в легендарном сборнике PDP [RM86].

На рис. 9.18 типичная сеть обратного распространения с входным слоем, скрытым выходным слой, и отображается выходной слой. Поскольку у текущее выходное значение $_{\text{П}}$ нейрона слоя x_j сравнивается с целевым выходным значением t, они рисуются параллельно $_{\text{П}}$ j другу другу. Все нейроны, кроме входных, вычисляют свое текущее значение x_j по правилу

$$x_j = f\left(\sum_{i=1}^{н} w_{ji} x_i\right) \tag{9.14}$$

Machine Translated by Google

где n — количество нейронов в предыдущем слое. Мыиспользуемсигмовидную функцию

$$f(x) = 1 \frac{1}{+ e\text{-}x}.$$

По аналогии с правилом приращения дельтывеса изменяются пропорционально отрицательному градиенту квадратичной функции ошибки, суммированной по выходным нейронам.

$$Ep(w) = 2 \frac{1}{k \text{ выход}} (т_к п - x_к^p)^2$$

для тренировочного шаблона p:

$$pwj\,i = \eta \frac{Эп}{wj\,я}.$$

Для вывода правила обучения приведенное выше выражение заменяется на Ep. В выражении xk заменяется на (9.14) на стр. 246. В уравнении выходыхи нейронов следующего, более глубокого слоя встречаются рекурсивно и т. д. Примногократном применении цепного правила (см. [RHR86] или [Zel94]) получаемправило обучения обратного распространения

$$pwj\,ij = \eta \delta p \; ик с пи,$$

с

$$\delta_j^p = \begin{cases} ик с п (1 \quad x^п)(т п - x^p) & \text{если j — выходной нейрон,} \\ джп x (1 \quad x_j^{jp}) \; jp \; \delta ik wkj, & \text{если j — скрытый нейрон,} \end{cases}$$

который также обозначается обобщенным дельта-правилом. Для всех нейронов формула изменения веса wj i от нейрона i к нейрону j (см. рис. 9.19 на стр. 248) содержит, как и правило Хебба, член ηx $\text{стр}x_{\text{я}}$ Джк. Новый коэффициент $п$) создает симмуникатор (1 x j try, отсутствующий в правиле Хебба, между активациями 0 и 1 нейрона. Для выходных нейронов коэффициент (tp j . $- x_{\text{дж}}^p$) заботится об изменении веса пропорционален ошибке. Для скрытых нейронов значение δ j $п$ нейрона j вычисляется рекурсивно от всех изменений δ $п_к$ нейронов следующего более высокого уровня.

Все выполнение процесса обучения показано на рис. 9.20 на стр. 248.

После расчета выхода сети (прямого распространения) для обучающего примера вычисляется ошибка аппроксимации. Затем это используется во время обратного распространения для изменения весов в обратном направлении от слоя к слою. Затем весь процесс применяется ко всем обучающим примерам и повторяется до тех пор, пока веса не перестанут изменяться или не будет достигнуто ограничение по времени.

Machine Translated by Google

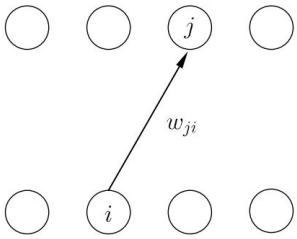


Рис. 9.19 Обозначение нейронов и весов для применения правила обратного распространения

ОБРАТНОЕ РАСПРОСТРАНЕНИЕ (Примерыобучения , η)
Инициализировать все веса wj случайными

значениями. Повторить для всех (q p ,t p)
для входного слоя 1. Применить TrainingExamples p
вектор запроса q 2. Прямое
распространение: Для всех слоев, начиная с первого
скрытого слоя вверх Для каждого нейрона слой
Рассчитать активацию xj = f (ni=1 wj ixi)
3. Расчет квадратичной ошибки Ep(w)
4. Обратное распространение: Для
всех уровней весов с последнего вниз Для каждого веса
wj i = wj i + ηδp
wj я
дж
x п
я

Пока w не сойдется или не будет достигнут предел времени

Рис. 9.20 Алгоритм обратного распространения

Если мыпостроимсеть хотя быс одним скрытым слоем, можно изучить нелинейные отображения . Без скрытых слоев выходные нейроныне мощнее линейного нейрона, несмотря на сигмовидную функцию . Причина этого в том, что сигмовидная функция строго монотонна. То же верно и для многоуровневых сетей, которые используют только линейную функцию в качестве функции активации, например, функцию идентификации.

Это связано с тем, что цепные выполнения линейных отображений являются линейными в совокупности.

Как и в случае с персептроном, класс функций, которые могут быть представлены также увеличиваются , если мыиспользуем переменную сигмовидную функцию

$$f(x) = 1 + \frac{1}{e(x \quad \Theta)}.$$

с порогом Θ. Это реализуется аналогично тому, как показано в разд. 8.2, в котором во входной слой и в каждый скрытый слой вставляется нейрон, активация которого всегда имеет значение единица и который связан с нейронами следующего по высоте уровня . Веса этих связей усваиваются нормально и представляют собой порог Θ нейронов-преемников.

Machine Translated by Google

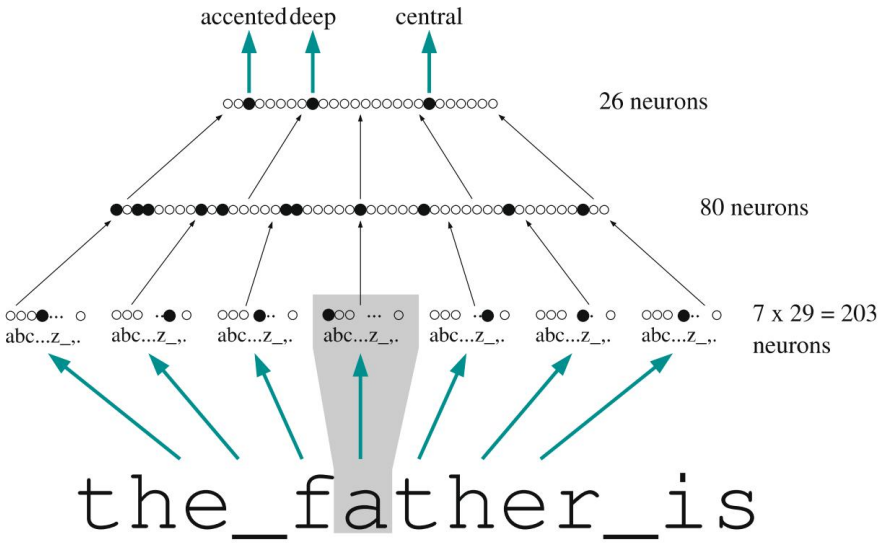


Рис. 9.21. Сеть NETtalk сопоставляет текст с его атрибутами произношения.

9.5.1 NETtalk: сеть учится говорить

Сейновски и Розенберг в 1986 г. весьма впечатляюще продемонстрировали, насколько способно обратное распространение [SR86]. Они построили систему, способную понятно читать вслух английский текст из текстового файла. Архитектура сети, показанная на рис. 9.21, состоит из входного слоя с 7 × 29 = 203 нейронами в котором закодированы текущая буква и три предыдущие буквы, а также три последующие буквы. Для каждой из этих семи букв зарезервировано 29 нейронов. символы «a...z ,.». Вход отображается на 26 выходных нейронов более 80. скрытые нейроны, каждый из которых соответствует определенной фонеме. Например, «а» в слове «отец» произносится глубоко, с ударением и посередине. Сеть прошла обучение с 1000 слов, которые применялись случайным образом одно за другим буква за буквой. Для каждой буквы вручную задавался целевой вывод по ее интонации. Чтобы преобразовать интонационные признаки в реальные звуки, часть системы синтеза речи Использовался DECtalk. Благодаря полной взаимосвязанности сеть содержит в общей сложности 203 × 80 + 80 × 26 = 18320 гирь.

Обучение системы проводилось на тренажере VAX 780 около 50 циклов. над всеми словами. Таким образом, примерно при 5 символах на слово в среднем около 5·50·1000 = Потребовалось 250 000 итераций алгоритма обратного распространения ошибки. При скорости около один символ в секунду, это означает примерно 69 часов вычислительного времени. разработчики наблюдали многие свойства системы, которые очень похожи на человеческие обучение. Сначала система может говорить только неразборчиво или использовать простые слова. С течением времени он продолжал улучшаться и, наконец, достиг 95% правильности произнесения букв.

Machine Translated by Google

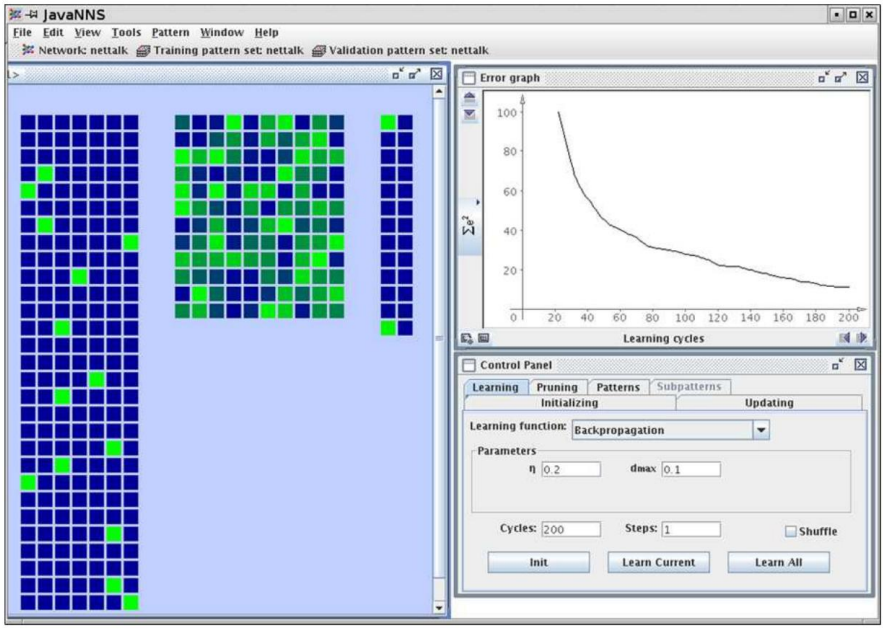


Рис. 9.22 Сеть NETtalk в JNNS. В левом окне слева направо мы видим 7 · 29 входных нейронов, 80 скрытых и 26 выходных нейронов. Из-за их большого количества веса в сети опущены. Вверху справа — кривая обучения, показывающая изменение квадрата ошибки с течением времени.

Для визуальных экспериментов с нейронными сетями рекомендуется Java Neural Network Simulator JNNS [Zel94]. Сеть NETtalk, загруженная и обученная с помощью JNNS, показана на рис. 9.22.

9.5.2 Изучение эвристики для средств доказательства теорем

В гл. В разделе 6 обсуждались алгоритмы эвристического поиска, такие как A -алгоритм и IDA -алгоритм. Чтобы добиться значительного сокращения пространства поиска, для реализации этих алгоритмов необходима хорошая эвристика. В разд. 4.1 была продемонстрирована проблема взрывающегося пространства поиска во время поиска доказательства доказательством теоремы. Эта проблема вызвана большим количеством возможных шагов вывода на каждом шаге.

Теперь мы попытаемся создать модули управления эвристическим доказательством, которые оценивают различные альтернативы для следующего шага, а затем выбирают альтернативу с лучшим рейтингом. В случае разрешения рейтинг доступных предложений может быть выполнен с помощью функции, которая на основе определенных атрибутов предложений, таких как количество литералов, сложность терминов и т. д., вычисляет значение для каждой пары разрешений. способные оговорки. В [ESS89, SE90] такая эвристика была изучена для средства доказательства теорем SETHEO с использованием обратного распространения ошибки. При изучении эвристики ат

Machine Translated by Google

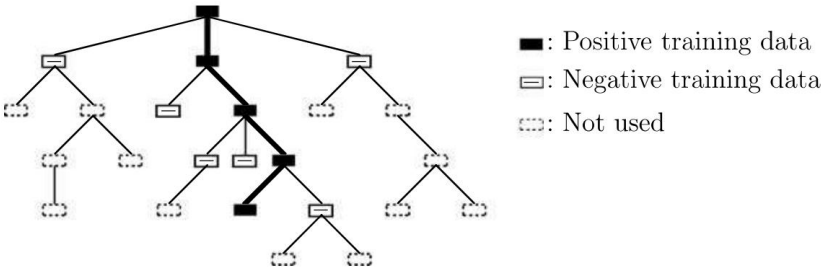


Рис. 9.23. Дерево поиска с успешным путем, узлы которого оценены как положительные, и отрицательно оцененными неуспешными ветвями

текущее предложение было сохранено в качестве обучающих данных для положительного класса для каждого найденного доказательства в каждом отделении во время поиска. На всех ответвления х, которые не привели к доказательству, атрибуты текущего предложения были сохранены как обучающие данные для отрицательного класса. Рисуется такое дерево с успешным путем и соответствующими рейтингами узлов. на рис. 9.23.

На этих данных была обучена сеть обратного распространения , а затем использована для оценки предложения в доказывающем. Для каждого предложения вычисляли, нормализовали и кодировали во входном нейроне 16 числовых признаков. Сеть обучалась на 25 скрытых нейронов и один выходной нейрон для класса (положительный/отрицательный).

Было показано, что количество попыток вывода может быть уменьшено во много раз. порядков для сложных задач , что в итоге сокращает время вычислений с часов до секунд. Тем самым стало возможным доказать теоремы которые без эвристики были недостаемы

9.5.3 Проблемы и улучшения

Обратному распространению уже 25 лет , и оно зарекомендовало себя в различных приложения х. например, в распознавании образов и в робототехнике. Однако его применение иногда проблематично. Особенно когда в сети многотысячные веса и необходимо изучить много обучающих данных, возникают две проблемы

Сеть часто сходится к локальным минимумам функции ошибок . Более того, обратное распространение часто сходится очень медленно. Это означает , что многие итерации в течение нужны все тренировочные шаблоны Для решения этих проблем было предложено множество усовершенствований. Как упоминалось в разд. 9.4.3, колебаний можно избежать , медленно уменьшая скорость обучения η, как показано на рис. 9.15 на странице 244.

Другой метод уменьшения колебаний - использование члена импульса, в то время как обновление весов, что гарантирует , что направление градиентного спуска не слишком резко меняется от одного шага к другому. Вот текущий вес к изменению pwj i(t) в момент времени t добавляется еще одна часть изменения pwj i(t 1) с предыдущего шага. Затем правило обучения меняется на

$$pwj\ i(t) = \eta\delta p \quad \text{Ик с} \quad \text{П} \quad + \gamma pwj\ i\ (t \quad 1)$$

Machine Translated by Google

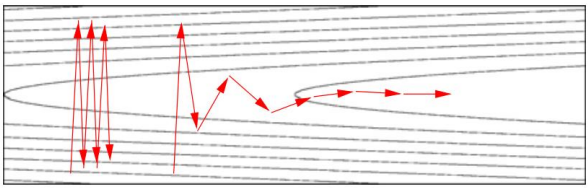


Рис. 9.24 Резкое направление изменения сглаживаются использование импульса срок. Итерация без член импульса (слева) в сравнение с итерацией с термин импульса (справа)

с параметром у между нулем и единицей. Это изображено в двухмерном пример на рис. 9.24.

Другая идея состоит в том, чтобы минимизировать линейную функцию ошибки вместо квадратичной. функция, сводящая проблему медленной сходимости к плоским долинам.

Градиентный спуск при обратном распространении в конечном счете основан на линейной аппроксимации функции ошибок. Алгоритм Quickprop, который использует квадратичное приближение к функции ошибок и, таким образом, обеспечивает более быструю сходимость, был создан Скотт Фалманн.

Путем умной унификации упомянутых улучшений и других эвристических приемов Мартин Ридмиллер добился дальнейшей оптимизации с помощью алгоритма RProp [RB93]. RProp заменил обратное распространение изменяется новым состоянием алгоритма аппроксимации нейронной сети с прямой связью искусства. В разд. 8.8 мы применили RProp к классификации данных аппендицита и добился ошибки, которая примерно того же размера, что и обученное дерево решений.

9.6 Машины опорных векторов

Нейронные сети с прямой связью только с одним слоем весов являются линейными. Линейность приводит к простым сетям и быстрому обучению с гарантированной сходимостью. Кроме того, опасность переобучения для линейных моделей невелика. Для многих приложений однако линейные модели недостаточно сильны, например, потому, что соответствующие классыне являются линейно разделимыми. Здесь используются многослойные сети, такие как обратное распространение, в результате чего локальные минимумы сходимость проблемы и может произойти переобучение.

Многообещающий подход, объединяющий преимущества линейных и нелинейных моделей, следует теории машин опорных векторов (SVM), которую мы будем грубо наметить, используя задачу двух классов.4

В случае двух линейно разделимых классов легко найти разделяющую гипер плоскости, например, с правилом обучения персептрона. Однако, как правило, бесконечно много таких плоскостей, как в двумерном примере на рис. 9.25 на стр. 253. Мы ищем плоскость, которая имеет наибольшее минимальное расстояние до обоих классы Эта плоскость обычно однозначно определяется несколькими точками в границной области. Эти точки, так называемые опорные векторы, находятся на одинаковом расстоянии от разделяющей поверхности.

4Машины опорных векторов не являются нейронными сетями. Однако из-за их исторического развития и математической связи с линейными сетями они обсуждаются здесь.

Machine Translated by Google

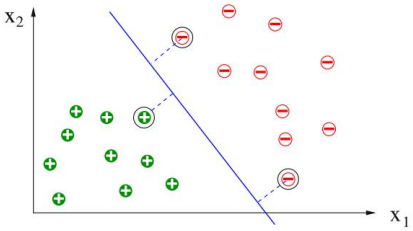


Рис. 9.25 Два класса с максимально разделяющей линией. Обведенные точки — опорные векторы

линия . Для нахождения опорных векторов существует эффективный алгоритм оптимизации. Интересно, что оптимальная разделяющая гиперплоскость определяется несколькими параметрами, а именно опорными векторами. Таким образом, опасность переоснащения мала.

Машиныопорных векторов применяют этот алгоритм к нелинейно разделимым задачам в двухэтапном процессе: на первом этапе к данным применяется нелинейное преобразование с тем свойством, что преобразованные данные являются линейно разделимыми. На втором этапе определяются опорные векторыв преобразованном пространстве.

Первый шаг очень интересен, но не совсем прост . На самом деле всегда можно сделать классы линейно разделимыми путем преобразования векторного пространства, если данные не содержат противоречий5. Такое разделение может быть достигнуто, например, введением нового (n + 1)-го измерения и Определение

$$x_{n+1} = \begin{cases} 1, \text{если } x \quad \text{класс 1,} \\ 0, \text{если } x \quad \text{класс 0.} \end{cases}$$

Однако эта формула не очень помогает , поскольку она неприменима к новым точкам неизвестного класса, подлежащим классификации. Таким образом, нам необходимо общее преобразование, максимально независимое от текущих данных. Можно показать , что такие общие преобразования существуют даже для границ деления классов произвольной формыв исходном векторном пространстве. В преобразованном пространстве данные становятся линейно разделимыми. Однако количество измерений нового векторного пространства растет экспоненциально с количеством измерений исходного векторного пространства.
Однако большое количество новых измерений не столь проблематично, поскольку при использовании опорных векторов разделяющая плоскость , как было сказано выше, определяется лишь несколькими параметрами.

Центральное нелинейное преобразование векторного пространства называется ядром, из-за чего машиныопорных векторов также известны как методыядра. Первоначальная теория SVM, разработанная для задач классификации, была расширена и теперь может использоваться и в задачах регрессии.

Используемая здесь математика очень интересна, но слишком обширна для первоначального введения . Чтобыглубже погрузиться в эту многообещающую молодую отрасль машинного обучения , мыотсылаем читателя к [SS02, Alp04] и [Bur98].

5 Точка данных противоречива, если она принадлежит обоимклассам.

Machine Translated by Google

9.7 Приложения

Помимо приведенных здесь примеров, существует бесчисленное множество приложений для работы с нейронными сетями во всех отраслях промышленности. Очень важной областью является распознавание образов во всех его формах, будь то анализ фотографий для распознавания людей или лиц, распознавание кожи кожи рыб по изображениям сонара, распознавание и классификация военной техники по скан радаров и многое другое. Но нейронные сети также можно научить распознавать устную речь и рукописный текст.

Нейронные сети используются не только для распознавания объектов и сцен. Их также обучают управлять простыми роботами с помощью данных датчиков, а также эвристически управлять поиском в компьютерах для игры в нарды и шахматы. Особый интерес в играх представляют техники обучения с подкреплением, разд. 10.8, например с нейронными сетями.

Нейронные сети уже давно, помимо статистических методов, успешно используются для прогнозирования фондовых рынков и оценки кредитоспособности клиентов банков.

Для многих из этих приложений также могут использоваться другие алгоритмы машинного обучения. Однако на сегодняшний день нейронные сети оказались более успешными и популярными, чем все другие алгоритмы машинного обучения. Однако историческое преобладание нейронных сетей постепенно отодвигается в пользу других методов из-за большого коммерческого успеха интеллектуального анализа данных и машин опорных векторов

9.8 Резюме и перспективы

С персептроном, дельта-правилом и обратным распространением мы представили самый важный класс нейронных сетей и их отношение к оценкам наивному Байесу, с одной стороны, но также и к методу наименьших квадратов. Затем мы увидели увлекательные сети Хопфилда, вдохновленные биологическими моделями, которыми, однако, трудно управлять на практике из-за их сложной динамики. На практике большее значение имеют различные модели ассоциативной памяти.

Во всех нейронных моделях информация хранится в распределении по множеству весов. Из-за этого отмирание нескольких нейронов в мозгу не оказывает заметного влияния на функцию мозга. Благодаря распределенному хранению данных сеть устойчива к небольшим сбоям. Это также связано со способностью распознавать шаблоны с ошибками, показанными в нескольких примерах.

Однако у распределенного представления знаний есть недостаток, заключающийся в том, что инженеру по знаниям трудно локализовать информацию. Практически невозможно проанализировать и понять множество весов в полностью обученной нейронной сети. Напротив, для дерева усвоенных решений легко понять полученные знания и даже представить их в виде логической формулы. Особенно выразительна и изящна логика предикатов, позволяющая формализовать отношения. Например, предикат бабушка (карен, клайд) легко понять. Такого рода отношения также можно изучить с помощью нейронных сетей, но невозможно даже локализовать «бабушкин нейрон» в сети. Поэтому до сих пор существуют проблемы с подключением нейронных сетей к системам обработки символов.

Machine Translated by Google

Из большого количества интересных нейромоделей многие здесь не поддаются обработке. Например, самоорганизующиеся карты, введенные Кохоненом, представляют собой биологически мотивированное отображение сенсорного слоя нейронов на второй слой нейронов со свойством адаптивности и сохранения сходства.

Проблема с представленными здесь сетями возникает во время поэтапного обучения. Если, например, полностью обученная сеть обратного распространения далее обучается с новыми шаблонами, то многие (и потенциально все) старые шаблоны быстро забываются. Чтобы решить эту проблему, Карпентер и Гроссберг разработали теорию адаптивного резонанса (APT), результатом которой стала целая линейка нейронных моделей.

В качестве дополнительной литературы рекомендуем учебники [Bis05, RMS92, Roj96, Zel94]. Те, кто заинтересован в чтении наиболее важных оригинальных работ в этой захватывающей области, могут обратиться к двум сборникам [AR88, APR90].

9.9 Упражнения

9.9.1 От биологии к моделированию

Упражнение 9.1. Покажите точечную симметрию сигмовидной функции относительно $(\theta, \frac{1}{2})$.

9.9.2 Сети Хопфилда

Упражнение 9.2. Используйте сетевой апплет Хопфилда по адресу http://www.cbu.edu/~pong/ai/hopfield/hopfieldapplet.html и проверьте объем памяти для коррелированных и некоррелированных шаблонов (со случайно установленными битами) с размером 10×10. размер сетки. Используйте шаблон с 10% шума для тестирования.

Упражнение 9.3. Сравните теоретический предел $N = 0{,}146\,n$ для максимального количества хранимых шаблонов, приведенный в разд. 9.2.2, с емкостью классической бинарной памяти того же размера.

9.9.3 Линейные сети с минимальными ошибками

Упражнение 9.4

(а) Напишите программу для обучения линейному отображению методом наименьших средних квадратов. Это довольно просто: нужно только составить нормальные уравнения по (9.12) на стр. 242, а затем решить систему уравнений.

(а) Примените эту программу к данным об аппендиците на веб-сайте этой книги и определите линейную оценку. Задайте ошибку как функцию порога, как показано на рис. 9.14 на стр. 243. (с) Теперь определите ROC-кривую для этой оценки.

Machine Translated by Google

9.9.4 Обратное распространение

Упражнение 9.5. Используя программу обратного распространения ошибки (например, в JNNS или KNIME), создайте сеть с восемью входными нейронами, восемью выходными нейронами и тремя скрытыми нейронами. Затем обучите сеть с помощью восьми парообучающихся с опорой, данных q, чтобы убедиться, что в m векторе q p-й бит равен единице, а все остальные биты равны нулю. Таким образом, перед сетью стоит простая задача изучения идентичного отображения. Такая сеть называется кодером 8-3-8. После процесса обучения наблюдайте за кодированием трех скрытых нейронов для ввода всех восьми изученных входных векторов. Что ты заметил? Теперь повторите эксперимент, уменьшив количество скрытых нейронов до двух, а затем до одного.

Упражнение 9.6 Чтобы показать, что сети обратного распространения могут разделять нелинейно разделимые множества, обучите функцию XOR. Обучающие данные для этого: ((0, 0), 0), ((0, 1), 1), ((1, 0), 1), ((1, 1), 0). (a) Создайте сеть с двумя
нейронами во входном слое и скрытом слое, и
один нейрон в выходном слое и обучить эту сеть.
(b) Теперь удалите скрытые нейроны и соедините входной слой напрямую с выходными нейронами.
Что вы наблюдаете?

Упражнение 9.7

(a) Покажите, что любая многослойная сеть обратного распространения с линейной функцией
активации обладает такой же мощностью, как и двухслойная. Для этого достаточно
показать, что последовательное выполнение линейных отображений
является линейным отображением. (b) Покажите, что двухслойная сеть обратного
распространения с любой строго монотонной функцией активации не более эффективна для
задач классификации, чем сеть без функции активации или с линейной функцией активации.

9.9.5 Машины опорных векторов

Упражнение 9.8 Два нелинейно разделимых двумерных набора обучающих данных M+ 2 2 = 1 и все + x
и M заданы. Все точки M+ лежат внутри единичной окружности x 2 2 2 1 R, что делает
точки в M находятся снаружи. Задайте преобразование координат
f : R для линейно разделимых данных. Дайте уравнение разделительной линии и нарисуйте два
пространства и точки данных.

Machine Translated by Google

Обучение с подкреплением

10

10.1 Введение

Все алгоритмы обучения, описанные до сих пор, кроме алгоритмов кластеризации, относятся к классу обучения с учителем. При обучении с учителем предполагается, что агент изучает отображение входных переменных в выходные переменные. Здесь важно, чтобы для каждого отдельного обучающего примера были предоставлены все значения как входных, так и выходных переменных. Другими словами, нам нужен учитель или база данных, в которой сопоставление, которое необходимо изучить, приблизительно определено для достаточного количества входных значений. Единственная задача алгоритма машинного обучения — отфильтровать шум из данных и найти функцию, которая хорошо аппроксимирует отображение даже между заданными точками данных.

При обучении с подкреплением ситуация иная и более сложная, потому что нет данных для обучения. Начнем с простого иллюстративного примера из робототехники, который затем используется как приложение для различных алгоритмов.

Обучение с подкреплением очень ценно в области робототехники, где задачи, которые необходимо выполнить, часто настолько сложны, что не поддаются кодированию в виде программ, а данные для обучения недоступны. Задача робота состоит в том, чтобы путем проб и ошибок (или успеха) выяснить, какие действия хороши в той или иной ситуации, а какие нет. Во многих случаях мы, люди, учимся очень похожим образом. Например, когда ребенок учится ходить, это обычно происходит без обучения, а просто через подкрепление. Успешные попытки ходить вознаграждаются продвижением вперед, а неудачные попытки наказываются часто болезненными падениями. Положительное и отрицательное подкрепление также являются важными факторами успешного обучения в школе и во многих видах спорта (см. рис. 10.1 на стр. 258).

В следующем примере изучается значительно упрощенная задача движения.

Пример 10.1 Робот, механизм которого состоит только из прямоугольного блока и руки с двумя шарнирами, дуги, показан на рис. 10.2 на стр. 258 (см. [KMK97]).
Единственными возможными действиями робота являются движения дуг вверх или вниз и дуг вправо или влево. Кроме того, мы разрешаем перемещение только фиксированных дискретных единиц (например, с шагом 10 градусов). Задача агента состоит в том, чтобы изучить политику

В. Эртель, Введение в искусственный интеллект, Темы бакалавриата по компьютерным наукам, DOI 10.1007/978-0-85729-299-5_10, © Springer-Verlag London Limited, 2011 г.

Machine Translated by Google



Рис. 10.1. «Может быть , в следующий раз мне следует нач ать поворач ивать ся немного ранше или двигать ся медленнее?» — Обуч ение на отриц ат ель ном под к реплении

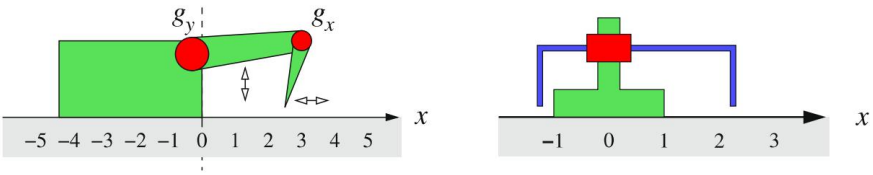


Рис. 10.2. Двигая обоими суст авами, ползающий робот в левой ч аст и изображения может двигат ь ся вперед и наз ад . Шагающий робот с правой ст ороны должен соот вет ст венно перемещат ь раму вверх и вниз или влево и вправо. Обрат ная свя зь для движения робот а положит ель на для движения вправо и отриц ат ель на для движения влево .

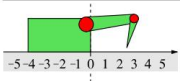
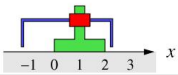
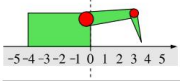
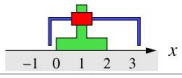
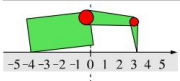
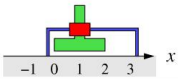
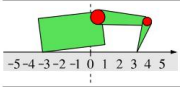
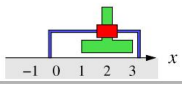
ч т о позволя ет ему двигат ь ся к ак можно быст рее вправо. Шагающий робот на рис. 10.2 работ ает аналогич но в том же двухмерном прост ранст ве сост оя ний.

Успешная последоват ель ност ь дейст вий пок аз ана в т аблиц е 10.1 на ст раниц е 259. Дейст вие в момент времени t = 2 приводит к т ому, ч т о нагруженная рук а перемещает т ело на одну ед иниц у д линывправо. Хорошие анимац ии эт ого примера можно найт и в [KMK97] и [Tok06].

Прежде ч ем мыперейдем к алгорит му обуч ения , мыдолжнысоот вет ст вующим образом смоделироват ь зад ач у мат емат ич еск и. Мыописываем сост оя ние робот а двумя переменными gx и gy для положения суст авов, к аждая из к от орых имеет к онеч ное ч исло д иск рет ных з нач ений. Т ак им образом, сост оя ние робот а к од ируется к ак век т ор (gx, gy). К олич ест во возможных совмест ных положений равно nx и ny или соот вет ст венно . Мыиспольз уем гориз онт аль ное положение т ела робот а, к от орое может принимат ь реаль ные з нач ения , для оц енк и дейст вий робот а. Движения вправо воз наг раждают ся

Machine Translated by Google

Таблица 10.1 Циклическая серия движений с систематическим движением вперед

Ползучий робот	Бегущий робот	Время	Состояние		Вознаграждение за действие	
		т	гы	gx	икс	в
		0	Вверх	Левый	0	Верно
		1	Вверх	Верно	0	Вниз
		2	Вниз вправо		0	Левый
		3	Вниз влево1			Вверх

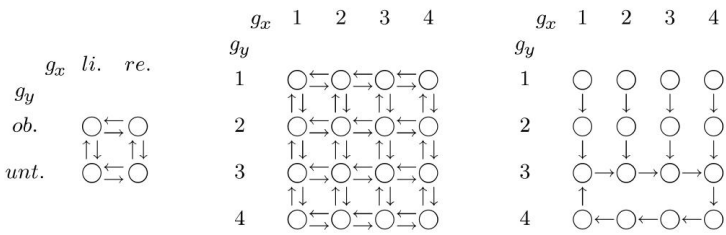


Рис. 10.3 Пространство состояний примера робота в случае двух возможных положений для каждого из суставов (слева) и в случае четырех горизонтальных и вертикальных положений для каждого (посередине). в правильное изображение дается оптимальная политика

с положительными изменениями x, а движения влево наказываются отрицательными изменениями .

На рис. 10.3 пространство состояний для двух вариантов этого показано в упрощенной форме1. В левом примере оба сустава имеют два положения , а в среднем примере они у каждого по четыре положения . Оптимальная политика представлена на рис. 10.3 справа.

10.2 Задача

Как показано на рис. 10.4 на стр. 260, мыразличаем агента и его окружение. В момент времени t мир, включающий в себя агента и его окружение, описывается состоянием st S. Множество S является абстракцией реальных возможных состояний мира, потому что, с одной стороны, мир не может быть точно описан, а с другой стороны, агент часто имеет только неполную информацию о фактическом

1 Пространство движения руки, состоящее из дуг, визуализируется как прямоугольная сетка.

Machine Translated by Google

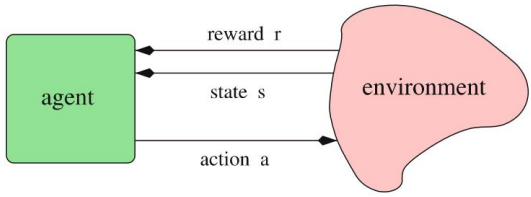


Рис. 10.4 Агент и его взаимодействие со средой

состояние из-за ошибок измерения. Затем агент выполняет действие в точке A в момент времени t. Это действие изменяет мир, таким образом, приводит к состоянию st+1 в момент времени t+1. Функция перехода состояний δ, определяемая средой, определяет новое состояние st+1 = δ(st, at). Агент не может повлиять на него.

Выполнив действие при, агент получает немедленное вознаграждение rt = r(st, at) (см. рис. 10.4). Немедленное вознаграждение rt = r(st, at) всегда зависит от текущего состояния и выполненного действия. r(st, at) = 0 означает, что агент не получает немедленной обратной связи за действие в. При обучении rt > 0 должно приводить к положительному, а rt < 0 к отрицательному подкреплению оценки действия в состоянии st. В частности, при обучении с подкреплением изучаются приложения, в которых в течение длительного времени не происходит немедленного вознаграждения. Шахматист, например, учится улучшать свою игру благодаря выигранным или проигранным партиям, даже если он не получает немедленного вознаграждения за все отдельные ходы. Здесь мы можем увидеть сложность присвоения вознаграждения в конце последовательности действий всем действиям в последовательности, которые привели к этой точке (проблема распределения кредитов).

В случае ползающего робота состояние состоит из положения двух суставов, то есть s = (gx, gy). Награда определяется пройденным расстоянием x.

Политика π : S A — это отображение состояний в действия. Цель обучения с подкреплением состоит в том, чтобы агент выучил оптимальную политику на основе своего опыта. Политика оптимальна, если она максимизирует вознаграждение в долгосрочной перспективе, то есть на многих этапах. Но что именно означает «максимальное вознаграждение»? Определяемость, или дисконтированное вознаграждение

$$B^{\pi}(st) = rt + \gamma\, rt+1 + \gamma^{2}\, p+2 + \cdots = \sum_{i=0} \gamma^{i}\, rt+i \qquad (10.1)$$

политики π, когда мы начинаем в начальном состоянии st. Здесь 0 γ < 1 — константа, которая гарантирует, что будущая обратная связь тем больше обесценивается, чем дальше в будущем она происходит. Немедленное вознаграждение rt взвешивается как самое сильное. Эта функция вознаграждения используется чаще всего. Альтернативой, которая иногда интересна, является среднее вознаграждение.

$$B^{\pi}(st) = \lim_{h \to \infty} \frac{1}{h} \sum_{i=0}^{h} rt+i \,. \qquad (10.2)$$

Политика π называется оптимальным, если для всех состояний s

$$B^{\pi}(c) \quad B^{\pi}(c). \qquad (10.3)$$

Machine Translated by Google

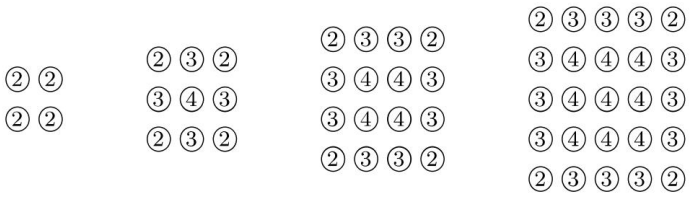


Рис. 10.5. Пространство состояний для примера со значениями 2, 3, 4, 5 для nx и ny. Количество возможных действий указано для каждого состояния в соответствующих кругах.

Таблица 10.2 Количество политик и для пространств состояний разного размера в примере

нкс, нью-йорк	Количество штатов	Количество политик
24 24 = 16		
39 243 44 = 5184		
4	16	243 84 4 2,7 × 107 243 124
5	25	9 2,2 × 1012

То есть он как минимум не уступает всем другим политикам в соответствии с заданным значением. функция. Для лучшей читаемости функция оптимального значения V π будет обозначаться V .

Обсуждаемые здесь агенты или их политики используют только информацию о текущем состоянии st для определения следующего состояния, а не предыдущую историю. Это оправдано, если награда за действие зависит только от текущего состояния и текущих действий.
Такие процессы называются марковскими процессами принятия решений (MDP). Во многих приложениях, особенно в робототехнике, фактическое состояние агента точно неизвестно, что еще больше затрудняет планирование действий. Причиной этого может быть зашумленный сигнал датчика. Мы называем такой процесс частично наблюдаемым марковским процессом принятия решений (POMDP).

10.3 Неинформированный комбинаторный поиск

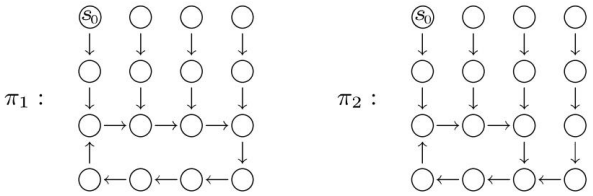
Простейшая возможность найти успешную политику — это комбинаторный перебор всех политик, как описано в гл. 6. Однако даже в простом примере 10.1 имеется очень много политик, из-за чего комбинаторный поиск связан с чрезвычайно высокими вычислительными затратами. На рис. 10.5 указано количество возможных действий для каждого состояния. Исходя из этого, количество возможных политик рассчитывается как произведение заданных значений, как показано в таблице 10.2.

Для произвольных значений nx и ny всегда есть четыре угловых узла с двумя возможными действиями, 2 (nx - 2) + 2 (ny - 2) краевых узла с тремя действиями и (nx - 2) (ny - 2) внутренние узлы. с четырьмя действиями. Таким образом, есть

4 2 3 2(nx 2)+2(ny 2) 4 (nx 2)(ny 2)

Machine Translated by Google

Рис. 10.6 Две разные политики для примера



разные политики для фиксированных nx и ny. Таким образом, количество политик растет экспоненциально с количеством состояний. В общем случае это верно, если в каждом состоянии имеется более одного возможного действия. Поэтому для практических приложений этот алгоритм бесполезен. Даже эвристический поиск, описанный в гл. 6, здесь нельзя использовать. Поскольку упрямое вознаграждение почти за все действия равно нулю, его нельзя использовать в качестве эвристической функции оценки.

Вычислительные затраты возрастают еще выше, если учесть, что (в дополнение к (s) должны быть перечисление всех политик), значение V^{π} рассчитаны для каждого сгенерированного (s) должны политика π и каждое начальное состояние s. Бесконечная сумма в V^{π} быть отсечены для использовать в практическом расчете; однако из-за экспоненциального уменьшения коэффициентов γ в (10.1) на стр. 260 это не представляет проблемы

В примере 10.1 на стр. 257 разница xt+1 xt может быть использована как немедленная награда за действие при, что означает, что каждое движение тела робота вправо вознаграждается 1, а каждое движение тела робота вправо левый называется -1. На рис. 10.6 показаны две политики. Здесь немедленное вознаграждение равно нулю везде, кроме нижней строки и пространства состояний. Левая политика π1 лучше в долгосрочной перспективе, потому что для длинных последовательностей действий средний прогресс на одно действие составляет 3/8 = 0,375 для π1 и 2/6 0,333 для π2. Если мы используем (10.1) (s), результатом будет следующая таблица с начальным состоянием s0 в точке
на стр. 260 для V^{π} вверху
слева и различных значений γ :

γ	0,9	0,8375	0,8
В π1 (c0)	2,52	1,156	В π2 0,77
(c0)	2,39	1,156	0,80

Здесь мы видим, что политика π1 лучше политики π2, когда gamma = 0,9, и наоборот, когда gamma = 0,8. Для γ 0,8375 обе политики одинаково хороши.
Мы можем сноситься, что более шее γ приводит к более шему временному горизонту для оценки политики.

10.4 Итерация значений и динамическое программирование

При простом подходе к перечислению всех политик выполняется много избыточной работы, потому что многие политики по большей части идентичны Они могут лишь незначительно отличаться. Тем не менее, каждая политика создается и оценивается полностью заново.
Это предполагает сохранение промежуточных результатов для частей политик и их повторное использование.

Machine Translated by Google

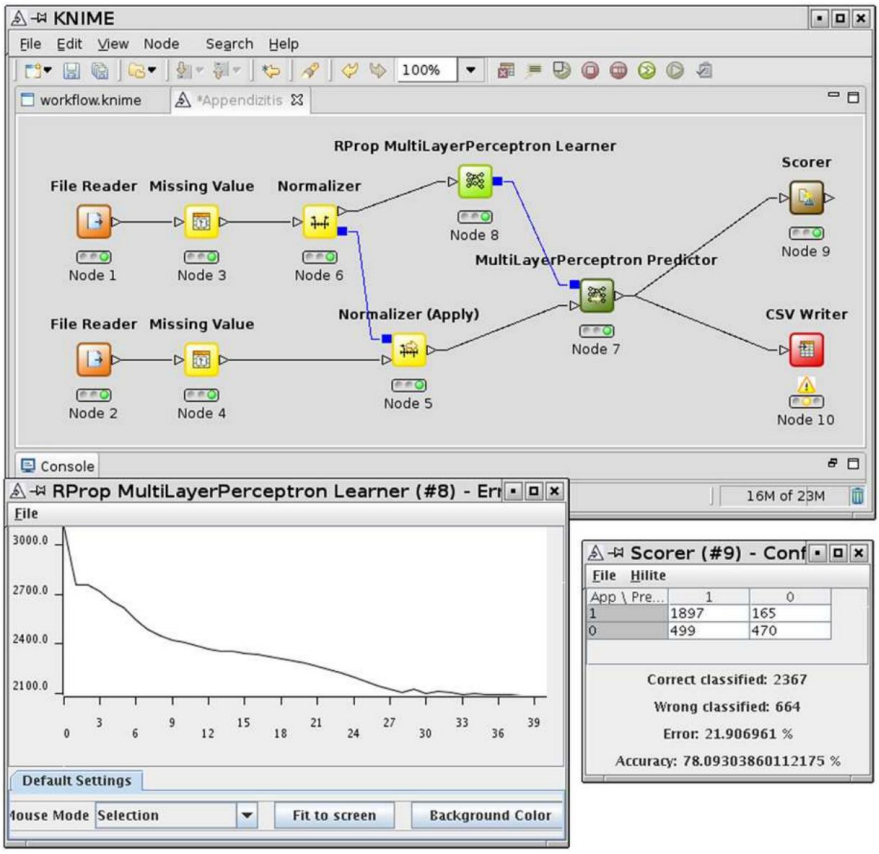


Рис. 8.36. Пользовательский интерфейс KNIME с окном рабочего процесса, кривой обучения и матрицей путаницы

После того, как поток полностью построен и все узлы настроены, произвольный узел может быть казнен. Он автоматически гарантирует выполнение узлов-предшественников, если необходимый. Узел J48 генерирует представление дерева решений, показанное справа. фигуры Это дерево идентично дереву, сгенерированному C4.5 в разд. 8.4.5, хотя здесь узел TRekt<=378 показан свернутым.

Для сравнения, проект обучения многослойного персептрона (см. раздел 9.5) показано на рис. 8.36. Это работает аналогично введенному ранее линейному персептрону, но также может разделять нелинейно разделимые классы Здесь течение несколько более сложный. Дополнительный узел необходим для обработки ошибок отсутствующих значений, когда предварительная обработка каждого из двух файлов. Мыустанавливаемего так, чтобыэтистроки были удалены Поскольку у нейронные сети не могут работать с произвольными значениями, значения всех переменных линейно масштабируются в интервале [0, 1] с помощью узла «нормализатор».

Machine Translated by Google

После применения обучающего модуля RProp улучшено обратное распространение (см. Разд. 9.5), мы можем проанализировать изменение во времени ошибки аппроксимации в показана кривая обучения. В матрице путаницы бомбардировщика выводит анализ тестовых данных. Узел «CSV Writer» в правом нижнем углу служит для экспорта файлы результатов, которые затем можно использовать извне для создания ROC-кривой, показанной на рис. Рис. 7.10 на стр. 142, для которого, к сожалению, нет инструмента KNIME (пока).

Подводя итог, можно сказать о KNIME (и подобных инструментах) следующее: для проекта с не слишком экзотическими требованиями к анализу данных, стоит работать с таким мощным верстаком для анализа данных. Пользователь уже сохраняет много времени на этапе предварительной обработки. Существует большой выбор легко используемых «инструментов общих», таких как классификаторы ближайших соседей, простые алгоритмы обучения байесовской сети, а также алгоритмы кластеризации k-средних (раздел 8.7.2). Оценка результатов, например перекрестную проверку, можно легко провести. Осталось Следует отметить, что помимо показанных существует множество других инструментов для визуализации данных. Кроме того, разработчики KNIME сделали расширение для Доступен KNIME, с помощью которого пользователь может программировать свои собственные инструменты на Java или Python.

Однако следует также упомянуть, что пользователь этого типа интеллектуального анализа данных система должна принести с собой прочные предварительные знания в области машинного обучения и использования методов интеллектуального анализа данных. Программное обеспечение само по себе не может анализировать данные, но в руке специалиста становится мощным инструментом извлечения знаний из данных.
Для новичка в увлекательной сфере машинного обучения такая система предлагает идеальная и простая возможность проверить свои знания на практике и сравнить различные алгоритмы Читатель может убедиться в этом в упражнении 8.21 на странице 220.

8.9 Резюме

Мы подробно рассмотрели несколько алгоритмов из устоявшейся области обучения с учителем, включая обучение по дереву решений, байесовские сети и ближайший алгоритм.
метод соседа. Эти алгоритмы стабильны и эффективно используются в различных приложениях и, таким образом, относятся к стандартному репертуару ИИ и интеллектуального анализа данных.
то же самое верно и для алгоритмов кластеризации, которые работают без «супервайзера» и можно найти, например, в приложениях поисковых систем. Обучение с подкреплением поскольку у другая область машинного обучения также не использует супервайзера. В отличие от обучения с учителем, когда учащийся получает правильные действия или ответы в качестве ярлыков. в обучающих данных, в обучении с подкреплением только время от времени поступает положительная или отрицательная обратная связь из окружающей среды В гл. 10 мы покажем, как это работает. Не такая сложная задача в полуконтролируемом обучении, молодой подобласти. машинного обучения, где только очень немногие из большого количества обучающих данных помечен.

Обучение с учителем в настоящее время является хорошо зарекомендовавшей себя область ю с большим количеством успешных приложений. Для контролируемого обучения данных с непрерывными метками может быть использован любой алгоритм аппроксимации функций. Таким образом, существует множество алгоритмов из различных областей математики и информатики. В разд. 9 мы представим различные типы нейронных сетей, алгоритмы наименьших квадратов и машины опорных векторов,

Machine Translated by Google

которые являются аппроксиматорами функций. В настоящее время гауссовыпроцессыочень популярны поскольку они очень универсальны и просты в применении и предоставляют пользователюоценку неопределенности выходных значений [RW06].

Следующая таксономия дает обзор наиболее важных алгоритмов обучения и их классификацию

Контролируемое обучение

- Ленивое обучение
  - метод k ближайших соседей (классификация + аппроксимация)
  - Локально взвешенная регрессия (аппроксимация)
  - Обучение на основе конкретных случаев (классификация + аппроксимация) •
  - Усердное обучение – Индукция деревьев решений (классификация)
  - Обучение байесовских сетей (классификация + аппроксимация)
  - Нейронные сети (классификация + аппроксимация)
  - Гауссовские процессы(классификация + аппроксимация)
  - вейвлеты сплайны радиальные базисные функции, . . .

Неконтролируемое обучение (кластеризация)

- Алгоритм ближайшего соседа •
- Алгоритм дальнего соседа • k-средних
- • Нейронные
- сети Обучение с
- подкреплением • Итерация
- значений • Q-
- обучение • TD-
- обучение • Методы
- градиента политик • Нейронные
- сети Однако сказанное

об обучении с учителем верно. , при работе с фиксированным набором известных атрибутов. Интересным, все еще открытым полем для интенсивных исследований является автоматический выбор признаков. В разд. 8.4, для обучения с деревьями решений мы представили алгоритм расчета прироста информации атрибутов, который сортирует атрибуты в соответствии с их релевантностью и использует только те, которые улучшают качество классификации. С помощью этого типа метода можно автоматически выбирать соответствующие атрибуты из потенциально большого базового набора.

Однако этот базовый набор необходимо выбирать вручную

До сих пор открытым и не вполне определенным является вопрос о том, как машина может находить новые атрибуты Представим себе робота, который должен собирать яблоки. Для этого он должен научиться различать спелые и незрелые яблоки и другие предметы
Традиционно мы определяли определенные атрибуты, такие как цвет и форма областей пикселей, а затем обучали алгоритм обучения , используя классифицированные вручную изображения .
Также возможно, что, например, нейронная сеть может быть обучена напрямую со всеми пикселями из изображения в качестве входных данных, что, однако, для высокого разрешения связано с серьезными проблемами времени вычислений. Здесь желательны подходы, которые автоматически предлагают соответствующие функции. Но это все еще научная фантастика.

Machine Translated by Google

Кластеризация обеспечивает один из подходов к выбору признаков. Перед обучением машины распознавания я блок мы запускаем кластеризацию данных. Для (контролируемого) обучения классов я блоки не я блоки входными данными больше не являются все пиксели, а только классы, найденные во время кластеризации, потенциально вместе с другими атрибутами.
Кластеризация в любом случае может быть использована для автоматического, творческого «обнаружения » признаков. Однако нея сно, имеют ли значение обнаруженные признаки.

Еще сложнее следующая задача: предположим, что видеокамера, используемая для распознавания я блок, передает только черно-белые изображения . Задача уже не может быть решена хорошо. Было бы неплохо, если бы машина занималась творчеством сама по себе, например, предлагая заменить камеру цветной камерой.
Сегодня это было бы слишком много.

Помимо специализированных работ по всем разделам машинного обучения , есть отличные учебники [Mit97, Bis06, Alp04, DHS01, HTF09]. Текущие результаты исследований можно найти в свободном доступе в Journal of Machine Learning Research (http://jmlr.csail.mit.edu), в журнале Machine Learning Journal, а также в материалах Международной конференции по машинному обучению(ICML).) Рекомендовано. Каждому разработчику алгоритмов обучения интересен репозиторий машинного обучения [DNM98] Калифорнийского университета в Ирвайне (UCI) с его большой коллекцией обучающих и тестовых данных для алгоритмов обучения и инструментов интеллектуального анализа данных. MLOSS, что означает программное обеспечение с открытым исходным кодом для машинного обучения , представляет собой отличный каталог ссылок на бесплатное программное обеспечение (www.mloss.org).

8.10 Упражнения

8.10.1 Введение

Упражнение 8.1

(а) Укажите задачу агента, который должен предсказать погоду на следующий день по измеренным значениям температуры, атмосферного давления и влажности. Погоду следует отнести к одному из трех классов: солнечная , облачная и дождливая .
б) Опишите структуру файла с обучающими данными для этого агента.

Упражнение 8.2. Покажите, что корреляционная матрица симметрична и что все диагональные элементы равны1.

8.10.2 Персептрон

Упражнение 8.3. Применение правила обучения персептрона к множествам

$$M+ = \{(0, 1,8), (2, 0,6)\} \text{ и } M = \{(1,2, 1,4), (0,4, 1)\}$$

из Примера 8.2 на стр. 172 и выдать результат значений весового вектора.

Machine Translated by Google

Упражнение 8.4. Имея таблицу справа с обучающими данными: (a) Используя график, покажите, что данные линейно разделимы (b) Вручную определить веса w1 и w2, а также порог Θ персептрона (с порогом), который правильно классифицирует данные.

номер	x1	x2	Класс
1	6	10	−
2	7	30	−
3	8	20	−
4	9	00	−
5	8	41	−
6	8	61	+
7	9	21	+
8	9	51	+

(c) Запрограммируйте правило обучения персептрона и примените вашу программу к таблице. Сравните обнаруженные веса с рассчитанными вручную

Упражнение 8.5

(a) Дайте визуальную интерпретацию эвристической инициализации.

$$w_0 = -\left(\frac{x_{i+} + x_{i-}}{2}\right)^T \frac{x_{i+} - x_{i-}}{\|x_{i+} - x_{i-}\|},$$

весового вектора, описанного в разд. 8.2.2.

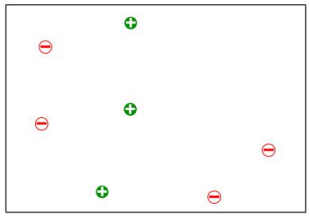
(b) Приведите пример линейно разделимого набора данных, для которого эта эвристика не дает разделительной линии.

8.10.3 Метод ближайших соседей

Упражнение 8.6

(a) Изобразите диаграмму Вороного для соседних наборов точек.

б) Затем нарисуйте линии деления классов.



Упражнение 8.7. Пусть дана таблица с обучающими данными из упражнения 8.4. В дальнейшем используйте манхэттенское расстояние d(a, b), определяемое как $d(a, b) = |a_1 - b_1| + |a_2 - b_2|$, чтобы определить расстояние d между двумя точками данных a = (a1, a2) и b = (b1, b2). (a) Классифицируйте вектор v = (8, 3.5) методом ближайшего соседа. (b) Классифицировать вектор v = (8, 3.5) методом k ближайших соседей для k = 2, 3, 5.

Упражнение 8.8

(a) Покажите, что в двумерном пространстве признаков разумно, как утверждается в (8.3) на стр. 180, взвешивать k ближайших соседей на величину, обратную квадрату расстояния.

(b) Почему взвешивание с использованием $w_i' = \frac{1}{1+\alpha d(x,x_i)}$ имеет меньше смысла?

Machine Translated by Google

8.10.4 Деревья решений

Упражнение 8.9. Покажите, что определение $0 \log_2 0 := 0$ разумно, другими словами, что функция $f(x) = x \log_2 x$ тем самым становится непрерывной в начале координат.

Упражнение 8.10. Определите энтропию для следующих распределений.

(a) (1, 0, 0, 0, 0)

(b) $\frac{1}{2}, \frac{1}{2}, 0, 0, 0$

(c) $\frac{1}{2}, \frac{1}{4}, \frac{1}{4}, 0, 0$

(d) $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16}$

(e) $\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}$

(f) $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \ldots$

Упражнение

8.11 (a) Покажите, что два разных определения энтропии из (7.9) на стр. 124 и определения 8.4 на стр. 188 отличаются только постоянным множителем, т. е. что

$$\sum_{i=1}^{n} p_i \log_2 p_i = c \sum_{i=1}^{n} p_i \ln p_i$$

и задайте константу c.

(b) Покажите, что для метода MaxEnt и для деревьев решений не имеет значения, какую из двух формул мы используем.

Упражнение 8.12. Разработайте дерево решений для набора данных D из упражнения 8.4 на стр. 217. (a)

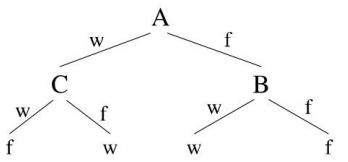
Рассматривайте оба атрибута как

дискретные. (b) Теперь считайте атрибут x2 непрерывным, а x1

дискретным. (c) Пусть C4.5 сгенерирует дерево с обоими вариантами. Используйте -m 1 -t 10 в

качестве параметров, чтобы получить разные предложения.

Упражнение 8.13. Учитывая следующее дерево решений и таблицы для обучающих и тестовых данных:



Тренировочные данные

ABC-класс
ттфт
тфт
ттф
ффф
ffff
фттт

Тестовые данные

ABC-класс
ттфт
тфт
ftff
ффф

(a) Укажите правильность дерева для обучающих и тестовых данных. (b)

Приведите формулу пропозициональной логики, эквивалентную

дереву. (c) Проведите обрезку дерева, нарисуйте получившееся дерево и укажите его правильность.

для обучающих и тестовых данных.

Machine Translated by Google

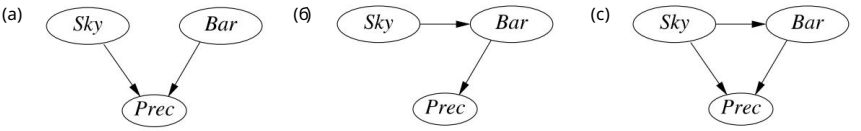
Упражнение 8.14

(a) При определении текущего атрибута алгоритм построения деревьев решений (рис. 8.26 на стр. 193) не исключает атрибуты, которые уже использовались выше по дереву. Несмотря на это, дискретный атрибут встречается в пути не более одного раза. Почему?

б) Почему непрерывные атрибуты могут встречаться несколько раз?

8.10.5 Изучение байесовских сетей

Упражнение 8.15. Используйте распределение, данное в упражнении 7.3 на стр. 158, и определите CPT для трех байесовских сетей:



(d) Определите распределение для двух сетей из (a) и (b) и сравните они с оригинальным дистрибутивом. Какая сеть «лучше»?

(e) Теперь определите распределение для сети (c). Что происходит с вами? Джустифай!

Упражнение 8.16. Покажите, что для бинарных переменных S1,...,Sn и бинарной переменной класса K линейная оценка вида

$$\text{решение} = \begin{cases} \text{положительно, если } w1S1 + \cdots + wnSn > \Theta, \text{ отрицательно} \\ \text{в противном случае} \end{cases}$$

одинаково выразительно по отношению к персептрону и наивному байесовскому классификатору, которые оба решают по формуле

$$\text{решение} = \begin{cases} \text{положительный, если } P(K|S1,...,Sn) > 1/2, \text{ отрицательный} \\ \text{в противном случае} \end{cases}$$

Упражнение 8.17. При реализации текстовой классификации с помощью наивного байесовского метода потеря значимости экспоненты может произойти быстро, потому что множители P (wi |K) (поя вляющиеся в (8.10) на стр. 206) обычно очень малы, что может привести к чрезвычайно малым значениям. Результаты Как мы можем смягчить эту проблему?

Упражнение 8.18. Напишите программу наивного байесовского анализа текста. Затем обучите и протестируйте его на текстовых тестах с помощью инструмента по вашему выбору. Подсчет частотыслов в тексте можно легко сделать в Linux с помощью команды

кошка <данные> | tr -d “[:punct:]” | tr -s «[:space:]» «\n» | сортировать | уникальный -ci

Machine Translated by Google

Получите данные Twenty Newsgroups Тома Митчелла в коллекции тестов машинного обучения UCI (репозиторий машинного обучения) [DNM98]. Там же вы найдете ссылку на наивную байесовскую программу для классификации текстов Митчелла.

8.10.6 Кластеризация

Упражнение 8.19. Покажите, что для алгоритмов, которые только сравнивают расстояния, применение строго монотонно возрастающей функции f к расстоянию не имеет значения. Другими словами, вы должны показать, что расстояние d1(x, y) и расстояние d2(x, y) := f (d1(x, y)) приводят к одному и тому же результату относительно отношения порядка.

Упражнение 8.20. Определите расстояния ds (скалярное произведение) следующих текстов друг от друга. x1: Мы

представим применение наивного Байеса к анализу текста в кратком

пример текста Алана Тьюринга из [Tur50].

x2: Мы можем надеяться, что машины со временем станут конкурировать с людьми во всех чисто интеллектуальных полях. Но с каких лучше начать?

x3: Опять же, я не знаю, каков правильный ответ, но я думаю, что оба подхода должны

пытался.

8.10.7 Интеллектуальный анализ данных

Упражнение 8.21. Используйте KNIME (www.knime.de) и (a)

Загрузите файл примера с данными Iris из каталога KNIME и поэкспериментируйте с различными

представления данных, особенно с диаграммами рассеяния.

(b) Сначала обучите дерево решений для трех классов, а затем обучите сеть RProp. (c) Загрузите данные об аппендиците на веб-сайт этой книги. Сравните качество классификации метода k

ближайших соседей с качеством сети RProp. Оптимизируйте k, а также количество скрытых

нейронов сети RProp. (d) Получите набор данных по вашему выбору из

коллекции данных UCI для интеллектуального анализа данных на http://kdd.ics.uci.edu или

для машинного обучения на http://mlearn.ics.uci.edu/MLRepository.html и поэкспериментируйте

с ним.

Machine Translated by Google

Нейронные сети

9

Нейронные сети представляют собой сети нервных клеток в мозге человека и животных. В человеческом мозгу около 100 миллиардов нервных клеток. Мы, люди, обязаны своим интеллектом и способностью обучаться различным двигательным и интеллектуальным способностям сложным механизмам и адаптивности мозга. На протяжении многих веков биологи, психологи и врачи пытались понять, как функционирует мозг. Примерно в 1900 году пришло революционное осознание того, что эти крошечные физические кирпичики мозга, нервные клетки и их связи, отвечают за осознание, ассоциации, мысли, сознание и способность к обучению.

Первый большой шаг к нейронным сетям в ИИ был сделан в 1943 году Мак Каллохом и Питтсом в статье под названием «Логическое исчисление идей, присущих нервной деятельности» [AR88]. Они первыми представили математическую модель нейрона как основного переключающего элемента мозга. Эта статья заложила основу для построения искусственных нейронных сетей и, следовательно, для этой очень важной области ИИ.

Мы могли бы рассматривать область моделирования и симуляции нейронных сетей как ветвь бионики[1] внутри ИИ. Почти все области ИИ пытаются воссоздать когнитивные процессы, такие как логика или вероятностные рассуждения. Однако инструменты, используемые для моделирования, а именно математика, язык программирования и цифровые компьютеры, имеют очень мало общего с человеческим мозгом. С искусственными нейронными сетями подход другой. Исходя из знаний о функциях естественных нейронных сетей, мы пытаемся моделировать, симулировать и даже реконструировать их в аппаратном обеспечении. Каждый исследователь в этой области сталкивается с увлекательной и захватывающей задачей сравнения результатов с человеческими способностями.

В этой главе мы попытаемся наметить историческую прогрессию, определив модель нейрона и его взаимосвязей, начиная с наиболее важных биологических открытий. Затем мы представим несколько важных и фундаментальных моделей: модель Хопфилда, две простые модели ассоциативной памяти и, что не менее важно, на практике алгоритм обратного распространения ошибки.

1Bionics занимается раскрытием «открытий живой природы» и ее инновационным преобразованием в технологии [Wik10].

В. Эртель, Введение в искусственный интеллект, Темы бакалавриата по компьютерным наукам, DOI 10.1007/978-0-85729-299-5_9, © Springer-Verlag London Limited 2011

Machine Translated by Google

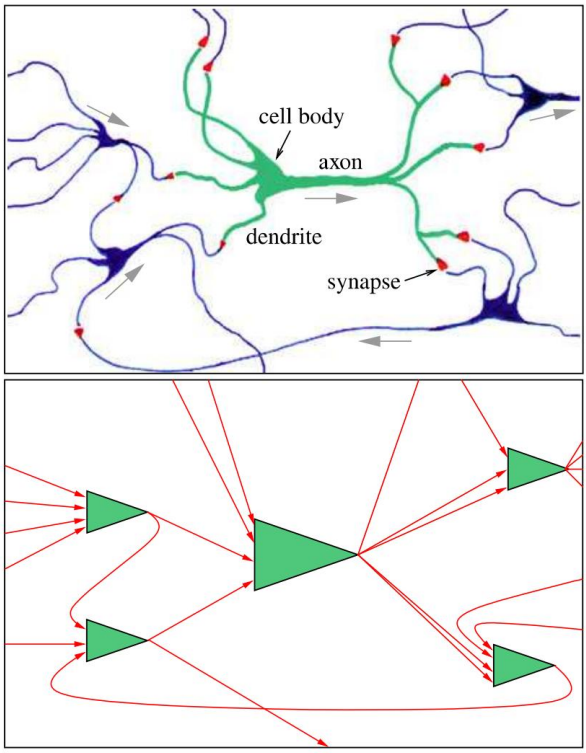


Рис. 9.1 Два этапа моделирования нейронной сети. Над биологической моделью под формальной моделью с нейронами и направленными связями между ними

9.1 От биологии к моделированию

Каждый из примерно 100 миллиардов нейронов в человеческом мозгу имеет, как показано в упрощенном представлении на рис. 9.1, следующую структуру и функцию Помимо тела клетки, у нейрона есть аксон, который может создавать локальные соединения с другими нейронами через дендриты Однако аксон может вырасти до метра в длину в виде нервного волокна, проходящего через тело.

Тело клетки нейрона может хранить небольшие электрические заряды подобно конденсатору или батарее. Это хранилище загружается поступающими электрическими импульсами от других нейронов. Чем больше поступает электрического импульса, тем выше напряжение. Если напряжение превышает определенный порог, нейрон срабатывает. Это означает, что он разгружает свой запас, посылая импульс по аксону и синапсам. Электрический ток делится и через синапсы достигает многих других нейронов, в которых происходит тот же процесс.

Теперь возникает вопрос о структуре нейронной сети. Каждый из примерно 1011 нейронов в мозгу связан примерно с 1000–10 000 других нейронов, что дает в общей сложности более 1014 соединений. Если мы далее учтем, что это гигантское количество через вышеупомянутых тонких соединений состоит из мягкой трехмерной ткани и что эксперименты над человеческим мозгом провести нелегко, то станет ясно, почему у нас нет подробной принципиальной схемы мозга. предположительно

Machine Translated by Google

мыник огда не сможем полность юпоня ть принц ипиаль нуюсхему нашего моз га, основьвая сь иск люн ит ель но на его огромных раз мерах.

С сегодня шней т оч к из рения уже не ст оит даже пыт ат ь ся сост авит ь полную принц ипиаль нуюсхему моз га, пот ому ч т о стр ук т ура моз га ад апт ивна. Он меня ет ся на лет у и приспособливает ся к дея т ель ност и ч еловек а и влия ниюок ружающей сред ы Ц ент раль ную роль з десь игр ают синапсы, соз д ающие свя з ь между нейронами. В т оч к е соед инения между двумя нейронами к ак быв ст реч ают ся два к абеля . Од нак о два вьвод а не обл ад ают ид еаль ной провод имост ь ю, ск орее имеет ся неболь шой з азор, ч ерез к от орый эл ек т роныне могут напря муюперепры нут ь . Эт от пробел з аполнен хим ич ск ими вещест вами, т ак наз ьваемыми нейрот рансмит т ерами. Они могут быт ь ионизированыприложенным напря жением, а з ат ем переносит ь з аря д ч ерез з азор. Провод имост ь эт ой щели з ависит от мног их парамет ров, например от к онц ент рац ии и хим ич ск ого сост ава нейрот рансмит т ера. Поуч ит ель но, ч т о функ ц ия моз га оч ень ч ут к о реаг ирует на из менения эт ой синапт ич ск ой свя з и, например, под влия ниемал к ого оля или д руг их нарк от ик ов.

К ак раб от ает обуч ение в т ак ой нейронной сет и? Уд ивит ель но з десь т о, ч т о ад апт ивны не сами ак т ивные ед иниц ы, а именно нейроны, а свя з и между ними, т о ест ь синапсы В ч аст ност и, эт о может изменит ь их провод имост ь . Мы з наем, ч т о синапс ст ановит ся силь нее от т ого, наск оль к о боль ший эл ек т рич ск ий т ок он д олжен нест и. Силь нее з десь оз нач ает , ч т о синапс имеет более выс ок уюпровод имост ь . Синапсы, к от орые исполь з ют ся , ч аст о получ ают все боль ший вес.
Для синапсов, к от орые исполь з ют ся неч аст о или вообще не ак т ивны, провод имост ь прод олж ает снижат ь ся . Эт о может д аже привест и к их от миранию

Все нейроныв моз гу раб от ают асинхронно и параллель но, но, по сравнениюс к омп ют ером, с оч ень низ к ой ск орост ь ю Время нервного имп ул са сост авля ет ок оло милл исек унд ы, ровно ст оль к о же, ск оль к о и время , необх од имое для т ранспорт а ионов ч ерез синапт ич ск ующель . В эт ом случ ае т ак т овая ч аст от а нейрона сост авля ет менее од ного к илогерц а и, т ак им образом, ниже, ч ем у современных к омп ют еров, в 106 раз . ст епень параллель ной обраб от к и в сет и нервных к лет ок .

Свя з ь с внешним миром осущест вля ет ся ч ерез ч увст вит ель нье нейроны, например, на сет ч ат к е глаза, или ч ерез нервные к лет к и с оч ень д линными ак сонами, к от орые т я нут ся от моз га к мышц ам, т ак им образом, могут вып олня т ь т ак ие дейст вия , к ак д вижение глаз а. ног а.

Од нак о д о сих пор нея сно, к ак обсужд аемые принц ипыд елают воз можным раз умное повед ение. К ак и мног ие исслед оват ели в обл аст и невр ологии, мыпыт аемся об ъя снит ь с помощь ю мод елирования прост ой мат емат ич ск ой мод ели, к ак ст ановя т ся воз можными к ог нит ивные з ад ач и, например распоз навание образ ов.

9.1.1 Мат емат ич ск ая мод ель

Снач ала мы з аменим непрерывнуюось времени ш к ал ой д иск рет ног о времени. Нейрон і вып олня ет след ующие вын исления на временном шаг е. «Заг руз к а» ак т иви-

Machine Translated by Google

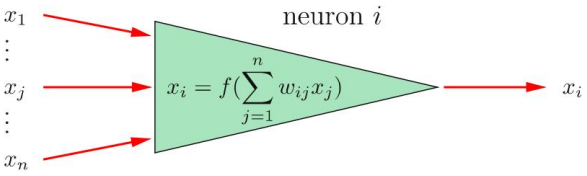


Рис. 9.2 Структура формального нейрона, который применяет функцию активации f к взвешенной сумме всех входов

потенциал достигается простым суммированием взвешенных выходных значений x1,...,xn всех входящих соединений по формуле

н

вид жхдж.

j=1

Эта взвешенная сумма рассчитывается большинством нейронных моделей. Затем к нему применяется функция активации f и результат

н

хи = е wij xj

j=1

передается соседним нейронам в виде выходных данных по синаптическим весам. На рис. 9.2 показан такой смоделированный нейрон. Для функции активации существует ряд возможностей. Самым простым является тождество: f(x) = x. Таким образом, нейрон вычисляет только взвешенную сумму входных значений и передает ее дальше. Однако это часто приводит к проблемам сходимости с нейронной динамикой, потому что функция f (x) = x неограничена, и значения функции могут выйти за все пределы с течением времени.

Напротив, очень хорошо ограничена пороговая функция (ступенчатая функция Хевисайда).

$$H(x) = \begin{cases} 0, & \text{если } x<\Theta, \\ 1 & \text{в противном случае.} \end{cases}$$

Затем весь нейрон вычисляет свой вывод как

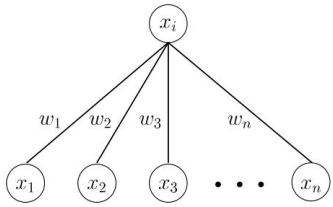
$$x_i = \begin{cases} 0 & \text{если } \sum_{j=1}^{n} w_{ij} x_j < \Theta, \\ 1 & \text{еще.} \end{cases}$$

Эта формула идентична (8.1) на стр. 173, другими словами, персептрону с порогом Θ (рис. 9.3 на стр. 225). Входные нейроны 1,...,n здесь имеют только функцию переменных, которые передают свои внешние заданные значения x1,...,xn без изменения.

Ступенчатая функция вполне разумна для бинарных нейронов, потому что активация нейрона в любом случае может принимать только значения ноль или единицу. Напротив, для непрерывного

Machine Translated by Google

Рис. 9.3 Нейрон со ступенчатой функцией работает как персептрон с порогом



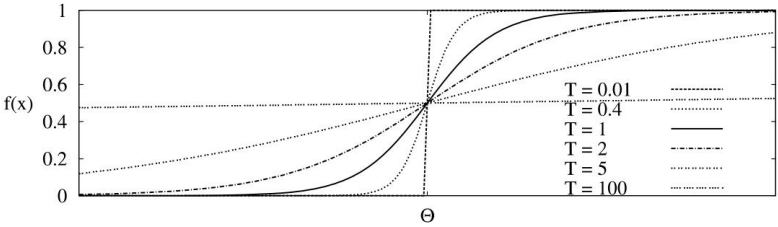


Рис. 9.4. Сигмовидная функция для различных значений параметра TT 0. . Мы видим, что в пределе Результаты ступенчатой функции

нейронов с активациями от 0 до 1, ступенчатая функция создает разрыв. Однако это можно сгладить сигмовидной функцией, такой как

$$\phi(x) = \frac{1}{1 + e^{\frac{x-\Theta}{T}}}$$

с графиком на рис. 9.4. Вблизи критической области около порога Θ эта функция ведет себя близко к линейной и имеет асимптотический предел. Сглаживание можно варьировать параметром T .

Моделирование обучения занимает центральное место в теории нейронных сетей. Как упоминалось ранее, одна из возможностей обучения состоит в укреплении синапса в зависимости от того, сколько электрических импульсов он должен передать . Этот принцип был постулирован Д. Хеббом в 1949 г. и известен как правило Хебба:

> Если между нейроном j и нейроном i существует связь wij и повторяющиеся сигналы посылаются от нейрона j к нейрону i, что приводит к одновременной активности обоих нейронов, то вес wij усиливается . Возможная формула для изменения веса wij :
>
> $$wij = \eta xixj$$
>
> с константой η (скорость юобучения), которая определяет размер отдельных шагов обучения .

Machine Translated by Google

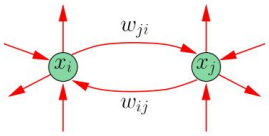


Рис. 9.5 Рекуррентные связи между двумя нейронами в сети Хопфилда

Существует множество модификаций этого правила, результатом которых становятся различные типы сетей или алгоритмов обучения. В следующих разделах мы познакомимся с некоторыми из них.

9.2 Сети Хопфилда

Глядя на правило Хебба, мы видим, что для нейронов со значениями от нуля до единицы веса могут только расти со временем. В соответствии с этим правилом нейрон не может ослабнуть или даже умереть. Это можно смоделировать, например, с помощью юконстанты затухания, которая ослабляет неиспользованный вес на постоянный коэффициент за временной шаг, например 0,99.

Совершенно иначе эта проблема решается в модели, представленной Хопфилдом в 1982 г. [Нор82]. Он использует бинарные нейроны, но с двумя значениями -1 для неактивных и 1 для активных. Используя правило Хебба, мы получаем положительный вклад в вес, когда два нейрона активны одновременно. Однако, если активен только один из двух нейронов, w_{ij} отрицателен.

На этой идее основана сеть Хопфилда, которые представляют собой прекрасный и наглядный пример автоассоциативной памяти. Паттерны могут храниться в автоассоциативной памяти. Для вызова сохраненного шаблона достаточно предоставить аналогичный шаблон. Затем магазин находит наиболее похожий сохраненный шаблон. Классическим применением этого является распознавание рукописного ввода.

На этапе обучения сети Хопфилда N двоично-кодированных шаблонов, сохраненных в векторы $q^1, \ldots, q^N$, должны быть изучены. Каждая компонента $q^k_j \in \{-1, 1\}$ вектора q^k представляет собой пиксель шаблона. Для векторов, состоящих из n пикселей, используется нейронная сеть с n нейронами, по одному на каждую позицию пиксель. Нейроны полностью связаны тем ограничением, что матрица весов симметрична и все диагональные элементы w_{ii} равны нулю. То есть между нейроном и самим собой нет никакой связи.

Полносвязная сеть включает в себя сложные петли обратной связи, так называемые рекуррентные сети (рис. 9.5).

N паттернов можно выучить, просто рассчитав все веса по формуле

$$w_{ij} = \frac{1}{n} \sum_{k=1}^{N} q^k_i q^k_j. \tag{9.1}$$

Эта формула указывает на интересную связь с правилом Хебба. Каждый шаблон, в котором пиксели i и j имеют одинаковое значение, вносит положительный вклад в вес w_{ij}.

Каждый другой паттерн вносит отрицательный вклад. Поскольку у каждого пикселя

Machine Translated by Google

соответствует нейрону, здесь усиливаются веса между нейронами, которые одновременно имеют одинаковое значение. Обратите внимание на это небольшое отличие от правила Хебба.

После сохранения всех шаблонов сеть можно использовать для распознавания образов. Задаем сети новый паттерн x и обновляем активации всех нейронов в асинхронном процессе по правилу

$$x_i = \begin{cases} 1, & \text{если } \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} x_j < 0, \\ 1 & \text{еще} \end{cases} \tag{9.2}$$

пока сеть не станет стабильной, то есть пока не перестанут меняться активации. В виде схемы программы это выглядит следующим образом:

```
ХОПФИЛДАССОЦИАТОР(q)
Инициализировать все нейроны
x = q
    Repeat i = Random(1, n)
    Обновить нейрон i согласно (9.2)
Пока x не сходится
Возврат (x)
```

9.2.1 Применение к примеру распознавания образов

Применим описанный алгоритм к простому примеру распознавания образов. Он должен распознавать цифры в поле размером 10× 10 пикселей. Таким образом, сеть Хопфилда состоит из 100 нейронов с общим количеством

$$\frac{100 \cdot 99}{2} = 4950$$

веса. Сначала обучаются шаблоны цифр 1, 2, 3, 4 на рис. 9.6 на стр. 228 выше. То есть веса рассчитываются по (9.1) на стр. 226. Затем мы вставляем шаблон с добавленным шумом и позволяем динамике Хопфилда работать до сходимости.
В строках со 2 по 4 на рисунке показаны пять снимков развития сети в процессе распознавания. При уровне шума 10 % все четыре выученных паттерна распознаются очень надежно. Шум выше примерно 20% алгоритм часто сходится к другим изученным шаблонам или даже к шаблонам, которые не были изучены. Несколько таких паттернов показаны на рис. 9.6 на стр. 228 ниже.

Теперь мы сохраняем цифры от 0 до 9 (рис. 9.7 на стр. 229 вверху) в той же сети и снова тестируем сеть с шаблонами, которые имеют случайное количество инвертированных пикселей около 10%. На рисунке мы сно видим, что итерация Хопфилда часто не сходится к наиболее похожему обученному состоянию даже для всего 10% шума. Очевидно

Machine Translated by Google

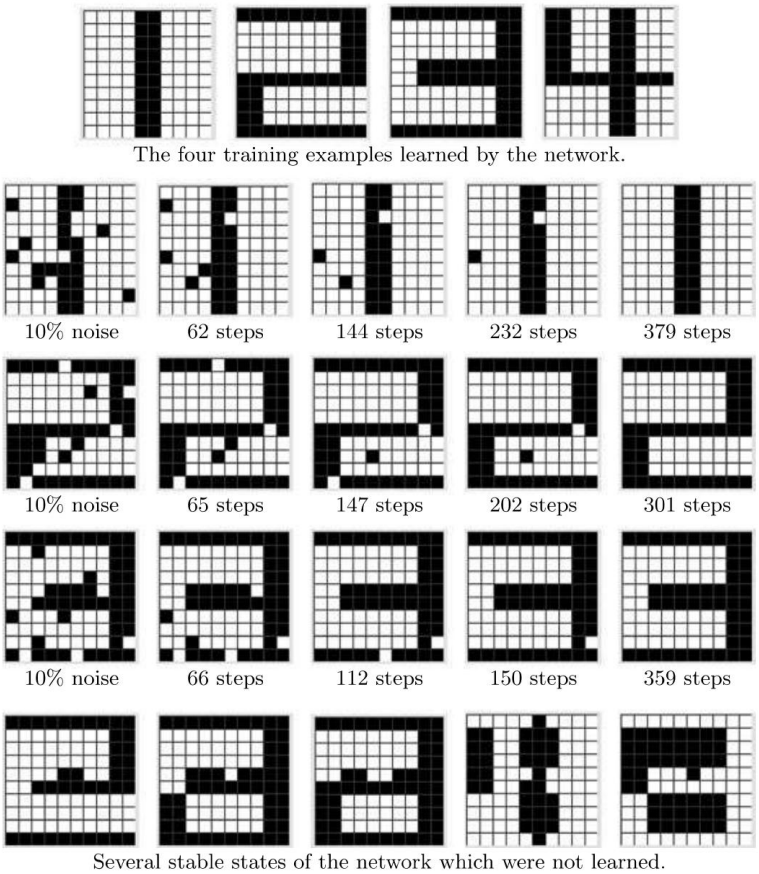


Рис. 9.6 Динамика сети Хопфилда. В строках 2, 3 и 4 мы можем легко увидеть, как сеть сходится, и изученный шаблон распознается примерно после 300–400 итераций. В последней строке показаны несколько устойчивых состояний, которые достигаются сетью, когда входной шаблон слишком сильно отклоняется от всех изученных шаблонов.

сеть может надежно сохранить и распознать четыре шаблона, но для десяти шаблонов ее объем памяти превышен. Чтобы лучше понять это, мы кратко рассмотрим теорию этой сети.

9.2.2 Анализ

В 1982 году Джон Хопфилд показал в [Hop82], что эта модель формально эквивалентна физической модели магнетизма. Маленькие элементарные магниты, так называемые спины, взаимно влияют друг на друга своими магнитными полями (см. рис. 9.8 на стр. 229). Если мы

Machine Translated by Google

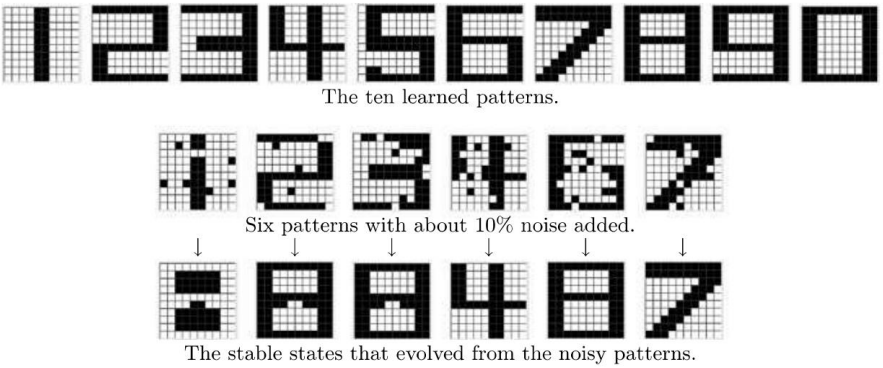


Рис. 9.7. Для десяти изученных состояний сеть показывает хаотичное поведение. Даже при небольшом шуме сеть сходится к неправильным паттернам или к артефактам

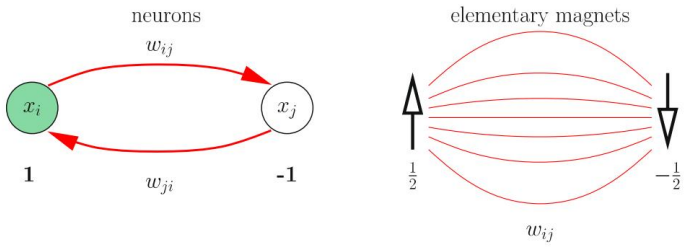


Рис. 9.8 Сравнение нейронной и физической интерпретации модели Хопфилда

наблюдаем два таких спина i и j , они взаимодействуют с постоянной величиной wij , и тогда полная энергия системы равна

$$E = -2 \frac{1}{\text{я,дж}} \text{вид}_{\text{ж}} x_i x_j .$$

К статье, и в физике wii = 0, потому что частицы не взаимодействуют друг с другом. Поскольку у физических взаимодействия симметричны, wij = wj i.

Физическая система, находящаяся в равновесии, переходит в (устойчивое) состояние с минимальной энергией и, таким образом, минимизирует E(x, y). Если такую систему привести в произвольное состояние, то она движется к состоянию минимальной энергии. Динамика Хопфилда, определенная в (9.2) на стр. 227 , точно соответствует этому принципу, потому что она обновляет состояние на каждой итерации таким образом, что из двух состояний 1 и 1 принимается то, которое имеет меньшую полную энергию Вклад нейрона i в общую энергию равен

$$\frac{1}{2} x_i \sum_{\text{дж=я}}^{\text{н}} \text{вид}_{\text{ж}} x_{\text{дж}} .$$

Machine Translated by Google

Если сейчас

$$\sum_{j=1}^{n} w_{ij} x_j < 0,$$

тогда $x_i = -1$ приводит к отрицательному вкладу в полную энергию, а $x_i = 1$ дает положительный вклад. При $x_i = -1$ сеть переходит в состояние с меньшей энергией, чем при $x_i = 1$. Аналогично можно утверждать, что в случае

$$\sum_{j=1}^{n} w_{ij} x_j \geq 0,$$

должно быть верно, что $x_i = 1$.

Если каждая отдельная итерация нейронной динамики приводит к уменьшению функции энергии, то полная энергия системы монотонно уменьшается со временем. Поскольку существует только конечное число состояний, сеть со временем переходит в состояние с минимальной энергией. Теперь у нас есть волнующий вопрос: что означают эти минимумы функции энергии?

Как мы видели в эксперименте по распознаванию образов, в случае нескольких выученных образов система сходится к одному из выученных образов. Выученные паттерны представляют собой минимумы энергетической функции в пространстве состояний. Однако если усвоено слишком много паттернов, то система сходится к минимумам, которые не соответствуют выученным паттернам. Здесь мы имеем переход от упорядоченной динамики к хаотической. один.

Хопфилд и другие физики исследовали именно этот процесс и показали, что фазовый переход действительно имеет место при критическом числе выученных паттернов.
Если количество выученных паттернов превышает это значение, то система переходит из упорядоченной фазы в хаотичную

В магнитной физике происходит такой переход от ферромагнитного режима, при котором все элементарные магнитыстремятся ориентироваться параллельно, к так называемому спиновому стеклу, в котором спины взаимодействуют хаотически. Более наглядным примером такого физического фазового перехода является таяние кристалла льда. Кристалл находится в состоянии высокого порядка, поскольку у молекулы H_2O строго упорядочены В жидкой воде, напротив, структура молекул растворена, а их положение более хаотично.

Затем в нейронной сети происходит фазовый переход от упорядоченного обучения и распознавания паттернов к хаотическому обучению и случае слишком большого количества паттернов, которые уже нельзя распознать наверняка. Здесь мы определенно видим параллели с эффектами, которые мы ранее испытывали сами.

Мы можем понять этот фазовый переход [RMS92], если мы поместим все нейроны а и вставим состояние шаблона, например q^1, из изученные веса из (9.1) на странице 226 , что актуально в терминах для обновления нейрона i. Это приводит к $\sum_{j=1, j\neq i}^{n} w_{ij} q_j$, в

Machine Translated by Google

$$\sum_{j=1,\, j\neq i}^{N} w_{ij} q_j \overset{\text{"="}}{=} \frac{1}{n} \sum_{j=1}^{N} \sum_{k=1,\, j\neq i}^{n} \xi_i^k \xi_j^k q_j \overset{\text{"="}}{=} \frac{1}{n} \sum_{j=1,\, j\neq i}^{N} \sum_{k=2}^{n} \xi_i^k \xi_j^k q_j \quad (\text{поскольку } \xi_i^1 \xi_j^1 q_j \approx q_i)$$

$$= q_i \left(1 + \frac{1}{n} \sum_{j=1}^{N} \sum_{k=2}^{n} \xi_i^k \xi_j^k q_j\right).$$

Здесь мы видим i-й компонент входного шаблона плюс сумму с (n 1)(N 1) элементами. Если все эти слагаемые статистически независимы, то мы можем описать сумму нормально распределенной случайной величиной со стандартным отклонением

$$\frac{1}{n} \sqrt{(n-1)(N-1)} \approx \sqrt{\frac{N-1}{n}}.$$

Статистическая независимость может быть достигнута, например, с помощью некоррелированных случайных закономерностей. Затем сумма генерирует шум, который является разрушительным до тех пор, пока N n, а это означает, что количество выученных паттернов остается намного меньше, чем количество нейронов. Если же N n, то влияние шума становится таким же большим, как и паттерн, и сеть реагирует хаотично. Более точный расчет фазового перехода дает критической точкой N = 0,146 n. Применительно к нашему примеру это означает, что для 100 нейронов можно сохранить до 14 паттернов. Однако, поскольку у закономерности в примере сильно коррелированы, критическое значение намного ниже, очевидно, между 0,04 и 0,1. Даже значение 0,146 намного меньше, чем емкость памяти традиционного списка памяти (упражнение 9.3 на стр. 255).

Сети Хопфилда в представленной форме хорошо работают только тогда, когда изученные шаблоны содержащие примерно 50% 1-битов. Если биты распределены очень асимметрично, то нейроны должны быть снабжены порогом [Roj96]. В физике это аналогично приложению внешнего магнитного поля, которое также вызывает асимметрию состояний со спином 1/2 и со спином -1/2.

9.2.3 Резюме и перспективы

Благодаря своему биологически правдоподобию, хорошо понятной математической модели и, прежде всего, впечатляющим симуляциям распознавания образов, модель Хопфилда способствовала волне интереса к нейронным сетям и росту нейроинформатики как важной ветви ИИ.2 В последствии было разработано много других сетевых моделей. С одной стороны, были исследованы сети без обратной связи, потому что их динамика значительно легче для понимания, чем рекуррентные сети Хопфилда. С другой стороны, были предприняты попытки улучшить емкость хранения данных в сетях, о чем мы поговорим в следующем разделе.

2Даже автора подхватила эта волна, унесшая его из физики в ИИ в 1987 году.

Machine Translated by Google

Особая проблема многих нейронных моделей уже проявилась в модели Хопфилда. Даже если существует гарантия сходимости, неясно, сойдется ли сеть в желаемое состояние или застрянет на локальном минимуме. Машина Больцмана с непрерывными значениями активации и вероятностным правилом обновления для ее сетевой динамики была разработана как попытка решить эту проблему. Используя параметр «температура», мы можем варьировать количество случайных изменений состояния и, таким образом, пытаться избежать локальных минимумов с целью найти устойчивый глобальный минимум. Этот алгоритм называется «имитация отжига». Отжиг — это процесс термической обработки металлов с целью сделать металл более прочным и «стабильным».

Модель Хопфилда осуществляет поиск минимума функции энергии в пространстве значений активации. Таким образом, он находит образец, сохраненный в весах и, таким образом, представленный в функции энергии. Динамика Хопфилда также может быть применена к другим функциям энергии, если матрица весов симметрична, а диагональные элементы равны нулю. Это было успешно продемонстрировано Хопфилдом и Танком на задаче коммивояжера [HT85, Zel94]. Задача здесь состоит в том, чтобы, зная n городов и их матрицу расстояний, найти кратчайший путь туда и обратно, который посещает каждый город ровно один раз.

9.3 Нейронная ассоциативная память

Традиционная память списка может в простейшем случае быть текстовым файлом, в котором строки цифр сохраняются строка за строкой. Если файл отсортирован построчно, то поиск элемента можно осуществить очень быстро за логарифмическое время даже для очень больших файлов.

Однако память списков также может использоваться для создания отображений. Например, телефонная книга представляет собой отображение набора всех введенных имен на набор всех телефонных номеров. Это сопоставление реализовано в виде простой таблицы, обычно сохраняемой в базе данных.

Аналогичная задача — контроль доступа в здание с помощью распознавания лиц. Здесь мы также могли бы использовать базу данных, в которой сохраняется фотография каждого человека вместе с именем человека и, возможно, другими данными. Затем камера на входе делает снимок человека и ищет в базе данных идентичную фотографию. Если фото найдено, то человек идентифицируется и получает доступ в здание. Однако в здании с такой системой управления не будет много посетителей, потому что вероятность того, что текущая фотография в точности совпадает с сохраненной фотографией, очень мала.

В этом случае недостаточно просто сохранить фото в таблице. Скорее, нам нужна ассоциативная память, способная не только присвоить нужное имя фотографии, но и любой из потенциально бесконечного набора «похожих» фотографий. Функциональность поиска сходства необходимо сгенерировать из конечного набора обучающих данных, а именно сохраненных фотографий, помеченных именами. Простым подходом для этого является метод ближайшего соседа, представленный в разд. 8.3. Во время обучения все фотографии просто сохраняются.

Для применения этой функции в базе данных должна быть найдена фотография, наиболее похожая на текущую. Для баз данных с большим количеством фотографий высокого разрешения этот процесс, в зависимости от используемой метрики расстояния, может потребовать очень большого времени вычислений и

Machine Translated by Google

поэтому не может быть реализован в этой простой форме. Поэтому вместо такого ленивого алгоритма мыпредпочтем тот, который передает данные в функцию, которая затем создает очень быструюассоциацию при ее применении.

Поиск подходящей метрики расстояния представляет собой еще одну проблему. Мыбы хотели, чтобыч еловека узнавали, даже если его лицо поя вляется в другом месте на фото (переводе), или если оно меньше, больше или даже повернуто. Уголобзора и освещение также могут различаться .

Именно здесь нейронные сети показывают свои сильные стороны Не требуя от разработчика думать о подходящей метрике сходства, они по-прежнему дают хорошие результаты Мыпредставим две простейшие модели ассоциативной памяти и начнем с модели Теувоо Кохонена, одного из пионеров в этой области.

Модель Хопфилда, представленнуюв предыдущей главе, было быслишком сложно использовать по двум причинам. Во-первых, это лишь автоассоциативная память, т. е. примерно тождественное отображение, сопоставляющее сходные объекты с выученным оригиналом. Во-вторых, сложной рекуррентной динамикой часто трудно управлять на практике. Поэтому сейчас мырассмотрим простые двухуровневые сети с прямой связью

9.3.1 Память корреляционной матрицы

В [Koh72] Кохонен представил ассоциативнуюмодель памяти, основаннуюна элементарной линейной алгебре. Это отображает векторызапроса $x^{\mu} \in \mathbb{R}^n$ в векторырезультатов $y \in \mathbb{R}^m$. Мы ищем матрицу W, которая правильно отображает набор обучающих данных.

$$T = \{(q^1, t^1), \ldots, (q^N, t^N)\}$$

с N запрос-ответ связывает все векторызапросов с их ответами.[3] То есть для $p = 1,\ldots,N$ должно быть так, что

$$t^p = W \cdot q^p, \tag{9.3}$$

или

$$t_i^p = \sum_{j=1}^{n} w_{ij} q_j^p \tag{9.4}$$

Для вычисления матричных элементов w_{ij} используется правило

$$w_{ij} = \sum_{p=1}^{N} t_i^p q_j^p \tag{9.5}$$

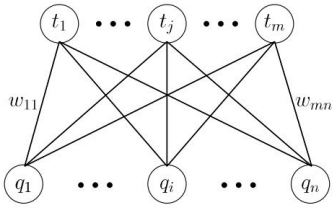
---

[3] Для четкого разграничения между обучающими данными и другими значениями нейрона в последующем обсуждении мыбудем называть вектор запроса q, а желаемый ответ — t (цель).

Machine Translated by Google

Рис. 9.9 Представление ассоциативной памяти

Кохонена в виде двухслойной нейронной сети



использует. Эти два линейных уравнения можно просто понимать как нейронную сеть, если мы определим, как на рис. 9.9, двухслойную сеть с q в качестве входного слоя и t в качестве выходного слоя. Нейроны выходного слоя имеют линейную функцию активации, а в качестве правила обучения используется (9.5), что в точности соответствует правилу Хебба.

Прежде чем мы покажем, что сеть распознает обучающие данные, нам понадобится следующее определение:

Определение 9.1 Два вектора x и y называются ортонормированными, если

$$T\ x \cdot y = \begin{cases} 1, \text{если } x = y, \\ 0 \text{ в противном случае.} \end{cases}$$

Таким образом

Теорема 9.1. Если все N векторов запроса q^p в обучающих данных ортонормированы, то каждый вектор q^p отображается в целевой вектор t^p путем умножения на матрицу w_{ij} из (9.5) на стр. 233.

Доказательство. Подставим (9.5) на стр. 233 в (9.4) на стр. 233 и получим

$$\begin{aligned}
(Вт \cdot к в^{п})_{я} &= \sum_{jj=1}^{н} w_{ij} q_{j}^{п} \\
&= \sum_{j=1}^{н} q_{j}^{п} \left(\sum_{г=p}^{Н} p_{кj} \right)_{i}^{p} q_{j}^{p} \\
&= \sum_{j=1}^{н} q_{j}^{п} \left(q_{j}^{п} + \sum_{г=p}^{Н} q_{j}^{п} \right)_{т_{я}}^{p} \\
&= т_{я}^{p} \sum_{j=1}^{н} д_{jq}^{п} j_{-}^{p} + \sum_{г=p}^{Н} т_{я}^{p} \sum_{j=1}^{н} д_{jq}^{p} j_{-}^{п} \\
&= т_{я}^{p} \underbrace{(q^{п})^{T}}_{=1} \underbrace{п q}_{} + т_{я}^{p} \sum_{г=p}^{Н} \underbrace{(q^{p})^{T}}_{=0} \underbrace{pp q}_{} = t_{i}
\end{aligned}$$

Machine Translated by Google

Таким образом, если векторы запроса ортонормированы, все шаблоны будут правильно сопоставлены с соответствующими целями. Однако ортонормированность является слишком сильным ограничением. В разд. 9.4 мы представим подход, преодолевающий это ограничение.

Поскольку линейные отображения непрерывны и инъективны, мы знаем, что отображение из векторов запроса в целевые векторы сохраняет сходство. Таким образом, аналогичные запросы сопоставляются с аналогичными целями из-за непрерывности. Однако в то же время мы знаем, что разные запросы сопоставляются с разными целями. Если сеть была обучена отображать лица в имена и если лицу присвоено имя Генри, то мы уверены, что на вход похожего лица будет выдан аналогичный «Генри», но сам «Генри» гарантированно не вычисляется. Если вывод можно интерпретировать как строку, то, например, это может быть «Genry» или «Gfnry». Чтобы получить наиболее похожий заученный случай, Кохонен использует двоичное кодирование для выходного нейрона. Вычисленный результат запроса округляется, если его значение не равно нулю или единице. Даже тогда у нас нет гарантии, что мы попадем в целевой вектор. В качестве альтернативы мы могли бы добавить последующее сопоставление вычисленного ответа с изученным целевым вектором с наименьшим расстоянием.

9.3.2 Псевдоинверсия

Существует другой подход, который мы можем использовать для вычисления весовой матрицы W: 1 думая обо всех векторах запросов как о столбцах матрицы $Q = (q_1 | \ldots | q_N)$ и аналогично целевые векторы как столбцы матрицы $T = (t_1 | \ldots | t_N)$, мы можем привести (9.3) на стр. 233 к форме

$$T = W \cdot Q. \tag{9.6}$$

Теперь попытаемся решить это уравнение относительно W. Формально инвертируем и получаем

$$W = T \cdot Q^{-1}. \tag{9.7}$$

Требованием для этого преобразования является обратимость Q. Для этого матрица должна быть квадратной и должна состоять из линейно независимых вектор-столбцов. То есть должно быть так, что $n = N$ и все n векторов запросов должны быть линейно независимыми. Это условие действительно неудобно, но все же не столь строго, как требуемая выше ортонормированность. Однако обучение по правилу Хебба имеет то преимущество, что мы можем просто применить его, даже если векторы запросов не являются ортонормированными, в надежде, что ассоциатор, тем не менее, будет работать сносно. Однако здесь это не так просто, ибо как инвертировать необратимую матрицу?

Матрица Q обратима тогда и только тогда, когда существует матрица Q^{-1} со свойством

$$Q \cdot Q^{-1} = I, \tag{9.8}$$

где I — единичная матрица. Если Q необратима (например, потому что Q неквадратная), то не существует матрицы Q^{-1} с таким свойством. Однако определенно существует матрица, которая близка к этому свойству. В этом смысле мы определяем

Machine Translated by Google

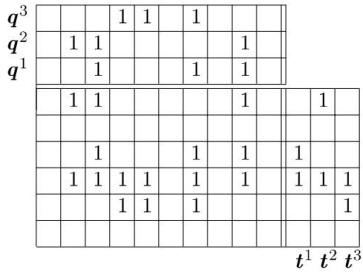


Рис. 9.10. Матрица W после сохранения трех пар $(q^1, t), (q^2, t), (q^3, t)$. Пустые поля соответствуют значению 0

Определение 9.2. Пусть Q — вещественная матрица размера $n \times m$. Матрица размера $m \times n$ Q^+ называется псевдообратной к Q, если она минимизирует

$$\|Q \cdot Q^+ - I\|.$$

Здесь $\|M\|$ — евклидова норма.

Теперь, используя

$$W = T \cdot Q^+ \tag{9.9}$$

мы можем рассчитать весовую матрицу, которая минимизирует перекрестные помехи (ошибки ассоциации) $\|T - W \cdot Q\|$. Существуют различные способы вычисления псевдообратного, например метод наименьших квадратов, который мы введем в разд. 9.4.1.

9.3.3 Бинарное правило Хебба

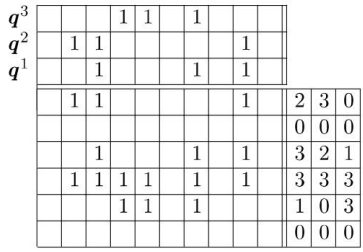
В контексте ассоциативной памяти было предложено так называемое бинарное правило Хебба. Это требует, чтобы шаблон был закодирован в двоичном виде. Это означает, что для всех образов q^p и t^p $\in \{0, 1\}^m$. Кроме того, сумма из (9.5) на стр. 233 равна $\{0, 1\}$ заменяется простым логическим ИЛИ, и мы получаем бинарное правило Хебба

$$w_{ij} = \bigvee_{p=1}^{N} t_i^p \wedge q_j^p. \tag{9.10}$$

Таким образом, весовая матрица также является двоичной, и матричный элемент w_{ij} равен единице, если он и только если хотя бы один из элементов $q_j^1, \ldots, q_j^N$ не равен нулю. Все остальные матрицы элементы равны нулю. Мысленно полагать, что при обучении здесь теряется много информации, потому что, когда элемент матрицы принимает значение 1 один раз, его нельзя изменить дополнительными шаблонами. На рис. 9.10 показано, как матрица заполняется единицами для примера с $n = 10$, $m = 6$ после изучения трех пар.

Machine Translated by Google

Рис. 9.11 Расчет произведения W q1 , W q2 , W q3



Чтобы получить сохраненные шаблоны, мы просто умножаем вектор запроса q на матрицу и посмотрите на результат W q. Проверим это на примере и получим рис. 9.11.

Мы видим, что в целевом векторе с правой стороны стоит значение 3 в том месте, где у изученного целевого вектора была единица. Правильные результаты будут получены при установке порогового значения 3. В общем случае мы выбираем количество единиц в векторе запроса в качестве порога. Таким образом, каждый выходной нейрон работает как персептрон, хотя и с переменным порогом.

Пока матрица весов разрежена, этот алгоритм работает хорошо. Однако, если сохраняется много разных паттернов, матрица становится все более и более плотной. В крайнем случае он содержит только единицы. Тогда, после установки порога, все ответы будут состоять только из единиц и больше не будут содержать никакой информации.

Этот случай возникает редко, пока количество битов, сохраняемых в матрице, не становится слишком большим. Матрица имеет размер mn элементов. Сохраняемая пара имеет m + n бит . Можно показать [Pal80], что количество запоминаемых паттернов Nmax определяется следующим условием:

$$\alpha = \frac{\text{количество сохраняемых битов}}{\text{количество элементов бинарной матрицы}}$$

$$\text{"="} \frac{(m+n)N_{max}}{mn}\ln 2 \quad 0{,}69. \tag{9.11}$$

Для памяти списка мы имеем α = 1. Ассоциативная память с двоичным правилом Хебба имеет максимальную эффективность памяти α = 0,69 по сравнению с α = 0,72 для ассоциативной памяти Кохонена и α = 0,292 для сетей Хопфилда [Pal80, Pal91]. Таким образом, емкость памяти бинарного правила Хебба на удивление высока по сравнению с моделью Кохонена с непрерывными нейронами.

Очевидно, что такая память «заполняется» менее быстро, когда вектор запроса и целевой вектор заполнены единицами редко. Это не единственная причина, по которой кодирование входных и выходных данных для ассоциативной памяти, как и для других нейронных сетей, очень важно для хорошей работы. Теперь мы продемонстрируем это на приложении этой памяти с соответствующим образом выбранными кодировками ввода и вывода.

Machine Translated by Google

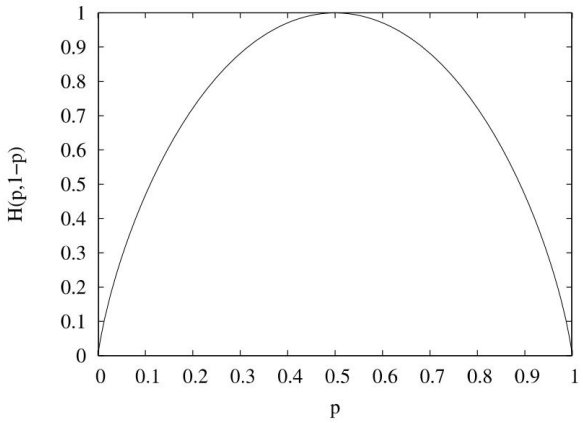


Рис. 8.24. Функция энтропии для случая двух классов. Мы видим максимум при p = 1/2 и симметрию относительно перестановки p и 1 p

Определение 8.4. Энтропия H как метрика неопределенности распределения вероятностей определяется формулой[7]

$$H(p) = H(p_1,\ldots,p_n) := \sum_{i=1}^{n} p_i \log_2 p_i .$$

Подробный вывод этой формулы можно найти в [SW76]. Если подставить некое событие p = (1, 0, 0,..., 0), то 0 log2 0, получится неопределенное выражение. Мы решаем эту задачу с помощью определения 0 log2 0 := 0 (см. упражнение 8.9 на стр. 218).

Теперь мы можем вычислить H (1, 0,..., 0) = 0. Покажем, что энтропия в гиперкубе [0, 1]^n при ограничении $\sum_{i=1}^{n} p_i = 1$ принимает максимальное значение $\log_2 n$ (равномерным распределением $\frac{1}{n},\ldots,\frac{1}{n}$). В случае события с двумя возможными исходы, соответствующие двум классам, результат

$$H(p) = H(p_1, p_2) = H(p_1, 1 - p_1) = (p_1 \log_2 p_1 + (1 - p_1)\log_2 (1 - p_1)).$$

Это выражение показано как функция p1 на рис. 8.24 с максимумом при p1 = 1/2.

Поскольку каждому классифицированному набору данных D присваивается распределение вероятностей p^D путем оценки вероятностей класса, мы можем расширить понятие энтропии на данные с помощью

[7] В (7.9) на стр. 124 для определения энтропии используется натуральный логарифм, а не log2 . Поскольку здесь, как и в случае с методом MaxEnt, только сравниваются энтропии, эта разница не играет роли. (см. упражнение 8.11 на стр. 218).

Machine Translated by Google

Определение

$$H(D) = H(p).$$

Теперь, поскольку информационное содержание I(D) набора данных D должно быть противоположным неопределенности. Таким образом, мы определяем:

Определение 8.5 Информационное содержание набора данных определяется как

$$I(D) := 1 \quad H(D). \tag{8.6}$$

8.4.3 Получение информации

Если мы применим формулу энтропии к примеру, результат будет

$$H(6/11, 5/11) = 0{,}994$$

При построении дерева решений набор данных дополнительно подразделяется по каждому новому атрибуту. Чем больше атрибут повышает информативность распределения за счет разделения данных, тем лучше этот атрибут. Соответственно определяем:

Определение 8.6 Прирост информации G(D, A) за счет использования атрибута A определяется разностьюсредней информативности набора данных D = D1 D2 ⋯ Dn, деленной на n-значение атрибута A и информативность I(D) неразделенного набора данных, что дает

$$G(D, A) = \sum_{i=1}^{n} \frac{|Di|}{|D|} + I(Di) \quad I(D).$$

С учетом (8.6) отсюда получаем

$$G(D, A) = \sum_{i=1}^{n} \frac{|Di|}{|D|} I(Di) - I(D) = \sum_{i=1}^{n} \frac{|Di|}{|D|} (1 - H(Di)) - (1 - H(D))$$

$$= 1 - \sum_{i=1}^{n} \frac{|Di|}{|D|} H(Di) - 1 + H(D)$$

$$= H(D) - \sum_{i=1}^{n} \frac{|Di|}{|D|} + H(Di). \tag{8.7}$$

Machine Translated by Google

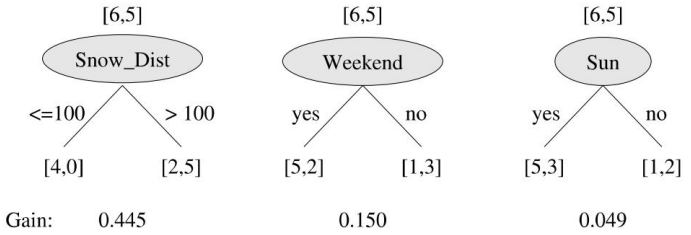


Рис. 8.25 Расчетный коэффициент усиления для различных атрибутов отражает, приводит ли разделение данных по соответствующему атрибуту к лучшему разделению классов. Чем больше распределения, генерируемые атрибутом, отклоняются от равномерного распределения, тем выше прирост информации.

Применительно к нашему примеру для атрибута Snow_Dist это дает

$$G(D, \text{Snow_Dist}) = H(D) - \frac{4}{11}H(D \le 100) + \frac{7}{11}H(D > 100)$$

$$= 0{,}994 - \frac{4}{11} \cdot 0 + \frac{7}{11} \cdot 0{,}863 = 0{,}445.$$

Аналогично получаем

$$G(\text{Дневные}) = 0{,}150$$

и

$$G(D, \text{Солнце}) = 0{,}049.$$

Атрибут Snow_Dist теперь становится корневым узлом дерева решений. Ситуация с выбором этого атрибута еще раз поясняется на рис. 8.25.

Два значения атрибута 100 и >100 генерируют два ребра в дереве, которые соответствуют подмножествам D 100 и D>100. Для подмножества D 100 классификация однозначна да. таким образом, дерево заканчивается здесь. В другой ветке D>100 однозначного результата нет. Таким образом, алгоритм повторяется рекурсивно. Из двух все еще доступных атрибутов, Sun и Weekend, необходимо выбрать лучший. Мы рассчитываем

$$G(D>100, \text{Дневные}) = 0{,}292$$

и

$$G(D>100, \text{Солнце}) = 0{,}170.$$

Таким образом, узлу назначается атрибут Weekend . Для Weekend = no дерево завершается решением Ski = no. Расчет усиления здесь возвращает значение 0. Для Weekend = yes, Sun дает прирост 0,171. Затем построение дерева прекращается, так как больше нет доступных атрибутов, хотя пример номер 7 классифицируется ложно. Готовое дерево уже знакомо по рис. 8.23 на стр. 186.

Machine Translated by Google

8.4.4 Применение C4.5

Дерево решений, которое мы только что сгенерировали, также может быть сгенерировано с помощью C4.5. Данные тренировки сохраняются в файле данных ski.data в следующем формате:

```
<=100, да, да, да <=100, да, да, да
<=100, да, нет, да <=100, нет, да, да
>100, да, да, да >100, да, да , да >100,
да, да, нет >100, да, нет, нет >100,
нет, да, нет >100, нет, да, нет >100,
нет,




                        нет нет
```

Информация об атрибутах и классах хранится в файле ski.names.
(строки, начинающиеся с «|», являются комментариями):

```
 |Классы: нет: не кататься на лыжах, да: кататься на лыжах |

нет да.
 |
 |Атрибуты |

Снег_Расстояние:          <=100,>100.
Выходные:                 нет да.
Солнце:                   нет да.
```

Затем C4.5 вызывается из командной строки Unix и генерирует дерево решений, показанное ниже, которое отформатировано с использованием отступов. Параметр -f предназначен для имени входного файла, а параметр -m указывает минимальное количество точек обучающих данных, необходимых для создания новой ветви в дереве. Поскольку количество точек обучающих данных в этом примере чрезвычайно мало, здесь уместна опция -m 1. Для больших наборов данных следует использовать значение не менее -m 10.

Machine Translated by Google


```
unixprompt> c4.5 -f ski -m 1

C4.5 [версия 8] генератор дерева решений

                                                Ср 23 Авг 10:44:49 2010
--------------------------------------------

    Параметры:
        Основа файла <ski>
        Разумный тест требует 2 ветвей с >=1 случаем

Считать 11 случаев (3 атрибута) из ski.data

Дерево решений:

Snow_Dist = <=100: ja (4.0)
Snow_Dist = >100:
 | Выходные = нет: нет (3.0)
 | Выходные = да:
 | | Солнце = нет: нет (1.0)
 | | Солнце = да: да (3,0/1,0)

Упрощенное дерево решений:

Snow_Dist = <=100: да (4.0/1.2)
Snow_Dist = >100: нет (7.0/3.4)

Оценка на обучающих данных (11 пунктов):

        Перед обрезкой                  После обрезки
        ----------------                -----------------------------
        Размер      Ошибки              Размер      Ошибки          Оценивать

          7         1(9,1%)               3         2(18,2%)        (41,7%) <<
```

Кроме того, дается упрощенное дерево только с одним атрибутом. Это дерево, которое был создан путем обрезки (см. раздел 8.4.7), будет важен для увеличения количества обучающих данных. В этом маленьком примере это пока не имеет особого смысла. Ошибка Также указана скорость для обоих деревьев на обучающих данных. Цифры в скобках после решений укажите размер базового набора данных и количество ошибок. Например, строка Sun = yes: yes (3.0/1.0) в верхнем дереве указывает что для этого листового узла Sun = yes существуют три обучающих примера, один из которых ложно классифицированы. Таким образом, пользователь может прочитать из этого, является ли решение статистически обоснованным и/или достоверным.

На рис. 8.26 на стр. 193 теперь мы можем представить схему алгоритма обучения . для построения дерева решений.

Теперь мы знакомы с основами автоматической генерации решений. деревья . Однако для практического применения необходимы важные расширения . Мы представит их с помощью уже знакомого приложения LEXMED .

Machine Translated by Google


```
СОЗДАТЬ ДЕРЕВО РЕШЕНИЙ(данные, узел)
Amax = Атрибут с максимальным информационным приростом.
Если G(Amax) = 0,
    то Узел становится конечным узлом с наиболее частым классом в данных. В
    противном случае присвойте узлу атрибут
        Amax. Для каждого значения a1,...,an Amax
                                        сгенерируйте последующий узел: K1,...,Kn
        Разделите данные на D1,...,Dn с Di = {x   Data|Amax(x) = ai}
        Для всех i   {1,...,n}
            Если все x   Di принадлежат одному классу Ci
                Затем сгенерируйте листовой узел Ki класса Ci
                Иначе GENERATEDECISIONTREE(Di, Ki)
```

Рис. 8.26 Алгоритм построения дерева решений

8.4.5 Обучение диагностике аппендицита

В исследовательском проекте LEXMED была разработана экспертная система для диагностики аппендицита на основе базы данных пациентов [ES99, SE00]. Система, работающая по методу максимальной энтропии, описана в разд. 7.3

Теперь мы используем базу данных LEXMED для создания дерева решений для диагностики аппендицита с C4.5. Симптомы, используемые в качестве атрибутов, определены в файле app.names:

```
|Определение классов и атрибутов| | Классы 0 =

отрицательный аппендицит | 1=положительный
аппендицит 0,1. | |Атрибуты|


Возраст:

непрерывный.
Пол_(1=м___2=ж): 1,2.
Боль_Квадрант1_(0=нет__1=да): 0,1.
Боль_Квадрант2_(0=нет__1=да): 0,1.
Боль_Квадрант3_(0=нет__1=да): 0,1.
Боль_Квадрант4_(0=нет__1=да): 0,1.
Локальная _охрана_(0=нет__1=да): 0,1.
Обобщенная _охрана_(0=нет_1=да): 0,1.
Отскок_нежность_(0=нет__1=да): 0,1.
Боль_при_постукивании_(0=нет__1=да): 0,1.
```

Machine Translated by Google


```
Боль_во время _ректального_обследования _(0=нет_1=да): 0,1.
Temp_axial: непрерывный.
Temp_rectal: непрерывный.
Лейкоциты: постоянно.
Сахарный диабет_(0=нет_1=да): 0,1
```

Мы видим, что помимо многих бинарных признаков, таких как различные болевые симптомы, также встречаются постоянные симптомы, такие как возраст и лихорадка. В следующем файле обучающих данных, app.data, в каждой строке описывается случай. В первой строке пациент 19 лет с болью в третьем квадранте (справа внизу, где аппендикс), двумя показателями лихорадки 36,2 и 37,8°C, лейкоцитами 13400 и положительным диагнозом, что есть воспаленный аппендикс.

```
19,1,0,0,1,0,1,0,1,1,0,362,378,13400,0,1 13,1,0,0,1,0,1,0,1,1,1,383,385,18100
,0,1 32,2,0,0,1,0,1,0,1,1,0,364,374,11800,0,1 18,2,0,0,1,1,0,0,0,0
,0,362,370,09300,0,0 73,2,1,0,1,1,1,0,1,1,1,376,380,13600,1,1 30,1,1,1,1,1,0,1
,1,1,1,377,387,21100,0,1 56,1,1,1,1,1,0,1,1,1,0,390,?,14100,0,1 36,1,0,0,1
,0,1,0,1,1,0,372,382,11300,0,1 36,2,0,0,1,0,0,0,1,1,1,370,379,15300,0,1 33,1,0
,0,1,0,1,0,1,1,0,367,376,17400,0,1 19,1,0,0,1,0,0,0,1,1,0,361,375,17600,0,1 12
,1,0,0,1,0,1,0,1,1,0,364,370,12900,0,0




...
```

Не вдаваясь в детали базы данных, важно отметить, что в базу данных включены только пациенты, у которых при поступлении в стационар возникло подозрение на аппендицит, а затем они были прооперированы. Мы видим в седьмой строке, что C4.5 также может иметь дело с пропущенными значениями. Данные содержат 9764 случая.

```
unixprompt> c4.5 -f приложение -u -m 100

C4.5 [версия 8] генератор деревьев решений
                                                    Ср 23 августа 13:13:15 2006
----------------------------------------


Считать 9764 случая (15 атрибутов) из app.data

Дерево решений:
```

Machine Translated by Google


```
Лейкоциты <= 11030:
| Rebound_tenderness = 0:
| | Temp_rectal > 381 : 1 (135,9/54,2)
| | Temp_rectal <= 381 :
| | | Local_guarding = 0:0 (1453,3/358,9)
| | | Local_guarding = 1:
| | | | Пол_(1=м___2=ж) = 1:1 (160,1/74,9)
| | | | Пол_(1=м___2=ж) = 2:0 (286,3/97,6)
| Rebound_tenderness = 1:
| | Лейкоциты <= 8600:
| | | Temp_rectal > 378 : 1 (176,0/59,4)
| | | Temp_rectal <= 378 :
| | | | Пол_(1=м___2=ж) = 1:
| | | | | Local_guarding = 0:0 (110,7/51,7)
| | | | | Local_guarding = 1:1 (160,6/68,5)
| | | | Пол_(1=м___2=ж) = 2:
| | | | | Возраст <= 14 : 1 (131,1/63,1)
| | | | | Возраст > 14 : 0 (398,3/137,6)
| | Лейкоциты > 8600:
| | | Пол_(1=м___2=ж) = 1:1 (429,9/91,0)
| | | Пол_(1=м___2=ж) = 2:
| | | | Local_guarding = 1:1 (311,2/103,0)
| | | | Local_guarding = 0:
| | | | | Temp_rectal <= 375 : 1 (125,4/55,8)
| | | | | Temp_rectal > 375 : 0 (118,3/56,1)
Лейкоциты > 11030:
| Отскок_нежность = 1:1 (4300,0/519,9)
| Rebound_tenderness = 0:
| | Лейкоциты > 14040 : 1 (826,6/163,8)
| | Лейкоциты <= 14040:
| | | Боль_при постукивании = 1:1 (260,6/83,7)
| | | Боль при постукивании = 0:
| | | | Local_guarding = 1:1 (117,5/44,4)
| | | | Local_guarding = 0:
| | | | | Temp_axial <= 368 : 0 (131,9/57,4)
| | | | | Temp_axial > 368 : 1 (130,5/57,8)

Упрощенное дерево решений:

Лейкоциты > 11030 : 1 (5767,0/964,1)
Лейкоциты <= 11030:
| Rebound_tenderness = 0:
| | Temp_rectal > 381 : 1 (135,9/58,7)
| | Temp_rectal <= 381 :
| | | Local_guarding = 0:0 (1453,3/370,9)
| | | Local_guarding = 1:
| | | | Пол_(1=м___2=ж) = 1:1 (160,1/79,7)
| | | | Пол_(1=м___2=ж) = 2:0 (286,3/103,7)
| Rebound_tenderness = 1:
| | Лейкоциты > 8600 : 1 (984,7/322,6)
| | Лейкоциты <= 8600:
```

Machine Translated by Google


```
. . . | Temp_rectal > 378 : 1 (176,0/64,3)
. . . | Temp_rectal <= 378 :
. . . . | Пол_(1=м___2=ж) = 1:
. . . . | | Local_guarding = 0:0 (110,7/55,8)
. . . . | | Local_guarding = 1:1 (160,6/73,4)
. . . . | Пол_(1=м___2=ж) = 2:
. . . . | | Возраст <= 14 : 1 (131,1/67,6)
. . . . | | Возраст > 14 : 0 (398,3/144,7)

Оценка на обучающих данных (9764 элемента):

До обрезки После обрезки
---------- ----------
Ошибки размера Оценка ошибок размера
37 2197(22,5%) 21 2223(22,8%) (23,6%) <<

Оценка на тестовых данных (4882 позиции):

До обрезки После обрезки
---------- ----------
Ошибки размера Оценка ошибок размера
37 1148(23,5%) 21 1153(23,6%) (23,6%) <<

 (a) (b) <-классифицируется как
---
758 885 (a): класс 0
268 2971 (б): класс 1
```

8.4.6 Непрерывные атрибуты

В деревьях, сгенерированных для диагностики аппендицита, есть узел Лейкоциты.
> 11030, что явно происходит от непрерывного атрибута Лейкоциты
установив пороговое значение на уровне 11030. Таким образом, C4.5 создал двоичный атрибут Лейкоциты
> 11030 из непрерывного атрибута Лейкоциты.

Порог $\Theta D,A$ для признака A определяется по следующему алгоритму:
для всех значений v, которые встречаются в обучающих данных D, бинарный атрибут A>v равен
генерируется и рассчитывается его информационный прирост. Затем порог $\Theta D,A$ устанавливается равным
значение v с максимальным приростом информации, то есть:

$$\Theta D,A = \text{argmax}_v \{G(D, A > v)\}.$$

Для такого атрибута, как количество лейкоцитов или возраст пациента, решение, основанное на
бинарная дискретизация, по-видимому, слишком неточна. Тем не менее нет необходимости
дискретизировать лучше, потому что каждый непрерывный атрибут проверяется на каждом вновь сгенерированном
узел и, таким образом, может повторяться в одном дереве с разным порогом $\Theta D,A$. Таким образом
в конечном итоге мы получаем очень хорошую дискретизацию, точность которой соответствует задаче.

Machine Translated by Google

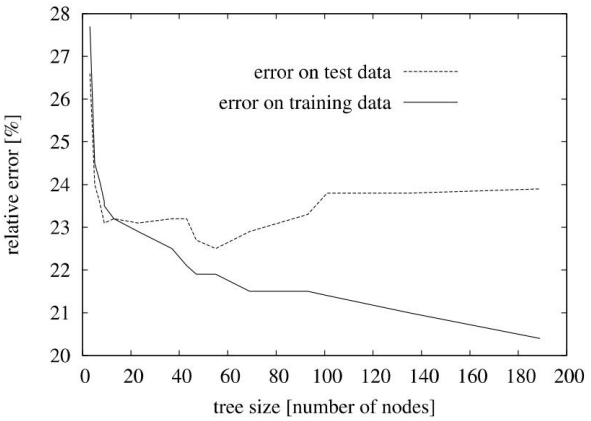


Рис. 8.27 Кривая обучения C4.5 на данных аппендицита. Мы я сно видим переоснащение деревье вс более чем 55 узлами.

8.4.7 Обрезка —спиливание дерева

Со времен Аристотеля было установлено, что из двух научных теорий, одинаково хорошо объясняющих одну и ту же ситуацию, предпочтение отдается более простой. Этот закон экономики, также известный как бритва Оккама, имеет большое значение для машинного обучения и интеллектуального анализа данных.

Дерево решений —это теория для описания обучающих данных. Другая теория описания данных —это сами данные. Если дерево классифицирует все данные без каких-либо ошибок, но гораздо более компактно и, следовательно, более понятно человеку, то оно предпочтительнее в соответствии с бритвой Оккама. То же верно и для двух деревье в решений разного размера. Таким образом, целью каждого алгоритма построения дерева решений должно быть создание наименьшего возможного дерева решений для данной частоты ошибок. Среди всех деревье в с фиксированной частотой ошибок всегда следует выбирать наименьшее дерево.

До сих пор мы не давали точного определения термину «частота ошибок». Как уже неоднократно упоминалось, важно, чтобы обученное дерево не просто запоминало обучающие данные, а хорошо обобщало. Чтобы проверить способность дерева к обобщению, мы делим доступные данные на набор обучающих данных и набор тестовых данных. Тестовые данные скрыты от алгоритма обучения и используются только для тестирования. Если доступен большой набор данных, например данные об аппендиците, то мы можем, например, использовать две трети данных для обучения, а оставшуюся треть —для тестирования.

Помимо лучшей понятности, у бритвы Оккама есть еще одно важное обоснование: способность к обобщению. Чем сложнее модель (в данном случае дерево решений), тем больше подробностей представлено, но в той же степени модель менее переносима на новые данные. Это соотношение показано на рис. 8.27. Деревья решений различных размеров были обучены на данных об аппендиците. На графике приведены ошибки классификации как на обучающих данных, так и на тестовых данных. Частота ошибок в обучающих данных монотонно уменьшается с размером дерева. При размере дерева до 55 узлов частота ошибок в тестовых данных также снижается. Однако если дерево растет дальше, то частота ошибок снова начинает увеличиваться! Этот эффект, который мы уже видели в методе ближайшего соседа, называется переоснащением.

Machine Translated by Google

Мы дадим это понятие, важное почти для всех процессов обучения, общее определение взято из [Mit97]:

> Определение 8.7. Пусть задан конкретный алгоритм обучения, то есть агент обучения. Мы называем агента A подогнанным к обучающим данным, если есть другой агент A', чья ошибка на обучающих данных больше, чем у A, но чья ошибка на всем распределении данных меньше, чем ошибка A.

Как мы теперь можем найти эту точку минимальной ошибки на тестовых данных? Наиболее очевидный алгоритм называется перекрестной проверкой. При построении дерева параллельно измеряется ошибка на тестовых данных. Как только ошибка значительно возрастает, сохраняется дерево с минимальной ошибкой. Этот алгоритм использует упомянутой ранее системой CART.

C4.5 работает несколько иначе. Во-первых, с помощью алгоритма GENERATEDECI SIONTREE, показанного на рис. 8.26 на стр. 193, создается дерево, которое обычно является переобученным. Затем, используя отсечение, он пытается отсечь узлы дерева до тех пор, пока ошибка на тестовых данных, оцениваемая по ошибке на обучающих данных, не начнет расти8. Подобно построению дерева, это также жадный алгоритм. Это означает, что после сокращения узла его нельзя вставить повторно, даже если позже это окажется лучше.

8.4.8 Отсутствующие значения

Часто в обучающих данных отсутствуют отдельные значения атрибутов. В наборе данных LEX MED встречается следующая запись:

56, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 390, ?, 14100, 0, 1,

при котором отсутствует одно из значений лихорадки. Тем не менее такие данные можно использовать при построении дерева решений. Мы можем присвоить атрибуту наиболее частое значение из всего набора данных или наиболее частое из всех точек данных того же класса. Еще лучше заменить отсутствующее значение атрибута распределением вероятностей всех значений атрибута и разбить обучающий пример на ветви в соответствии с этим распределением. Кстати, это является причиной появления нецелочисленных значений в выражениях в круглых скобках рядом с конечными узлами дерева C4.5.

Пропуски значений могут возникать не только при обучении, но и при классификации. Они обрабатываются так же, как и во время обучения.

---

8 Было бы лучше использовать ошибку непосредственно на тестовых данных. По крайней мере, когда объем обучающих данных достаточен для обоснования отдельного набора тестов.

Machine Translated by Google

8.4.9 Резюме

Изучение деревьев решений — излюбленный подход к задачам классификации. Причинами этого являются его простота применения и скорость. В наборе данных, состоящем примерно из 10 000 точек данных LEXMED с 15 атрибутами в каждой, для обучения C4.5 требуется около 0,3 секунды. Это очень быстро по сравнению с другими алгоритмами обучения.

Однако для пользователя также важно, чтобы дерево решений как изученная модель могла быть понята и потенциально изменена. Также нетрудно автоматически преобразовать дерево решений в серию операторов if-then-else и, таким образом, эффективно встроить его в существующую программу.

Поскольку для построения дерева, а также во время отсечения используется жадный алгоритм, деревья в целом не оптимальны. Обнаруженное дерево решений обычно имеет относительно небольшую частоту ошибок. Однако потенциально существует лучшее дерево, потому что эвристический жадный поиск C4.5 предпочитает маленькие деревья и атрибуты с высоким приростом информации в верхней части дерева. Для атрибутов со многими значениями представленная формула прироста информации показывает недостатки. Альтернативы этому приведены в [Mit97].

8.5 Изучение байесовских сетей

В разд. 7.4 показано, как построить байесовскую сеть вручную. Теперь мы введем алгоритмы индукции байесовских сетей. Подобно описанному ранее процессу обучения, байесовская сеть автоматически генерируется из файла, содержащего обучающие данные. Этот процесс обычно разбивается на две части.

1. Изучение структуры сети: для заданных переменных топология сети создается на основе обучающих данных. Этот первый шаг намного сложнее, и позже ему будет уделено более пристальное внимание.
2. Изучение условных вероятностей. Для известных сетевых топологий CPT должны быть заполнены значениями. Если доступно достаточно обучающих данных, все необходимые условные вероятности можно оценить, подсчитав частоты в данных.
   Этот шаг можно относительно легко автоматизировать.
   Теперь мы объясним, как байесовские сети обучаются, используя простой алгоритм из [Jen01].

8.5.1 Изучение структуры сети

При разработке байесовской сети (см. раздел 7.4.6) необходимо учитывать причинно-следственную зависимость переменных, чтобы получить простую сеть хорошего качества. Разработчик-человек полагается на фоновые знания, недоступные машине. Поэтому эту процедуру нельзя легко автоматизировать.

Поиск оптимальной структуры байесовской сети можно сформулировать как классическую задачу поиска. Пусть задан набор переменных V1,...,Vn и файл с обучающими данными. Ищем набор направленных ребер без циклов между узлами

Machine Translated by Google

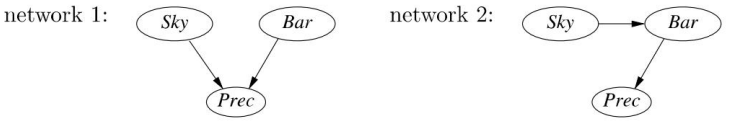


Рис. 8.28 Две байесовские сети для моделирования примера прогноза погоды из упражнения 7.3 на стр. 158

V1,...,Vn, то есть ориентированный ациклический граф (DAG), максимально точно воспроизводящий исходные данные.

Сначала мы наблюдаем пространство поиска. Количество различных DAG растет более чем экспоненциально с увеличением количества узлов. Для пяти узлов имеется 29281 и для девяти узлов около 1015 различных DAG [MDBM00]. Таким образом, неосведомленный комбинаторный поиск (см. п. 6.2) в пространстве всех графов с заданным набором переменных безнадежен, если число переменных растет. Поэтому необходимо использовать эвристические алгоритмы. Это ставит вопрос о функции оценки для байесовских сетей. Можно измерить ошибку классификации сети во время применения к набору тестовых данных, как это делается, например, в C4.5 (см. раздел 8.4). Однако для этого вероятности, рассчитанные байесовской сетью, должны быть сопоставлены с решением.

Прямое измерение качества сети можно провести по распределению вероятностей. Мы предполагаем, что до построения сети по данным мы могли определить (оценить) распределение. Затем начинаем поиск в пространстве всех DAG, оценивая по данным значение CPT для каждой DAG (т. е. для каждой байесовской сети), на основании чего вычисляем распределение и сравниваем его с известным данные. Для сравнения распределений нам, очевидно, понадобится метрика расстояния.

Давайте рассмотрим пример прогноза погоды из упражнения 7.3 на стр. 158 с тремя переменными Sky, Bar, Prec и распределением

P(Небо, Бар, Прек) = (0,40, 0,07, 0,08, 0,10, 0,09, 0,11, 0,03, 0,12).

На рис. 8.28 представлены две байесовские сети, которые мы сейчас сравним по их качеству. Каждая из этих сетей делает предположение о независимости, которое подтверждается тем, что мы определяем распределение сети, а затем сравниваем его с исходным распределением (см. упражнение 8.15 на стр. 219).

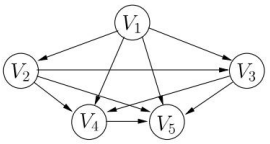
Поскольку для постоянных предопределенных переменных распределение явно представлено вектором постоянной длины, мы можем вычислить евклидову норму разности двух векторов как расстояние между распределениями. Мы определяем

$$dq(x, y) = \sum_{i} (x_i - y_i)^2$$

как сумму квадратов расстояний компонент вектора и вычислить расстояние dq (Pa,P) = 0,0029 распределения Ра сети 1 от исходного распределения. Для сети 2 мы вычисляем dq (Pb, P) = 0,014. Очевидно, что сеть 1 является лучшим приближением к распределению. Часто вместо квадратного расстояния

Machine Translated by Google

Рис. 8.29 Максимальная сеть с пятью переменными и ребер (Vi, Vj), удовлетворяющих условию i<j



так называемое расстояние Кульбака-Лейблера

$$dk(x, y) = \sum_i yi(\log 2\, yi - \log 2\, xi),$$

используется метрика теории информации. Сего помощью мы вычисляем dk(Pa,P) = 0,017 и dk(Pb,P) = 0,09 и приходим к тому же выводу, что и раньше. Следует ожидать, что сети с большим количеством ребер аппроксимируют распределение лучше, чем сети с небольшим количеством ребер. Если все ребра всети построены, то она становится очень запутанной и создает риск переобучения, как это имеет место во многих других алгоритмах обучения. Чтобы избежать переобучения, мы придаем небольшим сетям больший вес с помощью эвристической функции оценки.

$$f(N) = \text{Size}(N) + w \cdot dk(PN, P).$$

Здесь Size(N) —количество записей в CPT, а P N —распределение сети N. w —весовой коэффициент, который необходимо подобрать вручную.

Алгоритм обучения для байесовских сетей, таким образом, вычисляет эвристическую оценку f(N) для множества различных сетей, а затем выбирает сеть с наименьшим значением. Как упоминалось ранее, трудность состоит в сокращении пространства поиска для искомой топологии сети. В качестве простого алгоритма можно, начиная с (например, причинного) упорядочения переменных V1,...,Vn, включать в граф только те ребра, для которых i<j . Начнем с максимальной модели, удовлетворяющей этому условию. Эта сеть показана на рис. 8.29 для пяти упорядоченных переменных.

Теперь, например, в духе жадного поиска (ср. п. 6.3.1), одно ребро за другим удаляются до тех пор, пока значение f не перестанет уменьшаться .

Этот алгоритм не подходит для больших сетей в такой форме. Причинами этого являются большое пространство поиска, ручная настройка веса w и необходимое сравнение с целевым распределением P, потому что они могут просто стать слишком большими или доступный набор данных может быть слишком маленьким.

На самом деле исследования по обучению байесовских сетей все еще идут полным ходом, и существует большое количество предложенных алгоритмов, например алгоритм EM (см. раздел 8.7.2), методы Монте-Карло с цепями Маркова и выборка Гиббса [DHS01, Jor99, Jen01, HTF09]. Помимо представленного здесь пакетного обучения , при котором сеть генерируется один раз из всего набора данных, существуют также инкрементные алгоритмы, в которых каждый отдельный новый случай используется для улучшения сети. Также существуют реализации этих алгоритмов, такие как Hugin (www.hugin.com) и Bayesware (www.bayesware.com).

Machine Translated by Google

8.6 Наивный байесовский классификатор

На рис. 7.14 на стр. 152 диагноз аппендицита был смоделирован в виде байесовской сети. Поскольку направленные ребра начинаются в диагностическом узле и ни одно из них не заканчивается, для ответа на диагностический запрос необходимо использовать формулу Байеса. Для симптомов S1,...,Sn и k-значного диагноза D со значениями b1,...,bk вычисляем вероятность

$$P(D|S1,...,Sn) = P\frac{(S1,...,Sn|D)\cdot P(D)}{(S1,...,Sn)},$$

для постановки диагноза с учетом симптомов пациента. В худшем случае, т. е. при отсутствии независимых переменных, все комбинации всех симптомов и D необходимо было бы определить для всех 20 643 840 вероятностей распределения P(S1,...,Sn,D). Для этого потребуется огромная база данных. В случае с байесовской сетью LEXMED количество необходимых значений (в CPT) сокращено до 521. Однако сеть можно упростить еще больше, если предположить, что все переменные симптомов условно независимы при данном D, то есть:

$$P(S1,...,Sn|D) = P(S1|D)\cdot\cdot\cdot\cdot\cdot P(Sn|D).$$

Затем байесовская сеть для аппендицита упрощается до звезды, показанной на рис. 8.30 на стр. 203.

Таким образом, мы получаем формулу

$$S1,...,Sn) = \frac{P(D)n\ P(Si|D)\ i=1\ P(D|}{П(C1,...,Cн)}. \qquad (8.8)$$

Вычисленные вероятности преобразуются в решение простым наивным байесовским классификатором, который выбирает максимальное P(D = di |S1,...,Sn) из всех значений di в D. То есть он определяет

$$dНаивный\text{-}Байес = argmaxi\ \{1,...,k\}\ P(D = di\ |S1,...,Sn).$$

Поскольку знаменатель в (8.8) постоянный, его можно опустить при максимизации, что приведет к наивной формуле Байеса

$$dНаивный\text{-}Байес = argmaxi\ \{1,...,k\}\ P(D = di)\ \prod_{j=1}^{н} P(Sj\ |D).$$

Поскольку у некоторых узлов теперь меньше предков, количество значений, необходимых для описания распределения LEXMED в CPT, уменьшается, согласно (7.22) на стр. 151, до

$$6\cdot 4 + 5\cdot 4 + 2\cdot 4 + 9\cdot 4 + 3\cdot 4 + 10\cdot(1\cdot 4) + 1 = 141.$$

Machine Translated by Google

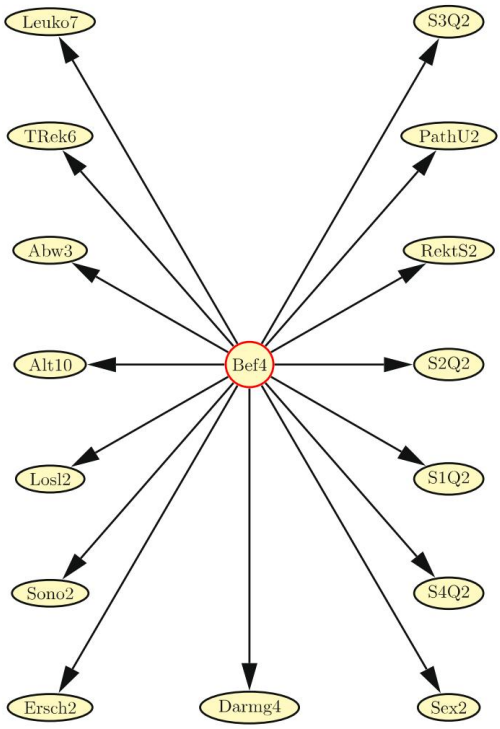


Рис. 8.30 Байесовская сеть для приложения LEXMED в предположении, что все симптомы условно независимы при данном диагнозе

Для медицинской диагностической системы, такой как LEXMED, такое упрощение неприемлемо. Но для задач со многими независимыми переменными наивный байесовский подход частично или даже очень хорошо подходит, как мы увидим на примере текстовой классификации.

Между прочим, наивная байесовская классификация столь же выразительна, как и линейная система оценок, описанная в разд. 7.3.1 (см. упражнение 8.16 на стр. 229). То есть, все оценки разделяющее в основе предположение, что все симптомы условно независимы с учетом диагноза. Тем не менее, счетыдо сих пор используются в медицине. Несмотря на то, что она была создана на основе более качественной репрезентативной базы данных, шкала Омана по сравнению с LEXMED на рис. 7.10 на стр. 142 имеет худшее качество диагностики. Его ограниченная выразительность, безусловно, является причиной этого. Например, как и в случае наивного Байеса, невозможно смоделировать зависимости между симптомами баллы.

Оценка вероятностей Если мы посмотрим на наивную формулу Байеса в (8.8) на стр. 202, мы увидим, что все выражение становится равным нулю, как только один из множителей P (Si |D) в правой части становится равным нулю Теоретически здесь нет ничего плохого. На практике, однако, это может привести к очень неприятным эффектам, если P (Si |D) малы, потому что они оцениваются путем подсчета частот и их подстановки.

Machine Translated by Google

в

$$P(Si = x|D = y) = \frac{|Si = x \quad D = y|}{|D = y|}.$$

Предположим, что для переменных Si : P (Si = x|D = y) = 0,01 и что имеется 40 обучающих случаев с D = y. Тогда с большой вероятностью обучающего случая с Si = x и D = y нет, и мы оцениваем P (Si = x|D = y) = 0. Для другого значения D = z предположим, что отношения расположены аналогично, но оценка приводит к значения м больше нуля для всех P (Si = x|D = z). Таким образом, всегда предпочтительнее значение D = z, которое не отражает фактического распределения вероятностей. Поэ тому при оценке вероятностей используется формула

$$П(A|B) \quad \frac{|A \quad B|}{|Б|} \text{ "=" } \frac{наб}{нБ}$$

заменена на

$$П(A|B) \quad \frac{нАБ + т.п.}{нБ + м},$$

где p = P (A) —априорная вероятность для A, а m —константа, которую можно свободно выбирать и которая известна как «эквивалентный размер данных». Чем больше становится m, тем больше вес априорной вероятности по сравнению со значением, определенным из измеренной частоты.

8.6.1 Классификация текста с помощью наивного Байеса

Наивный байесовский метод сегодня очень успешен и эффективен в классификации текстов. Его основным и в то же время очень важным применением является автоматическая фильтрация электронной почты нежелаемые и нежелательные или спам-письма. В спам-фильтрах, таких как SpamAssassin [Sch04], среди прочих методов используется наивный байесовский классификатор, который учится отделять нужные электронные письма от спама. SpamAssassin —это гибридная система, которая выполняет начальную фильтрацию с использованием черного и белого списков. Черные списки —это списки заблокированных адресов электронной почты от отправителей спама, чьи электронные письма всегда удаляются , а белые списки —это списки с отправителями, чьи электронные письма всегда доставляются . После такой предварительной фильтрации оставшиеся письма классифицируются наивным байесовским классификатором по их фактическому содержанию, то есть по тексту . Обнаруженное значение класса затем оценивается по баллу вместе с другими атрибутами из заголовка электронной почты, такими как домен отправителя , тип MIME и т. д., а затем, наконец, фильтр

Здесь весьма важна способность к обучению наивного байесовского фильтра. Для этого пользователь должен сначала вручную классифицировать большое количество писем как желаемое или как спам. Затем фильтр обучается . Чтобы оставаться в курсе , фильтр необходимо регулярно переобучать. Для этого пользователь должен корректно классифицировать все письма, ошибочно классифицированные фильтром, то есть поместить их в соответствующие папки. Затем фильтр постоянно переобучается с помощью этих писем.

Machine Translated by Google

Помимо фильтрации спама, существует множество других приложений для автоматической классификации текстов. Важные приложения включают в себя фильтрацию нежелательных записей на дискуссионных интернет-форумах и отслеживание веб-сайтов с криминальным содержанием, таким как боевики или террористическая деятельность, детская порнография или расизм. Его также можно использовать для настройки поисковых систем в соответствии с предпочтениями пользователя, чтобы лучше классифицировать результаты поиска. В промышленной и научной среде поиск в масштабах всей компании в базах данных или в литературе находится на переднем плане исследований. Благодаря своей способности к обучению фильтр может адаптироваться к привычкам и пожеланиям каждого отдельного пользователя.

Мы представим применение наивного Байеса к анализу текста на коротком примере. обширный текст Алана Тьюринга из [Tur50]:

> «Мы можем надеяться, что машины со временем будут конкурировать с людьми во всех чисто интеллектуальных областях. Но с каких лучше начать? Даже это трудное решение. Многие люди думают, что лучше всего подойдет очень абстрактная деятельность, например, игра в шахматы. Можно также утверждать, что лучше всего снабдить машину лучшими органами чувств, которые можно купить за деньги, а затем научить ее понимать и говорить по-английски. Этот процесс может следовать обычному обучению ребенка. Будут указаны и названы вещи и т. д. Опять же, я не знаю, какой правильный ответ, но я думаю, что следует попробовать оба подхода».

Предположим, что тексты, подобные приведенному, следует разделить на два класса: «I» — интересные и «¬I — неинтересные. Предположим также, что существует база данных уже классифицированных текстов. Какие атрибуты следует использовать? В классическом подходе к построению байесовской сети мы определяем набор атрибутов, таких как длина текста, средняя длина предложения, относительная частота определенных знаков препинания, частота некоторых важных слов, таких как «я». , «машины» и т. д. При классификации с использованием наивного Байеса, напротив, выбирается удивительно примитивный алгоритм. Для каждой из n позиций слов в тексте определяется атрибут si. Все слова, встречающиеся в тексте, допускаются в качестве возможных значений для всех позиций si. Теперь для классов I и ¬I значе

$$P(I \mid s1,\ldots,sn) = c \cdot P(I) \prod_{i=1}^{n} P(si \mid I) \qquad (8.9)$$

и $P(\neg I \mid s1,\ldots,sn)$ должны быть рассчитаны, а затем выбран класс с максимальным значением. В приведенном выше примере со 113 словами это дает

$$P(I \mid s1,\ldots,sn) = c \cdot P(I) \cdot P(s1 = \text{«Мы»} \mid I) \cdot P(s2 = \text{«может»} \mid I) \cdots\cdots P(s113 = \text{«должен»} \mid I)$$

и

$$P(\neg I \mid s1,\ldots,sn) = c \cdot P(\neg I) \cdot P(s1 = \text{«Мы»} \mid \neg I) \cdot P(s2 = \text{«может»} \mid \neg I) \cdots\cdots P(s113 = \text{«должен»} \mid \neg I).$$

Обучение здесь достаточно простое. Условные вероятности $P(si \mid I)$, $P(si \mid \neg I)$ и априорные вероятности $P(I)$, $P(\neg I)$ должны быть просто вычислены. Теперь дополнительно предположим, что $P(si \mid I)$ не зависит от позиции в тексте. Этот

Machine Translated by Google

означает, например, что

$$P(s61 = «и» \mid I) = P(s69 = «и» \mid I) = P(s86 = «и» \mid I).$$

Таким образом, мы могли бы использовать выражение $P(и|I)$ с новой двоичной переменной и как вероятность появления «и» в произвольной позиции.

Реализация может быть несколько ускорена, если найти частоту ni каждое слово wi, которое встречается в тексте, и используйте формулу

$$P(I|s1,...,sn) = c \cdot P(I) \prod_{i=1}^{л} P(wi|I)^{ni} \quad (8.10)$$

что эквивалентно (8.9) на стр. 205. Обратите внимание, что индекс i в произведении соответствует только количеству l различных слов, встречающихся в тексте.

Несмотря на свою простоту, наивный байесовский алгоритм дает отличные результаты для классификации текстов. Спам-фильтры, работающие с наивным байесовским алгоритмом, обеспечивают уровень ошибок менее одного процента. Системы DSPAM и CRM114 могут быть даже настолько хорошо обучены, что неправильно классифицируют только одно из 7000 или 8000 писем соответственно. Это соответствует точности почти 99,99%.

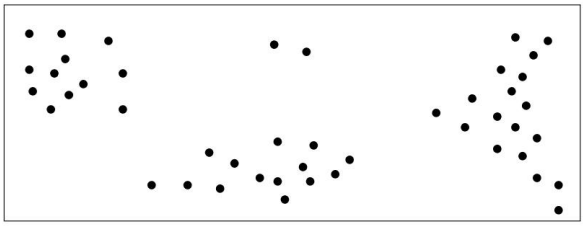
8,7 Кластеризация

Если мы будем искать в поисковой системе термин «марс», мы получим такие результаты, как «планета марс» и «конгломерат шоколада, кондитерских изделий и напитков», которые семантически совершенно разные. В наборе обнаруженных документов есть два заметно различающихся кластера. Google, например, по-прежнему выводит результаты в неструктурированном виде. Было бы лучше, если бы поисковая система разделяла кластеры и представляла их пользователю соответствующим образом, потому что пользователя обычно интересует только один из кластеров.

Отличие кластеризации от обучения с учителем заключается в том, что обучающие данные не помечены. Таким образом, предварительная структуризация данных руководителем отсутствует. Скорее, поиск структуры и есть весь смысл кластеризации. В пространстве обучающих данных должны быть найдены скопления данных, подобные тем, что показаны на рис. 8.31 на стр. 207 . В кластере расстояние между соседними точками обычно меньше, чем расстояние между точками разных кластеров. Поэтому выбор подходящей метрики расстояния для точек, т. е. для группируемых объектов и для кластеров, имеет принципиальное значение. Как и прежде, в дальнейшем мы предполагаем, что каждый объект данных описывается вектором числовых атрибутов.

Machine Translated by Google

Рис. 8.31. Простой двумерный пример с четырьмя четко разделенными кластерами



8.7.1 Метрики расстояния

Соответственно для каждого приложения различные метрики расстояния определяются для расстояния d между двумя векторами x и y в Rn: Наиболее распространенным является евклидово расстояние.

$$\text{де}(x, y) = \sqrt{\sum_{i=1}^{n} (xi - yi)^2}.$$

Несколько проще сумма квадратов расстояний

$$dq(x, y) = \sum_{i=1}^{n} (\text{кси} - \text{у и})^2,$$

что для алгоритмов, в которых сравниваются только расстояния, эквивалентно евклидову расстоянию (упражнение 8.19 на стр. 220). Также используются вышеупомянутые манхэттенские расстояния.

$$\text{дм}(x, y) = \sum_{i=1}^{n} |xi - yi|$$

а также расстояние максимальной составляющей

$$d(x, y) = \max_{i=1,\ldots,n} |xi - yi|$$

который основан на максимальной норме. Во время классификации текста нормированная проекция двух векторов друг на друга, то есть нормированное скалярное произведение

$$\frac{xy}{|x||y|}$$

часто вычисляется, где $|x|$ является евклидовой нормой x. Поскольку эта формула является метрикой подобия двух векторов, в качестве метрики расстояния обратная

$$\text{дс}(x, y) = \frac{|x||y|}{xy}$$

можно использовать или «>» и «<» можно поменять местами для всех сравнений. При поиске текста атрибуты x1,...,xn вычисляются аналогично наивному байесовскому методу как компоненты вектора x следующим образом. Для словаря с 50 000 слов значение xi

Machine Translated by Google

равно частоте i-го словарного слова в тексте. Поскольку обычно в таком векторе почти все компоненты равны нулю, при вычислении скалярного произведения почти все члены суммирования равны нулю. Используя такую информацию, можно значительно ускорить реализацию (упражнение 8.20 на стр. 220).

8.7.2 k-средних и алгоритм EM

Всякий раз, когда количество кластеров уже известно заранее, можно использовать алгоритм k-средних. Как следует из названия, k кластеров определяется их средним значением. Сначала случайным образом или вручную инициализируются k средних точек кластера μ1 ,...,μk . Затем повторно выполняются следующие два шага: • Классификация всех данных до их ближайшей средней точки кластера • Повторное вычисление средней точки кластера.
В качестве алгоритма получается следующая схема:

```
K-MEANS(x1,..., xn,k)
инициализируется μ1 ,...,μk (например, случайным образом
Повторить
    классификацию x1,..., xn до ближайшего μi
    пересчитать μ1 ,...,μk до
тех пор, пока не останется
изменений в μ1 ,...,μk Возврат(μ1 ,...,μk )
```

Вычисление середины кластера μ для точек x1,..., xl выполняется по формуле

$$м = \frac{1}{л} \sum_{я=1}^{л} xi .$$

Выполнение на примере показано на рис. 8.32 на стр. 209 для случая двух классов. Мы видим, как после трех итераций центры классов, которые сначала были выбраны случайным образом, стабилизируются. Хотя этот алгоритм не гарантирует сходимости, обычно он сходится очень быстро. Это означает, что количество шагов итерации обычно намного меньше количества точек данных. Его сложность равна O(ndkt), где n —общее количество точек, d —размерность пространства признаков, а t —количество шагов итерации.

Во многих случаях необходимость заранее указывать количество классов создает неудобное ограничение. Поэтому далее мы введем более гибкий алгоритм.

Однако перед этим мы упомянем алгоритм EM, который является непрерывным вариантом k-средних, так как он не делает жесткого распределения данных по классам, а скорее для каждой точки возвращает вероятность ее принадлежности различным классы

Machine Translated by Google

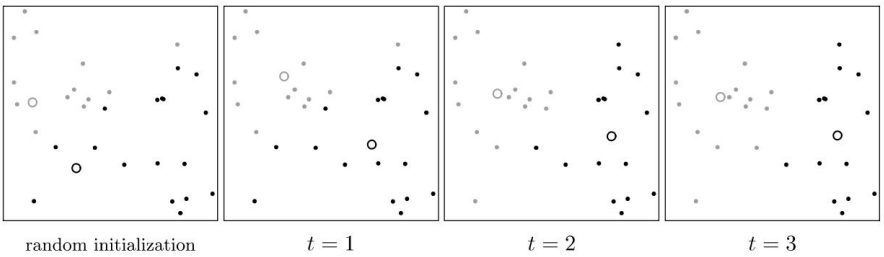


Рис. 8.32. k-средних с двумя классами (k = 2), примененными к 30 точкам данных. Крайний слева —набор данных с начальными центрами, а справа —кластер после каждой итерации. После трех итераций достигается сходимость

Здесь мы должны предположить, что тип распределения вероятностей известен. Часто используется нормальное распределение. Задача ЭМ-алгоритма состоит в определении параметров (среднего μi и ковариационных матриц Σi k многомерных нормальных распределений) для каждого кластера. Аналогично k-средним повторно выполняются два следующих шага:
Ожидание : для каждой
точки данных вычисляется вероятность P (Cj |xi) того, что она принадлежит каждому кластеру.

Максимизация : Используя вновь рассчитанные вероятности, пересчитываются параметры распределения .

Тем самым достигается более мягкая кластеризация , что во многих случаях приводит к лучшим результатам. Это чередование между ожиданием и максимизацией дало название алгоритму . Вдополнение к кластеризации, например, алгоритм EM используется для изучения байесовских сетей [DHS01].

8.7.3 Иерархическая кластеризация

В иерархической кластеризации мы начинаем с n кластеров, состоящих из одной точки каждый. Затем кластеры ближайших соседей объединяются до тех пор, пока все точки не будут объединены в один кластер или пока не будет достигнут критерий завершения . Получаем схему

```
ИЕРАРХИЧЕСКАЯ КЛАСТЕРИЗАЦИЯ(x1,..., xn,k)
инициализирует C1 = {x1},...,Cn = {xn}
Повторить
    Найти два кластера Ci и Cj с наименьшим расстоянием Объединить
    Ci и Cj до тех пор, пока
не будет достигнуто условие завершения
Возврат (дерево с кластерами)
```

Machine Translated by Google

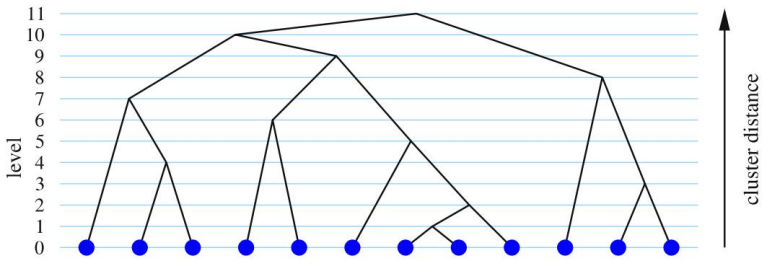


Рис. 8.33. При иерархической кластеризации на каждом шаге объединяются два кластера с наименьшим расстоянием

Условие завершения может быть выбрано, например, как желаемое количество кластеров или максимальное расстояние между кластерами. На рис. 8.33 этот алгоритм схематично представлен в виде бинарного дерева, в котором снизу вверх на каждом шаге, т. е. на каждом уровне, соединены два поддерева. На верхнем уровне все точки объединяются в один большой кластер.

Пока неясно, как рассчитываются расстояния между скоплениями. Действительно, в предыдущем разделе мы определили различные метрики расстояния для точек, но их нельзя использовать для кластеров. Удобной и часто используемой метрикой является расстояние между двумя ближайшими точками в двух кластерах Ci и Cj :

$$dmin(Ci, Cj) = \min d(x, y). \; x \in Ci, \; y \in Cj$$

Таким образом, мы получаем алгоритм ближайшего соседа, применение которого показано на рис. 8.34 на стр. 211. Дерево[9] 10. Мы видим, что этот алгоритм генерирует минимальный охват Кроме того, пример показывает, что два описанных алгоритма генерируют совершенно разные кластеры. Это говорит нам о том, что для графов с кластерами, которые четко не разделены, результат сильно зависит от алгоритма или выбранной метрики расстояния .

Для эффективной реализации этого алгоритма мы сначала создадим матрицу смежности, в которой сохраняются расстояния между всеми точками, что требует $O(n^2)$ времени и памяти. Если количество кластеров не имеет верхнего предела, цикл будет повторяться $n - 1$ раз, а асимптотическое время вычисления станет равным $O(n^3)$.

Чтобы вычислить расстояние между двумя кластерами, мы также можем использовать расстояние между двумя самыми дальними точками

$$dmax(Ci, Cj) = \max d(x, y) \; x \in Ci, y \in Cj$$

9 Алгоритм ближайшего соседа не следует путать с методом ближайшего соседа для классификация из разд. 8.3.

10 Минимальное остовное дерево — это ациклический неориентированный граф с минимальной суммой длин ребер.

Machine Translated by Google

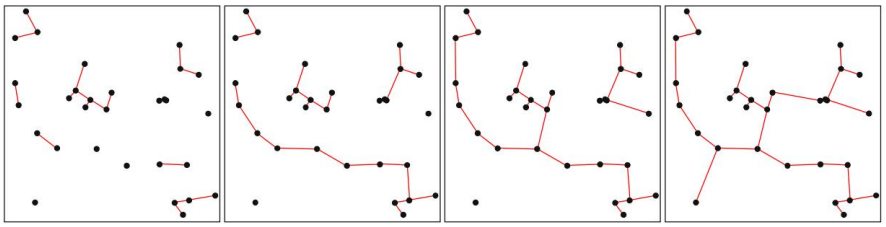


Рис. 8.34 Алгоритм ближайшего соседа, примененный к данным из рис. 8.32 на стр. 209 на разных уровнях с 12, 6, 3, 1 кластерами

и получить алгоритм самого дальнего соседа. В качестве альтернативы используется расстояние от средней точки кластера dμ(Ci, Cj) = d(μi,μj). Помимо алгоритма кластеризации представленные здесь, есть и многие другие, для ознакомления с которыми мы отправляем читателя к [DHS01] для дальнейшего изучения.

8.8 Интеллектуальный анализ данных на практике

Все алгоритмы обучения, представленные до сих пор, можно использовать в качестве инструментов для интеллектуального анализа данных. Для пользователю, однако, иногда довольно сложно привыкнуть к новому программному обеспечению инструменты для каждого приложения и, кроме того, поместить данные для анализа в соответствующий формат для каждого конкретного случая.

Ряд систем интеллектуального анализа данных решают эти проблемы. Большинство этих систем предлагают удобный графический пользовательский интерфейс с разнообразными инструментами для визуализации данных, для предварительной обработки, такой как манипулирование отсутствующими значениями, и для анализа. Для анализ, алгоритмы обучения, представленные здесь, используются, среди прочего.

Отдельного упоминания заслуживает все объемлющая библиотека Java с открытым исходным кодом WEKA. Он предлагает большое количество алгоритмов и упрощает разработку новых алгоритмов.

Свободно доступная система KNIME, которую мы кратко представим в следующий раздел предлагает удобный пользовательский интерфейс и все виды инструментов упомянутое выше. KNIME также использует модули WEKA. Кроме того, он предлагает простой способ управления потоком данных выбранных инструментов визуализации, предварительной обработки и анализа с помощью графического редактора. В то же время большое количество других систем предлагают весьма схожие функции, например, проект с открытым исходным кодом RapidMiner(www.rapidminer.com), система Clementine (www.spss.com/clementine), продаваемый SPSS, и аналитическая структура KXEN (www.kxen.com).

Machine Translated by Google

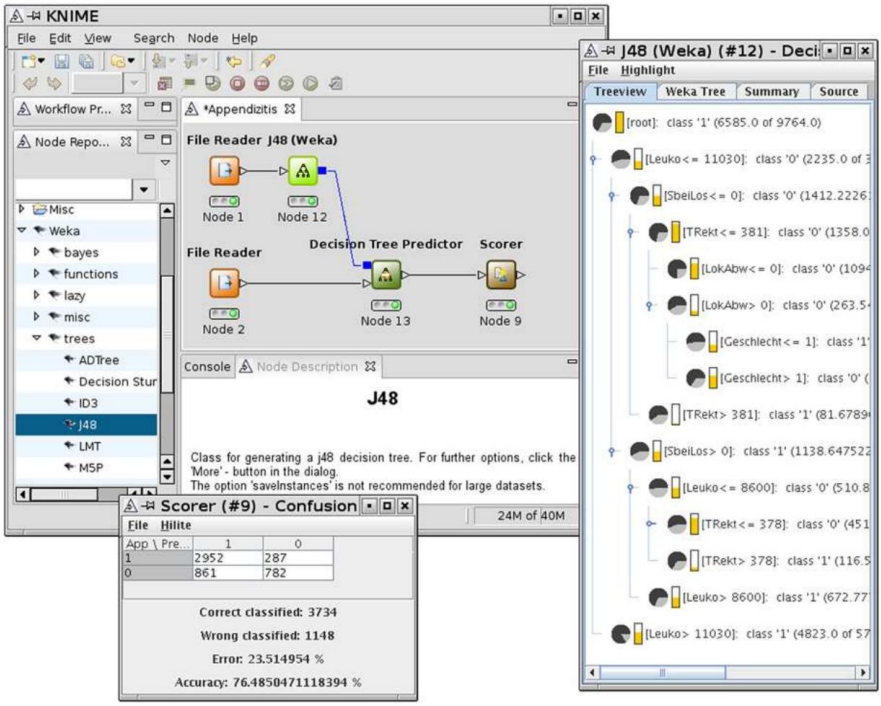


Рис. 8.35 Пользовательский интерфейс KNIME с двумя дополнительными представлениями, которые показывают дерево решений и матрицу путаницы

8.8.1 Инструмент интеллектуального анализа данных KNIME

Теперь, используя данные LEXMED, мы покажем, как извлекать знания из данных с помощью KNIME (Konstanz Information Miner, www.knime.org). Сначала мы создаем дерево решений, как показано на рис. 8.35. После создания нового проекта рабочий процесс строится графически. Для этого соответствующие инструменты просто достаются мышкой из репозитория узла и перетаскиваются в главное окно рабочего процесса.

Тренировочные и тестовые данные из файла C4.5 можно без проблем считывать с помощью двух узлов чтения файлов. Однако эти узлы также могут быть легко настроены для других форматов файлов. Боковой светофор под узлом показывает его статус (не готов, настроен, выполнен). Затем выбирается узел J48 из библиотеки WEKA [WF01], которая содержит Java-реализацию C4.5. Конфигурация для этого довольно проста. Теперь выбирается предикторный узел, который применяет сгенерированное дерево к тестовым данным. Он вставляет новый столбец в таблицу тестовых данных «Предсказание» с классификацией, сгенерированной деревом. Оттуда узел подсчета очков вычисляет матрицу путаницы, показанную на рисунке, которая дает количество правильно классифицированных случаев для обоих классов на диагонали, а также количество ложноположительных и ложноотрицательных точек данных.

Machine Translated by Google

Таблица 8.1 Данные обучения для агента сортировки яблок

Размер [см]	8	8 6 3. . .
Цвет	0,1 0,3 0,9 0,8 . . .	
Товарный класс	B	A A B . . .



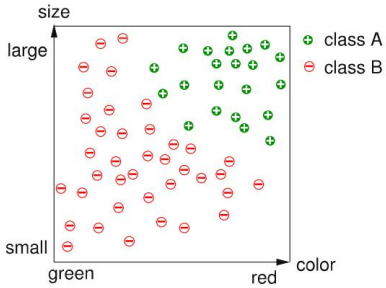
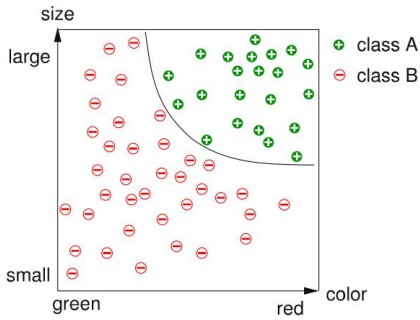


Рис. 8.2 Оборудование для сортировки яблок компании BayWa в Крессбронне и некоторые яблоки, классифицированные по товарным классам A и B в функциональном пространстве (Фото: BayWa)

Рис. 8.3 Кривая , нарисованная на диаграмме, разделяет классы и затем может быть применена к произвольным новым яблокам



Визуализация данных представлена в виде точек на диаграмме рассеяния справа на рис. 8.2.

Задача машинного обучения состоит в создании функции из собранных классифицированных данных, которая вычисляет значение класса (A или B) для нового яблока на основе двух характеристик размера и цвета. На рис. 8.3 такая функция показана разделительной линией, проведенной через диаграмму. Все яблоки с вектором признаков в левом нижнем углу линии помещаются в класс B, а все остальные — в класс A.

В этом примере по-прежнему очень просто найти такую разделительную линию для двух классов. Явно более трудная и, главное, гораздо менее наглядная задача, когда классифицируемые объекты описываются не двумя, а многими признаками. На практике обычно используется 30 и более признаков. Для n признаков задача состоит в том, чтобы найти n - 1-мерную гиперплоскость в n-мерном пространстве признаков , которая как можно лучше разделяет классы. «Хорошее» деление означает, что процент ошибочно классифицированных объектов как можно меньше.

Machine Translated by Google



Рис. 8.4 Функциональная структура обучающего агента для сортировки яблок (слева) и в целом (справа)

Классификатор сопоставляет вектор признаков со значением класса. Здесь он имеет фиксированное, обычно небольшое, количество альтернатив. Желаемое отображение также называется целевой функцией. Если целевая функция не отображается на конечную область, то это не классификация, а задача аппроксимации. Такой аппроксимационной задачей является определение рыночной стоимости акций по заданным характеристикам. В следующих разделах мы представим несколько обучающих агентов для обоих типов отображений.

Агент обучения Мы можем формально описать агент обучения как функцию, которая отображает вектор признаков в значение дискретного класса или, вообще говоря, в действительное число. Эта функция не запрограммирована, а возникает или изменяется на этапе обучения под влиянием данных обучения. На рис. 8.4 такой агент представлен в примере сортировки яблок. Во время обучения агент получает уже классифицированные данные из Таблицы 8.1 на стр. 163. После этого агент составляет как можно более точное отображение вектора признаков в значение функции (например, класс товаров).

Теперь мы можем попытаться подойти к определению термина «машинное обучение». Том Митчелл [Mit97] дает такое определение:

Машинное обучение — это изучение компьютерных алгоритмов, улучшающих автоматизацию. через опыт.

Опираясь на это, мы даем

Определение 8.1 Агент является обучающимся агентом, если он улучшает свою производительность (измеряемую с помощью подходящего критерия) на новых, неизвестных данных с течением времени (после того, как он увидел множество обучающих примеров).

Важно проверить способность алгоритма обучения к обобщению на неизвестных данных, на тестовых данных. В противном случае каждая система, которая только что сохранила тренировочные данные, будет работать оптимально, просто вызывая сохраненные данные. Агент обучения характеризуется следующими
терминами: Задача: задачей алгоритма обучения является изучение отображения. Это может быть, например, сопоставление размера и цвета яблока с его товарным классом, а также сопоставление 15 симптомов пациента с решением об удалении аппендикса.

Machine Translated by Google



Рис. 8.5 Интеллектуальный анализ данных

Переменный агент (точнее, класс агентов): здесь мы должны решить, с каким алгоритмом обучения будем работать. Если это было выбрано, таким образом определяется класс всех обучаемых функций.

Данные обучения (опыт): данные обучения (выборка) содержат знания, которые алгоритм обучения должен извлечь и изучить. При выборе обучающих данных необходимо убедиться, что это репрезентативная выборка для изучаемой задачи.

Тестовые данные: важны для оценки того, может ли обученный агент хорошо обобщать обучающие данные на новые данные.

Критерий эффективности: для устройства для сортировки яблок количество правильно классифицированных яблок. Он нужен нам для проверки качества агента. Знать показатель производительности обычно намного проще, чем знать функцию агента. Например, легко измерить производительность (время) бегуна на 10 000 метров. Однако это вовсе не означает, что судья, измеряющий время, может бежать так же быстро. Судья знает только, как измерить производительность, но не «функцию» агента, чью производительность он измеряет.

Что такое интеллектуальный анализ данных? Задача обучающей машины извлекать знания из обучающих данных. Часто разработчик или пользователь хотят, чтобы обучающаяся машина сделала извлеченные знания понятными для людей. Еще лучше, если разработчик может даже изменить знания. Процесс индукции деревьев решений в разд. 8.4 является примером такого метода.

Аналогичные проблемы связаны с электронным бизнесом и управлением знаниями. Здесь возникает классическая проблема: от действий посетителей его веб-портала,

Machine Translated by Google

владелец интернет-бизнеса хотел бы создать отношения между
характеристики покупателя и класс товаров, которые ему интересны
клиент. Тогда продавец сможет размещать рекламу для конкретных клиентов. Этот
красноречиво демонстрируется на сайте www.amazon.com, где покупателю рекомендуются товары,
аналогичные тем, что он видел при предыдущем посещении. Во многих
В областях рекламы и маркетинга, а также в управлении взаимоотношениями с клиентами (CRM) начинают
использоваться методы интеллектуального анализа данных. Всякий раз, когда большие суммы
данных доступны, можно попытаться использовать эти данные для анализа клиентов
предпочтения для показа рекламы, ориентированной на клиента. Развивающееся поле
обучения предпочтениям посвящена этой цели.

Процесс получения знаний из данных, а также их представление
и приложение, называется интеллектуальным анализом данных. Используемые методы обычно принимаются
из статистики или машинного обучения и должен быть применим к очень большим
объемы данных по разумной цене.

В контексте получения информации, например, в Интернете или в интрасети, интеллектуальный анализ текста играет все более важную роль. Типичные задачи включают в себя поиск похожего текста в поисковой системе или классификацию текстов, которые, например,
применяется в спам-фильтрах для электронной почты. В разд. 8.6.1 мы представим широко распространенный наивный байесовский алгоритм классификации текста. Относительно новая задача для интеллектуальный анализ данных — это извлечение структурной, статической и динамической информации из графовые структуры, такие как социальные сети, сети трафика или интернет-трафик.

Поскольку две описанные задачи машинного обучения и интеллектуального анализа данных формально очень похожи, основные методы, используемые в обеих областях, по большей части идентичны. Поэтому при описании алгоритмов обучения не будет различия.
сделано между машинным обучением и интеллектуальным анализом данных.

Благодаря огромному коммерческому влиянию методов интеллектуального анализа данных в настоящее время множество сложных оптимизаций и целая линейка мощных систем интеллектуального анализа данных, которые предлагают большую палитру удобных инструментов для извлечения знаний из данные. Такая система представлена в разд. 8.8.

8.1 Анализ данных

Статистика предоставляет несколько способов описания данных с помощью простых параметров. От из них мы выбираем несколько, которые особенно важны для анализа обучения
данные и протестировать их на подмножестве данных LEXMED из разд. 7.3. В этом примере
набор данных, симптомы x1,...,x15 у N = 473 пациентов, кратко описанные в Таблице 8.2 на стр. 167, а также метка класса, то есть диагноз (аппендицит
положительные/отрицательные) — перечислены. Пациент номер один, например, описан
вектор

$$x_1 = (26, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 37{,}9, 38{,}8, 23100, 0, 1)$$

Machine Translated by Google

Таблица 8.2 Описание переменные x1,...,x16. Немного другой формализация использовалась в Табл. 7.2 на стр. 133

Вар. число	Описание	Ценности
1	Возраст	Непрерывный
2	Пол (1=мужской, 2=женский)	1, 2
3	Квадрант боли 1	0, 1
4	Квадрант боли 2	0, 1
5	Квадрант боли 3	0, 1
6	Квадрант боли 4	0, 1
7	Местная мышечная защита	0, 1
8	Генерализованная мышечная защита	0, 1
9	Отскок нежности	0, 1
10	Боль при постукивании	0, 1
11	Боль во время ректального исследования	0, 1
12	Осевая температура	Непрерывный
13	Ректальная температура	Непрерывный
14	Лейкоциты	Непрерывный
15	Сахарный диабет	0, 1
16	Аппендицит	0, 1

и пациент номер два

$$^{2}x = (17, 2, 0, 0, 1, 0, 1, 0, 1, 1, 0, 36{,}9, 37{,}4, 8100, 0, 0)$$

У пациента номер два значение лейкоцитов $x_{14}^{2} = 8100$.

Для каждой переменной xi ее среднее значение $\bar{x}_i$ определяется как

$$\bar{x}_{я} := \frac{1}{H} \sum_{p=1}^{H} x_{я}^{п}$$

и стандартное отклонение si как мера его среднего отклонения от среднего значение как

$$си := \sqrt{\frac{1}{H-1} \sum_{p=1}^{H} (x_{я}^{опыт} - \bar{x}_i)^2}.$$

Вопрос о том, являются ли две переменные xi и xj статистически зависимыми (коррелированными), важен для анализа многомерных данных. Например, ковариация

$$\sigma_{ij} = \frac{1}{H-1} \sum_{p=1}^{H} (x_{я}^{опыт} - \bar{x}_i)(x_{дж}^{п} - \bar{x}_j)$$

Machine Translated by Google

Таблица 8.3 Матрица корреляции для 16 переменных аппендицита, измеренных в 473 случаях

1.	0,009	0,14	0,037	0,096	0,12	0,018	0,051	-0,034	-0,041	0,034	0,037	0,05	-0,037	0,37	0,012
0,009	1.	0,0074	0,019	0,06	0,063	0,17	0,0084	-0,17	-0,14	-0,13	-0,017	-0,034	-0,14	0,045	0,2
0,14	-0,0074	1.	0,55	0,091	0,24	0,13	0,24	0,045	0,18	0,028	0,02	0,045	0,03	0,11	0,045
0,037	0,019	0,55	1.	0,24	0,33	0,051	0,25	0,074	0,19	0,087	0,11	0,12	0,11	0,14	-0,0091
0,096	0,06	0,091	0,24	1.	0,059	0,14	0,034	0,14	0,049	0,057	0,064	0,058	0,11	0,017	0,14
0,12	0,063	0,24	0,33	0,059	1.	0,071	0,19	0,086	0,15	0,048	0,11	0,12	0,063	0,21	0,053
0,018	0,17	0,13	0,051	0,14	0,071	1.	0,16	0,4	0,28	0,2	0,24	0,36	0,29	0,0001	0,33
0,051	0,0084	0,24	0,25	0,034	0,19	0,16	1.	0,17	0,23	0,24	0,19	0,24	0,27	0,083	0,084
0,034	0,17	0,045	0,074	0,14	0,086	0,4	0,17	1.	0,53	0,25	0,19	0,27	0,27	0,026	0,38
0,041	0,14	0,18	0,19	0,049	0,15	0,28	0,23	0,53	1.	0,24	0,15	0,19	0,23	0,02	0,32
0,034	0,13	0,028	0,087	0,057	0,048	0,2	0,24	0,25	0,24	1.	0,17	0,17	0,22	0,098	0,17
0,037	0,017	0,02	0,11	0,064	0,11	0,24	0,19	0,19	0,15	0,17	1.	0,72	0,26	0,035	0,15
0,05	0,034	0,045	0,12	0,058	0,12	0,36	0,24	0,27	0,19	0,17	0,72	1.	0,38	0,044	0,21
0,037	0,14	0,03	0,11	0,11	0,063	0,29	0,27	0,27	0,23	0,22	0,26	0,38	1.	0,051	0,44
0,37	0,045	0,11	0,14	0,017	0,21	0,0001	0,083	0,026	0,02	0,098	0,035	0,044	0,051	1.	0,0055
0,012	0,2	0,045	0,0091	0,14	0,053	0,33	0,084	0,38	0,32	0,17	0,15	0,21	0,44	-0,0055	1.

дает информацию об этом. В этой сумме слагаемое возвращает положительную запись для
р-й вектор данных точно тогда, когда отклонения i-й и j-й компонент от
средние оба имеют одинаковый знак. Если они имеют разные знаки, то запись
отрицательный. Следовательно, ковариация σ12,13 двух различных значений лихорадки должна быть равна
вполне однозначно положительный.

Однако ковариация также зависит от абсолютного значения переменных,
что затрудняет сравнение значений. Чтобы можно было сравнить степень
зависимости в случае нескольких переменных, поэтому мы определяем корреляцию
коэффициент

$$\text{Кий} = \frac{\sigma ij}{\text{си} \cdot \text{сж}}$$

для двух значений xi и xj , что представляет собой не что иное, как нормализованную ковариацию. Матрица
K всех коэффициентов корреляции содержит значения от 1 до 1, является симметричным,
и все его диагональные элементы имеют значение 1. Корреляционная матрица для всех 16
переменных приведены в таблице 8.3.

Эта матрица становится несколько более читаемой, когда мы представляем ее как плотность
сюжет. Вместо числовых значений элементы матрицы на рис. 8.6 на стр. 169
заполнены серыми значениями. На правой диаграмме показаны абсолютные значения. Таким образом
мы можем очень быстро увидеть, какие переменные демонстрируют слабую или сильную зависимость. Мы
можно видеть, например, что переменные 7, 9, 10 и 14 демонстрируют самую сильную корреляцию с
переменной класса аппендицит и поэтому более важны для
диагноз, чем другая переменная. Мы также видим, однако, что переменные 9 и 10
сильно коррелированы. Это может означать, что одно из этих двух значений потенциально
достаточно для диагностики.

Machine Translated by Google

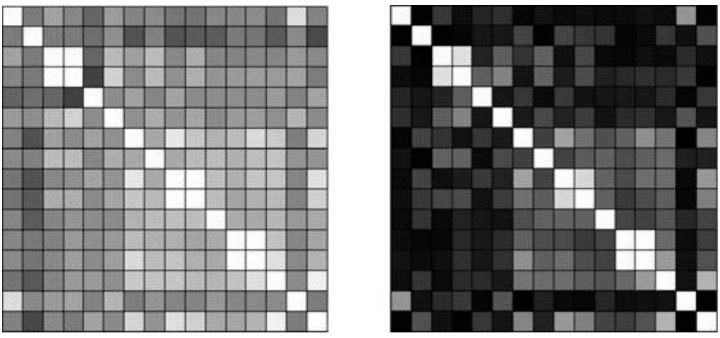
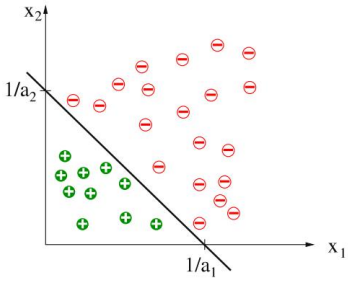


Рис. 8.6 Корреляционная матрица в виде частотного графика. На левой диаграмме темный цвет соответствует отрицательному, а светлый — положительному. На правом изображении были перечислены абсолютные значения. Здесь черный означает Kij 0 (некоррелированный), а белый |Kij | 1 (сильно коррелирует)

Рис. 8.7. Линейно разделимый двумерный набор данных. Уравнение разделительной прямой: a1x1 + a2x2 = 1



8.2 Персептрон, линейный классификатор

В примере классификации сортировки яблок между двумя классами на рис. 8.3 на стр. 163 проведена изогнутая разделительная линия. Более простой случай показан на рис. 8.7. Здесь двумерные обучающие примеры могут быть разделены прямой линией. Мы называем такой набор обучающих данных линейно разделимым. В n измерениях для разделения необходима гиперплоскость. Это представляет собой линейное подпространство размерности n - 1.

Поскольку каждая (n 1)-мерная гиперплоскость в уравнении R н можно описать с помощью

$$\sum_{i=1}^{n} a_i x_i = \theta$$

имеет смысл определить линейную отделимость следующим образом.

Machine Translated by Google

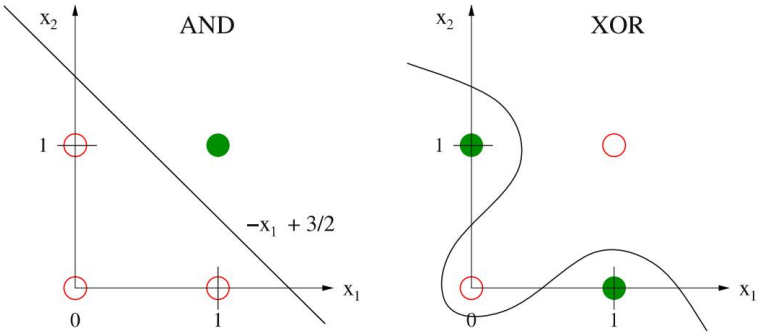


Рис. 8.8. Булева функция AND линейно разделима, а XOR — нет (ˆ= true, ˆ= ложь)

Определение 8.2 Два множества M1 R , н и M2 R н называются линейно разделимыми если существуют вещественные числа a1,...,an, θ с

н н
aixi > θ для всех x M1 и aixi θ для всех x M2.
я=1 я=1

Значение θ обозначает порог.

На рис. 8.8 мы видим, что функция И является линейно разделимой, а функция исключающее ИЛИ — нет. Для И, например, линия x1 + 3/2 разделяет истинную и неверную интерпретацию формулы x1 x2. Напротив, функция XOR не имеет прямой линии разделения. Очевидно, что в этом отношении функция XOR имеет более сложную структуру, чем функция AND.

С персептроном мы представляем очень простой алгоритм обучения, который может

арные линейно отделимые множества.

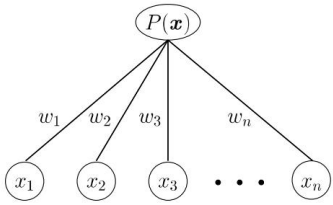
Определение 8.3. Пусть w = (w1,...,wn) R — н — весовой вектор и x R {0, н входной вектор. Персептрон представляет собой функцию P : R н 1}, что соответствует следующему правилу :

$$P(x) = \begin{cases} 1, \text{ если } wx = \sum_{i=1}^{n} w_i x_i > 0, \\ 0 \text{ еще.} \end{cases}$$

Персептрон [Рос58, МП69] — очень простой алгоритм классификации. Это эквивалентно двухслойной нейронной сети с активацией пороговой функцией, показанной на рис. 8.9 на стр. 171. Как показано в гл. 9 каждый узел в сети представляет собой нейрон, а каждое ребро — синапс. Однако пока мы будем только

Machine Translated by Google

Рис. 8.9 Графическое представление персептрона в виде двухслойной нейронной сети



рассматривайте персептрон как обучающий агент, то есть как математическую функцию, которая отображает вектор признаков в значение функции. Здесь входные переменные xi обозначены признаками.

Как видно из формулы $\sum_{i=1} w_i x_i > 0$, все точки x выше гиперплоскости $\sum_{i=1} w_i x_i = 0$ классифицируются как положительные (P(x)=1), а все остальные — как отрицательные (P(x) = 0). Разделяющая гиперплоскость проходит через начало координат, поскольку θ = 0.

Мы воспользуемся небольшой уловкой, чтобы показать, что отсутствие произвольного порога не означает ограничения мощности. Однако сначала мы хотим ввести простой алгоритм обучения для персептрона.

8.2.1 Правило обучения

С обозначениями M+ и M- для наборов положительных и отрицательных шаблонов обучения соответственно правило обучения персептрона читается как [MP69]

```
PERCEPTRONLEARNING[M+,M  ] w =
произвольный вектор действительных чисел
Повторить
    Для всех x   M+
        Если wx   0, то w = w + x
    Для всех x   M
        Если wx > 0, то w = w   x
Пока все x   M+   M   не будут правильно классифицированы
```

Перцептрон должен выводить значение 1 для всех x M+. По определению 8.3 на стр. 170 это верно, когда wx > 0. Если это не так, то x добавляется к вектору весов w, в результате чего вектор весов изменяется точно в правильном направлении.

Мы видим это, когда применяем персептрон к измененному вектору w + x, потому что

$$2\ (w + x) \cdot x = wx + x \qquad \cdot$$

Если этот шаг повторять достаточно часто, то в какой-то момент значение wx станет положительным, как и должно быть. Аналогично, мы видим, что для отрицательных обучающих данных

Machine Translated by Google

персептрон вычисляет все меньшее значение

$$(w \quad x) \cdot x = wx \quad x \quad ^2$$

который в какой-то момент становится отрицательным.2

Пример 8.2 Персептрон должен быть обучен на множествах M+ = {(0, 1,8), (2, 0,6)} и M = {(1,2, 1,4), (0,4, 1)}. w = (1, 1) использовался в качестве начального вектора весов. Данные обучения и линия, определяемая весовым вектором wx = x1 + x2 = 0, показаны на рис. 8.10 на стр. 173 на первом рисунке в верхнем ряду. Кроме того, вектор весов нарисован пунктирной линией. Поскольку wx = 0, это ортогонально прямой.

На первой итерации цикла алгоритма обучения единственное ложное пример классифицированного обучения (-1,2, 1,4), потому что

$$(\quad 1,2, 1,4) \cdot \quad _{1\,1} \quad = 0,2 > 0.$$

В результате w = (1, 1)(1,2, 1,4) = (2,2, 0,4), как показано на втором изображении в верхнем ряду на рис. 8.10 на странице 173. Другие изображения показывают , как после всего пять изменений, разделительная линия проходит между двумя классами. Таким образом, персептрон правильно классифицирует все данные. Мы ясно видим в примере, что каждая неправильно классифицированная точка данных из M+ «тянет» весовой вектор w в своем направлении, а каждая неправильно классифицированная точка из M «толкает» весовой вектор в противоположном направлении.

Было показано [MP69], что персептрон всегда сходится для линейного разделения. пахотные данные. У нас есть

Теорема 8.1. Пусть классы M+ и M линейно отделимы гиперплоскостью wx = 0. Тогда PERCEPTRONLEARNING сходится для каждой инициализации вектора w. Перцептрон P с рассчитанным таким образом весовым вектором делит классы M+ и M , то есть:

$$P(x) = 1 \quad x \quad M+$$

и

$$P(x) = 0 \quad x \quad M \quad .$$

Как мы можем ясно видеть в примере 8.2, персептроны, как определено выше, не могут делить произвольные линейно разделимые множества, а только те, которые делятся прямой, проходящей через начало координат, или в R^n — гиперплоскостью, проходящей через начало координат, потому что постоянный член θ равен отсутствует в уравнении $\sum_{i=1}^{n} w_i x_i = 0$.

2Осторожно! Это не доказательство сходимости правила обучения персептрона. Это показывает только то, что персептрон сходится, когда набор обучающих данных состоит из одного примера.

Machine Translated by Google

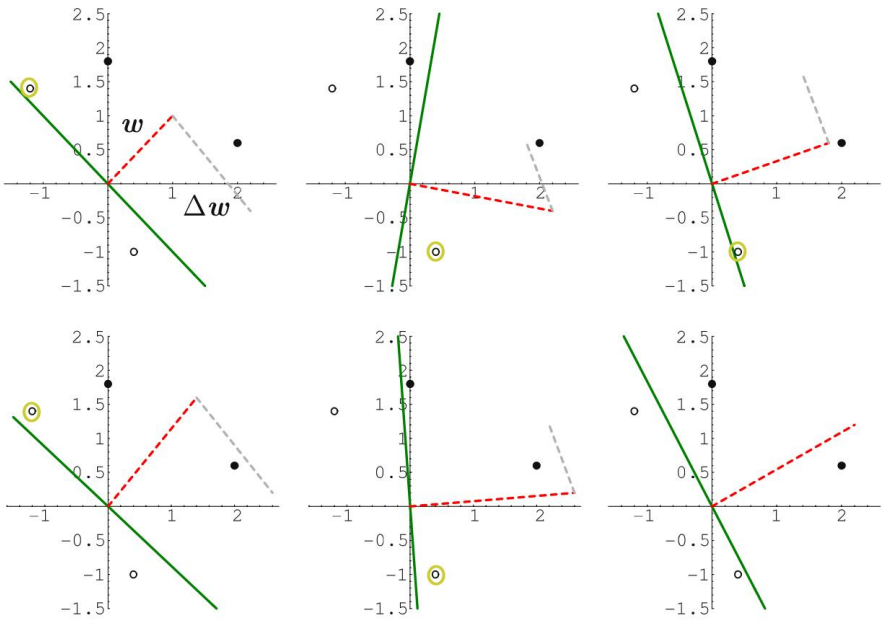


Рис. 8.10. Применение правила обучения персептрона к двум положительным (•) и двум отрицательным ( ) точкам данных. Сплошная линия показывает текущую разделительную линию wx = 0. Ортогональная пунктирная линия представляет собой весовой вектор w, а вторая пунктирная линия — добавляемый вектор изменения w = x или w = x, который вычисляется из текущей активной точки данных. окруженный серым цветом

С помощью следующего трюка мы можем сгенерировать постоянный член. Мы сохраняем последнюю компоненту xn входного вектора x постоянной и присваиваем ей значение 1. Теперь вес wn = θ работает как порог, потому что

$$\sum_{i=1}^{n} w_i x_i = \sum_{i=1}^{n-1} w_i x_i - \theta > 0 \quad \sum_{i=1}^{n-1} w_i x_i > \theta.$$

Такое постоянное значение xn = 1 на входе называется единицей смещения. Поскольку связанный вес вызывает постоянное смещение гиперплоскости, термин «смещение» подходит хорошо.

При применении алгоритма обучения персептрона к вектору обучающих данных добавляется бит с постоянным значением 1. Заметим, что вес wn, или порог θ , усваивается в процессе обучения.

Теперь было показано, что персептрон $P\theta : R^{n-1} \to \{0, 1\}$

$$P\theta(x_1,\ldots,x_{n-1}) = \begin{cases} 1 \text{ если } \sum_{i=1}^{n-1} w_i x_i > \theta, \\ 0 \text{ еще} \end{cases} \tag{8.1}$$

с произвольным порогом можно смоделировать с помощью персептрона $P : R^{n-1} \to \{0, 1\}$ с порог 0. Если сравнить (8.1) с определением линейно разделимого, то мы увидим, что оба утверждения эквивалентны. Подводя итог, мы показали, что:

Machine Translated by Google

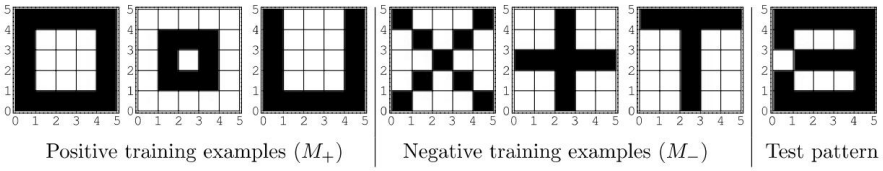
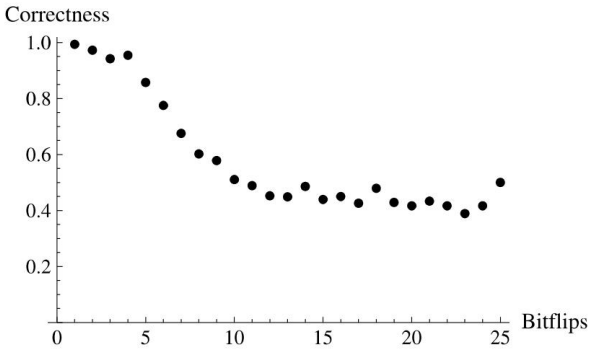


Рис. 8.11. Шесть паттернов, используемых для обучения. Весь правый шаблон является одним из 22 тестовых шаблонов для первого шаблона с последовательностью из четырех инвертированных битов.

Рис. 8.12 Относительная правильность персептрона в зависимости от количества инвертированных битов в тестовых данных



Теорема 8.2 Функция f : R {0, 1} может быть представлена персептроном тогда и только тогда, когда два множества положительных и отрицательных входных векторов линейно разделимы.

Пример 8.3 Теперь мы обучаем персептрон с порогом на шести простых графических двоичных шаблонах, представленных на рис. 8.11, размером 5 × 5 пикселей каждый.

Данные обучения могут быть изучены с помощью PERCEPTRONLEARNING за четыре итерации по всем шаблонам. Образцы с переменным числом инвертированных битов, представленных как шум, используются для проверки способности системы к обобщению. Инвертированные биты в тестовом шаблоне в каждом случае идут последовательно один за другим. На рис. 8.12 процент правильно классифицированных образов представлен как функция количества ложных битов.

Примерно после пяти последовательных инвертированных битов корректность резко падает, что неудивительно, учитывая простоту модели. В следующем разделе мы представим алгоритм, который работает намного лучше в этом случае.

8.2.2 Оптимизация и перспективы

Как один из простейших алгоритмов обучения на основе нейронных сетей, двухслойный персептрон может разделять только линейно разделимые классы. В разд. 9.5 мы увидим, что многоуровневые сети значительно мощнее. Несмотря на простоту конструкции,

Machine Translated by Google

персептрон в представленном виде сходится очень медленно. Его можно ускорить путем нормализации вектора изменения веса. Формулы w = w ± x заменяются на w = w ± x/|x|. Таким образом, каждая точка данных имеет одинаковый вес во время обучения, независимо от ее значения.

Скорость сходимости сильно зависит от инициализации вектора w.
В идеале его вообще не нужно было бы менять, и алгоритм сходился бы после одной итерации. Мы можем приблизиться к этой цели, используя эвристическую инициализацию

$$w0 = \frac{x -}{x \quad M+} \quad \frac{\text{Икс,}}{x \quad M-}$$

которые мы рассмотрим более подробно в упражнении 8.5 на стр. 217.

Если мы сравним формулу персептрона с методом оценки, представленным в разд. 7.3.1, мы сразу видим их эквивалентность. Кроме того, персептрон, как простейшая модель нейронной сети, эквивалентен наивной байесовской сети, простейшему типу байесовской сети (см. упражнение 8.16 на стр. 219). Таким образом, очевидно, что несколько весьма различных алгоритмов классификации имеют общее происхождение.

В гл. 9 мы познакомимся с обобщением персептрона в виде алгоритма обратного распространения, который может разделять нелинейно разделимые множества за счет использования нескольких слоев и обладает лучшим правилом обучения.

8.3 Метод ближайших соседей

Для персептрона знания, имеющиеся в обучающих данных, извлекаются и сохраняются в сжатом виде в весах wi . Тем самым информация о данных теряется. Однако это именно то, что требуется, если система должна обобщать обучающие данные на новые данные. Обобщение в данном случае является трудоемким процессом, целью которого является нахождение компактного представления данных в виде функции, которая максимально хорошо классифицирует новые данные.

Запоминание всех данных простым их сохранением — это совсем другое. Здесь обучение чрезвычайно простое. Однако, как было сказано ранее, сохраненные знания не так легко применимы к новым, неизвестным примерам. Такой подход совершенно не годится, например, для обучения катанию на лыжах. Новичок никогда не станет хорошим лыжником, просто просматривая видео с хорошими лыжниками. Очевидно, что при автоматическом выполнении обучающих движений такого типа происходит нечто подобное тому, что происходит в случае перцептрона. После достаточно длительной практики знания, хранящиеся в обучающих примерах, трансформируются во внутреннее представление в мозгу.

Однако есть удачные примеры запоминания, при которых возможно и обобщение. При диагностике сложного случая врач может попытаться вспомнить подобные случаи из прошлого. Если у него здоровая память, то он может наткнуться на это дело, поискать его в своих файлах и, наконец, прийти к похожему диагнозу. При таком подходе врач должен хорошо чувствовать сходство, чтобы помнить наиболее похожий случай. Если он нашел это, то он должен спросить себя, достаточно ли это похоже, чтобы оправдать такой же диагноз.

Machine Translated by Google

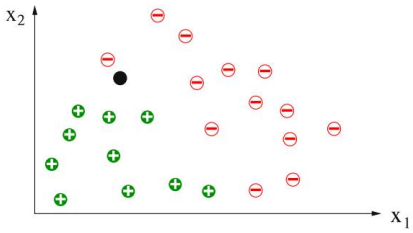


Рис. 8.13. В этом примере с отрицательными и положительными обучающими примерами метод ближайшего соседа группирует новую точку, отмеченную черным, в отрицательный класс

Что означает сходство в формальном контексте, который мы конструируем? Мы представляем обучающие выборки, как обычно, в многомерном пространстве признаков и определяем: Чем меньше их расстояние в пространстве признаков, тем более похожи два примера.

Применим теперь это определение к простому двумерному примеру на рис. 8.13. Здесь ближайший сосед черной точки является отрицательным примером. Таким образом, он отнесен к отрицательному классу.

Расстояние d(x, y) между двумя точками x R , н и y R н может быть например измеренное евклидовым расстоянием

$$d(x, y) = |x \quad y| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

Поскольку помимо этой метрики расстояния есть много других, имеет смысл подумать об альтернативах для конкретного приложения. Во многих приложениях одни функции важнее других. Поэтому часто имеет смысл по-разному масштабировать признаки с помощью весов wi . Затем формула читается

$$dw(x, y) = |x \quad y| = \sqrt{\sum_{i=1}^{n} w_i (x_i - y_i)^2}.$$

Следующая простая программа классификации ближайших соседей ищет в обучающих данных ближайшего соседа t для нового примера s, а затем классифицирует s точно так же , как t.

```
БЛИЖАЙШИЙ СОСЕД[M+,M ,s] t =
    argminx M+ M {d(s, x)}
    Если t M+ , то возврат («+»)
                Иначе Возврат(„–”)
```

3 Функционалы argmin и argmax определяют, аналогично min и max, минимум или максимум множества или функции. Однако вместо того, чтобы возвращать значение максимума или минимума, они выдают позицию, то есть аргумент, в котором появляется экстремум.

Machine Translated by Google

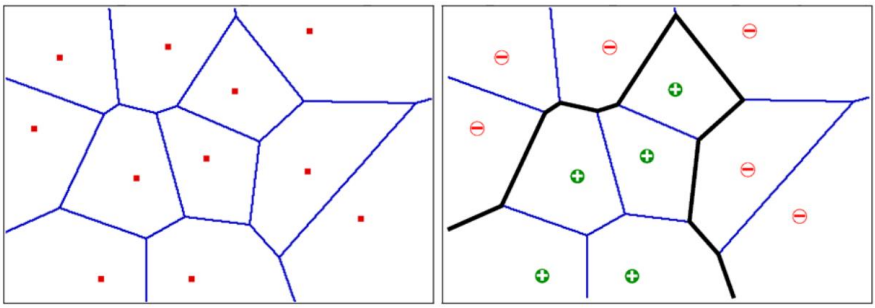


Рис. 8.14. Набор точек вместе с их диаграммой Вороного (слева) и разделительной линией, сгенерированной для двух классов M+ и M

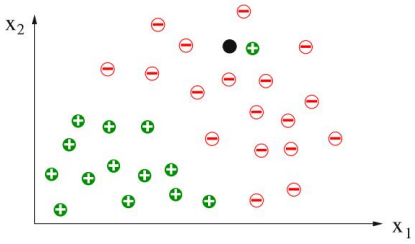


Рис. 8.15. Метод ближайшего соседа относит новую точку, отмеченную черным цветом , к неправильному (положительному) классу, поскольку ближайший сосед, скорее всего, классифицирован неправильно

В отличие от персептрона, метод ближайшего соседа не генерирует линию, разделяющую точки обучающих данных. Однако воображаемая линия, разделяющая два класса, определенно существует. Мы можем сгенерировать это, сначала сгенерировав так называемую диаграмму Вороного. На диаграмме Вороного каждая точка данных окружена выпуклым многоугольником, который, таким образом, определяет окрестности вокруг нее. Диаграмма Вороного обладает тем свойством, что для произвольной новой точки ближайшим соседом среди точек данных является точка данных, которая находится в той же окрестности. Если диаграмма Вороного для набора обучающих данных определена, то несложно найти ближайшего соседа для новой точки, подлежащей классификации. Затем членство в классе будет взято у ближайшего соседа.

На рис. 8.14 мы ясно видим, что метод ближайшего соседа значительно мощнее, чем персептрон. Он способен корректно реализовать сколь угодно сложные разделительные линии (в общем случае: гиперплоскости). Однако здесь есть опасность. Одна ошибочная точка может при определенных обстоятельствах привести к очень плохим результатам классификации. Такой случай имеет место на рис. 8.15 при классификации черной точки. Метод ближайшего соседа может классифицировать это неправильно. Если черная точка находится непосредственно рядом с положительной точкой, которая является выбросом из положительного класса, то она будет классифицироваться как положительная, а не отрицательная, как здесь предполагалось. Ошибочная подгонка к случайным ошибкам (шум) называется переподгонкой.

Machine Translated by Google


```
K-БЛИЖАЙШИЙ СОСЕД(M+,M   ,s)
    V = {k ближайших соседей в M+     M   }
    Если          |M+    V | > |M       V | Затем Возврат ("+")
    ИначеЕсли |M+    V | < |M       V | Затем Возврат(„-”)
    Еще                                    Возврат(Случайный("+", "-"))
```

Рис. 8.16. АЛГОРИТМ K-БЛИЖАЙШИХ СОСЕДЕЙ

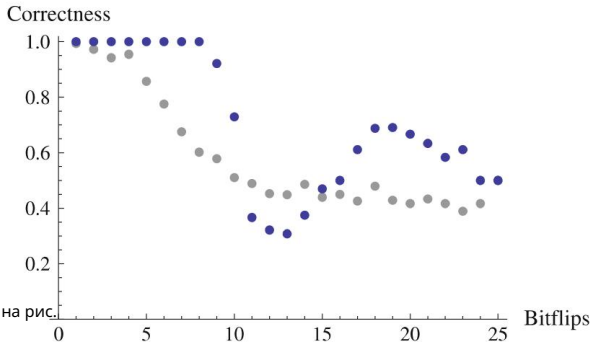


Рис. 8.17. Относительная правильность классификации ближайших соседей в зависимости от количества инвертированных битов. Структура кривой с минимумом в 13 и максимумом в 19 связана с особой структурой обучающих данных. Для сравнения значения персептрона из Примера 8.3 на стр. 174 показаны на рис. серый

Для предотвращения ложных классификаций из-за одиночных выбросов рекомендуется несколько сгладить поверхность деления. Этого можно добиться, например, с помощью алгоритма K-NEARESTNEIGHBOR на рис. 8.16, который принимает решение о большинстве среди k ближайших соседей.

Пример 8.4 Теперь применим NEARESTNEIGHBOR к примеру 8.3 на стр. 174. Поскольку мы имеем дело с двоичными данными, мы используем расстояние Хэмминга в качестве метрики расстояния4 . В качестве тестового примера мы снова используем модифицированные обучающие примеры с n последовательными инвертированными битами в каждом. На рис. 8.17 процент правильно классифицированных тестовых примеров показан как функция числа инвертированных битов b. До восьми инвертированных битов все шаблоны идентифицируются правильно. После этого количество ошибок быстро увеличивается. Это неудивительно, потому что тренировочный образец номер 2 на рис. 8.11 на странице 174 из класса M+ имеет расстояние Хэмминга 9 до двух тренировочных примеров, номера 4 и 5 из другого класса. Это означает, что тестовый образец, скорее всего, близок к образцам другого класса. Совершенно ясно мы видим, что классификация ближайших соседей превосходит персептрон в этом приложении до восьми ложных битов.

4Расстояние Хэмминга между двумя битовыми векторами — это количество различных битов двух векторов.

Machine Translated by Google

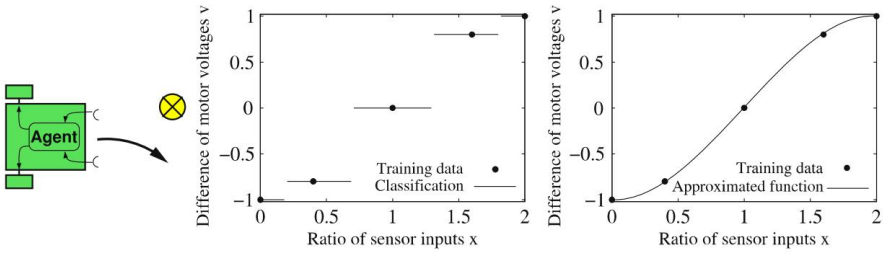


Рис. 8.18. Обучающийся агент, который должен избегать света (слева), представленный в виде классификатора (в центре) и в виде аппроксимации (справа)

8.3.1 Два класса, много классов, приближение

Классификация ближайших соседей также может применяться к более чем двум классам. Как и в случае с двумя классами, класс классифицируемого вектора признаков просто устанавливается как класс ближайшего соседа. Для метода k ближайших соседей класс должен быть определен как класс с наибольшим количеством членов среди k ближайших соседей.

Если количество классов велико, то, как правило, уже не имеет смысла использовать алгоритмы классификации, так как размер необходимых обучающих данных быстро растет с количеством классов. Кроме того, при определенных обстоятельствах важная информация теряется при классификации многих классов. Это станет ясно на следующем примере.

Пример 8.5 Предполагается, что автономный робот с простыми датчиками, подобными транспортным средствам Брайтенберга, представленным на рис. 1.1 на странице 2, учится уходить от света. Это означает, что он должен как можно оптимальнее научиться преобразовывать значения своих датчиков в сигнал рулевого управления, который управляет направлением движения. Робот оснащен двумя простыми датчиками света на передней стороне. Из сигналов двух датчиков (с sl для левого и sr для правого датчика) вычисляется соотношение x = sr/sl .

Чтобы управлять электродвигателями двух колес от этого значения x, разности v = Ur -Ul двух напряжений Ur и Ul левого и правого двигателей соответственно.
Теперь задача обучающегося агента состоит в том, чтобы избежать светового сигнала. Следовательно, он должен изучить 5- отображение f, которое вычисляет «правильное» значение v = f (x).

Для этого проведем эксперимент, в котором для нескольких измеренных значений x найдем как можно более оптимальное значение v. Эти значения нанесены в виде точек данных на рис. 8.18 и должны служить обучающими данными для агента обучения. При классификации ближайших соседей каждая точка в пространстве признаков (то есть на оси x) классифицируется точно так же, как ее ближайший сосед среди обучающих данных. Тогда функция управления двигателями представляет собой ступенчатую функцию с большими скачками (рис. 8.18, середина). Если нам нужны более тонкие шаги, мы должны предоставить соответственно больше обучающих данных. На

5 Чтобы сделать пример простым и читабельным, вектор признаков x был намеренно оставлен размерный.

Machine Translated by Google

с другой стороны, мы можем получить непрерывную функцию, если аппроксимируем гладкую функцию по пяти точкам (рис. 8.18 на странице 179 справа). Требование непрерывности функции f приводит к очень хорошим результатам даже без дополнительных точек данных

Для аппроксимации функций в точках данных существует множество математических методов, таких как полиномиальная интерполяция, сплайн-интерполяция или метод наименьших квадратов. Применение этих методов становится проблематичным в более высоких измерениях. Особая трудность в ИИ заключается в том, что необходимы безмодельные методы аппроксимации. То есть хорошая аппроксимация данных должна быть сделана без знания особых свойств данных или приложения. Очень хорошие результаты были достигнуты здесь с помощью нейронных сетей и других аппроксиматоров нелинейных функций, которые представлены в гл. 9.

Метод k ближайших соседей можно просто применить к задаче аппроксимации. В алгоритме K-NEARESTNEIGHBOR после определения множества V = {x1, x2,..., xk } среднее значение функции k ближайших соседей

$$f(x) = \frac{1}{k} \sum_{i=1}^{k} \phi(\text{кси}) \tag{8.2}$$

вычисляется и принимается как аппроксимация f^ для вектора запроса x. Чем больше становится k, тем более гладкой становится функция f^.

8.3.2 Расстояние имеет значение

При практическом применении как дискретных, так и непрерывных вариантов метода k ближайших соседей часто возникают проблемы. Когда k становится большим, обычно существует больше соседей с большим расстоянием, чем соседей с малым расстоянием. Таким образом, при вычислении f^ преобладают соседи, находящиеся далеко. Чтобы предотвратить это, k соседей взвешиваются таким образом, чтобы более удаленные соседи оказывали меньшее влияние на результат. При мажоритарном решении в алгоритме K-БЛИЖАЙШИЙ СОСЕД «голоса» взвешиваются с весом

$$\text{ви} = \frac{1}{1 + \alpha d(x, xi)^2}, \tag{8.3}$$

которая убывает пропорционально квадрату расстояния. Константа α определяет скорость убывания весов. Уравнение (8.2) теперь заменяется на

$$f(x) = \frac{\text{ки=1}\text{ж}\phi(\text{xi})}{\text{ки=1 ш}}.$$

Для равномерно распределенной концентрации точек в пространстве признаков это гарантирует, что влияние точек асимптотически стремится к нулю по мере увеличения расстояния. Таким образом, становится возможным использовать многие или даже все обучающие данные для классификации или аппроксимации заданного входного вектора.

Machine Translated by Google

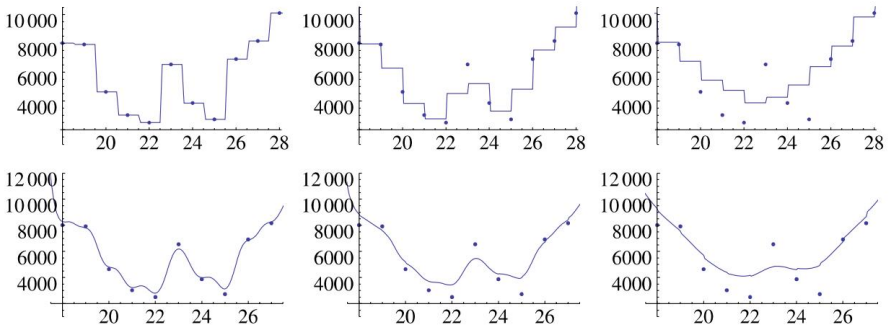


Рис. 8.19. Сравнение метода k-ближайших соседей (верхний ряд) с k = 1 (слева), k = 2 (в центре) и k = 6 (справа) с его взвешенным по расстоянию вариантом (нижний ряд) с α = 20 (слева), α = 4 (в центре) и α = 1 (справа) в одномерном наборе данных

Чтобы получить представление об этих методах, на рис. 8.19 метод k-ближайших соседей (в верхнем ряду) сравнивается с его взвешенной по расстоянию оптимизацией. Благодаря усреднению оба метода могут обобщать или, другими словами, компенсировать шумы, если число соседей для k-ближайшего соседа или параметр α заданы соответствующим образом. Диаграммы хорошо показывают, что метод взвешенного расстояния дает гораздо более гладкую аппроксимацию, чем метод k-ближайшего соседа. По качеству аппроксимации этот очень простой метод может хорошо конкурировать со сложными алгоритмами аппроксимации, такими как нелинейные нейронные сети, машины опорных векторов и гауссовы процессы.

Существует множество альтернатив весовой функции (также называемой ядром), приведенной в (8.3) на стр. 180. Например, можно использовать функцию Гаусса или аналогичную колоколообразную функцию. Для большинства приложений результаты не очень разумны при выборе ядра. Однако параметр ширины α, который для всех этих функций необходимо задавать вручную, оказывает большое влияние на результаты, как показано на рис. 8.19. Чтобы избежать такой неудобной ручной адаптации, были разработаны методы оптимизации для автоматической установки этого параметра [SA94, SE10].

8.3.3 Время вычислений

Как упоминалось ранее, обучение выполняется во всех вариантах метода ближайшего соседа путем простого сохранения всех обучающих векторов вместе с их метками (значениями класса) или значением функции f(x). Таким образом, нет другого алгоритма обучения, который учится так же быстро. Однако ответ на запрос о классификации или аппроксимации вектора x может стать очень дорогим. Простое нахождение k ближайших соседей для n обучающих данных требует затрат, которые растут линейно с n. Для классификации или аппроксимации есть дополнительная стоимость, линейная по k. Таким образом, общее время вычислений растет как Θ (n + k). Для больших объемов обучающих данных это может привести к проблемам.

Machine Translated by Google

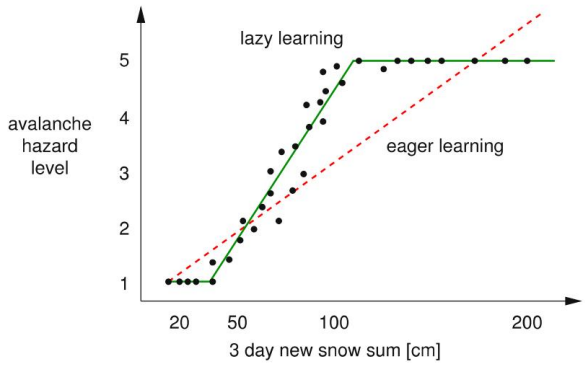


Рис. 8.20. Для определения лавинной опасности функция аппроксимируется по обучающим данным. Здесь для сравнения локальная модель (сплошная линия) и глобальная модель (штриховая линия)

8.3.4 Резюме и перспективы

Поскольку на этапе обучения представленных методов ближайшего соседа ничего не происходит, такие алгоритмы также обозначаются как ленивое обучение, в отличие от энергичного обучения, при котором этап обучения может быть дорогостоящим, но применение к новым примерам очень эффективно. Персептрон и все другие нейронные сети, обучение дереву решений и обучение байесовских сетей — это активные методы обучения. Поскольку ленивым методам обучения требуется доступ к памяти со всеми обучающими данными для аппроксимации нового ввода, их также называют обучением на основе памяти.

Для сравнения этих двух классов процессов обучения возьмем в качестве примера задачу определения текущей лавинной опасности по количеству свежевыпавшего снега в определенном районе Швейцарии6. На рис. 8.20 внесены определенные экспертами значения , которые мы хотим использовать в качестве обучающих данных. Во время применения алгоритма энергичного обучения, который выполняет линейную аппроксимацию данных, вычисляется пунктирная линия, показанная на рисунке. Из-за ограничения на прямую линию ошибка относительно велика и составляет максимум около 1,5 уровня опасности. При ленивом обучении ничего не вычисляется до поступления запроса о текущем уровне опасно Затем ответ вычисляется из нескольких ближайших соседей, то есть локально. Это может привести к кривой, показанной на рисунке, которая составлена из отрезков прямой и показывает гораздо меньшие ошибки. Преимуществом ленивого метода является его локальность. Приближение берется локально из текущего нового уровня снега, а не глобально. Таким образом, для принципиально равных классов функций (например, линейных функций) ленивые алгоритмы лучше.

Методы ближайших соседей хорошо подходят для всех проблемных ситуаций, в которых требуется хорошая локальная аппроксимация, но не предъявляются высокие требования к быстродействию системы. Упомянутый здесь предсказатель лавин, который запускается один раз в день, является таким приложением. Методы ближайшего соседа не подходят, когда описание знаний, извлеченных из данных, должно быть понятно человеку, что сегодня имеет место во многих приложениях интеллектуального анализа данных (см. раздел 8.4) . В

6 Общее количество снегопадов за три дня на самом деле является важной характеристикой для определения уровня опасности. Однако на практике используются дополнительные атрибуты [Bra01]. Используемый здесь пример упрощен.

Machine Translated by Google

Особенность	Запрос	Корпус из корпуса
Дефектная часть:	Задние фары	Передний свет
Модель велосипеда:	Марин Пайн Маунтин	ВСФ Т400
Год:	1993 г.	2001 г.
Источник питания:	Аккумулятор в	Динамо
Состояние лампы:	порядке	хорошо
Состояние светового кабеля:	?	хорошо
	Решение	
Диагноз:	?	Отсутствует передний электрический контакт
Ремонт:	?	Установить передний электрический контакт

Рис. 8.21 Пример простой диагностики для запроса и соответствующего случая из базы случаев

В последние годы эти методы обучения на основе памяти становятся популярными, и различные улучшенные варианты (например, локально взвешенная линейная регрессия) применил [Cle79].

Чтобы иметь возможность использовать описанные методы, должны быть доступны обучающие данные. в виде векторов целых или действительных чисел. Таким образом, они не подходят для приложения, в которых данные представлены символически, например, как первый логические формулы порядка. Сейчас мы кратко обсудим это.

8.3.5 Рассуждение на основе прецедентов

В рассуждении на основе прецедентов (CBR) метод ближайшего соседа распространяется на символические описания проблем и их решения. CBR используется для диагностики технических проблем в обслуживании клиентов или для телефонных горячих линий. Показанный пример на рис. 8.21 о диагностике выключения велосипедного фонаря иллюстрирует этот тип ситуация.

Ищется решение по запросу клиента с неисправным задним велосипедом свет. В правом столбце приведен случай, аналогичный запросу в среднем столбце. Это вытекает из базы прецедентов, которая соответствует обучающим данным в ближайшем метод соседа. Если мы просто возьмем наиболее похожий случай, как мы это делаем в ближайшем метод соседа, то мы закончим тем, что попытаемся отремонтировать передний фонарь, когда задний фонарь разбит. Таким образом, нам необходимо обратное преобразование решения обнаруженной аналогичной задачи обратно в запрос. Самые важные шаги в решении к случаю CBR выполняются на рис. 8.22 на стр. 184. Преобразование в этом пример прост: задний свет отображается на передний свет.

Какими бы красивыми и простыми ни казались эти методы в теории, на практике построение систем диагностики CBR является очень сложной задачей. Три главные трудности являются:

Моделирование Предметы приложения должны быть смоделированы в формальном контексте. Здесь логическое однообразие, известное нам из гл. 4, представляет трудности. Может разработчик предсказывает и отображает все возможные частные случаи и варианты проблемы?

Machine Translated by Google

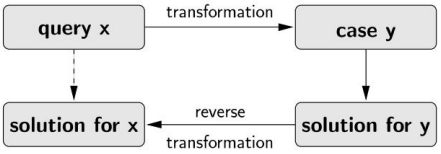


Рис. 8.22. Если для случая x найден аналогичный случай y, то для получения решения для x необходимо определить преобразование и применить его обратное к найденному случаю y

Сходство Поиск подходящей метрики сходства для символических, нечисловых признаков тур.

Преобразование Даже если подобный случай будет найден, пока неясно, как должно выглядеть отображение преобразования и обратное ему.

Действительно, сегодня существуют практические системы CBR для диагностических приложений. Однако по указанным причинам они остаются далеко позади людей-экспертов по производительности и гибкости. Интересной альтернативой CBR являются байесовские сети, представленные в гл. 7. Часто символическое представление задачи также может быть достаточно хорошо отображено в дискретные или непрерывные числовые признаки. Тогда можно успешно использовать упомянутые индуктивные методы обучения, такие как деревья решений или нейронные сети.

8.4 Обучение дереву решений

Обучение дереву решений — чрезвычайно важный алгоритм для ИИ, потому что он очень мощный, но также простой и эффективный для извлечения знаний из данных.
По сравнению с двумя уже представленными алгоритмами обучения он имеет важное преимущество. Извлеченные знания не только доступны и могут использоваться как функция черного ящика, но и могут быть легко поняты, интерпретированы и проконтролированы людьми в форме удобочитаемого дерева решений. Это также делает изучение дерева решений важным инструментом для интеллектуального анализа данных.

Мы обсудим функцию и применение обучения дерева решений с использованием алгоритма C4.5 . C4.5 был представлен в 1993 году австралийцем Россом Куинланом и является усовершенствованием своего предшественника ID3 (Iterative Dichotomiser 3, 1986). Он находится в свободном доступе для некоммерческого использования [Qui93]. Дальнейшее развитие, которое работает еще эффективнее и может учитывать затраты на решения, — это C5.0 [Qui93].

Система CART (Деревья классификации и регрессии, 1984 г.), разработанная Лео Брейманом [BFOS84], работает аналогично C4.5. Имеет удобный графический интерфейс, но стоит очень дорого.

Двадцатью годами ранее, в 1964 г., Дж. Сонквист и Дж. Морган представили систему CHAID (Chi-square Automatic Interaction Detectors), которая может автоматически генерировать деревья решений. У него есть примечательная особенность, заключающаяся в том, что он останавливает рост дерева до того, как оно станет слишком большим, но сегодня это уже не актуально.

Machine Translated by Google

Таблица 8.4 Переменные для задачи классификации лыж

Переменная	Ценить	Описание
Лыжи (целевая переменная)	Да нет	Должен ли я ехать на ближайший горнолыжный курорт при достаточном количестве снега?
Солнце (особенность)	Да нет	Сегодня есть солнце?
Snow_Dist (функция)	100, >100	Расстояние до ближайшего горнолыжного курорта с хорошие снежные условия (больше/меньше 100 км)
Выходные (особенность)	Да нет	Сегодня выходные?

Также интересен инструмент интеллектуального анализа данных KNIME (Konstanz Information Miner), который имеет дружественный пользовательский интерфейс и, используя библиотеку WEKA Java, также делает Возможна индукция деревьев решений. В разд. 8.8 мы представим KNIME.

Теперь мы сначала покажем на простом примере, как можно построить дерево решений. из обучающих данных, чтобы затем проанализировать алгоритм и применить его к более комплексный пример LEXMED для медицинской диагностики.

8.4.1 Простой пример

Преданный лыжник, живущий рядом с высокой сьеррой, красивым горным хребтом в Калифорнии, хочет, чтобы дерево решений помогло ему решить, стоит ли ехать. свою машину на горнолыжный курорт в горах. Таким образом, у нас есть проблемная лыжа с двумя классами да/нет на основе переменных, перечисленных в таблице 8.4.

На рис. 8.23 на стр. 186 показано дерево решений для этой проблемы. Дерево решений представляет собой дерево, внутренние узлы которого представляют признаки (атрибуты). Каждое ребро обозначает значение атрибута. В каждом листовом узле задается значение класса.

Данные, использованные для построения дерева решений, приведены в таблице 8.5 на стр. 186. Каждая строка в таблице содержит данные за один день и, таким образом, представляет образец. При ближайшем рассмотрении мы видим, что ряд 6 и ряд 7 противоречат друг другу. Таким образом, ни один алгоритм детерминированной классификации не может правильно классифицировать все данные. Таким образом, количество ложно классифицированных данных должно быть 1. Дерево на рис. 8.23 на стр. 186, таким образом, оптимально классифицирует данные.

Как создается такое дерево из данных? Чтобы ответить на этот вопрос, мы будем в сначала ограничимся дискретными атрибутами с конечным числом значений. Поскольку количество атрибутов также конечно, и каждый атрибут может встречаться не более одного раза на пути, существует конечное число различных деревьев решений. Простой и понятный алгоритм построение дерева просто генерировало бы все деревья, затем для каждого дерева вычислялось бы количество ошибочных классификаций данных, а в конце выбрать дерево с минимальным количеством ошибок. Таким образом, у нас был бы даже оптимальный алгоритм (в смысле ошибок для обучающих данных) для обучения дерева решений.

Очевидным недостатком этого алгоритма является неприемлемо большое время вычислений, так как количество признаков становится несколько больше. Мы будем теперь разработайте эвристический алгоритм, который, начиная с корня, рекурсивно строит

Machine Translated by Google

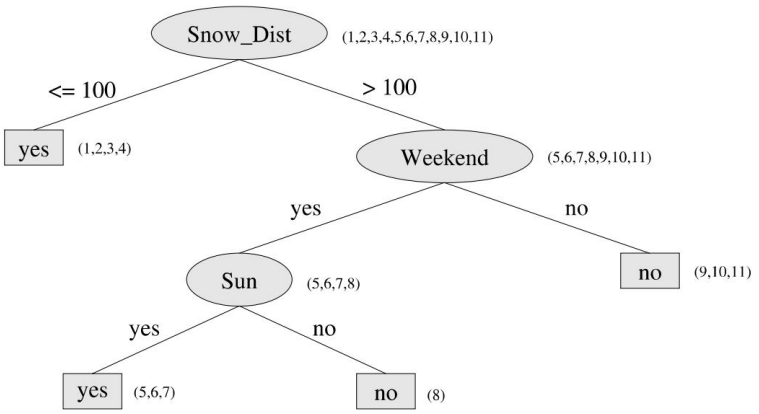


Рис. 8.23. Дерево решений для задачи классификации лыж. В списках справа от узлов даны номера соответствующих обучающих данных. Обратите внимание, что для листовых узлов солнечно = да только два из трех примеров классифицированы правильно

Таблица 8.5 Набор данных для проблема классификации лыж

День	Snow_Dist	Выходные	Солнце	Катание на лыжах
1	100	Да	Да	Да
2	100	Да	Да	Да
3	100	Да	Нет	да
4	100	Нет	Да	Да
5	>100	Да	Да	Да
6	>100	Да	Да	Да
7	>100	Да	да	нет
8	>100	Да	Нет	нет
9	>100	Нет	да	нет
10	>100	Нет	да	нет
11	>100	Нет	Нет	нет

дерево решений. Сначала атрибут с наибольшим информационным приростом (Snow_Dist) . выбирается для корневого узла из набора всех атрибутов. Для каждого значения атрибута (100, >100) в дереве есть ветвь. Теперь для каждой ветки этот процесс повторяется рекурсивно. При генерации узлов всегда выбирается атрибут с наибольшим информационным приростом среди еще не использованных атрибутов, т.е. дух жадной стратегии.

8.4.2 Энтропия как метрика информационного содержания

Описанный нисходящий алгоритм построения дерева решений, на каждом шаг выбирает атрибут с наибольшим приростом информации. Теперь мы представляем

Machine Translated by Google

энтропия как метрика информационного содержания набора тренировочных данных D. Если мы посмотрим только на бинарную переменную лыжи в приведенном выше примере, то D можно описать как

$$D = (\text{да}, \text{да}, \text{да}, \text{да}, \text{да}, \text{да}, \text{нет}, \text{нет}, \text{нет}, \text{нет}, \text{нет})$$

с предполагаемыми вероятностями

$$p1 = P(\text{да}) = 6/11 \text{ и } p2 = P(\text{нет}) = 5/11.$$

Здесь мы, очевидно, имеем распределение вероятностей p = (6/11, 5/11). В общем, для задачи n-класса это читается

$$p = (p1,\ldots,pn)$$

с

$$\sum_{i=1}^{n} pi = 1.$$

Чтобы представить информативность распределения, мы наблюдаем два крайних случая. Сначала пусть

$$p = (1, 0, 0, \ldots, 0). \tag{8.4}$$

То есть первое из n событий обязательно произойдет, а все остальные — нет. Таким образом, неопределенность относительно исхода событий минимальна. Напротив, для равномерного распределения

$$p = \frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n} \tag{8.5}$$

неопределенность максимальна, потому что ни одно событие нельзя отличить от других. Здесь Клод Шеннон задался вопросом, сколько битов потребуется для кодирования такого события. В определенном случае (8.4) нулевые биты необходимы, потому что мы знаем, что случай i = 1 всегда имеет место. В случае равномерного распределения (8.5) имеется n равновероятных возможностей. Для двоичного кодирования здесь требуется log2 n бит. Поскольку все отдельные вероятности равны pi = 1/n, для этого кодирования необходимы логарифмические $\frac{1}{1\ pi}$ 2 бита.

В общем случае p = (p1,...,pn), если вероятности элементарных событий отклоняются от равномерного распределения, то вычисляется математическое ожидание H для количества битов. С этой целью мы взвесим все значения $\frac{1}{1\ pi}$ log2 = log2 pi с их вероятностями и получим

$$H = \sum_{i=1}^{n} pi(-\log_2 pi) = \sum_{i=1}^{n} pi \log_2 pi .$$

Чем больше битов нам нужно для кодирования события, тем выше неопределенность результата. Поэтому мы определяем: