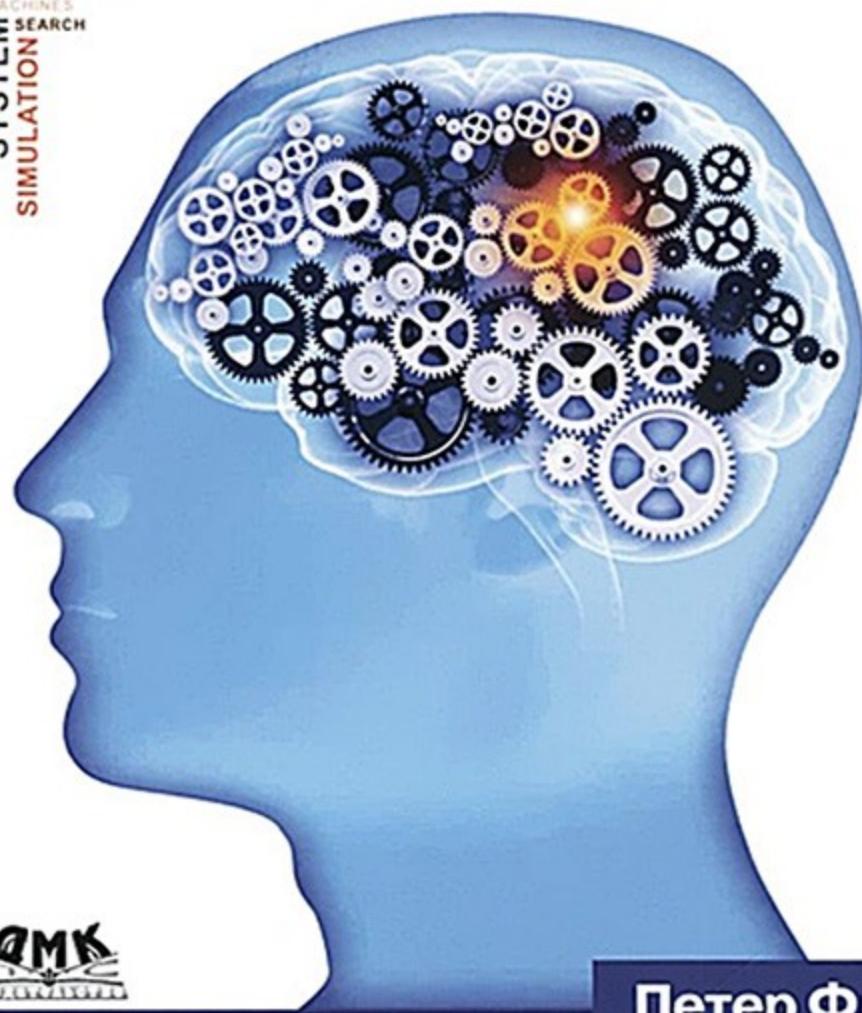


INTL SOFTWARE
KNOWLEDGE SCIENCE
INTELLIGENCE
ARTIFICIAL AI
LOGIC
SYSTEM SIMULATION
MACHINES SEARCH

Цветное издание



Петер Флах

Машинное обучение

Наука и искусство построения
алгоритмов, которые извлекают
знания из данных

Peter Flach

Machine Learning

**The Art and Science of Algorithms
that Make Sense of Data**



CAMBRIDGE
UNIVERSITY PRESS

Петер Флах

Машинное обучение

**Наука и искусство построения алгоритмов,
которые извлекают знания из данных**



Москва, 2015

**УДК 004.4
ББК 32.972
Ф70**

- Флах П.
- Ф70** Машинаное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2015. – 400 с.: ил.

ISBN 978-5-97060-273-7

Перед вами один из самых интересных учебников по машинному обучению – разделу искусственного интеллекта, изучающего методы построения моделей, способных обучаться, и алгоритмов для их построения и обучения. Автор воздал должное невероятному богатству предмета и не упустил из виду объединяющих принципов. Читатель с первых страниц видит машинное обучение в действии, но без не нужных на первых порах технических деталей. По мере изучения предмета тщательно подобранные примеры, сопровождаемые иллюстрациями, постепенно усложняются.

В книге описан широкий круг логических, геометрических и статистических моделей, затрагиваются и такие находящиеся на переднем крае науки темы, как матричная факторизация и анализ РХП. Особое внимание уделено важнейшей роли признаков. Устоявшаяся терминология дополняется введением в рассмотрение новых полезных концепций. В конце каждой главы приводятся ссылки на дополнительную литературу с авторскими комментариями.

Благодаря всему этому книга задает новый стандарт изучения такой сложной дисциплины как машинное обучение.

Цветные рисунки к книге размещены на нашем сайте www.dmkpress.com

**УДК 004.4
ББК 32.972**

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press. First published 2012.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-1-107-09639-4 (анг.)
ISBN 978-5-97060-273-7 (рус.)

© Peter Flach 2012
© Издание, перевод, ДМК Пресс, 2015

Посвящается Хэссел Флах
(1923–2006)

Содержание

Предисловие	11
Как читать эту книгу.....	11
Благодарности.....	12
Пролог: пример машинного обучения.....	14
1 Ингредиенты машинного обучения.....	25
1.1 Задачи: проблемы, решаемые методами машинного обучения	25
В поисках структуры	27
Оценка качества решения задачи.....	30
1.2 Модели: результат машинного обучения	32
Геометрические модели.....	33
Вероятностные модели	37
Логические модели.....	44
Группировка и ранжирование	49
1.3 Признаки: рабочая лошадка машинного обучения.....	50
Два способа использования признаков.....	52
Отбор и преобразование признаков.....	54
Взаимодействие между признаками	56
1.4 Итоги и перспективы.....	59
Что будет в книге дальше.....	61
2 Бинарная классификация и родственные задачи	62
2.1 Классификация	65
Оценка качества классификации	66
Наглядное представление качества классификации.....	70
2.2 Оценивание и ранжирование	75
Оценка и визуализация качества ранжирования	78

Преобразование ранжировщика в классификатор	84
2.3. Оценивание вероятностей классов.....	87
Качество оценивания вероятностей классов.....	88
Преобразование ранжировщиков в оценки вероятностей классов.....	91
2.4 Бинарная классификация и родственные задачи: итоги и дополнительная литература	93
3 За пределами бинарной классификации	96
3.1 Когда классов больше двух	96
Многоклассовая классификация	96
Многоклассовые оценки и вероятности	101
3.2 Регрессия	105
3.3 Обучение без учителя и дескриптивные модели	108
Прогностическая и дескриптивная кластеризация	109
Другие дескриптивные модели	114
3.4 За пределами бинарной классификации: итоги и литература для дальнейшего чтения.....	116
4 Концептуальное обучение	118
4.1 Пространство гипотез	119
Наименее обобщение	120
Внутренняя дизъюнкция.....	122
4.2 Пути в пространстве гипотез	124
Наиболее общие непротиворечивые гипотезы.....	128
Замкнутые концепты.....	130
4.3 За пределами конъюнктивных концептов	130
Применение логики первого порядка.....	135
4.4 Обучаемость	136
4.5 Концептуальное обучение: итоги и литература для дальнейшего чтения.....	139
5 Древовидные модели.....	142
5.1 Решающие деревья	146
5.2 Деревья ранжирования и оценивания вероятностей	151
Чувствительность к асимметричному распределению по классам.....	156
5.3 Обучение деревьев как уменьшение дисперсии	161
Деревья регрессии	161
Кластеризующие деревья.....	165

5.4 Древовидные модели: итоги и литература для дальнейшего чтения	168
6 Модели на основе правил.....	170
6.1 Обучение упорядоченных списков правил.....	170
Списки правил для ранжирования и оценивания вероятностей.....	176
6.2 Обучение неупорядоченных множеств правил.....	179
Применение множеств правил для ранжирования и оценивания вероятностей.....	183
Более пристальный взгляд на перекрытие правил	187
6.3 Обучение дескриптивных моделей на основе правил	189
Обучение правил для выявления подгрупп	190
Добыча ассоциативных правил.....	194
6.4 Обучение правил первого порядка.....	199
6.5 Модели на основе правил: итоги и литература для дальнейшего чтения.....	203
7 Линейные модели.....	206
7.1 Метод наименьших квадратов	208
Многомерная линейная регрессия	212
Регуляризованная регрессия	216
Применение регрессии по методу наименьших квадратов к задаче классификации	217
7.2 Перцептрон	218
7.3 Метод опорных векторов	223
Метод опорных векторов с мягким зазором.....	228
7.4 Получение вероятностей от линейных классификаторов.....	231
7.5 За пределами линейности – ядерные методы.....	236
7.6 Линейные модели: итоги и литература для дальнейшего чтения.....	239
8 Метрические модели	242
8.1 Так много дорог.....	242
8.2 Соседи и эталоны	248
8.3 Классификация по ближайшему соседу	253
8.4 Метрическая кластеризация	256
Алгоритм K средних	259
Кластеризация вокруг медоидов	261
Силуэты.....	262
8.5 Иерархическая кластеризация	264

	9
8.6 От ядер к расстояниям.....	269
8.7 Метрические модели: итоги и литература для дальнейшего чтения.....	270
9 Вероятностные модели.....	273
9.1 Нормальное распределение и его геометрические интерпретации.....	277
9.2 Вероятностные модели для категориальных данных	284
Использование наивной байесовской модели для классификации.....	286
Обучение наивной байесовской модели.....	289
9.3 Дискриминантное обучение путем оптимизации условного правдоподобия	293
9.4 Вероятностные модели со скрытыми переменными	297
ЕМ-алгоритм	299
Гауссовые смесевые модели	300
9.5 Модели на основе сжатия	304
9.6 Вероятностные модели: итоги и литература для дальнейшего чтения.....	306
10 Признаки.....	310
10.1 Виды признаков	310
Вычисления с признаками.....	311
Категориальные, порядковые и количественные признаки	315
Структурированные признаки.....	317
10.2 Преобразования признаков.....	318
Задание порога и дискретизация.....	319
Нормировка и калибровка.....	325
Неполные признаки	333
10.3 Конструирование и отбор признаков	334
Преобразование и разложение матриц	336
10.4 Признак: итоги и литература для дальнейшего чтения	339
11 Ансамбли моделей	342
11.1 Баггинг и случайные леса	343
11.2 Усиление.....	345
Обучение усиленных правил.....	349
11.3 Карта ансамблевого ландшафта	350
Смещение, дисперсия и зазоры.....	350
Другие ансамблевые методы.....	352
Метаобучение.....	352

11.4 Аnsamбли моделей: итоги и литература для дальнейшего чтения	353
12 Эксперименты в машинном обучении	355
12.1 Что измерять	356
12.2 Как измерять	360
12.3 Как интерпретировать.....	362
Интерпретация результатов, полученных на нескольких наборах данных.....	365
12.4 Эксперименты в машинном обучении: итоги и литература для дальнейшего чтения.....	368
Эпилог: что дальше?	371
Что нужно запомнить.....	373
Библиография.....	376
Предметный указатель.....	387

Предисловие

Идея этой книги появилась летом 2008 года, когда Бристольский университет, где я в то время работал, предоставил мне годичную стипендию для проведения научных исследований. Написать общее введение в машинное обучение я решил по двум причинам. Во-первых, существовала очевидная потребность в такой книге, которая дополняла бы многочисленные специальные издания на эту тему, а во-вторых, это позволило бы мне самому изучить что-то новое – ведь, как известно, лучший способ чему-то научиться – начать это преподавать.

Перед любым автором, желающим написать вводный учебник по машинному обучению, стоит трудная задача – осветить невероятно богатый материал, не упустив из виду объединяющих его принципов. Стоит уделить чрезмерно много внимания разнообразию дисциплины – и вы рискуете получить сборник слабо связанных между собой «рецептов». А увлекшись какой-нибудь из своих излюбленных парадигм, вы оставите за бортом массу других не менее интересных вещей. В конце концов, методом проб и ошибок я остановился на подходе, который позволяет подчеркнуть как единство, так и разнообразие. Единство достигается путем разделения *задач* и *признаков* – и то, и другое присутствует в любом подходе к машинному обучению, но зачастую принимается как само собой разумеющееся. А разнообразие обеспечивается рассмотрением широкого круга логических, геометрических и вероятностных моделей.

Понятно, что сколько-нибудь глубоко охватить весь предмет машинного обучения на 400 страницах нет никакой надежды. В эпилоге перечисляется ряд важных дисциплин, которые я решил не включать. На мой взгляд, машинное обучение – это сочетание статистики и представления знаний, и темы для книги были отобраны соответственно. Поэтому сначала довольно подробно рассматриваются решающие деревья и обучение на основе правил, а затем я перехожу к материалу, основанному на применении математической статистики. В книге постоянно подчеркивается важность интуиции, подкрепленная многочисленными примерами и иллюстрациями, многие из которых взяты из моих работ по применению РХП-анализа¹ в машинном обучении.

Как читать эту книгу

Печатный текст по природе своей линеен, поэтому материал организован так, чтобы книгу можно было читать последовательно. Но это не значит, что выборочное чтение невозможно, – я старался строить изложение по модульному принципу.

Например, если вы хотите поскорее приступить к первому алгоритму обучения, можете начать с раздела 2.1, где описывается бинарная классификация,

¹ Рабочая характеристика приемника (ROC – receiver operating characteristic). – Прим. перев.

а затем сразу перейти к главе 5, посвященной решающим деревьям, – без существенного нарушения непрерывности изложения. Прочитав раздел 5.1, можно перескочить к первым двум разделам главы 6, чтобы узнать о классификаторах на основе правил.

С другой стороны, читатель, интересующийся линейными моделями, может после раздела 2.1 перейти к разделу 3.2 о регрессионных задачах, а затем к главе 7, которая начинается с рассмотрения линейной регрессии. В порядке следования глав 4–9, посвященных логическим, геометрическим и вероятностным моделям, есть определенная логика, но по большей части их можно читать независимо; то же самое относится к главам 10–12 о признаках, ансамблях моделей и экспериментах в машинном обучении.

Отмечу также, что пролог и глава 1 носят вступительный характер и в значительной мере независимы от всего остального: в прологе есть кое-какие технические детали, но все они должны быть понятны даже человеку, не обучавшемуся в университете, а глава 1 содержит сжатый общий обзор материала, изложенного в книге. Обе главы можно бесплатно скачать с сайта книги по адресу www.cs.bris.ac.uk/~flach/mlbook; со временем будут добавлены и другие материалы, в частности лекционные слайды. Поскольку в книге такого объема неизбежны мелкие погрешности, на сайте имеется форма, с помощью которой мне можно отправить извещения о замеченных ошибках и опечатках.

Благодарности

Работа над книгой одного автора всегда была уделом одиночек, но мне повезло – я пользовался помощью и поддержкой многочисленных друзей и коллег. Тим Ковач (Tim Kovacs) в Бристоле, Люк де Редт (Luc De Raedt) в Лёвене и Карла Бродли (Carla Brodley) в Бостоне организовали группы читателей, от которых я получал весьма полезные отзывы. Своими замечаниями со мной любезно поделились также Хендрик Блокиль (Hendrik Blockeel), Натали Япковиц (Nathalie Japkowicz), Николас Лашиш (Nicolas Lachiche), Мартин ван Оттерло (Martijn van Otterlo), Фабрицио Ригуци (Fabrizio Riguzzi) и Мохак Шах (Mohak Shah). Тем или иным способом мне помогали и многие другие – спасибо всем.

Хосе Хернандес-Оралло (Jose Hernandez-Orallo), выйдя далеко за пределы служебных обязанностей, внимательно прочитал рукопись и высказал много конструктивных критических замечаний, которые я учел в той мере, в какой позволяло время. Хосе, с меня причитается.

Большое спасибо моим коллегам по Бристольскому университету: Тареку Абудавуду (Tarek Abudawood), Рафалу Богачу (Rafal Bogacz), Тило Бургхардту (Tilo Burghardt), Нелло Кристианини (Nello Cristianini), Тийл де Бье (Tijl De Bie), Бруно Голениа (Bruno Golenia), Саймону Прайсу (Simon Price), Оливеру Рэю (Oliver Ray) и Себастьяну Шпиглеру (Sebastian Spiegler) – за совместную работу и поучительные дискуссии. Благодарю также своих зарубежных коллег Иоханнеса Фурнкранца (Johannes Furnkranz), Цезаря Ферри (Cesar Ferri), Томаса

Гартнера (Thomas Gartner), Хосе Хернандес-Оралло, Николаса Лашиша (Nicolas Lachiche), Джона Ллойда (John Lloyd), Эдсона Мацувара (Edson Matsubara) и Рональдо Прати (Ronaldo Prati) за разрешение использовать в книге результаты нашей совместной работы и за иную помощь. В те моменты, когда проекту требовалось резкое ускорение, мне любезно обеспечивали уединение Керри, Пол, Дэвид, Рэни и Тринти.

Дэвид Транах (David Tranah) из издательства Кэмбридж Юниверсити Пресс всемерно содействовал запуску проекта и предложил пуантилистскую метафору для «извлечения знаний из данных», которая нашла отражение в рисунке на обложке книги (по мысли Дэвида, это «обобщенный силуэт», не обладающий портретным сходством ни с кем конкретно). Мэйри Сазерленд (Mairi Sutherland) тщательно отредактировала рукопись перед сдачей в набор.

Я посвящаю свой труд покойному отцу, который, безусловно, откупорил бы бутылку шампанского, узнав, что «этая книга» наконец закончена. Его трактовка проблемы индукции наводила на интересные, хотя и мрачноватые, размышления: та же рука, что каждый день кормит курицу, однажды сворачивает ей шею (приношу извинения читателям-вегетарианцам). Я благодарен своим родителям за то, что они снабдили меня всем необходимым для поиска собственного жизненного пути.

И наконец, не выразить словами все, чем я обязан своей жене Лайзе. Я начал писать эту книгу вскоре после нашей свадьбы, и мы даже представить не могли, что на эту работу уйдет почти четыре года. Ретроспективный взгляд – отличная штука: например, таким образом можно установить, что попытка закончить книгу одновременно с организацией международной конференции и капитальным ремонтом дома – не самая здравая мысль. Но наградой Лайзе за поддержку, ободрение и молчаливые страдания стал тот факт, что все три вещи все-таки приближаются к успешному завершению. Спасибо, любимая!

Петер Флах, Бристоль

Пролог: пример машинного обучения

Очень может быть, что вы, сами того не подозревая, давно уже пользуетесь технологиями машинного обучения. В большинство почтовых клиентов встроены алгоритмы определения и фильтрации спама, или нежелательной почты. Первые фильтры спама основывались на технике сопоставления с образцами, например с регулярными выражениями, причем сами образцы кодировались вручную. Однако скоро стало понятно, что этот подход непригоден для сопровождения и зачастую не обладает достаточной гибкостью – ведь что для одного спама, то для другого желанное послание¹! Необходимая адаптивность и гибкость достигается с помощью методов машинного обучения.

SpamAssassin – широко известный фильтр спама с открытым исходным кодом. Он вычисляет оценку входящего почтового сообщения, опираясь на ряд встроенных правил, или «критериев» и, если оценка оказывается не менее 5, включает в заголовки сообщения признак «спам» и сводный отчет. Вот пример отчета для полученного мной сообщения:

-0.1 RCVD_IN_MXRATE_WL RBL:	MXRate recommends allowing [123.45.6.789 listed in sub.mxrate.net]
0.6 HTML_IMAGE_RATIO_02 BODY:	HTML has a low ratio of text to image area
1.2 TVD_FW_GRAPHIC_NAME_MID BODY:	TVD_FW_GRAPHIC_NAME_MID
0.0 HTML_MESSAGE BODY:	HTML included in message
0.6 HTML_FONT_FACE_BAD BODY:	HTML font face is not a word
1.4 SARE_GIF_ATTACH FULL:	Email has a inline gif
0.1 BOUNCE_MESSAGE	MTA bounce message
0.1 ANY_BOUNCE_MESSAGE	Message is some kind of bounce message
1.4 AWL	AWL: From: address is in the auto white-list

Слева направо идут: оценка, вычисленная для данного критерия, идентификатор критерия и краткое описание, содержащее ссылку на релевантную часть сообщения. Как видим, оценка критерия может быть как отрицательной (свидетельство в пользу того, что сообщение не является спамом), так и положительной. Поскольку итоговая оценка равна 5.3, то сообщение может быть спамом. На самом деле это конкретное сообщение было уведомлением от промежуточного сервера о том, что какое-то другое сообщение, с чудовищной оценкой 14.6, было сочтено спамом и отвергнуто. В сообщение о недоставке было включено и исходное сообщение, а значит, оно унаследовало кое-какие характеристики по-

¹ Слово spam произошло от слияния двух слов: «spiced ham» (пряная ветчина). Так называлось мясное изделие, получившее недобрую славу после высмеивания в сериале «Летающий цирк Монти Пайтона». Поэтому в оригинале употреблена игра слов «one person's spam is another person's ham» – «что одному спам, то другому ветчина». – Прим. перев.

следнего, в частности низкое отношение текста к графике; отсюда и оценка, превышающая пороговое значение 5.

А вот пример важного письма, которого я долго ждал и в конце концов обнаружил в папке для спама:

```
2.5 URI_NOVOWEL URI:           URI hostname has long non-vowel sequence  
3.1 FROM_DOMAIN_NOVOWEL        From: domain has series of non-vowel letters
```

Это письмо касалось работы, которую мы с коллегой отправили на Европейскую конференцию по машинному обучению (ECML) и на Европейскую конференцию по принципам и практическим методам выявления знаний в базах данных (European Conference on Principles and Practice of Knowledge Discovery in Databases – PKDD), которые, начиная с 2001 года, проводятся совместно. Для освещения этих конференций, состоявшихся в 2008 году, в Интернете был создан домен www.ecmlpkdd2008.org, пользующийся заслуженно высокой репутацией у специалистов по машинному обучению. Однако в имени домена подряд идут одиннадцать согласных – вполне достаточно, чтобы возбудить подозрения у SpamAssassin! Этот пример наглядно демонстрирует, что ценность критериев SpamAssassin для разных пользователей может быть различна. Машинное обучение – великолепный способ создания программ, адаптирующихся к пользователю.



Как SpamAssassin вычисляет оценки, или «веса», для каждого из нескольких десятков критериев? Вот тут-то и приходит на помощь машинное обучение. Допустим, что имеется большой «обучающий набор» почтовых сообщений, которые были вручную помечены как «спам» или «неспам», и для каждого сообщения известны результаты по каждому критерию. Наша цель – вычислить вес для каждого критерия таким образом, чтобы все спамные сообщения получили итоговую оценку выше 5, а все хорошие – оценку ниже 5. В этой книге мы будем обсуждать различные способы решения этой задачи. А пока на простом примере проиллюстрируем основную идею.

Пример 1 (линейная классификация). Предположим, что есть только два критерия и четыре обучающих сообщения, одно из которых – спам (см. табл. 1). Для спамного сообщения оба критерия удовлетворяются, для одного хорошего не удовлетворяется ни один критерий, для второго удовлетворяется только первый, а для третьего – только второй. Легко видеть, что, приспав каждому критерию вес 4, мы сможем правильно «классифицировать» все четыре сообщения. В математической нотации, описываемой в замечании 1, этот классификатор можно было бы записать в виде $4x_1 + 4x_2 > 5$ или $(4,4) \cdot (x_1, x_2) > 5$. На самом деле при любом весе между 2.5 и 5 порог будет превышен, только когда удовлетворяются оба критерия. Можно было бы даже назначить критериям разные веса – при условии, что каждый из них меньше 5, а сумма больше 5; правда, не понятно, зачем нужны такие сложности при имеющихся обучающих данных.

Сообщение	x_1	x_2	Спам?	$4x_1 + 4x_2$
1	1	1	1	8
2	0	0	0	0
3	1	0	0	4
4	0	1	0	4

Таблица 1. Небольшой обучающий набор для фильтра SpamAssassin. В столбцах x_1 и x_2 приведены результаты применения критериев к каждому сообщению. В четвертом столбце указано, какие сообщения являются спамом. И из последнего столбца мы видим, что, сравнивая значение функции $4x_1 + 4x_2$ с 5, мы можем отделить спамные сообщения от хороших

Но, спросите вы, какое отношение все это имеет к обучению? Ведь это обычная математическая задача. Так-то оно так, но разве нельзя сказать, что SpamAssassin обучается распознавать почтовый спам на примерах и контрпримерах? И чем больше доступно обучающих данных, тем лучше SpamAssassin будет справляться с задачей. Идея о том, что качество решения возрастает с накоплением опыта, является главной для большинства, если не для всех форм машинного обучения. Мы будем использовать следующее общее определение: *машинным обучением называется систематическое обучение алгоритмов и систем, в результате которого их знания или качество работы возрастают по мере накопления опыта*. В случае фильтра SpamAssassin под «опытом», на котором он учится, понимается набор правильно размеченных обучающих данных, а под «качеством работы» – способность распознавать почтовый спам. На рис. 2 схематически показано место машинного обучения в задаче классификации почтового спама. В других задачах машинного обучения опыт может принимать иную форму, например исправление ошибок, вознаграждение в случае достижения определенной цели и т. д. Отметим также, что, как и при обучении человека, целью машинного обучения не обязательно является повышение качества решения определенной задачи; желаемым результатом может быть и расширение знаний.

Существует много полезных способов описать классификатор SpamAssassin математически. Если обозначить результат i -го критерия применительно к данному сообщению в виде x_i , где $x_i = 1$, если критерий удовлетворяется, и 0 в противном случае, а вес i -го критерия обозначить w_i , то итоговую оценку сообщения можно записать в виде $\sum_{i=1}^n w_i x_i$, отразив тот факт, что w_i дает вклад в сумму, только если $x_i = 1$, то есть если i -й критерий удовлетворяется. Если обозначить t пороговую величину, при превышении которой сообщение классифицируется как спам (в нашем примере 5), то «решающее правило» можно записать в виде $\sum_{i=1}^n w_i x_i > t$.

Отметим, что левая часть этого неравенства линейно зависит от переменных x_i , а это означает, что при увеличении какой-либо переменной x_i на некоторую величину δ сумма изменится на величину $(w_i \delta)$, не зависящую от значения x_i . Это уже не так, если зависимость от x_i квадратичная или вообще степенная с показателем степени, отличным от 1.

Описанную нотацию можно упростить, применив линейную алгебру. Обозначим \mathbf{w} вектор весов (w_1, \dots, w_n) , а \mathbf{x} – вектор результатов вычисления критериев (x_1, \dots, x_n) . Приведенное выше

неравенство можно записать с помощью скалярного произведения: $\mathbf{w} \cdot \mathbf{x} > t$. Если заменить неравенство равенством, $\mathbf{w} \cdot \mathbf{x} = t$, то получится «решающая граница», отделяющая спам от неспама. В силу линейности левой части решающей границы является плоскость в пространстве, натянутая на переменные x_i . Вектор \mathbf{w} перпендикулярен этой плоскости и направлен в сторону спама. На рис. 1 это наглядно показано в случае двух переменных.

Иногда удобно еще упростить запись, введя дополнительную постоянную «переменную» $x_0 = 1$ с фиксированным весом $w_0 = -t$. Тогда получается точка в расширенном пространстве данных $\mathbf{x}^* = (1, x_1, \dots, x_n)$ и расширенный вектор весов $\mathbf{w}^* = (-t, w_1, \dots, w_n)$. При этом решающее правило примет вид $\mathbf{w}^* \cdot \mathbf{x}^* > 0$, а уравнение решающей границы — $\mathbf{w}^* \cdot \mathbf{x}^* = 0$. В этих так называемых однородных координатах решающая граница проходит через начало расширенной системы координат ценой добавления лишнего измерения (отметим, однако, что на данные это не влияет, потому что все точки и «истинная» решающая граница расположены в плоскости $x_0 = 1$).

Замечание 1. SpamAssassin в математических обозначениях. В таких абзацах я буду напоминать о полезных концепциях и обозначениях. Если что-то окажется незнакомым, то для полного понимания изложенного в книге материала придется потратить некоторое время на изучение — обратитесь к другим книгам и сетевым ресурсам, например www.wikipedia.org или mathworld.wolfram.com.

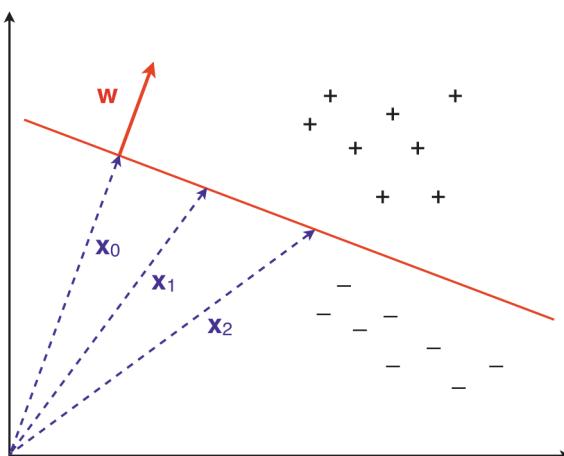


Рис. 1. Пример линейной классификации в двух измерениях. Прямая линия отделяет положительные результаты от отрицательных. Она определена уравнением $\mathbf{w} \cdot \mathbf{x}_i = t$, где \mathbf{w} — вектор, перпендикулярный решающей границе и направленный в сторону положительных результатов, t — порог принятия решения, а \mathbf{x}_i — вектор, оканчивающийся на решающей границе. Вектор \mathbf{x}_0 направлен туда же, куда и \mathbf{w} , откуда следует, что $\mathbf{w} \cdot \mathbf{x}_0 = \|\mathbf{w}\| \|\mathbf{x}_0\| = t$ ($\|\mathbf{x}\|$ обозначает длину вектора \mathbf{x}). Следовательно, решающую границу можно также описать уравнением $\mathbf{w} \cdot (\mathbf{x} - \mathbf{x}_0) = 0$, и иногда такая запись оказывается удобнее. В частности, из этого уравнения сразу видно, что положение решающей границы определяется только направлением \mathbf{w} , но не его длиной.

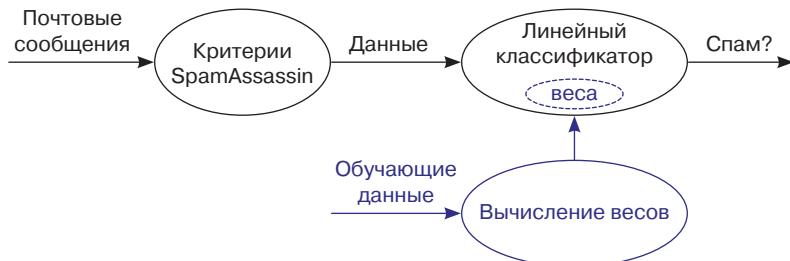


Рис. 2. В верхней части показано, как SpamAssassin подходит к задаче классификации почтового спама: текст каждого сообщения представляется точкой в пространстве данных, определяемой результатами вычисления встроенных критериев, а для решения «спам или неспам» применяется линейный классификатор. В нижней части показано то, что относится к машинному обучению

Мы уже видели, что у задачи машинного обучения, даже у такой простой, как в примере 1, может быть несколько решений. Тогда возникает вопрос: какое решение выбрать? И тут нужно понимать, что качество решения на обучающих данных нас не волнует – мы и так знаем, какие из предъявленных сообщений – спам! А волнует нас, как поведет себя классификатор на *будущих* сообщениях. На первый взгляд, получается порочный круг: чтобы узнать, правильно ли классифицировано сообщение, мне нужно знать его истинный класс, но если истинный класс известен, то классификатор уже не нужен. Однако важно помнить, что хорошее качество работы на обучающих данных – лишь средство к достижению цели, а не сама цель. На самом деле стремление добиться исключительно хорошего качества на обучающих данных легко может привести к удивительному и потенциально опасному явлению – *переобучению*.

Пример 2 (переобучение). Представьте, что вы готовитесь к экзамену по основам машинного обучения. По счастью, профессор Флах выложил в Сеть вопросы, которые задавались на предыдущем экзамене, и ответы на них. Вы начинаете отвечать на старые вопросы и сравнивать свои ответы с опубликованными. К сожалению, вы слишком увлеклись и тратите все свое время на запоминание ответов на старые вопросы. Если на предстоящем экзамене будут задаваться только старые вопросы, то все у вас сложится прекрасно. Но если материал останется тем же, а вопросы будут другими, то окажется, что ваша методика никуда не годится, и оценка будет гораздо ниже той, что вы заслужили бы при традиционной подготовке. В таком случае можно сказать, что вы переобучились на вопросах прошлых лет и приобретенные знания не обобщаются на вопросы будущего экзамена.

Обобщение – это, пожалуй, самая фундаментальная концепция машинного обучения. Если знания, которые SpamAssassin получил из предъявленных когда-то обучающих данных, переносятся – обобщаются – на ваши почтовые сообщения, вы довольны; если нет, вы начинаете искать более качественный фильтр

спама. Однако переобучение – не единственная возможная причина низкого качества работы на новых данных. Возможно, программисты SpamAssassin использовали обучающие данные, не репрезентативные для тех почтовых сообщений, которые приходят вам. К счастью, у этой проблемы есть решение: взять другие обучающие данные с такими же характеристиками, как у вашей почты. Машинное обучение – замечательная технология, позволяющая адаптировать поведение программы к конкретным обстоятельствам, и есть немало фильтров почтового спама, которые допускают обучение на пользовательских данных.

Таким образом, если существует несколько решений, то нужно выбрать то, которое не переобучено. Ниже мы обсудим, как это сделать, и даже разными способами. Ну а что сказать о противоположной ситуации, когда не существует ни одного решения, идеально классифицирующего обучающие данные? Представим, например, что сообщение 2 из примера 1, на котором оба критерия не удовлетворяются, – это спам, тогда не найдется ни одной прямой линии, разделяющей спам и неспам (можете убедиться в этом, изобразив все четыре сообщения в виде точек на плоскости, так что x_1 лежит на одной оси, а x_2 – на другой). В этом случае есть несколько вариантов действий. Первый – просто игнорировать это сообщение; возможно, оно нетипично или неправильно помечено (так называемый *шум*). Второй – взять более выразительный классификатор. Например, можно ввести второе решающее правило для фильтрации спама: в дополнение к условию $4x_1 + 4x_2 > 5$ добавить альтернативное условие $4x_1 + 4x_2 < 1$. Отметим, что при этом придется вычислить также другой порог и, возможно, другой вектор весов. Этот вариант можно рассматривать, только если обучающих данных достаточно для надежного вывода дополнительных параметров.



Линейная классификация в духе SpamAssassin может послужить полезным введением, но если бы она была единственным способом машинного обучения, то книга получилась бы куда тоньше. А что, если результатом обучения должны быть не только веса, но и сами критерии? Как решить, является ли отношение текстового материала к графическому хорошим критерием? Да и вообще, откуда этот критерий взялся? В этой области машинному обучению есть что предложить. Вероятно, вы уже заметили, что рассмотренные до сих пор критерии SpamAssassin не принимали во внимание *содержание* почтового сообщения. А ведь такие слова и словосочетания, как «виагра», «бесплатный iPod» или «подтвердите данные вашего счета», – верные индикаторы спама, тогда как другие – например, использование прозвища, известного только вашим друзьям, – свидетельствуют, что сообщение хорошее. Поэтому многие фильтры спама применяют методы классификации текста. Грубо говоря, хранят словарь слов и словосочетаний, являющихся признаками спама и неспама. Для каждого включенного в словарь элемента собирается статистика по обучающему набору. Пусть, например, слово «виагра» встретилось в четырех спамных сообщениях и одном хорошем. Если затем в новом сообщении нам попадется слово «виагра», то можно будет

заключить, что оно с вероятностью 0.80 (4:1) является спамом, а с вероятностью 0.20 – неспамом (см. основные понятия теории вероятностей в замечании 2).

Правда, ситуация несколько сложнее, чем кажется на первый взгляд, потому что мы должны принимать в расчет частоту спама. Предположим для определенности, что в среднем я получаю одно спамное сообщение на каждые шесть хороших (ах, если бы!). Это означает, что у следующего сообщения шанс оказаться спамным составляет 1:6, то есть не очень высокий, хотя и не пренебрежимо малый. Если затем выясняется, что это сообщение содержит слово «виагра», которое в спаме встречается в четыре раза чаще, чем в хороших сообщениях, то нужно каким-то образом учсть обе вероятности. Ниже мы узнаем о правиле Байеса, которое говорит, что их нужно просто перемножить: если умножить 1:6 на 4:1, то получится 4:6, то есть вероятность, что сообщение является спамом, равна 0.4. Иными словами, несмотря на наличие слова «виагра», сообщение, скорее всего, является хорошим. Но это же бред какой-то! Или не бред?

С вероятностями связаны «случайные величины», которые описывают исходы «событий». События часто гипотетические, и потому вероятности приходится оценивать. Взять, к примеру, утверждение «42% населения Великобритании одобряет премьер-министра». Единственный способ узнать, верно оно или неверно, состоит в том, чтобы задать вопрос каждому жителю Великобритании. Очевидно, это нереально. Вместо этого опрашивается некоторая выборка (хочется надеяться, репрезентативная), поэтому правильнее было бы сформулировать утверждение так: «42% опрошенной выборки из населения Великобритании одобряет премьер-министра». Отметим, что утверждения сформулированы в терминах процентной доли, или «относительной частоты»; в терминах вероятностей то же утверждение звучало бы так: «вероятность того, что случайно выбранный житель Великобритании одобряет премьер-министра, оценивается как 0.42». В данном случае событием является «данный случайно выбранный человек одобряет премьер-министра». Условной вероятностью $P(A|B)$ называется вероятность того, что событие A произойдет, если известно, что событие B произошло. Например, возможно, что мужчины и женщины одобряют премьер-министра с разной частотой. Если обозначить $P(\text{PM})$ вероятность того, что случайно выбранный человек одобряет премьер-министра, а $P(\text{PM}|\text{женщина})$ – вероятность того, что случайно выбранная женщина одобряет премьер-министра, то $P(\text{PM}|\text{женщина}) = P(\text{PM}, \text{женщина}) / P(\text{женщина})$, где $P(\text{PM}, \text{женщина})$ – вероятность «совместного события», заключающегося в том, что случайно выбранный человек одобряет премьер-министра и одновременно является женщиной, а $P(\text{женщина})$ – вероятность того, что случайно выбранный человек является женщиной (то есть доля женщин в населении Великобритании).

Приведем еще два полезных тождества: $P(A,B) = P(A|B)P(B) = P(B|A)P(A)$ и $P(A|B) = P(B|A)P(A)/P(B)$. Последнее, называемое «правилом Байеса», будет играть важную роль в этой книге. Отметим, что многие из этих тождеств обобщаются на случай, когда случайных величин больше двух, например «цепное правило исчисления вероятностей»: $P(A,B,C,D) = P(A|B,C,D)P(B|C,D)P(C|D)P(D)$.

Два события A и B называются независимыми, если $P(A|B) = P(A)$, то есть от того, что мы знаем, что B произошло, вероятность A не изменяется. Эквивалентная формулировка: $P(A,B) = P(A)P(B)$. В общем случае перемножение вероятностей основывается на предположении о независимости соответствующих событий.

«Шанс» события – это отношение вероятности того, что событие произойдет, к вероятности того, что оно не произойдет. Иначе говоря, если вероятность некоторого события равна p , то его шанс равен $o = p/(1-p)$. И наоборот, $p = o/(o+1)$. Таким образом, вероятность 0.8 соответствует шансу 4:1, а противоположному шансу 1:4 соответствует вероятность 0.2. Если же событие мо-

жет с равным успехом как произойти, так и не произойти, то его вероятность равна 0.5, а шанс – 1:1. И хотя по большей части мы будем использовать вероятностную нотацию, шансы иногда удобнее, потому что выражаются в виде отношения.

Замечание 2. Основные понятия теории вероятностей

В применении к рассматриваемой задаче нужно ясно понимать, что имеются два независимых свидетельства: частота спама и вхождение слова «виагра». Они действуют в противоположных направлениях, и потому важно оценить их относительную силу. Числа говорят нам, что для преодоления того факта, что спам – относительно редкое явление, необходимо, чтобы шанс был не ниже 6:1. Шанс появления слова «виагра» оценивается как 4:1, этого недостаточно для перевеса в сторону спама, то есть мы не можем заключить, что сообщение действительно является спамом. Наличие слова «виагра» позволяет лишь сказать, что утверждение «это сообщение хорошее» стало гораздо менее вероятным, поскольку его вероятность снизилась с $6/7 = 0.86$ до $6/10 = 0.60$.

Схема «байесовской» классификации хороша тем, что ее можно повторить при наличии дополнительных свидетельств. Пусть, например, шансы в пользу спама при наличии словосочетания «голубая таблетка» оцениваются как 3:1 (то есть среди сообщений, содержащих это словосочетание, спама в три раза больше, чем неспама), и еще предположим, что наше сообщение содержит как «виагра», так и «голубая таблетка». Тогда произведение шансов 4:1 и 3:1 дает 12:1, и этого вполне достаточно, чтобы перевесить шанс 1:6, ассоциированный с низкой частотой спама (результатирующий шанс равен 2:1, то есть вероятность спама повысилась до 0.67 против 0.40 без «голубой таблетки»).

Из того, что нам нет необходимости оценивать совместные вероятности событий, следует, что можно ввести в рассмотрение большее число переменных. На самом деле словарь типичного байесовского фильтра спама или классификатора текстов может содержать порядка 10 000 терминов¹. Поэтому, вместо того чтобы вручную составлять небольшой набор «признаков», которые эксперты считают релевантными, или обладающими предсказательной силой, мы включаем куда больший набор и поручаем классификатору определить, какие признаки важны и в каких сочетаниях.



Следует отметить, что, перемножая шансы «виагры» и «голубой таблетки», мы неявно предполагаем, что соответствующие события независимы. Очевидно, что это не так: зная, что сообщение содержит словосочетание «голубая таблетка», мы не удивимся, встретив также и «виагру». В терминах вероятностей это формулируется следующим образом:

¹ В действительности словосочетания, содержащие несколько слов, обычно разлагаются на отдельные слова, то есть $P(\text{голубая таблетка})$ оценивается как $P(\text{голубая}) P(\text{таблетка})$.

- ☞ вероятность $P(\text{виагра}|\text{голубая таблетка})$ близка к 1;
- ☞ следовательно, совместная вероятность $P(\text{виагра}, \text{голубая таблетка})$ близка к $P(\text{голубая таблетка})$;
- ☞ следовательно, шанс сообщения оказаться спамом вследствие вхождения слов «виагра» и «голубая таблетка» мало отличается от шанса оказаться спамом вследствие вхождения одной лишь «голубой таблетки».

Иначе говоря, перемножая эти шансы, мы дважды учитываем одну и ту же информацию. Результирующее произведение 12:1 почти наверняка является завышенной оценкой, а истинный шанс вряд ли будет больше, чем 5:1.

Похоже, мы загнали себя в угол. Чтобы избежать завышения оценки, мы должны принимать во внимание совместное вхождение словосочетаний, но с вычислительной точки зрения задача практически разрешима, только если считать их независимыми.

Кажется, в действительности нам нужно нечто, близкое к такой основанной на правилах модели:

1. Если почтовое сообщение содержит слово «виагра», то его шанс оказаться спамом оценивается как 4:1.
2. Иначе, если оно содержит словосочетание «голубая таблетка», то его шанс оказаться спамом оценивается как 3:1.
3. Иначе его шанс оказаться спамом оценивается как 1:6.

Под первое правило подпадают все сообщения, содержащие слово «виагра» вне зависимости от того, встречается в них словосочетание «голубая таблетка» или нет, поэтому завышения оценки не происходит. Под второе правило подпадают *только* сообщения, содержащие словосочетание «голубая таблетка» без слова «виагра», – это гарантируется союзом «иначе». И под третье правило подпадают все остальные сообщения: не содержащие ни «виагры», ни «голубой таблетки».

Существование таких классификаторов, основанных на правилах, состоит в том, что сообщения рассматриваются не единообразно, а индивидуально. В каждом случае выделяются только наиболее релевантные признаки. Случай можно определить с помощью нескольких вложенных признаков.

1. Сообщение содержит слово «виагра»?
 - (a) если да: сообщение содержит словосочетание «голубая таблетка»?
 - i. если да: оценить шанс спама как 5:1.
 - ii. если нет: оценить шанс спама как 4:1.
 - (b) если нет: сообщение содержит слово «лотерея»?
 - i. если да: оценить шанс спама как 3:1.
 - ii. если нет: оценить шанс спама как 1:6.

Эти четыре случая характеризуются логическими условиями вида «сообщение содержит слово “виагра”, но не содержит словосочетание “голубая таблетка”». Существуют эффективные алгоритмы выявления комбинаций признаков, обладающих наибольшей предсказательной силой, и организации их в виде правил или деревьев. Мы познакомимся с ними ниже.



Мы рассмотрели три практических примера применения машинного обучения для распознавания почтового спама. Специалисты называют рассмотренную задачу бинарной классификацией, поскольку требуется отнести объекты (почтовые сообщения) к одному из двух классов: спам или неспам. Для решения задачи каждое сообщение описывается с помощью ряда переменных, или признаков. В случае фильтра SpamAssassin признаки вручную отбирались экспертом по фильтрации спама, а в случае байесовской классификации текстов использовался большой словарь. Вопрос теперь заключается в том, как использовать признаки, чтобы отличить спам от неспама. Нам необходимо каким-то образом установить связь между признаками и классом – специалисты называют такую связь *моделью*, – путем анализа *обучающего набора* сообщений, уже помеченных правильным классом.

- ☞ В примере фильтра SpamAssassin мы вывели линейное уравнение вида $\sum_{i=1}^n w_i x_i > t$, где x_i – «булевы» признаки, которые принимают значение 0 или 1 и показывают, удовлетворяет сообщение i -му критерию или нет, w_i – веса признаков, выведенные на основе анализа обучающего набора, а t – пороговая величина, при превышении которой сообщение классифицируется как спам.
- ☞ В примере байесовской классификации мы воспользовались решающим правилом, которое можно записать в виде $\prod_{i=0}^n o_i > 1$, где величины $o_i = P(\text{спам}|x_i)/P(\text{неспам}|x_i)$, $1 \leq i \leq n$ – шанс, что сообщение окажется спамом, ассоциированный с каждым словом x_i из словаря, и $o_0 = P(\text{спам})/P(\text{неспам})$ заранее вычислены на основе данных из обучающего набора.
- ☞ В примере классификатора на основе правил мы построили логические условия, которые выделяют подмножества данных, настолько похожих, что им можно сопоставить определенную метку.

Итак, налицо основные ингредиенты машинного обучения: задачи, модели и признаки. На рис. 3 показано, как эти ингредиенты соотносятся. Сравнив этот рисунок с рис. 2, мы увидим, что модель занимает центральное место, а не просто является набором параметров классификатора, который во всех остальных отношениях определен признаками. Эта гибкость необходима для включения широкого спектра моделей, применяемых в машинном обучении. Подчеркнем различие между задачами и проблемами обучения: *задачи решаются с помощью моделей, а проблемы обучения – алгоритмами обучения, которые порождают модели*. И хотя это различие все понимают, терминология может отличаться: например, некоторые авторы употребляют выражение «задача обучения» вместо принятого нами «проблема обучения».

Подводя итог, можно сказать, что *предметом машинного обучения является использование нужных признаков для построения моделей, подходящих для решения правильно поставленных задач*. Я называю эти составные части «ингредиентами», чтобы подчеркнуть, что они могут принимать различные формы, и для получе-

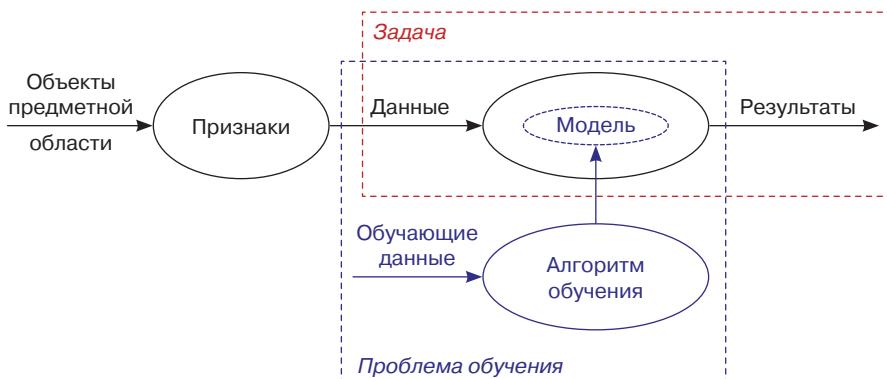


Рис. 3. Как машинное обучение применяется для решения поставленной задачи. Задача (**красный** прямоугольник) нуждается в подходящем отображении – модели – данных, описываемых признаками, на результаты. Получение такого отображения из обучающих данных и является предметом обучения (**синий** прямоугольник)

ния удачного «блюда» их необходимо правильно подобрать и смешать в нужной пропорции. Специалисты по машинному обучению называют эту деятельность приложением (построение модели для решения практической задачи с применением средств машинного обучения, если использовать данные из предметной области). Нельзя стать хорошим шеф-поваром без знания имеющихся на кухне ингредиентов, и то же самое относится к специалисту по машинному обучению. Наши основные ингредиенты – задачи, модели и признаки – подробно обсуждаются, начиная с главы 2. Но сначала насладимся небольшим «дегустационным меню», в котором я сервирую несколько примеров, чтобы вы почувствовали эти ингредиенты на вкус.



Ингредиенты машинного обучения

Смысл машинного обучения состоит в использовании нужных признаков для построения моделей, подходящих для решения правильно поставленных задач, – этой фразой, наглядно представленной на рис. 3, мы закончили пролог. По существу, *признаки* определяют «язык», на котором описываются объекты предметной области, будь то почтовые сообщения или сложные органические молекулы. Имея представление в виде признаков, мы уже можем не возвращаться к самим объектам предметной области; именно поэтому признаки играют такую важную роль в машинном обучении. В разделе 1.3 мы присмотримся к ним поближе. *Задача* – это абстрактное представление проблемы с участием объектов предметной области, которую мы хотим решить. Чаще всего требуется классифицировать объекты, то есть отнести каждый объект к одному из двух или более классов, но на страницах этой книги мы встретим и другие задачи. Многие задачи можно представить в виде отображения исходных данных на результаты. Такое отображение, или *модель*, само является результатом алгоритма машинного обучения, примененного к обучающим данным. Как мы увидим в разделе 1.2, разнообразие моделей весьма широко.

Мы начнем эту главу с обсуждения задач – проблем, которые можно решить методами машинного обучения. Различных моделей очень много, но, как выясняется, все они предназначены для решения узкого круга задач и пользуются немногими типами признаков. Можно сказать, что *модели обеспечивают разнообразие предмета машинного обучения, тогда как задачи и признаки придают ему единство*.

1.1 Задачи: проблемы, решаемые методами машинного обучения

В прологе было описано распознавание почтового спама. Это задача бинарной классификации – пожалуй, самая распространенная в машинном обучении, она будет встречаться нам не раз. Очевидное обобщение – задачи классификации с числом классов больше двух. Так, может быть интересно распознавать различные виды хороших почтовых сообщений, например относящиеся к работе и личные. К этой задаче можно было бы подойти как к комбинации двух задач бинарной классификации: сначала отделить спам от неспама, а затем разбить хорошие

сообщения на рабочие и личные. Однако при этом теряется потенциально полезная информация, поскольку некоторые спамные сообщения выглядят скорее как личные, нежели как рабочие. Поэтому зачастую выгодно рассматривать **многоклассовую классификацию** как самостоятельную задачу машинного обучения. На первый взгляд, ничего особенного: ведь все равно придется обучить модель, чтобы связать класс с признаками. Но при такой, более общей постановке некоторые концепции нуждаются в переосмыслении: например, понятие решающей границы не столь очевидно, как в случае с двумя классами.

Иногда естественно вообще отказаться от дискретных классов, а предсказывать некое вещественное число. Например, что, если требуется оценить срочность входящего сообщения по некоторой шкале? Такая задача называется **регрессией**, суть ее состоит в том, чтобы построить вещественную функцию на основании обучающих примеров, помеченных истинными значениями функции. Например, я мог бы построить такой обучающий набор, случайно выбрав несколько сообщений из ящика «Входящие» и приписав им оценку срочности по шкале от 0 (игнорировать) до 10 (требуется немедленная реакция). Обычно для решения этой задачи выбирают некий класс функций (скажем, функции, линейно зависящие от каких-то числовых признаков) и строят функцию из этого класса, которая минимизирует разность между предсказанными и истинными значениями. Отметим, что существует тонкое различие между этим подходом и обучением SpamAssassin, поскольку в последнем случае обучающий набор был помечен классами, а не «истинными» оценками спама. Это означает, что в распоряжении SpamAssassin меньше исходной информации, но также позволяет нам интерпретировать выданную SpamAssassin оценку как отдаленность сообщения от решающей границы и, следовательно, как меру уверенности SpamAssassin в собственном предсказании. В задаче регрессии понятие решающей границы бессмысленно, поэтому нужно найти другие способы выразить доверие к предсказаниям модели – вещественным числам.

Как классификация, так и регрессия предполагают наличие обучающего набора примеров, помеченных истинными классами или значениями функции. Разметка набора данных нередко оказывается трудоемкой и дорогостоящей работой. Можно ли научиться отличать спам от неспама или рабочую почту от личной, не размечая обучающий набор? Да, до некоторой степени. Задача группировки данных в отсутствие априорной информации о группах называется **кластеризацией**. Обучение на неразмеченных данных называется **обучением без учителя**, оно принципиально отличается от **обучения с учителем**, для которого требуются размеченные обучающие данные. Работа типичного алгоритма кластеризации основана на оценивании сходства между объектами (теми, что мы пытаемся кластеризовать, например почтовыми сообщениями) и помещении похожих экземпляров в один кластер, а непохожих – в разные.

Пример 1.1 (измерение сходства). Если почтовые сообщения описываются признаками в виде вхождения слов, как в примере классификации текста, то схожесть сообщений можно измерить в терминах общих слов. Например, взять количество общих слов в двух сообщениях и разделить его на общее количество слов в обоих сообщениях (такая мера называется *коэффициентом Жаккара*). Пусть одно сообщение содержит 42 (различных) слова, другое – 112 слов, и у этих сообщений имеется 23 общих слова. Тогда мера их сходства равна $23 / (42 + 112 - 23) = 23 / 130 = 0.18$. Далее мы можем разбить сообщения на группы так, чтобы среднее сходство сообщения со всеми остальными сообщениями в его группе было намного больше, чем среднее сходство с сообщениями в любой другой группе. И хотя не стоит ожидать, что таким образом удастся получить два четко разделенных кластера, содержащих спам и неспам, – чудеса не бывает – кластеры все же могут вскрыть интересные и полезные особенности строения данных. Возможно, удалось бы выявить какой-то особый вид спама, если бы образовалась подгруппа, в которой состав слов или язык отличается от всех остальных сообщений.

Существует еще много закономерностей, которые можно обнаружить в данных в результате обучения без учителя. *Ассоциативные правила* – вид закономерностей, популярных в маркетинговых приложениях, а результатом можно полюбоваться на сайтах многих интернет-магазинов. Например, когда я искал на сайте www.amazon.co.uk книгу John Shawe-Taylor, Nello Cristianini «Kernel Methods for Pattern Analysis», то в разделе «Люди, купившие эту книгу, покупали также» мне предложили также следующие издания:

- ☞ Nello Cristianini, John Shawe-Taylor «An Introduction to Support Vector Machines and Other Kernel-based Learning Methods»;
- ☞ Christopher Bishop «Pattern Recognition and Machine Learning»;
- ☞ Trevor Hastie, Robert Tibshirani, Jerome Friedman «The Elements of Statistical Learning: Data Mining, Inference and Prediction»;
- ☞ Richard Duda, Peter Hart, David Stork «Pattern Classification»

и еще 34 книги. Такие ассоциации обнаруживаются алгоритмами добычи данных, которые уделяют внимание объектам, часто встречающимся вместе. Обычно в подобных алгоритмах рассматриваются только объекты, встречающиеся не менее некоторого минимального числа раз (поскольку никому не нужны рекомендации, основанные на приобретении всех 39 книг одним покупателем!). Более интересные ассоциации можно найти, анализируя элементы в вашем заказе. Существует много других видов ассоциаций, которые можно выявить и исследовать, например корреляция между вещественными переменными.

В поисках структуры

Как и все прочие модели машинного обучения, закономерности – это проявление особенностей структуры данных. Иногда структура предстает в виде единственной *скрытой*, или *латентной*, *переменной*, наподобие ненаблюдаемых, но тем не менее объяснимых физических величин, скажем, энергии. Рассмотрим следующую матрицу:

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \end{pmatrix}.$$

Предположим, что каждая строка содержит оценки по четырехбалльной шкале (от 0 до 3), поставленные шестью разными людьми четырем разным фильмам – скажем, «Побег из Шоушенка», «Подозрительные лица», «Крестный отец» и «Большой Лебовски» (представлены столбцами слева направо). Похоже, что «Крестный отец» – самый популярный фильм из этой четверки со средней оценкой 1.5, а «Побег из Шоушенка» – самый непопулярный со средней оценкой 0.5. Вы видите какую-нибудь структуру в этой матрице?

Если вы готовы ответить «нет», то попробуйте поискать столбцы или строки, являющиеся комбинациями других столбцов или строк. Например, оказывается, что третий столбец – сумма первого и второго. Аналогично четвертая строка – сумма первой и второй. Это означает, что четвертый человек суммировал оценки первого и второго. Точно так же оценки «Крестного отца» – это сумма оценок первых двух фильмов. Это станет наглядно видно, если записать матрицу в виде следующего произведения:

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Возможно, вам кажется, что я только сделал хуже – вместо одной матрицы у нас теперь целых три! Заметим, однако, что первая и третья матрицы в правой части содержат только нули и единицы, а средняя – диагональная (все элементы вне главной диагонали равны нулю). И что важнее, у этих матриц есть очень естественная интерпретация в терминах *жанров* фильмов. Самая правая матрица ассоциирует фильмы (по столбцам) с жанрами (по строкам): «Побег из Шоушенка» и «Подозрительные лица» относятся к двум разным жанрам, назовем их драма и криминальный фильм, «Крестный отец» – к тому и другому, а «Большой Лебовски» – это одновременно криминальный фильм и представитель еще одного жанра (комедии). Высокая матрица размерности 6×3 описывает предпочтения людей в терминах жанров: первый, четвертый и пятый рецензенты любят драмы, второй, четвертый и пятый – криминальные фильмы, а третий, пятый

и шестой – комедии. И наконец, средняя матрица говорит, что криминальный жанр в два раза важнее остальных двух, если оценивать его с точки зрения предпочтений зрителей.

Методы обнаружения таких скрытых переменных, как жанры фильмов, демонстрируют свои истинные возможности, когда количество значений скрытой переменной (здесь количество жанров) намного меньше количества строк и столбцов в исходной матрице. Например, на момент написания книги на сайте www.imdb.com было представлено примерно 630 000 оцененных фильмов, за которые проголосовало 4 миллиона посетителей, но категорий фильмов было всего 27 (включая и вышеупомянутые). Хотя было бы наивно полагать, что оценка фильма определяется исключительно жанром, – границы жанров часто размыты, к тому же кому-то, возможно, нравятся только комедии, снятые братьями Коэн, – такой вид *разложения матрицы* часто вскрывает полезную скрытую структуру. Мы еще вернемся к этой теме в главе 10.

Настал подходящий момент, чтобы ввести терминологию, которой мы будем пользоваться в этой книге. Мы уже видели различие между обучением с учителем по размеченным данным и обучением без учителя по неразмеченным данным. Аналогично можно провести различие в зависимости от того, подразумевает модель вычисление некоей целевой переменной или нет: если да, модель будет называться *прогностической*, иначе *дескриптивной*. Таким образом, мы получаем четыре разновидности машинного обучения, перечисленные в табл. 1.1.

- ☞ Чаще всего встречаются прогностические модели, обученные с учителем, – на самом деле именно это обычно и имеется в виду, когда говорят об обучении с учителем. Типичные задачи – классификация и регрессия.
- ☞ Размеченный обучающий набор можно использовать также для построения дескриптивной модели, цель которой – не спрогнозировать значение целевой переменной, а, скажем, выявить подмножества данных, которые обладают специфическим поведением относительно целевой переменной. Этот конкретный пример обучения дескриптивной модели с учителем называется *выявлением подгрупп*; мы займемся им в разделе 6.3.
- ☞ Дескриптивные модели можно естественно обучить без учителя, и такие примеры нам уже встречались (кластеризация, выявление ассоциативных правил и разложение матрицы). Часто именно это имеют в виду, говоря об обучении без учителя.
- ☞ Типичный пример обучения прогностической модели без учителя – кластеризация данных с намерением использовать кластеры для присвоения меток классов новым данным. Мы будем называть это *прогностической кластеризацией*, чтобы отличить от *дескриптивной* формы кластеризации.

Существует еще и пятая разновидность – обучение прогностических моделей с *частичным привлечением учителя*, хотя в этой книге мы ее касаться не будем. Во многих предметных областях получить просто данные легко, но вот получить размеченные данные дорого. Например, для классификации веб-страниц в вашем распоряжении вся Всемирная паутина, но чтобы построить размеченный

	Прогностическая модель	Дескриптивная модель
Обучение с учителем	Классификация, регрессия	Выявление подгрупп
Обучение без учителя	Прогностическая кластеризация	Дескриптивная кластеризация, выявление ассоциативных правил

Таблица 1.1. Разновидности машинного обучения. В строках указано, помечены ли обучающие данные целевой переменной, а в столбцах – используются ли модели для предсказания целевой переменной или для описания имеющихся данных

обучающий набор, придется приложить немало усилий. Один из возможных подходов – взять небольшой размеченный набор для построения начальной модели, а затем уточнить ее с помощью неразмеченных данных; это и называется обучением с частичным привлечением учителя. Например, начальную модель можно было бы использовать для предсказания характеристик неразмеченных данных, взять самые надежные предсказания в качестве новых обучающих данных и переобучить модель на расширенном обучающем наборе.

Оценка качества решения задачи

Говоря о проблемах машинного обучения, важно понимать, что у них нет одного «правильного» решения. Этим они отличаются от многих других знакомых вам проблем информатики. Например, у задачи сортировки записей адресной книги по алфавиту есть единственный правильный результат (если только в ней не встречаются два человека с одинаковой фамилией, но тогда для устранения неопределенности можно взять еще какое-то поле, скажем, имя или возраст). Это не значит, что существует единственный способ достижения результата, – напротив, имеется множество алгоритмов сортировки: сортировка вставками, пузырьковая сортировка, быстрая сортировка и т. д. При сравнении качества работы этих алгоритмов мы принимали бы во внимание их скорость и максимальный объем обрабатываемых данных. Например, можно было бы экспериментально проверить алгоритм на реальных данных или проанализировать его методами теории вычислительной сложности. Однако мы не учитывали бы при сравнении правильность результата, потому что алгоритм, который иногда упорядочивает данные, а иногда нет, вообще бесполезен в качестве алгоритма сортировки.

Совсем иначе обстоит дело в машинном обучении (и не только в нем, см. замечание 1.1). Можно с уверенностью предположить, что идеального фильтра почтового спама не существует – иначе спамеры немедленно подвергли бы его обратному конструированию и нашли способы обмануть фильтр, заставив его воспринимать спам как желательную почту. Во многих случаях данные «зашумлены», например неправильно помечены или содержат ошибочные признаки, – и тогда было бы даже вредно чересчур усердно стараться найти модель, которая корректно классифицирует обучающие данные, поскольку она оказалась бы переобученной и не обобщалась бы на новые данные. Бывает, что признаки, использованные для описания данных, дают лишь предположительное указание

о возможном классе, но не содержат достаточно сильного «сигнала», который позволил бы с уверенностью предсказать класс. По этим и другим причинам специалисты по машинному обучению очень серьезно подходят к оценке качества работы алгоритмов обучения, и мы также уделим этому вопросу много внимания. Нам нужно составить хоть какое-то представление о том, как алгоритм будет работать на новых данных, не с точки зрения быстродействия или потребления памяти (хотя и это важно), а в терминах правильности классификации (если перед нами стоит задача классификации).

Допустим, требуется выяснить, насколько хорошо работает наш свежеобученный фильтр спама. Можно, например, подсчитать количество правильно классифицированных сообщений, спамных и хороших, и разделить его на общее число предъявленных примеров. В результате мы получим величину, называемую *верностью* (accuracy) классификатора. Однако она ничего не говорит о наличии переобученности. Лучше было бы поступить иначе: использовать только часть имеющихся данных (скажем, 90%) для обучения, а оставшуюся часть – в качестве *тестового набора*. Если имеет место переобучение, то на тестовом наборе алгоритм покажет гораздо худшие результаты, чем на обучающем. Но даже если выбирать данные для тестового набора случайно, нельзя исключить, что нам повезет, – большинство элементов тестового набора похожи на элементы обучающего набора – или, наоборот, не повезет – все элементы тестового набора окажутся нетипичными или зашумленными. Поэтому на практике разбиение данных на обучающие и тестовые повторяется, этот процесс, называемый *перекрестной проверкой*, мы рассмотрим подробнее в главе 12. Идея такова: мы случайно разбиваем данные на десять групп одинакового размера, из них девять групп используются для обучения, а одна – для тестирования. Обучение повторяется десять раз, и каждый раз для тестирования берется новая группа. В конце мы вычисляем среднее значение качества работы на тестовом наборе (и обычно также его стандартное отклонение, поскольку оно позволяет сказать, значимы ли небольшие расхождения в вычисленном среднем значении показателя качества для различных алгоритмов обучения). Перекрестную проверку можно применять и к другим проблемам обучения с учителем, но качество методов обучения без учителя обычно оценивается по-другому.

В главах 2 и 3 мы гораздо подробнее рассмотрим различные задачи, которые поддаются решению с помощью методов машинного обучения. В каждом случае мы поставим задачу и изучим различные варианты. Мы будем обращать особое внимание на оценку качества моделей, обученных для решения задач, поскольку это позволит глубже понять природу самих задач.

Задолго до появления машинного обучения философы понимали, что переход от частных случаев к общим правилам – это не четко поставленная задача, имеющая однозначное решение. Способ вывода путем обобщения называется *индукцией* в противоположность *дедукции* – способу рассуждений, применимому к задачам, имеющим четко определенное правильное решение.

Существует много вариантов так называемой *проблемы индукции*. Один из них был сформулирован шотландским философом XVIII века Дэвидом Юмом, который утверждал, что единственное обоснование индукции само по себе носит индуктивный характер: из того, что методика работает для некоторых индуктивных проблем, делается вывод, что она будет работать и для всех таких проблем. Здесь утверждается не просто, что индукцию невозможно обосновать дедуктивно, а что ее обоснование ведет к порочному кругу, – и это гораздо хуже.

Близкой проблеме посвящена *теорема о бесплатных завтраках*, которая утверждает, что ни один алгоритм не может быть лучше всех остальных на любой возможной задаче классификации и что средняя эффективность любого алгоритма обучения на множестве всех возможных проблем обучения не лучше случайной догадки. Рассмотрим, к примеру, вопрос типа «угадай следующее число», популярный в психологических тестах: как продолжить последовательность 1, 2, 4, 8, ...? Если все числовые последовательности равновероятны, то нет никакой надежды, что мы сможем как-то улучшить – в среднем – случайную догадку (лично я на такие вопросы всегда отвечаю 42). Разумеется, некоторые последовательности выглядят намного вероятнее прочих, по крайней мере в психологических тестах. Точно так же и распределение проблем обучения в реальном мире весьма далеко от равномерного. И чтобы уйти от проклятия теоремы о бесплатных завтраках, нужно выяснить как можно больше об этом распределении и воспользоваться этим знанием при выборе алгоритма обучения.

Замечание 1.1. Проблемы индукции и бесплатные завтраки

1.2 Модели: результат машинного обучения

Модели – центральная концепция машинного обучения, поскольку это именно то, что порождается в результате обучения на данных с целью решения поставленной задачи. Разнообразие моделей чрезвычайно – даже бескрайне – велико. И одна из причин этого феномена – разнообразие задач, решаемых методами машинного обучения: классификация, регрессия, кластеризация, выявление ассоциаций – и это только небольшая выборка. Примеры встречаются практически в любой отрасли науки и техники. Математики, инженеры, психологи, специалисты по информатике и многие другие ученые открывали – а иногда и переоткрывали – способы решения таких задач. Все они принесли на общий алтарь свои специфические навыки, и потому принципы, лежащие в основе моделей, существенно разнятся. Лично я считаю, что это хорошо, поскольку благодаря разнообразию машинное обучение оказывается более эффективной и интересной научной дисциплиной. Однако это отнюдь не облегчает задачу, стоящую перед автором книги по машинному обучению! К счастью, можно выделить несколько общих тем, что позволяет мне внести некоторую систематичность в обсуждение моделей машинного обучения. Я буду рассматривать три группы моделей: геометрические, вероятностные и логические. Они не являются взаимно исключающими, иногда встречаются модели, допускающие как геометрическую, так и вероятностную интерпретацию. Но в качестве отправной точки для наших изысканий такой классификации вполне достаточно.

Геометрические модели

Пространством объектов называется множество всех возможных или описываемых объектов вне зависимости от того, присутствуют они в имеющемся наборе данных или нет. Обычно это множество имеет некую геометрическую структуру. Например, если все признаки числовые, то каждый признак можно рассматривать как точку в декартовой системе координат. *Геометрическая модель* строится в пространстве экземпляров с применением таких геометрических понятий, как прямые, плоскости и расстояния. Например, линейный классификатор, изображенный на рис. 1, является геометрическим классификатором. Главное достоинство геометрических классификаторов заключается в простоте их визуализации, по крайней мере в двухмерном и трехмерном пространстве. Однако важно понимать, что размерность декартова пространства объектов равна количеству признаков, а их могут быть десятки, сотни, тысячи и даже больше. Пространства столь высокой размерности наглядно представить сложно, но в машинном обучении они встречаются сплошь и рядом. В названиях геометрических понятий, потенциально применимых к многомерным пространствам, обычно присутствует префикс «гипер», например решающая граница называется *гиперплоскостью*.

Если существует линейная решающая граница, разделяющая два класса, то говорят, что данные *линейно разделимы*. Как мы видели, линейная решающая граница определяется уравнением $\mathbf{w} \cdot \mathbf{x} = t$, где \mathbf{w} – вектор, перпендикулярный решающей границе, а t – порог принятия решения. Вектор \mathbf{w} удобно представлять себе как вектор, соединяющий «центр масс» отрицательных объектов \mathbf{n} с центром масс положительных объектов \mathbf{p} . Иными словами, \mathbf{w} пропорционален (или равен) $\mathbf{p} - \mathbf{n}$. Вычислить положения центров масс можно с помощью усреднения. Например, если P – множество n положительных примеров, то можно определить $\mathbf{p} = (1/n) \sum_{x \in P} \mathbf{x}$, и аналогично для \mathbf{n} . Задав подходящим образом порог принятия решения, мы сможем сделать так, чтобы точка пересечения с отрезком, соединяющим \mathbf{n} с \mathbf{p} , находилась точно посередине (рис. 1.1). В этом случае мы будем говорить о *базовом линейном классификаторе*¹. Его достоинством является простота, поскольку он определяется только в терминах сложения, вычитания и умножения векторов-примеров на число (иначе говоря, \mathbf{w} – *линейная комбинация* примеров). На самом деле, как мы увидим ниже, при некоторых дополнительных предположениях о данных это лучшее, на что можно надеяться. Но если эти предположения не выполняются, то базовый линейный классификатор ведет себя плохо – например, он может неидеально отделять отрицательные объекты от положительных, даже если данные линейно разделимы.

Поскольку данные обычно зашумлены, линейная разделимость на практике встречается редко, если только данные не являются сильно разреженными, как в случае классификации текстов. Напомним, что мы использовали большой словарь, порядка 10 000 слов, и каждому слову соответствует булев признак, описы-

¹ Это упрощенный вариант линейных дискриминантов.

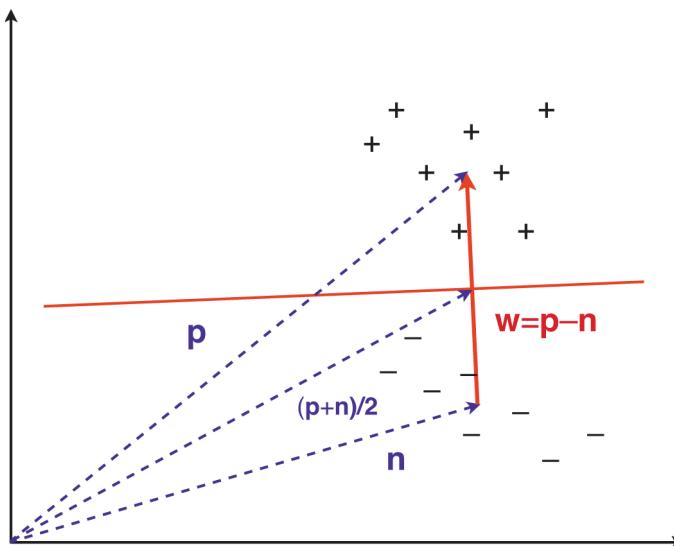


Рис. 1.1. Базовый линейный классификатор строит решающую границу путем проведения прямой через середину отрезка, соединяющего центры масс положительных и отрицательных примеров. Он описывается уравнением $w \cdot x = t$, где $w = p - n$; порог принятия решения можно найти, заметив, что вектор $(p + n)/2$ находится на решающей границе, поэтому $t = (p - n) \cdot (p + n)/2 = (\|p\|^2 - \|n\|^2)/2$, где $\|x\|$ — длина вектора x

вающий, встречается это слово в документе или нет. Это означает, что размерность пространства объектов равна 10 000, но для любого документа лишь очень небольшая доля признаков отлична от нуля. Поэтому между объектами много «пустого места», что повышает шансы на линейную разделимость. Но поскольку решающая граница в случае линейно разделимых данных определена неоднозначно, то возникает следующий вопрос: какую из бесконечного множества решающих границ выбрать? Естественно было бы предпочесть классификаторы с большим зазором, где под зазором линейного классификатора понимается расстояние между решающей границей и ближайшим к ней объектом. *Метод опорных векторов*, рассматриваемый в главе 7, — пример алгоритма линейной классификации, который ищет решающую границу с максимальным зазором (рис. 1.2).

Геометрические идеи, в частности линейные преобразования, могут оказаться весьма полезны для понимания сходства и различия методов машинного обучения (замечание 1.2). Например, мы ожидаем, что большинство алгоритмов обучения (если не все) инвариантно относительно параллельного переноса, то есть не зависит от того, где поместить начало координат. Некоторые алгоритмы инвариантны также относительно вращения, например линейные классификаторы и метод опорных векторов. Но далеко не все — скажем, байесовский классификатор к таковым не относится. Существуют также алгоритмы, чувствительные к непропорциональному масштабированию.

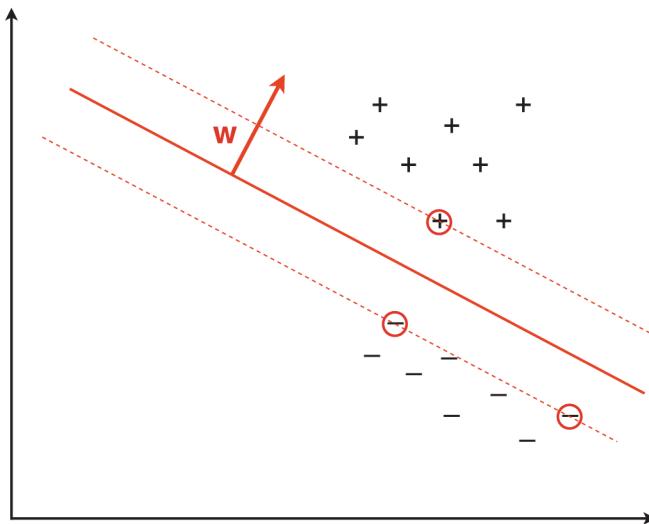


Рис. 1.2. Решающая граница, найденная методом опорных векторов для линейно разделимых данных, показанных на рис. 1.1. При таком выборе решающей границы максимизируется зазор, обозначенный пунктирными линиями. Точки, обведенные кружочками, – опорные векторы

Очень полезно в машинном обучении геометрическое понятие *расстояния*. Если расстояние между двумя объектами мало, то эти объекты похожи с точки зрения значений их признаков, и можно ожидать, что близким объектам будет сопоставлен один и тот же класс или что они попадут в один и тот же кластер. В декартовой системе координат для измерения расстояния можно взять *евклидову метрику* – квадратный корень из суммы квадратов расстояний вдоль каждой координаты¹: $\sqrt{\sum_{i=1}^d (x_i - y_i)^2}$. Очень простой метрический классификатор работает следующим образом: чтобы классифицировать новый экземпляр, мы находим самый похожий на него пример из обучающего набора (то есть объект, для которого евклидово расстояние до классифицируемого объекта минимально) и назначаем классифицируемому объекту тот же класс, что у этого примера. Это *классификатор по ближайшему соседу*. На эту тему существуют бесчисленные вариации: можно взять k самых похожих примеров и организовать голосование (метод k ближайших соседей); можно присвоить голосу каждого соседа вес, обратно пропорциональный расстоянию до него; можно применить ту же идею к задачам регрессии, усреднив значения функции для обучающих примеров, и т. д. У всех этих методов есть общая черта – их предсказания локальны в том

¹ В векторных обозначениях это можно записать в виде $\|\mathbf{x} - \mathbf{y}\| = \sqrt{(\mathbf{x} - \mathbf{y}) \cdot (\mathbf{x} - \mathbf{y})} = \sqrt{\mathbf{x} \cdot \mathbf{x} - 2\mathbf{x} \cdot \mathbf{y} + \mathbf{y} \cdot \mathbf{y}} = \sqrt{\|\mathbf{x}\|^2 - 2\|\mathbf{x}\|\|\mathbf{y}\|\cos\theta + \|\mathbf{y}\|^2}$, где θ – угол между \mathbf{x} и \mathbf{y} .

смысле, что основаны на нескольких обучающих примерах, а не на глобальной модели, построенной по всему набору данных.

Преобразования в d -мерной декартовой системе координат удобно представлять в матричной форме. Пусть \mathbf{x} – d -мерный вектор, представляющий точку, и \mathbf{t} – еще один d -мерный вектор, тогда $\mathbf{x} + \mathbf{t}$ – точка, полученная *параллельным переносом* на вектор \mathbf{t} . Параллельный перенос множества точек на вектор \mathbf{t} – то же самое, что параллельный перенос начала координат на вектор $-\mathbf{t}$. В *однородных координатах* (получаются путем добавления дополнительной координаты, равной 1) параллельный перенос можно записать в виде умножения на матрицу. Например, в двухмерном пространстве имеем:

$$\mathbf{x}^\circ = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}; \quad \mathbf{T} = \begin{pmatrix} 1 & 0 & 0 \\ t_1 & 1 & 0 \\ t_2 & 0 & 1 \end{pmatrix}; \quad \mathbf{T}\mathbf{x}^\circ = \begin{pmatrix} 1 \\ x_1 + t_1 \\ x_2 + t_2 \end{pmatrix}.$$

Вращение, или *поворот*, определяется матрицей \mathbf{D} размерности $d \times d$, для которой транспонированная матрица совпадает с обратной (то есть матрица ортогональна), а определитель равен 1. В двухмерном пространстве матрицу поворота вокруг начала координат на угол θ по часовой стрелке можно записать в виде $\mathbf{R} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$. Например, поворот на 90° по часовой стрелке описывается матрицей $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$.

Масштабирование определяется диагональной матрицей; в двухмерном пространстве $\mathbf{S} = \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix}$.

В случае пропорционального масштабирования ко всем измерениям применяется один и тот же масштабный коэффициент s , так что матрицу можно записать в виде $s\mathbf{I}$, где \mathbf{I} – единичная матрица. Отметим, что пропорциональное масштабирование с коэффициентом -1 совпадает с поворотом (на 180° в двухмерном случае).

Преобразования обычно используются следующим образом. Если дана матрица \mathbf{X} размерности $n \times d$, представляющая n точек в d -мерном пространстве, то сначала вычисляется центр масс, или средний вектор μ , путем нахождения среднего арифметического значений в каждом столбце. Затем этот центр делается началом координат путем прибавления $-\mu$ к каждой строке, что соответствует параллельному переносу. Далее мы вращаем данные, так чтобы рассеяние (мера «разброса» данных в некотором направлении) по возможности было сосредоточено вдоль осей координат; это можно сделать с помощью преобразования матрицы, которое называется *методом главных компонент* (см. главу 10). И наконец, мы масштабируем данные, так чтобы расстояние вдоль каждой оси было равно единице.

Замечание 1.2. Линейные преобразования

Имеется интересная связь между евклидовым расстоянием и средней точкой, или центроидом множества точек: не существует точки, для которой сумма квадратов расстояний до каждой точки множества была бы меньше, чем та же сумма для центроида множества (доказательство см. в теореме 8.1). Следовательно, центроид множества точек можно считать *эталоном* этого множества. Предположим, что требуется распределить данные по K кластерам, и у нас есть предва-

рительное предположение о том, как это распределение должно выглядеть. Тогда мы вычисляем центроиды начальных кластеров и переносим каждую точку в тот кластер, к центроиду которого она ближе всего. Если наша гипотеза была неточна, то в результате некоторые кластеры изменятся. Поэтому мы повторяем описанные шаги (вычисление центроидов и перераспределение точек между кластерами) до тех пор, пока не прекратятся изменения. Этот алгоритм кластеризации, подробно обсуждаемый в главе 8, называется *методом K средних* и широко применяется для решения различных задач кластеризации. Остается еще определить, как строится начальная гипотеза. Обычно это делается случайным образом: либо множество данных случайно распределяется по K «кластерам», либо случайно выбираются K «центроидов кластеров». И хотя начальные «кластеры» или «центроиды кластеров», скорее всего, имеют мало общего с реальностью, это не страшно, потому что после нескольких итераций алгоритма все придет в норму.

Подведем итог. Геометрические понятия – плоскости, параллельный перенос, поворот и расстояние – оказываются весьма полезны в машинном обучении, потому что позволяют интуитивно осмыслить многие ключевые концепции. Геометрические модели, основанные на этих интуитивных представлениях, просты, эффективны и допускают многочисленные вариации. Так, вместо евклидова расстояния, чувствительного к выбросам, можно использовать другие метрики, например манхэттенское расстояние – сумму расстояний вдоль осей: $\sum_{i=1}^d |x_i - y_i|$.

Вероятностные модели

Существуют также модели вероятностного характера, как, например, рассмотренный выше байесовский классификатор. Многие из них основаны на следующей идее. Обозначим X множество известных нам переменных, например значения признаков объектов, а Y – множество интересующих нас *целевых переменных*, например классы объектов. Основной вопрос машинного обучения – как смоделировать взаимосвязь между X и Y . Статистический подход заключается в том, чтобы предположить существование некоего случайного процесса, который порождает значения целевых переменных, подчиняющиеся вполне определенному, но неизвестному нам распределению вероятностей. Мы хотим использовать имеющиеся данные, чтобы как можно больше узнать об этом распределении. Но сначала подумаем, как можно было бы воспользоваться распределением, после того как мы что-то узнаем о нем.

Поскольку X для конкретного объекта известно, а Y , возможно, нет, то нам особенно интересны условные вероятности $P(Y|X)$. Например, Y может указывать, является почтовое сообщение спамом или нет, а X – информация о том, содержит ли сообщение слова «виагра» и «лотерея». Тогда нас интересует вероятность $P(Y|\text{виагра, лотерея})$, где «виагра» и «лотерея» – булевы переменные, вместе составляющие вектор признаков X . Для любого конкретного сообщения значения этих признаков известны, поэтому можно написать $P(Y|\text{виагра} = 1, \text{лотерея} = 0)$, если сообщение содержит слово «виагра», но не содержит слова «лотерея». Эта

величина называется *апостериорной вероятностью*, потому что используется *после* наблюдения признаков X .

В табл. 1.2 приведен пример возможного распределения этих вероятностей. Из него можно сделать вывод, что если сообщение не содержит слова «виагра», то наблюдение слова «лотерея» повышает вероятность спама с 0.31 до 0.65; но если сообщение содержит слово «виагра», то одновременное наблюдение слова «лотерея» понижает вероятность спама с 0.80 до 0.40. В примере таблица мала, но очень быстро она становится неприемлемо большой (при n булевых переменных необходимо различать 2^n случаев). Поэтому обычно полное совместное распределение нам недоступно и приходится аппроксимировать его, вводя дополнительные допущения.

Виагра	Лотерея	$P(Y = \text{спам} \text{виагра, лотерея})$	$P(Y = \text{неспам} \text{виагра, лотерея})$
0	0	0.31	0.69
0	1	0.65	0.35
1	0	0.80	0.20
1	1	0.40	0.60

Таблица 1.2. Пример апостериорного распределения. «Виагра» и «лотерея» – булевые признаки; Y – переменная класса, принимающая значения «спам» и «неспам». В каждой строке наиболее вероятный класс выделен полужирным шрифтом

В предположении, что X и Y – единственные переменные, до которых нам есть дело, апостериорное распределение $P(Y|X)$ позволяет ответить на много интересных вопросов. Например, чтобы классифицировать новое сообщение, мы смотрим, встречаются ли в нем слова «виагра» и «лотерея», ищем соответствующую вероятность $P(Y = \text{спам} | \text{виагра, лотерея})$ и классифицируем сообщение как спам, если эта вероятность превышает 0.5, и как неспам в противном случае. Такой способ предсказания значения Y по значениям X и апостериорному распределению $P(Y|X)$ называется *решающим правилом*. Необязательно даже знать все значения X , как показывает следующий пример.

Пример 1.2 (отсутствующие значения). Предположим, что мы бегло просмотрели сообщение и увидели, что в нем встречается слово «лотерея», но на анализ вхождения слова «виагра» у нас не хватило времени. Таким образом, мы не знаем, на основе какой строки табл. 1.2 – второй или четвертой – делать предсказание. Это проблема, потому что если сообщение содержит слово «виагра», то мы сочли бы его спамом (вторая строка), если не содержит – то неспамом (четвертая строка).

Решение заключается в том, чтобы усреднить эти две строки, используя вероятность вхождения слова «виагра» в произвольное сообщение (не важно, спам это или нет):

$$P(Y|\text{лотерея}) = P(Y|\text{виагра} = 0, \text{лотерея})P(\text{виагра} = 0) + \\ + P(Y|\text{виагра} = 1, \text{лотерея})P(\text{виагра} = 1).$$

Предположим для определенности, что слово «виагра» встречается в одном из десяти сообщений, тогда $P(\text{виагра} = 1) = 0.10$ и $P(\text{виагра} = 0) = 0.90$. По формуле выше получаем $P(Y = \text{спам}| \text{лотерея} = 1) = 0.65 \cdot 0.90 + 0.40 \cdot 0.10 = 0.625$ и $P(Y = \text{неспам}| \text{лотерея} = 1) = 0.35 \cdot 0.90 + 0.60 \cdot 0.10 = 0.375$. Поскольку слово «виагра» встречается в сообщениях довольно редко, получившееся распределение лишь немного отклоняется от второй строки в табл. 1.2.

На самом деле статистики очень часто работают с различными условными вероятностями, описываемыми *функцией правдоподобия* $P(X|Y)$ ¹. На первый взгляд, ситуация противоречит интуиции: зачем нам знать вероятность события, которое достоверно произошло (X), при условии, о котором мы ничего не знаем (Y)? Я поясню это, поставив мысленный эксперимент: если бы кто-то захотел отправить мне спамное сообщение, то с какой вероятностью оно содержало бы в точности те слова, которые встречаются в сообщении, которое я читаю? А как изменилась бы вероятность, если бы это сообщение было неспамным? Наверное, вы ответите «в обоих случаях вероятность мала» – и будете правы: при таком богатом выборе слов вероятность любой конкретной комбинации крайне мала. Но на самом деле важна не абсолютная величина этих вероятностей, а их отношение: насколько вероятнее наблюдать данную комбинацию слов в спаме, чем в неспаме? Предположим, к примеру, что в некотором сообщении, описываемом признаками X , $P(X|Y = \text{спам}) = 3.5 \cdot 10^{-5}$ и $P(X|Y = \text{неспам}) = 7.4 \cdot 10^{-6}$, тогда X будет наблюдаться в спамных сообщениях с вероятностью, почти в пять раз большей, чем в хороших. Отсюда вытекает следующее решающее правило: классифицировать сообщение как спам, если коэффициент правдоподобия больше 1, и как неспам – в противном случае.

Так что же использовать: апостериорные вероятности или функцию правдоподобия? Как выясняется, одно можно преобразовать в другое с помощью *правила Байеса*:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}.$$

Здесь $P(Y)$ – *априорная вероятность*, в случае классификации она говорит, насколько вероятен каждый класс априори, то есть до наблюдения данных X . $P(X)$ – вероятность данных, которая не зависит от Y , в большинстве случаев ее можно проигнорировать (или устраниТЬ на шаге нормировки, поскольку она равна $\sum_y P(X|Y)P(Y)$). Согласно первому решающему правилу, мы должны выбрать класс с максимальной апостериорной вероятностью, по правилу Байеса это мож-

¹ Она называется функцией правдоподобия, а не «распределением правдоподобия», потому что при фиксированном X $P(X|Y)$ – это отображение Y на множество вероятностей, но сумма значений этих вероятностей не равна 1, и потому нельзя говорить о распределении вероятности на Y .

но следующим образом переписать в терминах функции правдоподобия:

$$y_{\text{MAP}} = \arg \max_Y P(Y|X) = \operatorname{argmax}_Y \frac{P(X|Y)P(Y)}{P(X)} = \arg \max_Y P(X|Y)P(Y).$$

Обычно это решающее правило называют правилом *апостериорного максимума* (MAP). Если предположить, что априорное распределение равномерно (то есть $P(Y)$ принимает одно и то же значение для всех Y), то получается решающее правило *максимального правдоподобия* (ML):

$$y_{\text{ML}} = \arg \max_Y P(X|Y).$$

Полезно следующее эвристическое правило: *пользуйтесь правдоподобием, если желательно игнорировать априорное распределение или предположить, что оно равномерно, и апостериорными вероятностями – в противном случае.*

Если есть только два класса, то удобно работать с отношениями апостериорных вероятностей, или с коэффициентами правдоподобия. Если мы хотим узнать, в какой из двух классов данные попадают чаще, то можно вычислить *апостериорный шанс*, например:

$$\frac{P(Y = \text{спам} | X)}{P(Y = \text{неспам} | X)} = \frac{P(X | Y = \text{спам})}{P(X | Y = \text{неспам})} \frac{P(Y = \text{спам})}{P(Y = \text{неспам})}.$$

Словами: *апостериорный шанс* равен произведению *коэффициента правдоподобия* и *априорного шанса*. Если шанс больше 1, то мы делаем вывод, что класс в числителе более вероятен, если меньше 1 – то более вероятен класс в знаменателе. В очень многих случаях априорный шанс – просто постоянный множитель, который можно задать вручную, оценить на основе данных или оптимизировать для получения максимальной эффективности на тестовом наборе.

Пример 1.3 (апостериорные шансы). Используя данные из табл. 1.2 и предполагая, что априорное распределение равномерно, мы вычисляем апостериорные шансы следующим образом:

$$\begin{aligned} \frac{P(Y = \text{спам} | \text{виагра} = 0, \text{лотерея} = 0)}{P(Y = \text{неспам} | \text{виагра} = 0, \text{лотерея} = 0)} &= \frac{0.31}{0.69} = 0.45; \\ \frac{P(Y = \text{спам} | \text{виагра} = 1, \text{лотерея} = 0)}{P(Y = \text{неспам} | \text{виагра} = 1, \text{лотерея} = 0)} &= \frac{0.40}{0.60} = 0.67; \\ \frac{P(Y = \text{спам} | \text{виагра} = 0, \text{лотерея} = 1)}{P(Y = \text{неспам} | \text{виагра} = 0, \text{лотерея} = 1)} &= \frac{0.65}{0.35} = 1.9; \\ \frac{P(Y = \text{спам} | \text{виагра} = 1, \text{лотерея} = 0)}{P(Y = \text{неспам} | \text{виагра} = 1, \text{лотерея} = 0)} &= \frac{0.80}{0.20} = 4.0. \end{aligned}$$

По правилу апостериорного максимума (которое в данном случае совпадает с правилом максимального правдоподобия, потому что по предположению априорное распределение равномерно)

мы предсказываем неспам в первых двух случаях и спам – в двух последних. Поскольку с точки зрения статистики полное апостериорное распределение – все, что мы можем сказать о предметной области, эти предсказания – лучшее, на что можно рассчитывать; они *оптимальны по Байесу*.

Y	$P(\text{виагра}=1 Y)$	$P(\text{виагра}=0 Y)$	Y	$P(\text{лотерея}=1 Y)$	$P(\text{лотерея}=0 Y)$
Спам	0.40	0.60	Спам	0.21	0.79
Неспам	0.12	0.88	Неспам	0.13	0.87

Таблица 1.3. Примеры маргинального правдоподобия

Из проведенного анализа ясно, что функция правдоподобия играет важную роль в статистическом машинном обучении. Она устанавливает так называемую *порождающую модель*: вероятностную модель, из которой мы можем отбирать значения всех участвующих переменных. Представьте, к примеру, ящик с двумя кнопками: «спам» и «неспам». Нажатие кнопки «неспам» порождает случайное сообщение с вероятностью $P(X|Y = \text{неспам})$, а нажатие кнопки «спам» – случайное сообщение с вероятностью $P(X|Y = \text{спам})$. Теперь вопрос: что поместить внутрь ящика? Попробуем модель настолько простую, что это даже смешно. В предположении, что имеется словарь из 10 000 слов, возьмем два мешка по 10 000 монет в каждом – по одной монете на каждое слово. Для порождения случайного сообщения мы берем тот или иной мешок – в зависимости от того, какая кнопка нажата, – и подбрасываем каждую из 10 000 находящихся в нем монет, чтобы решить, какие слова попадут в сообщение (пусть орел означает, что слово попало, а решка – что не попало).

В терминах статистики каждая монета – неизбежательно правильная – представляет один параметр модели, так что всего у нас 20 000 параметров. Если слово «виагра» есть в словаре, то монета с надписью «виагра» в мешке «спам» представляет условную вероятность $P(\text{виагра}|Y = \text{спам})$, а монета с надписью «виагра» в мешке «неспам» – условную вероятность $P(\text{виагра}|Y = \text{неспам})$. Вместе эти две монеты соответствуют левой части табл. 1.3. Отметим, что, используя разные монеты для каждого слова, мы молчаливо предполагаем, что вероятности слов внутри одного и того же класса независимы. Если это действительно так, то мы сможем разложить функцию совместного правдоподобия в произведение *маргинальных правдоподобий*:

$$P(\text{виагра}, \text{лотерея}|Y) = P(\text{виагра}|Y)P(\text{лотерея}|Y).$$

По существу, предположение о независимости означает, что знание о том, встречается некое слово в сообщении или нет, ничего не говорит о правдоподобии других слов. Вероятности в правой части называются маргинальными правдоподобиями, потому что получены «вытеснением на обочину» некоторых переменных в совместном распределении: $P(\text{виагра}|Y) = \sum_{\text{лотерея}} P(\text{виагра}, \text{лотерея}|Y)$.

Пример 1.4 (использование маргинального правдоподобия). В предположении, что оценки взяты из табл. 1.3, мы можем вычислить коэффициенты правдоподобия (шансы, ранее вычисленные по полному апостериорному распределению, показаны в скобках):

$$\frac{P(\text{виагра} = 0 | Y = \text{спам})}{P(\text{виагра} = 0 | Y = \text{неспам})} \frac{P(\text{лотерея} = 0 | Y = \text{спам})}{P(\text{лотерея} = 0 | Y = \text{неспам})} = \frac{0.60}{0.88} \frac{0.79}{0.87} = 0.62 \quad (4.45)$$

$$\frac{P(\text{виагра} = 0 | Y = \text{спам})}{P(\text{виагра} = 0 | Y = \text{неспам})} \frac{P(\text{лотерея} = 1 | Y = \text{спам})}{P(\text{лотерея} = 1 | Y = \text{неспам})} = \frac{0.60}{0.88} \frac{0.21}{0.13} = 1.1 \quad (1.9)$$

$$\frac{P(\text{виагра} = 1 | Y = \text{спам})}{P(\text{виагра} = 1 | Y = \text{неспам})} \frac{P(\text{лотерея} = 0 | Y = \text{спам})}{P(\text{лотерея} = 0 | Y = \text{неспам})} = \frac{0.40}{0.12} \frac{0.79}{0.87} = 3.0 \quad (4.0)$$

$$\frac{P(\text{виагра} = 1 | Y = \text{спам})}{P(\text{виагра} = 1 | Y = \text{неспам})} \frac{P(\text{лотерея} = 1 | Y = \text{спам})}{P(\text{лотерея} = 1 | Y = \text{неспам})} = \frac{0.40}{0.12} \frac{0.21}{0.13} = 5.4 \quad (0.67)$$

Мы видим, что благодаря решающему правилу максимального правдоподобия наша простейшая модель дает оптимальные по Байесу предсказания в первых трех случаях, но не в четвертом (когда встречаются оба слова – «виагра» и «лотерея»), где маргинальное правдоподобие оказывается очень далеко от истины. Возможное объяснение заключается в том, что одновременное появление этих двух слов в почтовом сообщении крайне маловероятно, но вероятность встретить их в хорошем сообщении все же чуть выше, чем в спаме, – например, я мог бы отправить вот этот самый текст кому-нибудь по почте!

Предположение о независимости, позволившее нам разложить функцию совместного правдоподобия в произведение маргинальных правдоподобий, можно было бы счесть «наивным» – и именно поэтому в машинном обучении такой упрощенный классификатор называется *наивным байесовским классификатором*. Но никакого уничижительного намека здесь нет – напротив, название иллюстрирует важнейший для машинного обучения принцип: *делать настолько просто, насколько возможно, но не проще*¹. В контексте статистики этот принцип означает, что нужно использовать простейшую порождающую модель, которая решает задачу. Например, мы можем остановиться на наивном байесовском классификаторе на том основании, что случаи, когда маргинальное правдоподобие действительно вводит в заблуждение, на практике крайне маловероятны и потому вряд ли смогут снизить качество модели, обученной на реальных данных.

Итак, мы теперь понимаем, на что похожа вероятностная модель, но как такую модель обучить? Во многих случаях это сводится к оцениванию параметров мо-

¹ Это высказывание часто приписывают Эйнштейну, хотя на самом деле его автор точно неизвестен. Существуют и другие афоризмы в том же духе: «не умножать сущности без необходимости» (этот принцип называют *бритвой Оккама* по имени его автора, Уильяма из Окхема), «не следует допускать причин больше, чем достаточно для объяснения видимых природных явлений» (Исаак Ньютон), «ученый должен пользоваться простейшими средствами для достижения результатов и исключить все, что не воспринимается органами чувств» (Эрнст Мах). О том, являются ли эти принципы чем-то большим, чем методологические правила, и не указывают ли они на какие-то фундаментальные свойства природы, ведутся оживленные дебаты.

дели по данным, что обычно достигается непосредственным подсчетом. Например, в модели распознавания спама путем подбрасывания монеты у нас было по две монеты для каждого слова w_i в словаре; одну из них мы подбрасывали для порождения спамного сообщения, другую – для порождения хорошего. Предположим, что монета для спама выпадает орлом с вероятностью Θ_i^{\oplus} , а монета для неспама – с вероятностью Θ_i^{\ominus} , тогда через эти параметры можно выразить все правдоподобия:

$$\begin{aligned} P(w_i = 1 | Y = \text{спам}) &= \Theta_i^{\oplus} & P(w_i = 0 | Y = \text{спам}) &= 1 - \Theta_i^{\oplus} \\ P(w_i = 1 | Y = \text{неспам}) &= \Theta_i^{\ominus} & P(w_i = 0 | Y = \text{неспам}) &= 1 - \Theta_i^{\ominus} \end{aligned}$$

Чтобы оценить параметры Θ_i^{\pm} , нам понадобится обучающий набор сообщений, снабженных метками «спам» или «неспам». Мы возьмем спамные сообщения и подсчитаем, во скольких из них встречается слово w_i ; поделив это значение на

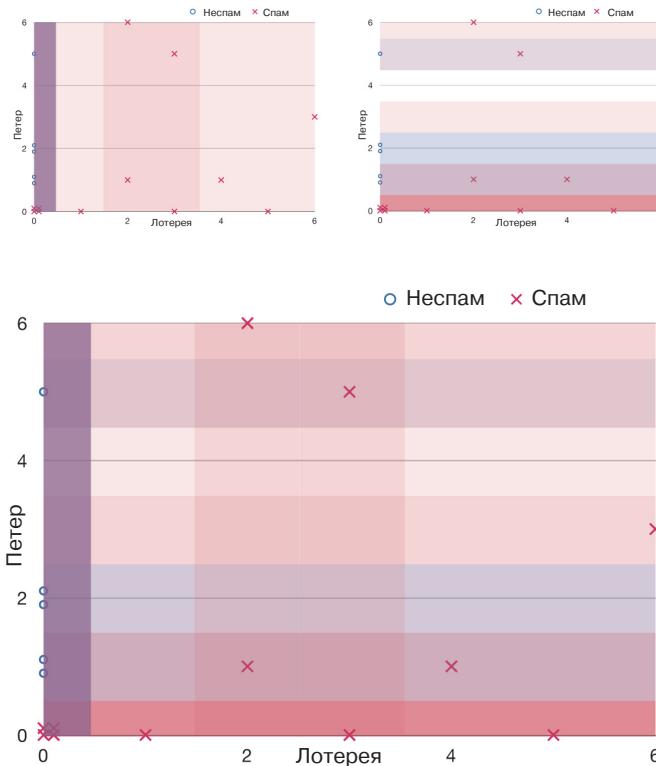


Рис. 1.3. (Сверху) Наглядное представление двух маргинальных правдоподобий, вычисленных по небольшому набору данных. Цветом обозначена классификация на основе правдоподобия: спам или неспам. **(Снизу)** Комбинирование двух маргинальных правдоподобий дает картину, похожую на шотландский плед. Цвет ячейки образован наложением цветов в строке и столце.

общее число спамных сообщений, мы получим оценку θ_i^{\oplus} . Повторение этой процедуры для хороших сообщений даст оценку θ_i^{\ominus} . Вот, собственно, и все!¹

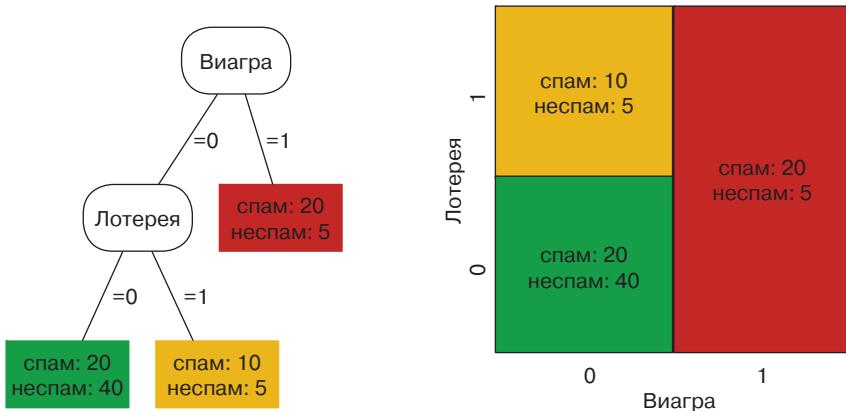


Рис. 1.4. (Слева) Дерево, состоящее из комбинаций двух булевых признаков. Каждый внутренний узел помечен признаком, а исходящие из него ребра – значениями этого признака. Поэтому каждый листовой узел соответствует уникальной комбинации значений признаков. Кроме того, в каждом листовом узле указано распределение по классам, полученное на основе обучающего набора. **(Справа)** Дерево признаков разбивает пространство объектов на прямоугольные области, соответствующие листьям. Видно, что большинство неспамных сообщений находится в левом нижнем углу

На рис. 1.3 это наглядно представлено для рассмотренного выше варианта наивного байесовского классификатора. В данном случае мы запоминаем, сколько раз каждое слово встречалось в сообщении, а не просто встречалось оно или нет. Поэтому нам нужен параметр $p_{ij\pm}$ для каждого правдоподобия $P(w_i=j|Y=\pm)$, где $j = 0, 1, 2, \dots$. Например, мы видим, что существуют два спамных сообщения, в которых слово «лотерея» встречается дважды, и одно хорошее сообщение, в котором пять раз встречается слово «Peter». Объединив оба набора маргинальных правдоподобий, мы получим картину, напоминающую клетчатый плед (рис. 1.3 снизу), почему я и называю такой наивный байесовский классификатор «шотландским». Это наглядное напоминание о том, что многомерная наивная байесовская модель *разлагается* на несколько одномерных. Мы еще не раз вернемся к вопросу о разложении на страницах этой книги.

Логические модели

Модели третьего типа по своей природе более алгоритмичны, они заимствуют идеи из информатики и технических дисциплин. Я называю их «логическими»,

¹ Иногда приходится немного корректировать счетчики слов, встречающихся слишком часто или слишком редко, мы поговорим об этом в разделе 2.3.

потому что модели такого типа легко выразить на языке правил, понятных человеку, например: if виагра = 1 then класс = Y = спам. Такие правила легко организовать в виде дерева, подобного тому, что изображено на рис. 1.4. Я буду называть его *деревом признаков*. Идея в том, что признаки используются для итеративного разбиения пространства объектов. В результате листьям дерева соответствуют прямоугольные области пространства объектов (в общем случае гиперпрямоугольники), которые мы будем называть *сегментами пространства объектов*, или просто сегментами. В зависимости от решаемой задачи листья можно пометить классами, вероятностями, вещественными значениями и т. д. Деревья признаков, листья которых помечены классами, принято называть *решающими деревьями*.

Пример 1.5 (разметка дерева признаков). Листья дерева на рис. 1.4 можно было бы пометить слева направо следующим образом: неспам – спам – спам, применяя простое решающее правило *мажоритарного класса*. Альтернативно их можно было бы пометить в соответствии с долей спамных сообщений в каждом листе: слева направо это будет 1/3, 2/3 и 4/5. Либо если мы решаем задачу регрессии, то можно было бы пометить листья предсказанными вещественными значениями или даже линейными функциями от каких-то других вещественных признаков.

Деревья признаков – весьма гибкий инструмент, и в этой книге они будут играть важную роль. Даже модели, которые на первый взгляд не имеют ничего общего с деревьями, можно интерпретировать как построенные на основе дерева признаков. Возьмем, к примеру, рассмотренный выше наивный байесовский классификатор. Поскольку в нем используются маргинальные правдоподобия типа тех, что показаны в табл. 1.3, он разбивает пространство объектов на столько областей, сколько существует комбинаций значений признаков. Это означает, что классификатор можно представлять себе как *полное* дерево признаков, содержащее все признаки, по одному на каждом уровне дерева (рис. 1.5). Отметим, кстати, что самый правый лист соответствует случаю, когда наивный байесовский классификатор дает неверное предсказание. Поскольку в этот лист попадает только один пример, существует опасность, что дерево переобучено на конкретных данных, и модель, представленная предыдущим деревом, лучше. При обучении решающих деревьев часто применяется техника *редукции*, позволяющая удалить такие внутренние узлы.

Списком признаков называется двоичное дерево признаков, в котором все ветви направлены в одну сторону – налево или направо. Дерево на рис. 1.4 – это левосторонний список признаков. Такие списки можно записать в виде вложенных предложений if–then–else, знакомых любому человеку с минимальным опытом программирования. Например, если бы мы пометили листья дерева на рис. 1.4 мажоритарными классами, то получился бы такой *решающий список*:

```
if виагра = 1 then класс = Y = спам
else if лотерей = 1 then класс = Y = спам
else класс = Y = неспам
```

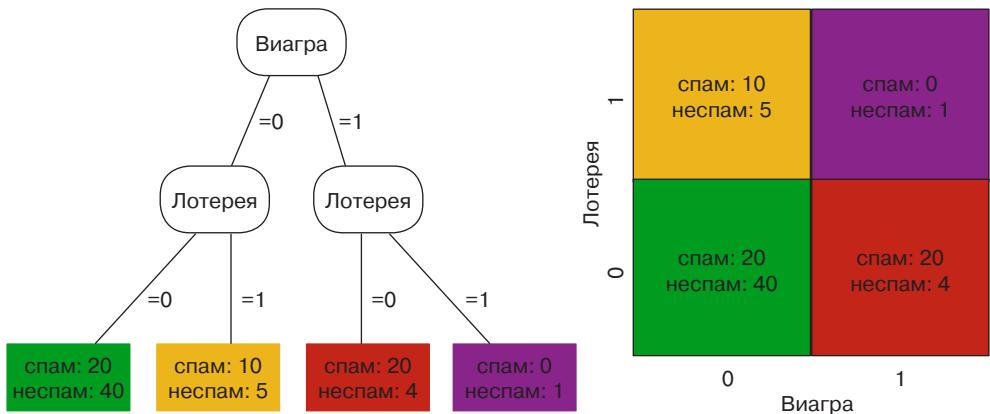


Рис. 1.5. (Слева) Полное дерево признаков, построенное по двум булевым признакам.
(Справа) Соответствующее разбиение пространства объектов - самое мелкое из всех возможных для этих двух признаков.

У логических моделей часто бывают различные, но эквивалентные формулировки. Вот, например, две альтернативные формулировки для этой модели:

```
if виагра = 1 ∨ лотерея = 1 then класс = Y = спам
else класс = Y = неспам
```

```
if виагра = 0 ∧ лотерея = 0 then класс = Y = неспам
else класс = Y = спам
```

В первой формулировке два правила из исходного решающего списка объединены операцией *дизъюнкции* («или»), обозначаемой \vee . При этом в пространстве объектов выбирается одна непрямоугольная область. Во второй формулировке условия объединены операцией *конъюнкции* («и»), обозначаемой \wedge , при выполнении условия выбирается противоположный класс (неспам), а все остальное считается спамом.

Ту же модель можно представить и правилами без ветви else:

```
if виагра = 1 then класс = Y = спам
if виагра = 1 ∧ лотерея = 1 then класс = Y = спам
if виагра = 0 ∧ лотерея = 0 then класс = Y = неспам
```

При такой записи каждый путь от корня к листу транслируется в одно правило. Это означает, что хотя в правилах из одного и того же поддерева есть общие условия (например, *виагра = 0*), в каждой паре правил обязательно имеются взаимно исключающие условия (скажем, *лотерея = 1* во втором правиле и *лотерея = 0* в третьем). Однако так бывает не всегда: правила могут и перекрываться.

Пример 1.6 (перекрывающиеся правила). Рассмотрим следующие правила:

```
if лотерея = 1 then класс = Y = спам
if Петер = 1 класс = Y = неспам
```

Как видно по рис. 1.6, эти правила перекрываются по условию $\text{лотерея} = 1 \wedge \text{Петер} = 1$ и в этом случае дают противоречивые предсказания. Кроме того, они не дают вообще никакого предсказания, когда выполняется условие $\text{лотерея} = 0 \wedge \text{Петер} = 0$.

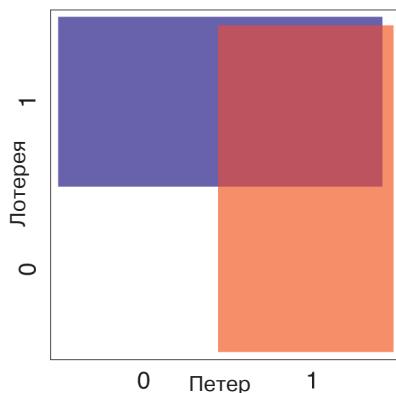


Рис. 1.6. Результат действия перекрывающихся правил в пространстве объектов. Эти два правила дают противоречивые предсказания в правом верхнем углу и вообще не дают предсказания в левом нижнем углу

Логик сказал бы, что эта система правил одновременно *противоречива* и *неполна*. Для устранения неполноты мы могли бы добавить правило по умолчанию, которое будет возвращать, скажем, мажоритарный класс, если никакое другое правило не применимо. Существует много стратегий работы с перекрывающимися правилами, мы рассмотрим их в главе 6.

Алгоритм построения решающих деревьев обычно работает сверху вниз. Первым делом нужно найти хороший признак, по которому разделяются верхние ветви дерева. Цель – сделать так, чтобы чистота узлов на следующем уровне улучшилась; узел считается тем чище, чем больше принадлежащих ему обучающих примеров попадают в один и тот же класс. После того как алгоритм найдет такой признак, обучающий набор разбивается на подмножества, по одному для каждого узла, получившегося в результате разделения. Для каждого из этих подмножеств мы снова ищем оптимальный для разделения признак и т. д. Таким образом, на каждом шаге алгоритм разбивает задачу на меньшие подзадачи – в информатике алгоритмы такого типа называют «разделяй и властвуй». Разделение узла прекращается, когда все принадлежащие ему обучающие примеры оказываются в одном классе. Большинство алгоритмов поиска решающих правил также

работает сверху вниз. Мы выводим правило, итеративно добавляя условия до тех пор, пока не окажется, что все примеры, удовлетворяющие этому правилу, принадлежат одному и тому же классу. Затем мы удаляем из обучающего набора эти примеры и повторяем процесс сначала. Иногда такой подход называют «отделяй и властуй».

У логических моделей есть интересная особенность, которая отличает их от геометрических и вероятностных моделей: они в какой-то степени *объясняют*, как получено предсказание. Так, предсказание, выданное решающим деревом, можно объяснить, перечислив условия на пути от корня к листу, выдавшему предсказание. Для человека такие модели легко обозримы, поэтому иногда их называют *декларативными*. Декларативные модели необязательно должны ограничиваться рассмотренными до сих пор простыми правилами. Система вывода логических правил *Progol* построила следующий набор условий, предсказывающих, будет ли данное молекулярное соединение канцерогенным.

1. Даёт положительный результат в исследовании с сальмонеллами.
2. Даёт положительный результат на сцепленную с полом рецессивную летальную мутацию у дрозофилы, или
3. Даёт отрицательный результат на хромосомную аберрацию, или
4. Включает углерод в шестичленном ароматическом цикле с частичным зарядом -0.13 , или
5. Включает первичную аминогруппу и не включает вторичных и третичных аминогрупп, или
6. Включает ароматический (или резонансный) водород с частичным зарядом ≥ 0.168 , или
7. Включает кислород в гидроксильной группе с частичным зарядом ≥ -0.616 и ароматический (или резонансный) водород, или
8. Включает бром, или
9. Включает тетраэдрический углерод с частичным зарядом ≤ -0.144 и даёт положительный результат на встроенные в *Progol* правила определения мутагенности¹.

В первых трех условиях речь идет о неких тестах, которые были выполнены для всех молекул и результаты которых запечатлены в данных в виде булевых признаков. Остальные же шесть правил касаются структуры молекулы и были выведены самой системой *Progol*. Например, правило 4 предсказывает, что молекула, содержащая атом углерода с определенными свойствами, будет канцерогенной. Это условие отличается от первых трех тем, что не было ранее зафиксировано в данных, а выведено в процессе обучения *Progol*, поскольку позволяет объяснить экспериментальные данные.

¹ Мутагенные молекулы вызывают мутации ДНК и часто являются канцерогенными. Это последнее правило ссылается на правило предсказания мутагенности, которому *Progol* была обучена ранее.

Группировка и ранжирование

Мы рассмотрели три типа моделей: геометрические, вероятностные и логические. Я отметил, что каких-то фундаментальных принципов, относящихся к моделям каждого типа, не существует, а деление произведено в основном ради удобства. Прежде чем переходить к третьему из основных ингредиентов машинного обучения, признакам, я хотел бы ввести еще одну важную, но несколько более абстрактную характеристику, которая в некотором смысле ортогональна делению моделей на геометрические, вероятностные и логические. Это различие между *группирующими* и *ранжирующими* моделями. Различаются они прежде всего способом работы с пространством объектов.

Группирующие модели разбивают пространство объектов на группы, или *сегменты*, число которых определяется на этапе обучения. Можно сказать, что у группирующих моделей фиксированное конечное «разрешение», которое накладывает ограничения на возможность различения объектов. На уровне своего максимального разрешения группирующая модель делает обычно что-то совсем простое, например назначает всем попавшим в один сегмент объектам мажоритарный класс. При обучении группирующей модели основной упор делается на определении правильных сегментов, чтобы эта простейшая схема пометки на уровне локального сегмента давала удовлетворительные результаты. С другой стороны, в ранжирующих моделях понятия сегмента нет. Вместо применения очень простых локальных моделей формируется одна глобальная модель над всем пространством объектов. Поэтому ранжирующие модели (обычно) способны различить произвольные объекты, как бы похожи они ни были. Теоретически они обладают бесконечным разрешением, особенно при работе в декартовом пространстве объектов.

Примером группирующих моделей являются только что рассмотренные модели на основе деревьев. Принцип их работы – постепенное разбиение пространства объектов на меньшие подмножества. Поскольку деревья обычно имеют ограниченную глубину и не содержат всех доступных признаков, подмножества, оказавшиеся на уровне листьев, образуют разбиение пространства объектов с конечной разрешающей способностью. Все объекты, попавшие в один листовый узел, трактуются одинаково, независимо от наличия не отраженных в дереве признаков, с помощью которых их можно было бы различить. Метод опорных векторов и другие геометрические классификаторы – это примеры ранжирующих моделей. Работая в декартовом пространстве объектов, они способны представить и уловить мельчайшие различия между объектами. Отсюда следует, что тестовый объект вполне может получить оценку, которая не присваивалась никакому ранее предъявленному объекту.

Различие между группирующими и ранжирующими моделями носит скорее относительный, чем абсолютный характер, в некоторых моделях свойства тех и других сочетаются. Так, хотя линейные классификаторы – типичный пример ранжирующей модели, легко привести примеры объектов, которые линейная модель распознать не в состоянии, а именно объекты, расположенные на прямой

или плоскости, параллельной решающей границе. Проблема не в том, что нет ни одного сегмента, а в том, что их бесконечно много. На другом конце спектра находятся деревья регрессии, которые, как мы впоследствии увидим, сочетают свойства группирующих и ранжирующих моделей. Таким образом, общая картина выглядит, как на рис. 1.7. А на рис. 1.8¹ показана таксономия восьми моделей, обсуждаемых в этой книге – в главах 4–9.

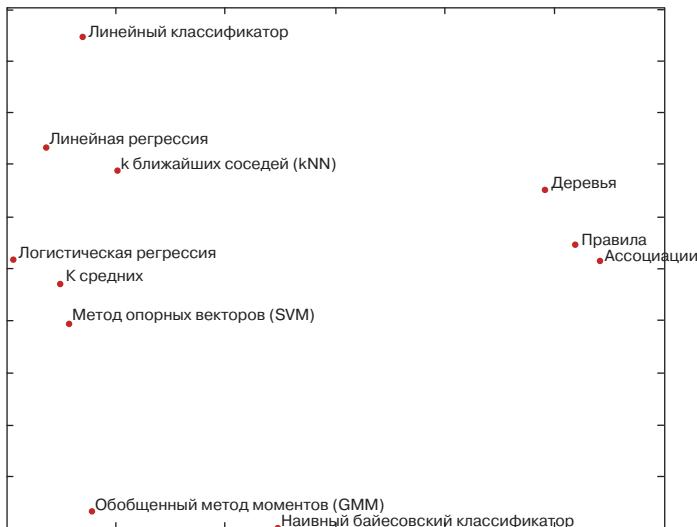


Рис. 1.7. Карта некоторых моделей, рассматриваемых в этой книге. Модели, обладающие общими характеристиками, расположены рядом: логические модели – справа, геометрические – слева вверху, вероятностные – слева внизу. В горизонтальном направлении левее находятся ранжирующие модели, а правее – группирующие

1.3 Признаки: рабочая лошадка машинного обучения

Теперь, познакомившись с примерами задач и моделей машинного обучения, обратимся к третьему, и последнему, ингредиенту. Именно признакам обязано успехом или неудачей приложение машинного обучения, потому что модель хороша лишь настолько, насколько удачно выбраны признаки. Можно считать, что признак – это своего рода измерение, которому можно подвергнуть любой объект. С точки зрения математики, признаки представляют собой функции из пространства объектов во множество, которое называется *областью значений* признака. Поскольку результаты измерения обычно числовые, чаще всего в роли

¹ Эти рисунки были созданы на основе данных из примера 1.7 ниже.

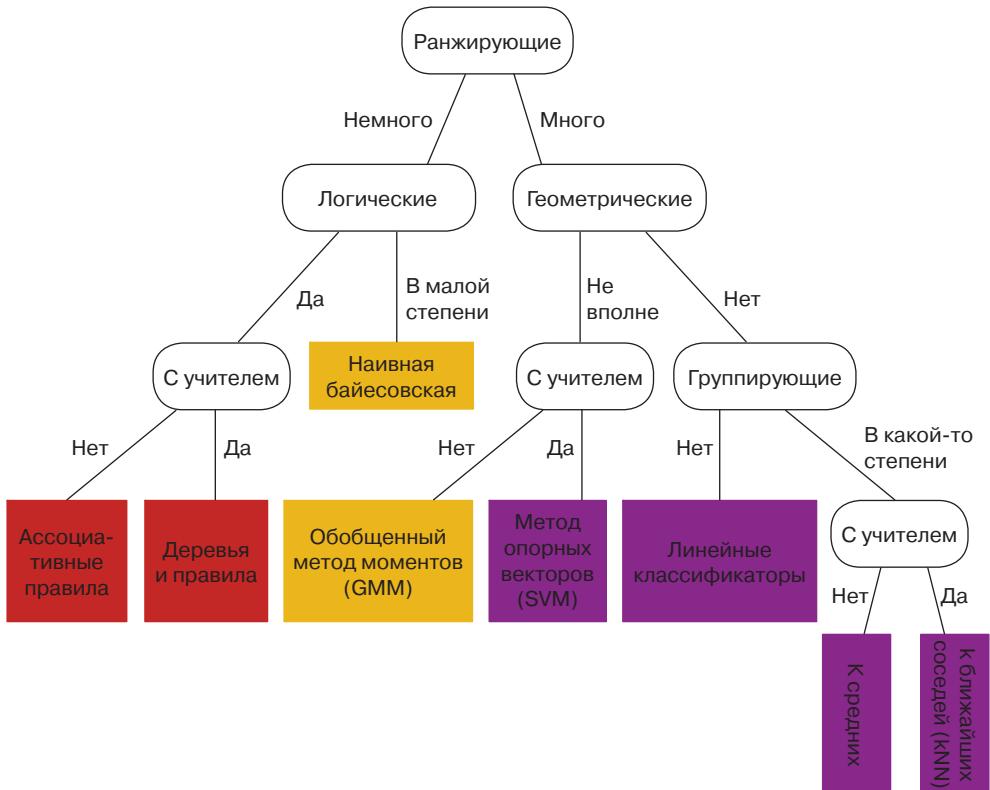


Рис. 1.8. Таксономия методов машинного обучения: группирующие и ранжирующие модели, логические, геометрические и комбинированные, с учителем и без. Цвет показывает тип модели: логическая (красный), вероятностная (оранжевый) или геометрическая (фиолетовый)

области значений признака выступает некоторое множество вещественных чисел. Впрочем, встречаются и другие области значений: множества целых чисел (например, когда признаком выступает счетчик чего-либо, скажем, вхождений некоторого слова); булевые величины (если признаком является утверждение, которое для каждого объекта может быть истинным или ложным, например «это почтовое сообщение адресовано Петеру Флаху»); произвольные конечные множества, например множество цветов или геометрических фигур.

Пример 1.7 (набор данных ММО). Предположим, что имеется ряд моделей обучения, которые мы хотим описать в терминах некоторых свойств:

- ☞ к какому типу больше тяготеет модель: геометрическому, вероятностному или логическому;
- ☞ является модель группирующей или ранжирующей;

Модель	<i>Геом</i>	<i>Стат</i>	<i>Лог</i>	<i>Групп</i>	<i>Ранж</i>	<i>Дискр</i>	<i>Вещ</i>	<i>С учит</i>	<i>Без учит</i>	<i>Многокласс</i>
Деревья	1	0	3	3	0	3	2	3	2	3
Правила	0	0	3	3	1	3	2	3	0	2
Наивная байесовская	1	3	1	3	1	3	1	3	0	3
kNN	3	1	0	2	2	1	3	3	0	3
Линейный классификатор	3	0	0	0	3	1	3	3	0	0
Линейная регрессия	3	1	0	0	3	0	3	3	0	1
Логистическая регрессия	3	2	0	0	3	1	3	3	0	0
Метод опорных векторов	2	2	0	0	3	2	3	3	0	0
К средних	3	2	0	1	2	1	3	0	3	1
Обобщенный метод моментов (GMM)	1	3	0	0	3	1	3	0	3	1
Ассоциации	0	0	3	3	0	3	1	0	3	1

Таблица 1.4. Набор данных ММО, описывающий свойства моделей машинного обучения. На основе этих данных были созданы рис. 1.7 и 1.8

- ☞ насколько хорошо модель приспособлена к обработке дискретных и (или) вещественных признаков;
- ☞ модель обучается с учителем или без;
- ☞ в какой мере модель может использоваться для решения многоклассовых задач.

Первые два свойства можно описать с помощью дискретных признаков с тремя или двумя значениями соответственно, или если классификация не столь однозначна, то можно оценивать по какой-то числовой шкале. Проще всего измерять величину свойства по целочисленной шкале от 0 до 3, как показано в табл. 1.4. Эта таблица определяет набор данных, в котором каждая строка представляет объект, а каждый столбец – свойство. Например, согласно этой (очень упрощенной) классификации, некоторые модели являются чисто группирующими (деревья, ассоциации) или чисто ранжирующими (линейные модели, логистическая регрессия и GMM), тогда как другие – комбинированными. Мы видим также, что деревья и правила по большинству признаков очень похожи, тогда как GMM и ассоциации почти во всем различаются.

Этот небольшой набор данных будет далее использоваться в нескольких примерах. На самом деле изображенная на рис. 1.8 таксономия была получена ручной модификацией решающего дерева, построенного в результате обучения на этом наборе данных с использованием моделей в качестве классов. А график на рис. 1.7 был построен путем применения метода снижения размерности, который по возможности сохраняет попарные расстояния.

Два способа использования признаков

Стоит отметить, что признаки и модели тесно связаны – и не только потому, что модели определяются в терминах признаков, но и потому, что одиничный

признак можно преобразовать в так называемую *одномерную модель*. Поэтому можно выделить два способа использования признаков, отражающих различие между группирующими и ранжирующими моделями. Очень часто, особенно в логических моделях, признаки используются для более пристального изучения какой-то области пространства объектов. Пусть признак f – количество вхождений слова «виагра» в почтовое сообщение, и пусть x – произвольное сообщение, тогда условие $f(x) = 0$ отбирает сообщения, которые не содержат слова «виагра», условие $f(x) \neq 0$, или $f(x) > 0$ – сообщения, которые содержат это слово, условие $f(x) \geq 2$ – сообщения, которые содержат его не менее двух раз, и т. д. Такие условия называются *двоичным*, или *бинарным, разделением*, потому что разбивают пространство объектов на две группы: удовлетворяющие и не удовлетворяющие условию. Возможно также недвоичное разделение: например, если g – признак, принимающий значение «твит» для сообщений, содержащих не более 20 слов, «короткое» – для сообщений, содержащих от 21 до 20 слов, «среднее» – для сообщений, содержащих от 51 до 200 слов, и «длинное» – для сообщений, содержащих более 200 слов, то выражение $g(x)$ определяет разделение пространства на четыре части. Как мы уже видели, такие разделения можно представить в виде дерева признаков, на основе которого построить модель.

Второй способ использования признаков встречается при обучении с учителем. Напомним, что в работе линейного классификатора используется решающее правило вида $\sum_{i=1}^n w_i x_i > t$, где x_i – числовые признаки¹. Линейность этого правила означает, что каждый признак дает независимый вклад в оценку объекта. Этот вклад зависит от веса w_i : если он большой и положительный, то положительное значение x_i увеличивает оценку; если $w_i < 0$, то положительное значение x_i уменьшает оценку, а если $w_i \approx 0$, то влиянием x_i можно пренебречь. Таким образом, признак вносит точный и поддающийся измерению вклад в окончательное предсказание. Отметим также, что при вычислении оценки объекта принимаются во внимание не «пороговые значения» отдельных признаков, а их совокупная «разрешающая способность». Эти две ипостаси признаков – «признаки как способ разделения» и «признаки как прогностические факторы» – часто используются совместно в рамках одной модели.

Пример 1.8 (два способа использования признаков). Пусть требуется аппроксимировать функцию $y = \cos \pi x$ на отрезке $-1 \leq x \leq 1$. Линейная аппроксимация тут не подойдет, поскольку наилучшее соответствие дает прямая $y = 0$. Но если разбить ось x на два отрезка: $-1 \leq x < 0$ и $0 \leq x \leq 1$, – то на каждом можно будет найти приемлемую линейную аппроксимацию. С этой целью мы используем x одновременно как разделяющий признак и как переменную регрессии (рис. 1.9).

¹ Обратите внимание на две разные системы обозначения признаков: иногда мы пишем $f(x)$, если удобнее рассматривать признак как функцию от объекта x , а иногда x_i – если удобнее рассматривать объект как вектор значений признаков.

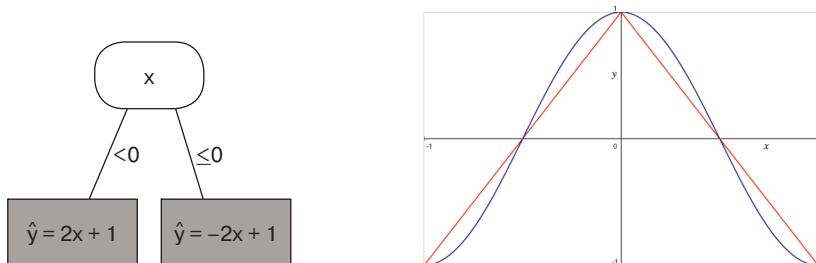


Рис. 1.9. (Слева) Дерево регрессии, в котором дерево признаков с одним разделением сочетается с моделями линейной регрессии в листьях. Обратите внимание, что x используется одновременно как разделяющий признак и как переменная регрессии. **(Справа)** Функция $y = \cos \pi x$ на отрезке $-1 \leq x \leq 1$ и ее кусочно-линейная аппроксимация с помощью показанного дерева регрессии

Отбор и преобразование признаков

В машинном обучении манипуляциям с признаками уделяется немало внимания. В примере фильтра спама и – более общо – классификации текстов входные сообщения, или документы, не содержат встроенных признаков; их должен подобрать разработчик приложения машинного обучения. Этот процесс **отбора признаков** – важнейшее условие успеха приложения. Индексирование почтового сообщения по встречающимся в нем словам (оно называется представлением в виде **мешка слов**, потому что порядок слов игнорируется) – это тщательно продуманное представление, которое позволяет усилить «сигнал» и ослабить «шум» в задаче фильтрации почтового спама и сходных задачах классификации. Однако нетрудно указать проблему, в которой поступать так было бы совершенство противопоказано: например, если наша цель – обучить классификатор различать правильно и неправильно построенные предложения, то порядок слов, очевидно, является сигналом, а не шумом, поэтому необходимо иное представление.

Часто бывает естественно строить модель в терминах заданных признаков. Однако никто не мешает нам как угодно изменять признаки и даже вводить новые. Например, вещественные признаки нередко содержат слишком много деталей, которые можно устраниТЬ с помощью **дискретизации**. Допустим, мы хотим проанализировать вес тела для сравнительно небольшой группы людей, скажем из 100 человек, и для этого рисуем гистограмму. Если измерять вес в килограммах с одним знаком после запятой (то есть с точностью до 100 граммов), то гистограмма окажется разреженной и игольчатой. На основании такой гистограммы трудно сделать какие-либо общие выводы. Гораздо полезнее было бы дискретизировать результаты измерения, поместив их в интервалы шириной 10 килограммов. Если стоит задача классификации, например мы пытаемся соотнести вес тела с заболеваемостью диабетом, то можно было бы связать с каждым столбиком гистограммы долю больных диабетом среди людей, вес которых попадает

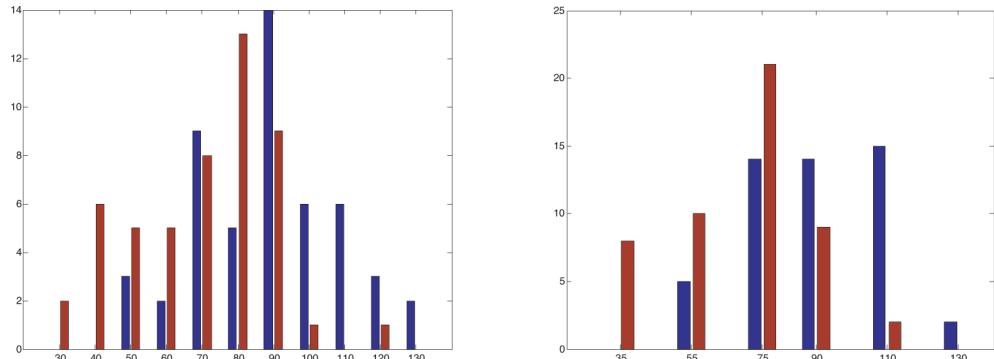


Рис. 1.10. (Слева) Искусственно подобранные данные, иллюстрирующие гистограмму распределения веса тела для людей, больных (**синие** столбики) и не больных (**красные** столбики) диабетом. Мы взяли одиннадцать фиксированных интервалов шириной 10 кг. (Справа) Объединив первый и второй, третий и четвертый, пятый и шестой, а также восьмой, девятый и десятый интервалы, мы получаем дискретизацию, в которой доля больных диабетом возрастает слева направо. Эта дискретизация увеличивает полезность признака для прогнозирования диабета

в соответствующий интервал. На самом деле в главе 10 мы увидим, что интервалы можно выбрать так, что эта доля будет монотонно возрастать (рис. 1.10).

Предыдущий пример дает еще одну иллюстрацию того, как в случае конкретной задачи – классификации – можно улучшить отношение сигнал–шум для признака. В исключительных случаях для отбора признаков преобразованию подвергается все пространство объектов. Взгляните на рис. 1.11: очевидно, что данные в левой части линейно неразделимы, но, перейдя из пространства объектов в новое «пространство признаков» путем возведения значений исходных признаков в квадрат, мы обнаруживаем, что данные стали почти линейно разделимыми. Фактически, добавив третий признак, мы можем проделать потрясающий трюк: построить этот классификатор в пространстве признаков, не конструируя самого пространства признаков.

Пример 1.9 (трюк с ядром). Пусть $\mathbf{x}_1 = (x_1, y_1)$ и $\mathbf{x}_2 = (x_2, y_2)$ – две точки. Рассмотрим отображение $(x, y) \mapsto (x^2, y^2, \sqrt{2}xy)$ в трехмерное пространство признаков. Точкам \mathbf{x}_1 и \mathbf{x}_2 соответствуют точки $\mathbf{x}'_1 = (x_1^2, y_1^2, \sqrt{2}x_1y_1)$ и $\mathbf{x}'_2 = (x_2^2, y_2^2, \sqrt{2}x_2y_2)$ в пространстве признаков. Скалярное произведение этих двух векторов признаков равно

$$\mathbf{x}'_1 \cdot \mathbf{x}'_2 = x_1^2 x_2^2 + y_1^2 y_2^2 + 2x_1 y_1 x_2 y_2 = (x_1 x_2 + y_1 y_2)^2 = (\mathbf{x}_1 \cdot \mathbf{x}_2)^2.$$

Таким образом, возведя в квадрат скалярное произведение в исходном пространстве, мы получаем скалярное произведение в новом пространстве *без фактического построения векторов признаков!* Функция, которая вычисляет скалярное произведение в пространстве признаков непосредственно по векторам в исходном пространстве, называется **ядром** – в данном случае ядро $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^2$.

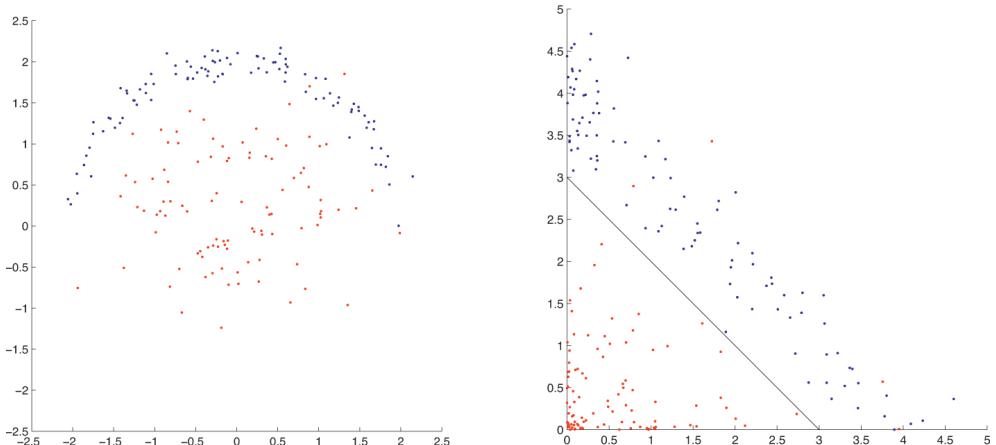


Рис. 1.11. (Слева) На таких данных линейный классификатор будет работать плохо. **(Справа)** После преобразования каждой точки (x, y) в точку $(x', y') = (x^2, y^2)$ данные становятся более «линейными», и решающая граница – прямая $x' + y' = 3$ – довольно хорошо их разделяет. В исходном пространстве этой прямой соответствует окружность радиуса $\sqrt{3}$ с центром в начале координат

Этот *трюк с ядром* (kernel trick) мы можем применить к базовому линейному классификатору, если изменим способ вычисления решающей границы. Напомним, что простой линейный классификатор находит решающую границу в виде уравнения $\mathbf{w} \cdot \mathbf{x} = t$, где $\mathbf{w} = \mathbf{p} - \mathbf{n}$ – разность между средним положительных примеров и средним отрицательных примеров. Пусть для определенности $\mathbf{n} = (0, 0)$ и $\mathbf{p} = (0, 1)$ и, предположим, что положительное среднее было получено по двум примерам $\mathbf{p}_1 = (-1, 1)$ и $\mathbf{p}_2 = (1, 1)$. Тогда $\mathbf{p} = \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2)$, и решающую границу можно записать в виде $\frac{1}{2}\mathbf{p}_1 \cdot \mathbf{x} + \frac{1}{2}\mathbf{p}_2 \cdot \mathbf{x} - \mathbf{n} \cdot \mathbf{x} = t$. Применив трюк с ядром, мы получим такую решающую границу: $\frac{1}{2}\kappa(\mathbf{p}_1, \mathbf{x}) + \frac{1}{2}\kappa(\mathbf{p}_2, \mathbf{x}) - \kappa(\mathbf{n}, \mathbf{x}) = t$. Используя определенное выше ядро, имеем $\kappa(\mathbf{p}_1, \mathbf{x}) = (-x + y)^2$, $\kappa(\mathbf{p}_2, \mathbf{x}) = (x + y)^2$ и $\kappa(\mathbf{n}, \mathbf{x}) = 0$, откуда находим решающую границу $\frac{1}{2}(-x + y)^2 + \frac{1}{2}(x + y)^2 = x^2 + y^2 = t$, то есть окружность радиуса \sqrt{t} с центром в начале координат. На рис. 1.11 эта ситуация показана на относительно большом наборе данных.

Соль такого «перехода к ядру» в случае простого линейного классификатора заключается в том, что мы не сводим обучающие данные к положительному и отрицательному среднему, а сохраняем все обучающие данные (в данном случае \mathbf{p}_1 , \mathbf{p}_2 и \mathbf{n}), так что при классификации нового объекта мы можем вычислить его ядро в паре с каждым обучающим примером. В обмен на более трудоемкие вычисления мы получаем возможность строить гораздо более точные решающие границы.

Взаимодействие между признаками

Одна из чарующих особенностей признаков, имеющая к тому же разнообразные проявления, – тот факт, что они могут взаимодействовать между собой. Иногда

такое взаимодействие можно обратить себе на пользу, иногда просто игнорировать, а подчас из-за него возникают проблемы. Мы уже видели пример взаимодействия признаков, когда обсуждали байесовскую фильтрацию спама. Понятно, что, обнаружив в сообщении слово «виагра», мы несильно удивимся, встретив также словосочетание «голубая таблетка». Игнорируя это взаимодействие – как поступает байесовский классификатор, – мы тем самым переоцениваем количество информации, содержащееся в наблюдении обоих словосочетаний в одном сообщении. Допустимо ли это, зависит от задачи: при классификации почтового спама проблема не слишком серьезна, если не считать, что порог принятия решения, возможно, стоит сдвинуть с учетом этого эффекта.

Другие примеры взаимодействия признаков видны из табл. 1.4. Рассмотрим признаки «ранж» и «вещ»; первый оценивает, в какой мере модель относится к ранжирующим, а второй – в какой мере она может справляться с вещественными признаками. Возможно, вы заметили, что значения этих признаков отличаются не более чем на 1 для всех моделей, кроме одной. Статистики говорят, что между этими признаками имеется положительная корреляция (см. замечание 1.3). Еще одна пара признаков с положительной корреляцией – «лог» и «дискр», то есть принадлежность к логическим моделям и способность обрабатывать дискретные признаки. Имеются также и признаки с отрицательной корреляцией, когда возрастание одного сопровождается убыванием другого: это естественно для признаков «групп» и «ранж», поскольку модели бывают преимущественно группирующими или преимущественно ранжирующими; то же самое относится и к признакам «лог» и «ранж». Наконец, между признаками «без учит» (модели, обучаемые без учителя) и «многокласс» (модели, способные работать более чем с двумя классами) нет корреляции, как и между признаками «диск» и «с учит». При классификации признаки могут по-разному коррелировать в зависимости от класса. Например, вполне можно себе представить, что для человека по имени Хилтон (Hilton), который работает в городском совете Парижа (Paris), сообщения, содержащие только слово «Paris» или только слово «Hilton», хорошие, тогда как сообщения, содержащие оба слова, вполне могут оказаться спамом¹. Иными словами, внутри класса спама между этими признаками имеется положительная корреляция, а в классе неспама – отрицательная. В таком случае игнорирование подобных взаимодействий может снизить качество классификации. В других случаях корреляция признаков может завуалировать истинную модель – с примерами мы встретимся ниже. С другой стороны, иногда корреляция признаков помогает выяснить представляющую интерес часть пространства объектов.

Случайные величины описывают возможные исходы случайного процесса. Они могут быть как дискретными (например, множество возможных исходов бросания игральной кости – {1, 2, 3, 4, 5, 6}), так и непрерывными (например, возможные исходы измерения веса тела в килограммах).

¹ Paris Hilton (Пэрис Хилтон) – светская львица, не блещущая интеллектом, одно упоминание которой наводит на мысль о спаме. – Прим. перев.

Областью определения случайной величины необязательно является множество целых или вещественных чисел, но при таком предположении математика оказывается гораздо проще, так что будем его придерживаться.

Если X – дискретная случайная величина с распределением вероятности $P(X)$, то **математическим ожиданием** X называется $\mathbb{E}[X] = \sum_x x P(x)$. Например, математическое ожидание бросания правильной кости равно $1 \cdot (1/6) + 2 \cdot (1/6) + \dots + 6 \cdot (1/6) = 3.5$. Отметим, что в точности такой исход невозможен. В случае непрерывной случайной величины сумму нужно заменить интегралом, а распределение вероятности – функцией плотности вероятности: $\mathbb{E}[X] = \int_{-\infty}^{+\infty} x p(x) dx$. Идея этой довольно абстрактной концепции в том, что если взять выборку x_1, \dots, x_n исходов случайного процесса, то математическое ожидание – это ожидаемое значение **выборочного среднего** $\bar{x} = (1/n) \sum_{i=1}^n x_i$ – знаменитый **закон больших чисел**, впервые доказанный Якобом Бернулли в 1713 году. Поэтому математическое ожидание иногда еще называют **средним по генеральной совокупности**, но важно понимать, что последнее значение – теоретическое, тогда как выборочное среднее – это эмпирическая **оценка** теоретического значения.

Оператор математического ожидания можно применить и к функциям случайных величин. Например, теоретическая **дисперсия** дискретной случайной величины определяется как $\mathbb{E}[(X - \mathbb{E}[X])^2] = \sum_x (x - \mathbb{E}[X])^2 P(x)$ – она измеряет разброс распределения относительно математического ожидания. Отметим, что

$$\mathbb{E}[(X - \mathbb{E}[X])^2] = \sum_x (x - \mathbb{E}[X])^2 P(x) = \mathbb{E}[X^2] - \mathbb{E}[X]^2.$$

Аналогично можно определить **выборочную дисперсию**: $\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$, что можно также записать в виде $\frac{1}{n} \sum_{i=1}^n x_i^2 - \bar{x}^2$. Иногда можно встретить такое определение выборочной дисперсии: $\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$. Деление на $n-1$ вместо n дает чуть более высокую оценку, компенсируя тот факт, что мы вычисляем разброс относительно выборочного среднего, а не среднего по генеральной совокупности.

Теоретическая **ковариация** между двумя дискретными случайными величинами X и Y определяется по формуле $\mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] = \mathbb{E}[XY] - \mathbb{E}[X] \cdot \mathbb{E}[Y]$. Дисперсия X – частный случай этой формулы, когда $Y = X$. В отличие от дисперсии, ковариация может быть как положительной, так и отрицательной. Положительная ковариация означает, что обе величины имеют тенденцию увеличиваться или уменьшаться вместе; отрицательная – что увеличение одной величины сопровождается уменьшением другой. Если взять выборку пар значений X и Y , то **выборочная ковариация** определяется по формуле $\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) = \frac{1}{n} \sum_{i=1}^n x_i y_i - \bar{x} \bar{y}$. Поделив ковариацию между X и Y на $\sqrt{\sigma_X^2 \sigma_Y^2}$, мы получим **коэффициент корреляции** – число от -1 до $+1$.

Замечание 1.3. Ожидания и оценки

Между признаками могут быть и другие соотношения. Рассмотрим следующие три признака молекулярного соединения, которые могут быть истинными или ложными:

- 1) соединение содержит шестичленный ароматический цикл;
- 2) соединение содержит углерод с частичным зарядом -0.13 ;
- 3) соединение содержит углерод в шестичленном ароматическом цикле с частичным зарядом -0.13 .

Говорят, что третий признак более **специфичный** (или менее **общий**), чем два остальных, потому что если третий признак принимает значение истина, значит,

то же самое справедливо для первого и второго. Обратное, однако, неверно: если первый и второй признаки истинны, то третий может быть и ложным (потому что углерод в шестичленном ароматическом цикле необязательно то же самое, что углерод с частичным зарядом -0.13). Такие соотношения можно использовать при поиске признаков, которые стоит включить в модель. Например, если обнаруживается, что третий признак принимает значение истина для конкретного отрицательного примера, который мы хотели бы исключить, то нет смысла рассматривать более общие первый и второй признаки, потому что они тоже не помогут исключить отрицательный пример. Аналогично если выясняется, что первый признак принимает значение ложь для некоторого положительного примера, который мы хотели бы включить, то ни к чему рассматривать более специфичный третий признак. Иными словами, эти связи помогут внести структуру в поиск признаков, обладающих предсказательной силой.

1.4 Итоги и перспективы

В этой главе я ставил себе целью показать вам восхитительный ландшафт машинного обучения, чтобы вызвать интерес и побудить читать дальше. Вот что мы рассмотрели.

- ☞ Предметом машинного обучения является использование правильных признаков для построения правильных моделей, призванных решить правильно поставленные задачи. К таким задачам относятся: двоичная и многоклассовая классификация, регрессия, кластеризация и дескриптивное моделирование. Модели для первых задач из этого списка обучаются с учителем, то есть требуются размеченные обучающие данные. Например, чтобы обучить фильтр спама, нам понадобится размеченный набор почтовых сообщений, каждое из которых помечено как спам или неспам. Чтобы узнать, насколько хороша модель, также понадобятся размеченные тестовые данные, но не те же самые, что обучающие, поскольку оценка модели на тех же данных, на которых она обучалась, покажет слишком благостную картину; тестовый набор необходим для выявления феномена переобученности.
- ☞ С другой стороны, обучение без учителя проводится на неразмеченных данных, поэтому тестовые данные как таковые не нужны. Например, чтобы оценить конкретное разбиение данных на кластеры, нужно вычислить среднее расстояние до центра кластера. Существуют и другие формы обучения без учителя, например выявление ассоциаций (явлений, которые обычно наблюдаются совместно) и идентификация скрытых переменных, таких как жанры фильмов. Переобучение свойственно и обучению без учителя; например, если поместить каждую точку в отдельный кластер, то среднее расстояние до центра кластера обратится в нуль, но такое разбиение, очевидно, не слишком полезно.

- ☞ С точки зрения результатов мы различаем прогностические модели, которые выводят значение целевой переменной, и дескриптивные модели, которые выявляют интересные структуры в данных. Как правило, прогностические модели обучаются с учителем, а дескриптивные – без учителя, но существуют также примеры обучения дескриптивных моделей с учителем (например, обнаружение подгрупп, ставящее целью выявление областей с необычным распределением по классам) и обучения прогностических моделей без учителя (например, прогностическая кластеризация, когда выявленные кластеры интерпретируются как классы).
- ☞ Мы выделили три вида моделей машинного обучения: геометрические, вероятностные и логические. Геометрические модели строятся в декартовом пространстве объектов с применением таких геометрических понятий, как плоскости и расстояние. Прототипом всех геометрических моделей является линейный классификатор, который строит плоскость решений, ортогональную прямой, соединяющей центроиды множеств положительных и отрицательных примеров. В случае вероятностных моделей обучение рассматривается как процесс уменьшения неопределенности с помощью данных. Например, в байесовском классификаторе моделируется апостериорное распределение $P(Y|X)$ (или его антагонист, функция правдоподобия $P(X|Y)$), которое говорит о распределении по классам Y после наблюдения значений признаков X . Логические модели – самые «декларативные» из трех, в них фигурируют правила «если – то», построенные из логических условий, а цель – выделить однородные области в пространстве объектов.
- ☞ Мы ввели также различие между группирующими и ранжирующими моделями. Группирующие модели разбивают пространство объектов на сегменты, определяемые на этапе обучения, и потому их разрешающая способность конечна. В каждом сегменте обычно применяется какая-нибудь очень простая модель, например «всегда предсказывать такой-то класс». Ранжирующие модели по своей природе более глобальны, они производят ранжирование по местоположению объекта в пространстве объектов (обычно, но не всегда, декартовом). Логические модели – типичные примеры группирующих моделей, а геометрические обычно ближе к ранжирующими, хотя граница размыта. Сейчас слова о разделении звучат абстрактно, но эта мысль станет яснее, когда в следующей главе мы будем обсуждать кривые покрытия.
- ☞ И напоследок мы обсудили роль признаков в машинном обучении. Никакая модель не может существовать без признаков, и иногда для построения модели хватает единственного признака. Данные не всегда изначально оснащены признаками, и зачастую признаки приходится преобразовывать и даже конструировать. Поэтому машинное обучение обычно представляет собой итеративный процесс: мы узнаем, что отобрали правильные признаки, только после того, как модель построена, и если модель работает плохо,

то необходимо проанализировать, в чем причины и нельзя ли улучшить состав признаков.

Что будет в книге дальше

В следующих девяти главах мы будем следовать намеченному плану, детализируя его пункты:

- ☞ задачи машинного обучения в главах 2 и 3;
- ☞ логические модели: концептуальное обучение в главе 4, древовидные модели в главе 5, модели на основе правил в главе 6;
- ☞ геометрические модели: линейные модели в главе 7, метрические модели в главе 8;
- ☞ вероятностные модели в главе 9;
- ☞ признаки в главе 10.

Глава 11 посвящена методам обучения «ансамблей» моделей, имеющих определенные преимущества, по сравнению с одиночными моделями. В главе 12 мы рассмотрим ряд методов, которые специалисты называют «экспериментами», они подразумевают обучение и оценку моделей на реальных данных. Наконец, в эпилоге мы подведем итоги и обсудим, что дальше.



Бинарная классификация и родственные задачи

В этой и в следующей главе мы с высоты птичьего полета бросим взгляд на широкий круг задач, решаемых методами машинного обучения. Под словом «задача» здесь понимается любая деятельность, эффективность которой призвано улучшить машинное обучение (вспомните определение машинного обучения выше), например распознавание почтового спама. Поскольку речь пойдет о задаче классификации, нам потребуется обучить подходящий классификатор на обучающих данных. Существует много типов классификаторов: линейные, байесовский, метрические – и это еще далеко не полный перечень. Мы будем называть эти типы моделями, они станут предметом рассмотрения в главах 4–9. Классификация – лишь одна из обширного спектра задач, для решения которых можно обучить модель; в этой главе мы также кратко рассмотрим оценивание вероятности класса и ранжирование. А в следующей обсудим регрессию, кластеризацию и дескриптивное моделирование. Для каждой задачи мы расскажем, что это такое, какие существуют варианты, как можно оценивать качество ее решения и как она соотносится с другими задачами. Но прежде всего введем обозначения, которые будут использоваться в этой и последующих главах (см. краткое описание соответствующих математических понятий в защемании 2.1).

Предмет рассмотрения в машинном обучении называется *объектом*. Множество всех возможных объектов называется *пространством объектов* и обозначается буквой \mathcal{X} . Например, \mathcal{X} могло бы быть множеством всех возможных почтовых сообщений, которые можно записать буквами латиницы¹. Мы будем также различать *пространство меток* \mathcal{Y} и *пространство выходов* \mathcal{U} . Пространство меток используется в машинном обучении с учителем для пометки примеров. Чтобы решить задачу, нам понадобится *модель*: отображение пространства объектов в пространство выходов. Например, в случае классификации простран-

¹ Наверное, стоит отметить, что такое пространство объектов невообразимо велико (например, множество всех текстовых сообщений длиной 160 знаков, среди которых могут встречаться только строчные буквы, пробелы и точки, состоит из 28^{160} элементов, такое число далеко превосходит возможности большинства карманных калькуляторов), и лишь малая доля этого множества сообщений несет какой-то разумный смысл.

ством выходов является множество классов, а в задаче регрессии это множество вещественных чисел. Для обучения модели необходим *обучающий набор* Tr *помеченных объектов* $(x, l(x))$, называемых также *примерами*. Здесь $l: \mathcal{X} \rightarrow \mathcal{L}$ – помечающая функция.

В табл. 2.1, опирающейся на эти термины и обозначения, перечислены различные виды прогностических моделей, обучаемых с учителем. Чаще всего в машинном обучении встречается случай, когда пространство меток совпадает с пространством выходов. Иными словами, $\mathcal{Y} = \mathcal{L}$, а мы пытаемся обучить аппроксимацию $\hat{l}: \mathcal{X} \rightarrow \mathcal{L}$ истинной помечающей функции l , имея в своем распоряжении только метки, присвоенные обучающим данным. Под эти условия подпадает как классификация, так и регрессия. Когда пространства меток и выходов различаются, обычно целью является обучение модели, которая порождает на выходе не просто метку; например, могут возвращаться оценки для каждой возможной метки. В таком случае имеем $\mathcal{Y} = \mathbb{R}^k$, где $k = |\mathcal{L}|$ – количество меток.

Ситуация может осложняться наличием *шума*, который может представлять в обличье *меточного шума*, – вместо функции $l = l(x)$ мы наблюдаем искаженную метку l' , или *объектного шума*, – вместо x наблюдается искаженный объект x' . Вследствие зашумленности данных в общем случае не рекомендуется добиваться точного соответствия обучающим данным, поскольку это может привести к переобучению на шуме. Часть помеченных данных обычно резервируют для оценки или проверки классификатора, эта часть называется *тестовым набором* и обозначается Te . Мы будем использовать верхние индексы для ограничения обучающего или тестового набора на конкретный класс, например $Te^\oplus = \{(x, l(x)) | x \in Te, l(x) = \oplus\}$ – это множество положительных тестовых примеров, а Te^\ominus – множество отрицательных тестовых примеров

Простейшее пространство объектов возникает, когда объекты описываются фиксированным множеством *признаков*, которые также называют атрибутами, прогностическими параметрами, объясняющими переменными или независимыми переменными. Обозначив *область значений* признака \mathcal{F}_p , будем иметь $\mathcal{X} = \mathcal{F}_1 \times \mathcal{F}_2 \times \dots \times \mathcal{F}_d$ то есть каждый объект является вектором длины d , состоящим из значений признаков. Иногда признаки очевидны, а иногда их приходится отбирать. Например, в примере фильтра почтового спама в прологе мы отобрали много признаков – количество вхождений каждого имеющегося словарного слова в сообщение. Но даже когда признаки заданы явно, нам зачастую хочется каким-то образом преобразовать их и тем повысить полезность для решения задачи. Мы подробно обсудим эту тему в главе 10.

Коротко напомним некоторые важные понятия дискретной математики. *Множеством* называется собрание объектов, обычно одного и того же вида (например, множество натуральных чисел \mathbb{N} или множество вещественных чисел \mathbb{R}). Мы пишем $x \in A$, если x является элементом множества A , и $A \subseteq B$, если все элементы A являются также элементами B (в частности, A и B могут быть

одним и тем же множеством, и тогда $A \subseteq B$ и $B \subseteq A$). **Пересечение** и **объединение** двух множеств определяются следующим образом: $A \cap B = \{x | x \in A \text{ и } x \in B\}$, $A \cup B = \{x | x \in A \text{ или } x \in B\}$. **Разностью** двух множеств называется множество $A \setminus B = \{x | x \in A \text{ и } x \notin B\}$. Принято фиксировать **универсальное множество** U – такое, что все рассматриваемые множества являются подмножествами U . **Дополнением** множества A называется множество $\bar{A} = U \setminus A$. Два множества называются **непересекающимися**, или **дизъюнктными**, если их пересечение пусто: $A \cap B = \emptyset$. **Кардинальным числом** множества A (обозначается $|A|$) называется количество элементов в нем. **Степенным множеством** множества A называется множество всех его подмножеств – $2^A = \{B | B \subseteq A\}$; его кардинальное число равно $|2^A| = 2^{|A|}$. **Характеристической функцией** множества A называется функция $f: U \rightarrow \{\text{true}, \text{false}\}$ – такая, что $f(x) = \text{true}$, если $x \in A$ и $f(x) = \text{false}$, если $x \in U \setminus A$.

Если A и B – множества, то **декартовым произведением** $A \times B$ называется множество всех пар $\{(x, y) | x \in A \text{ и } y \in B\}$; это понятие легко обобщается на произвольное количество множеств. **Отношением** (бинарным) называется множество пар $R \subseteq A \times B$ для некоторых множеств A и B ; если $A = B$, то говорят об отношении над A . Вместо $(x, y) \in R$ можно также писать xRy . Отношение над A называется (i) **рефлексивным**, если xRx для всех $x \in A$; (ii) **симметричным**, если из xRy следует, что yRx для всех $x, y \in A$; (iii) **антисимметричным**, если из xRy и yRx следует, что $x = y$ для всех $x, y \in A$; (iv) – **транзитивным**, если из xRy и yRz следует, что xRz для всех $x, y, z \in A$, (v) – **полным**, если для всех $x, y \in A$ имеет место либо xRy , либо yRx .

Частичным порядком называется бинарное отношение, обладающее свойствами рефлексивности, антисимметричности и транзитивности. Например, отношение подмножества \subseteq является частичным порядком. **Полным порядком** называется бинарное отношение, являющееся полным (и, следовательно, рефлексивным), антисимметричным и транзитивным. Отношение \leq на множестве вещественных чисел является отношением полного порядка. Если xRy или yRx , то говорят, что x и y **сравнимы**, в противном случае – **несравнимы**. **Отношением эквивалентности** называется бинарное отношение \equiv , являющееся рефлексивным, симметричным и транзитивным. Классом эквивалентности x называется множество $[x] = \{y | x \equiv y\}$. Например, бинарное отношение «содержит столько же элементов, сколько» над любым множеством является отношением эквивалентности. Любые два класса эквивалентности не пересекаются, а объединение всех классов эквивалентности образует **разбиение** множества. Если A_1, \dots, A_n – разбиение множества A , то есть $A_1 \cup \dots \cup A_n = A$ и $A_i \cap A_j = \emptyset$ для всех $i \neq j$, то пишут $A = A_1 \uplus \dots \uplus A_n$.

Для иллюстрации предположим, что T – дерево признаков, и определим отношение $\sim_T \subseteq \mathcal{X} \times \mathcal{X}$, так что $x \sim_T x'$ тогда и только тогда, когда x и x' приписаны одному и тому же листу дерева T . Тогда \sim_T является отношением эквивалентности, а классы эквивалентности по этому отношению в точности совпадают с сегментами пространства объектов, ассоциированными с T .

Замечание 2.1. Полезные понятия дискретной математики

Задача	Пространство меток	Пространство выходов	Проблема обучения
Классификация	$\mathcal{L} = \mathcal{C}$	$\mathcal{Y} = \mathcal{C}$	Обучить аппроксимацию $\hat{c}: \mathcal{X} \rightarrow \mathcal{C}$ истинной помечающей функции c
Оценка и ранжирование	$\mathcal{L} = \mathcal{C}$	$\mathcal{Y} = \mathbb{R}^{ \mathcal{C} }$	Обучить модель, которая выводит вектор оценок принадлежности к классам
Оценивание вероятности	$\mathcal{L} = \mathcal{C}$	$\mathcal{Y} = [0,1]^{ \mathcal{C} }$	Обучить модель, которая выводит вектор вероятностей принадлежности к классам
Регрессия	$\mathcal{L} = \mathbb{R}$	$\mathcal{Y} = \mathbb{R}$	Обучить аппроксимацию $\hat{f}: \mathcal{X} \rightarrow \mathbb{R}$ истинной помечающей функции f

Таблица 2.1. Сценарии обучения прогностических моделей

Эта глава посвящена первым трем сценариям, перечисленным в табл. 2.1: классификация рассматривается в разделе 2.1, оценка и ранжирование – в разделе 2.2, оценивание вероятностей принадлежности к классам – в разделе 2.3. Чтобы текст оставался обозримым, мы ограничимся в основном двухклассовыми задачами, а задачи с большим числом классов отложим до главы 3. Регрессия, обучение без учителя и обучение дескриптивных моделей также рассматриваются здесь.

В этой главе я буду иллюстрировать ключевые концепции, используя в качестве примеров простые модели, подобные обсуждавшимся в прологе. Это либо простые древовидные модели – как представители группирующих моделей, либо линейные модели – как представители ранжирующих моделей. Иногда мы даже будем конструировать модели с единственным признаком, такой сценарий называется *одномерным машинным обучением*. Вопрос о том, как *обучать* такие модели, рассматривается, начиная с главы 4.

2.1 Классификация

Классификация – самая часто встречающаяся задача машинного обучения. *Классификатором* называется отображение $\hat{c}: \mathcal{X} \rightarrow \mathcal{C}$, где $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ – конечное и обычное небольшое множество *меток классов*. Иногда под C_i мы будем также понимать множество примеров соответствующего класса. Знак «крышки» означает, что $\hat{c}(x)$ – оценка истинной, но неизвестной функции $c(x)$. Примеры для классификатора имеют вид $(x, c(x))$, где $x \in \mathcal{X}$ – объект, а $c(x)$ – истинный класс этого объекта. Под обучением классификатора понимается построение функции \hat{c} , которая как можно лучше аппроксимирует c (и не только на обучающем наборе, но в идеале и на всем пространстве объектов \mathcal{X}).

В простейшем случае есть всего два класса, которые обычно называют *положительным* и *отрицательным*, \oplus и \ominus или $+1$ и -1 . Двухклассовую классификацию часто называют также *бинарной классификацией* (или *концептуальным обучением*, если положительный класс можно с должным основанием назвать концептом). Фильтрация почтового спама – хороший пример бинарной классификации, в котором спам традиционно считается положительным классом, а не-спам – отрицательным (очевидно, что слово «положительный» в данном случае не означает «хороший»!). Из других примеров бинарной классификации назовем медицинскую диагностику (положительным классом здесь является конкретное заболевание) и обнаружение мошенничества с кредитными картами.

Дерево признаков на рис. 2.1 слева можно преобразовать в классификатор, пометив каждый его листовый узел классом. Проще всего это сделать, сопоставив каждому листу *мажоритарный класс*, тогда получится решающее дерево, изображенное на рис. 2.1 справа. Классификатор работает следующим образом: если почтовое сообщение содержит слово «виагра», то оно классифицируется как спам (самый правый лист), в противном случае решение зависит от того, содер-

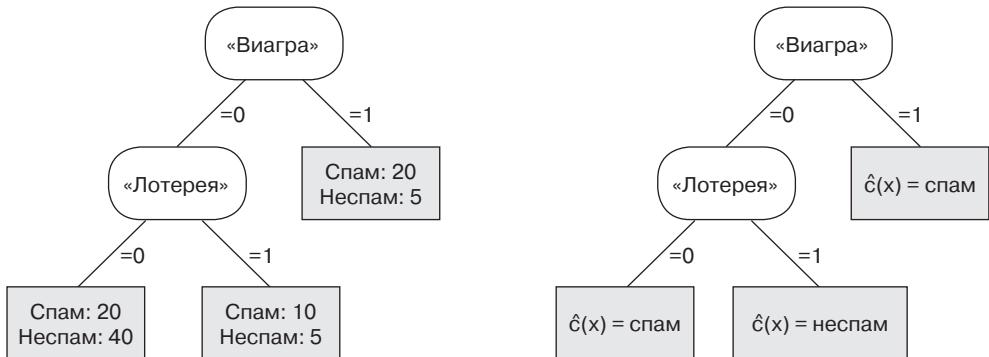


Рис. 2.1. (Слева) Дерево признаков, в котором листья представляют распределение обучающего набора по классам. **(Справа)** Решающее дерево, полученное с помощью правила мажоритарного класса

жит ли сообщение слово «лотерея»¹. Числа на рис. 2.1 дают общее представление о качестве этого классификатора. Самый левый узел правильно определяет 40 хороших сообщений, но при этом ошибочно относит к хорошим 20 спамных сообщений, не содержащих ни одного из слов «виагра» или «лотерея». Средний лист правильно классифицирует 10 спамных сообщений, но ошибочно помечает 5 хороших сообщений как спам. Критерий «виагра» правильно отбирает 20 спамных сообщений, но вместе с ними 5 хороших. В общем получается, что из 50 спамных сообщений 30 классифицированы правильно, а из 50 хороших сообщений таковыми сочтено только 40.

Оценка качества классификации

Оценить качество таких классификаторов можно с помощью таблицы, которая называется *таблицей сопряженности*, или *матрицей неточностей* (табл. 2.2 слева). Каждая строка этой таблицы соответствует фактическим классам из тестового набора, а каждый столбец – классам, предсказанным классификатором. Так, из первой строки видно, что тестовый набор содержит 50 положительных примеров, из которых 30 классифицированы правильно, а 20 – неправильно. В последнем столбце и в последней строке находятся *маргиналы* – суммы элементов в соответствующем столбце или строке. Маргиналы важны, потому что позволяют оценить статистическую значимость. Например, в таблице сопряженности (табл. 2.2 справа) маргиналы точно такие же, но классификатор, очевидно, относил примеры к положительному или отрицательному классу случайным образом – в результате распределение фактических положительных и отрицательных примеров

¹ Если вам не терпится узнать, как такое решающее дерево получается из обучающих данных, можете бегло просмотреть алгоритм 5.1 на стр. 145.

в любом предсказанном классе такое же, как их общее распределение (в данном случае – равномерное).

	Предсказано \oplus	Предсказано \ominus		\oplus	\ominus
Фактически \oplus	30	20	50	20	30
Фактически \ominus	10	40	50	20	30
	40	60	100	40	60

Таблица 2.2. (Слева) Двухклассовая таблица сопряженности (она же – матрица неточностей) иллюстрирует качество решающего дерева на рис. 2.1. Числа на главной диагонали – правильные предсказания, а на побочной – ошибки. **(Справа)** Матрица сопряженности с такими же маргиналами, но независимыми строками и столбцами

По таблице сопряженности мы можем вычислить ряд показателей качества. Простейшим из них является *верность* – доля правильно классифицированных тестовых примеров. В обозначениях, введенных в начале главы, верность на тестовом наборе Te определяется по формуле

$$acc = \frac{1}{|Te|} \sum_{x \in Te} I[\hat{c}(x) = c(x)]. \quad (2.1)$$

Здесь $I[\cdot]$ – *индикаторная функция*, которая равна 1, если ее аргумент принимает значение истина, и 0 в противном случае. В данном случае она дает удобный способ подсчитать количество правильно классифицированных тестовых примеров (то есть случаев, когда оценочная метка класса $\hat{c}(x)$ совпадает с истинной меткой класса $c(x)$). Так, в табл. 2.2 слева верность классификатора равна 0.70, или 70%, а в табл. 2.2 справа – только 0.50. Можно также вычислить *частоту ошибок* – долю неправильно классифицированных примеров, в данном случае она равна 0.30 и 0.50 соответственно. Понятно, что сумма верности и частоты ошибок равна 1.

Верность на тестовом наборе можно рассматривать как *оценку* вероятности того, что произвольный объект $x \in \mathcal{X}$ классифицирован правильно; точнее, это оценка вероятности

$$P_x(\hat{c}(x) = c(x)).$$

(Я пишу P_x , чтобы подчеркнуть тот факт, что это распределение вероятности по пространству объектов \mathcal{X} ; я часто буду опускать нижние индексы, если они ясны из контекста.) Как правило, истинные классы известны только для малой части пространства объектов, так что оценка – лучшее, на что можно рассчитывать. Поэтому тестовый набор должен быть максимально репрезентативным. Обычно эта мысль формализуется в виде предположения о том, что встречаемость объектов в реальном мире – то есть насколько вероятным или типичным является конкретное почтовое сообщение – описывается неизвестным распределением вероятности на \mathcal{X} и что тестовый набор Te сгенерирован в соответствии с этим распределением.

Часто бывает удобно, если не сказать – необходимо, различать качество классификации для отдельных классов. Поэтому нам понадобится дополнительная терминология. Правильно классифицированные положительные и отрицательные объекты называют, соответственно, *истинно положительными* и *истинно отрицательными* результатами. Неправильно классифицированные положительные объекты называют, пожалуй, не вполне интуитивно, *ложноотрицательными*, а неправильно классифицированные отрицательные объекты – *ложноположительными*. Запомнить это проще всего, считая, что слова «положительный» и «отрицательный» относятся к предсказанию классификатора, а «истинный» и «ложный» – к правильности предсказания. Тогда ложноположительный означает нечто, что было неправильно определено как положительный и, следовательно, на самом деле являющееся отрицательным (например, хорошее сообщение неправильно классифицировано как спам или у здорового пациента неправильно определено заболевание). В табл. 2.2 слева мы имеем 30 истинно положительных результатов, 20 ложноотрицательных, 40 истинно отрицательных и 10 ложноположительных.

Частотой истинно положительных результатов называется доля правильно классифицированных положительных объектов, математически ее можно определить формулой

$$tpr = \frac{\sum_{x \in T_p} I[\hat{c}(x) = c(x) = \oplus]}{\sum_{x \in T_e} I[c(x) = \oplus]}. \quad (2.2)$$

Частота истинно положительных результатов – это оценка вероятности того, что произвольно взятый положительный объект классифицирован правильно, то есть оценка величины $P_x(\hat{c}(x) = \oplus | c(x) = \oplus)$. Аналогично *частотой истинно отрицательных результатов* называется доля правильно классифицированных отрицательных объектов (математическое определение см. в табл. 2.3 ниже), она оценивает вероятность $P_x(\hat{c}(x) = \ominus | c(x) = \ominus)$. Эти частоты, которые иногда называют *чувствительностью* и *специфичностью*, можно рассматривать как верность применительно к отдельному классу. Имея таблицу сопряженности, частоты истинно положительных и истинно отрицательных результатов можно вычислить, разделив число на главной диагонали (где находятся правильно предсказанные результаты) на итоговую сумму по строке. Можно также говорить о частотах ошибок на уровне класса, то есть *частоте ложноотрицательных результатов* для положительных объектов (доле неправильно классифицированных положительных объектов – ложноотрицательных результатов – от общего количества положительных объектов) и *частоте ложноположительных результатов* для отрицательных объектов (ее иногда называют *частотой ложных тревог*). Эти частоты можно найти, разделив число на побочной диагонали (где находятся неправильно предсказанные результаты) на итоговую сумму по строке.

В табл. 2.2 слева частота истинно положительных результатов равна 60%, частота истинно отрицательных – 80%, частота ложноотрицательных – 40% и частота ложноположительных результатов – 20%. В табл. 2.2 справа частота истинно

положительных результатов равна 40%, частота истинно отрицательных – 60%, частота ложноотрицательных – 60% и частота ложноположительных результатов – 40%. Отметим, что в обоих случаях верность равна среднему арифметическому частоты истинно положительных и частоты истинно отрицательных результатов (а частота ошибок – среднему арифметическому частоты ложноположительных и частоты ложноотрицательных результатов). Однако это справедливо лишь тогда, когда тестовый набор содержит одинаковое число положительных и отрицательных объектов, – в общем случае нужно вычислять *взвешенное* среднее, в котором веса пропорциональны числу положительных и отрицательных объектов в тестовом наборе.

Пример 2.1 (верность как взвешенное среднее). Предположим, что на тестовом наборе классификатор дал предсказания, показанные в следующей таблице.

	Предсказано \oplus	Предсказано \ominus	
Фактически \oplus	60	15	75
Фактически \ominus	10	15	25
	70	30	100

Из этой таблицы видно, что частота истинно положительных результатов $tpr = 60/75 = 0.80$, а частота истинно отрицательных результатов $tnr = 15/25 = 0.60$. Общая верность $acc = (60+15)/100 = 0.75$, и теперь это уже не среднее арифметическое частоты истинно положительных и частоты истинно отрицательных результатов. Однако, приняв во внимание долю положительных примеров $pos = 0.75$ и долю отрицательных примеров $neg = 1 - pos = 0.25$, мы увидим, что

$$acc = pos \cdot tpr + neg \cdot tnr. \quad (2.3)$$

Эта формула имеет место в общем случае: если количество положительных и отрицательных объектов одинаково, то получается невзвешенное среднее из предыдущего примера ($acc = (tpr + tnr)/2$).

Формула 2.3 имеет интуитивно понятную интерпретацию: хорошее качество классификации на любом отдельно взятом классе дает вклад в общую верность классификации, но чем чаще встречается класс, тем его вклад больше. Для достижения хорошей верности классификатор должен уделять больше внимания *мажоритарному классу*, особенно если распределение объектов по классам не сбалансировано. Однако часто бывает, что мажоритарный класс как раз наименее интересен. Для иллюстрации предположим, что мы посыпаем запрос интернет-поисковику¹ и что этому запросу релевантна всего одна страница из тысячи.

¹ Если зафиксировать запрос, то интернет-поисковик можно рассматривать как бинарный классификатор с классами «релевантно» и «нерелевантно» или «интересно» и «неинтересно». На практике такая фиксация не слишком реалистична, но для наших целей аналогия полезна.

Теперь рассмотрим «упрямый» поисковик, который не возвращает *никаких* ответов, то есть классифицирует все страницы как нерелевантные запросу. Следовательно, частота истинно положительных результатов для него равна 0%, а частота истинно отрицательных – 100%. Поскольку $pos = 1/1000 = 0.1\%$ и $neg = 99.9\%$, то верность упрямого поисковика очень высока (99.9%). По-другому это можно выразить так: если случайным образом (равномерно) выбрать одну страницу из множества всех веб-страниц, то вероятность получить положительный результат составляет всего 0.001, и это именно те страницы, на которых упрямый поисковик дает ошибку. Однако обычно мы выбираем веб-страницы неравномерно, поэтому верность в этом контексте несущественна. У сколько-нибудь полезной поисковой системы частота истинно положительных результатов должна быть гораздо выше, а это обычно сопровождается соответственным уменьшением частоты истинно отрицательных результатов (и, следовательно, верности классификации).

Из этого примера можно заключить, что если интерес представляет миноритарный класс и он очень мал, то верность и качество классификации для мажоритарного класса – не те величины, которые следует оптимизировать. Поэтому в таких случаях обычно рассматривается альтернатива частоте истинно отрицательных результатов – *точность*. Точность дополняет частоты истинно положительных результатов в следующем смысле: частота истинно положительных результатов есть доля предсказанных положительных результатов во множестве действительно положительных объектов, тогда как точность – это доля действительно положительных объектов во множестве предсказанных положительных результатов. В примере 2.1 точность классификатора на тестовом наборе равна $60/70 = 85.7\%$. В примере упрямого поисковика не только частота истинно положительных результатов (которая в этом контексте называется *полнотой*) равна 0, но точность равна 0, что наглядно демонстрирует проблему поисковой системы, которая не возвращает никаких ответов. В табл. 2.3 приведена сводка оценочных показателей, введенных в этом разделе.

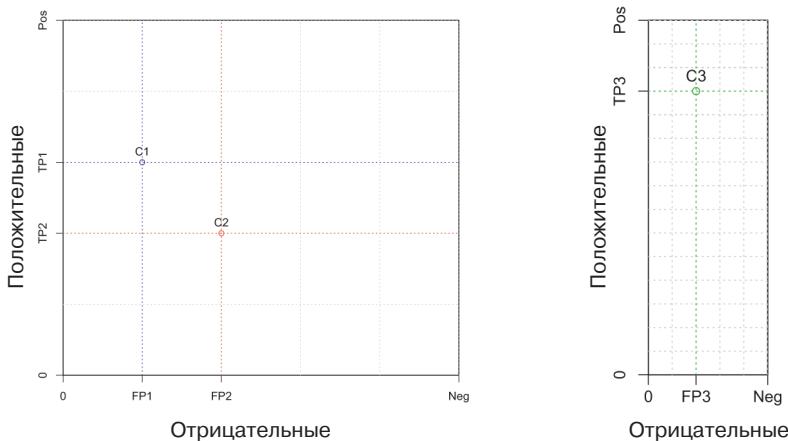
Наглядное представление качества классификации

Теперь я познакомлю вас с важным средством наглядного представления качества классификаторов и других моделей, которое называется *графиком покрытия*. Внимательное изучение двухклассовых таблиц сопряженности (табл. 2.2) приводит к выводу, что хотя таблица содержит девять чисел, только четыре из них можно выбрать произвольно. Например, если определены истинно (ложно) положительные (отрицательные) результаты, то маргиналы оказываются фиксированными. Или если известно количество истинно положительных результатов, количество истинно отрицательных результатов, общее количество положительных примеров и размер тестового набора, то все остальные элементы таблицы можно вычислить. В статистике говорят, что таблица имеет четыре *степени свободы*¹.

¹ В общем случае k -классовая таблица сопряженности состоит из $(k+1)^2$ элементов и имеет k^2 степеней свободы.

Показатель	Определение	Равен	Оценивает
Количество положительных примеров	$Pos = \sum_{x \in Te} I[c(x) = \oplus]$		
Количество отрицательных примеров	$Neg = \sum_{x \in Te} I[c(x) = \ominus]$	$ Te - Pos$	
Количество истинно положительных результатов	$TP = \sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]$		
Количество истинно отрицательных результатов	$TN = \sum_{x \in Te} I[\hat{c}(x) = c(x) = \ominus]$		
Количество ложноположительных результатов	$FP = \sum_{x \in Te} I[\hat{c}(x) = \oplus, c(x) = \ominus]$	$Neg - TN$	
Количество ложноотрицательных результатов	$FN = \sum_{x \in Te} I[\hat{c}(x) = \ominus, c(x) = \oplus]$	$Pos - TP$	
Доля положительных примеров	$pos = \frac{1}{ Te } \sum_{x \in Te} I[c(x) = \oplus]$	$Pos / Te $	$P(c(x) = \oplus)$
Доля отрицательных примеров	$neg = \frac{1}{ Te } \sum_{x \in Te} I[c(x) = \ominus]$	$1 - pos$	$P(c(x) = \ominus)$
Класс отношения	$clr = pos/neg$	Pos/Neg	
(*) Верность	$acc = \frac{1}{ Te } \sum_{x \in Te} I[\hat{c}(x) = c(x)]$		$P(\hat{c}(x) = c(x))$
(*) Частота ошибок	$err = \frac{1}{ Te } \sum_{x \in Te} I[\hat{c}(x) \neq c(x)]$	$1 - acc$	$P(\hat{c}(x) \neq c(x))$
Частота истинно положительных результатов, чувствительность, полнота	$tpr = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]}{\sum_{x \in Te} I[c(x) = \oplus]}$	TP/Pos	$P(\hat{c}(x) = \oplus c(x) = \oplus)$
Частота истинно отрицательных результатов, специфичность, отрицательная полнота	$tnc = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \ominus]}{\sum_{x \in Te} I[c(x) = \ominus]}$	TN/Neg	$P(\hat{c}(x) = \ominus c(x) = \ominus)$
Частота ложноположительных результатов, частота ложных тревог	$fpr = \frac{\sum_{x \in Te} I[\hat{c}(x) = \oplus, c(x) = \ominus]}{\sum_{x \in Te} I[c(x) = \ominus]}$	$FP/Neg = 1 - tnc$	$P(\hat{c}(x) = \oplus c(x) = \ominus)$
Частота ложноотрицательных результатов	$fnr = \frac{\sum_{x \in Te} I[\hat{c}(x) = \ominus, c(x) = \oplus]}{\sum_{x \in Te} I[c(x) = \oplus]}$	$FN/Pos = 1 - tpr$	$P(\hat{c}(x) = \ominus c(x) = \oplus)$
Точность, доверие	$prec = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]}{\sum_{x \in Te} I[\hat{c}(x) = \oplus]}$	$TP/(TP + FP)$	$P(c(x) = \oplus \hat{c}(x) = \oplus)$

Таблица 2.3. Сводка различных количественных и оценочных показателей качества классификаторов на тестовом наборе Te . Символами, начинающимися с заглавной буквы, обозначены абсолютные частоты (счетчики), а символами, начинающимися со строчной буквы, – относительные частоты (коэффициенты). Все показатели, кроме помеченных звездочкой (*), определены только для бинарной классификации. В правом столбце указаны вероятности в пространстве объектов, оценками которых служат соответствующие частоты



(Слева) График покрытия, изображающий обе таблицы сопряженности, приведенные в табл. 2.2. График квадратный, потому что распределение по классам равномерно.
(Справа) График покрытия для примера 2.1, в котором коэффициент распределения по классам $clr = 3$

Часто нас особенно интересуют следующие четыре числа, полностью определяющие таблицу сопряженности: количество положительных примеров Pos , количество отрицательных примеров Neg , количество истинно положительных результатов TP и количество ложноотрицательных результатов FP . На графике покрытия эти четыре числа представлены точкой в прямоугольной системе координат. Представьте себе прямоугольник высотой Pos и шириной Neg . Пусть все положительные объекты расположены на оси y этого прямоугольника, а все отрицательные – на оси x . Нас не интересует реальный порядок расположения положительных и отрицательных объектов на осях при условии, что *положительные предсказания предшествуют отрицательным*. Это дает достаточно информации для изображения всей таблицы сопряженности в виде одной точки внутри прямоугольника (рис. 2.2).

Рассмотрим два классификатора, обозначенных $C1$ и $C2$ на рис. 2.2 слева. Из графиков покрытия мы сразу видим (и это одна из причин их полезности), что $C1$ лучше $C2$. Почему так? Потому что $C1$ дает больше истинно положительных и меньше ложноположительных результатов, чем $C2$, то есть лучше в обоих отношениях. Иначе говоря, качество $C1$ лучше на *обоих* классах. Если один классификатор лучше другого на всех классах, то мы говорим, что первый *доминирует* над вторым¹. Однако не всегда все так просто. Рассмотрим третий классификатор $C3$, который лучше $C1$ на положительных объектах и хуже – на отрицательных

¹ Эта терминология заимствована из *многокритериальной оптимизации*. Доминируемое решение – то, что не лежит на *фронтне Парето*.

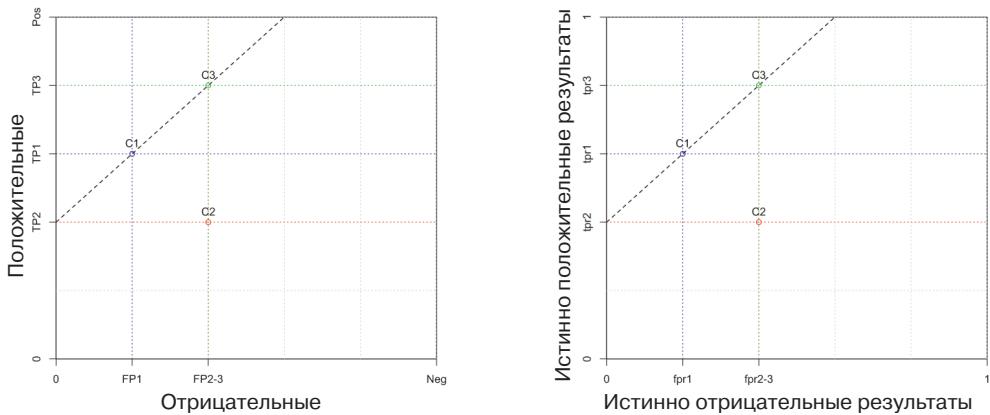


Рис. 2.3. (Слева) С1 и С3 доминируют над С2, но не доминируют друг над другом. Диагональная прямая показывает, что С1 и С3 достигают одинаковой верности. **(Справа)** Тот же график с нормированными осями. Его можно интерпретировать как результат объединения двух графиков покрытия на рис. 2.2 с использованием нормировки для учета различных распределений по классам. Теперь диагональная прямая показывает, что у С1 и С3 однааковая средняя полнота

(рис. 2.3 слева). Хотя и С1, и С3 доминируют над С2, ни один из них не доминирует над другим. Какой предпочтеть, зависит от того, что нам важнее: положительные или отрицательные результаты.

Можно уточнить эту мысль. Заметим, что для отрезка прямой, соединяющей С1 и С3, угловой коэффициент равен 1. Представим, что мы поднимаемся вверх по этой прямой: выигрывая в истинно положительных результатах, мы одновременно теряем в истинно отрицательных (или выигрываем в ложноположительных, что одно и то же). Но сумма истинно положительных и истинно отрицательных результатов остается одной и той же, а потому и верность не изменяется, пока мы остаемся на этой прямой. Отсюда следует, что верность С1 и С3 одинакова. *На графике покрытия классификаторы с одинаковой верностью соединены отрезками прямых с угловым коэффициентом 1.* Если истинно положительные и истинно отрицательные результаты одинаково важны, то все равно, какой из классификаторов С1 и С3 выбрать; если истинно положительные результаты важнее, то следует предпочесть С3; если же важнее истинно отрицательные результаты, то лучше выбрать С1.

Теперь рассмотрим правую часть рис. 2.3. Здесь я перенормировал оси, поделив значения на оси x на Neg, а значения на оси y – на Pos. В результате весь график оказался внутри единичного квадрата в системе координат, где по оси y отложена частота истинно положительных результатов, а по оси x – частота истинно отрицательных. В данном случае исходный график покрытия уже был квадратным ($Pos = Neg$), поэтому нормировка не повлияла на относительное расположение классификаторов. Однако, поскольку нормированный график будет квадратным вне зависимости от формы исходного, нормировка позволяет объ-

единять графики покрытия разной формы и, следовательно, объединять результаты, полученные на тестовых наборах с разным распределением по классам. Допустим, что мы нормировали классификатор на рис. 2.2 справа; поскольку частоты истинно- и ложноположительных результатов равны соответственно 80% и 40% (см. пример 2.1), то его положение на нормированном графике будет точно таким же, как у классификатора С3 на рис. 2.3 справа! Иными словами, классификаторы, занимающие разные точки в различных пространствах покрытия (например, С3 на рис. 2.2 справа и С3 на рис. 2.3 слева), на нормированном графике могут оказаться в одной точке.

В чем смысл диагональной прямой, которая соединяет классификаторы С1 и С3 на рис. 2.3 справа? Она не может иметь тот же смысл, что на графике покрытия, потому что после нормировки мы знаем частоты истинно- и ложноположительных результатов, но не знаем распределение по классам, а значит, не можем вычислить верность (вернитесь к формуле 2.3 на стр. 69, чтобы понять, почему). Эта прямая определяется уравнением $tpr = fpr + y_0$, где y_0 – точка ее пересечения с осью y . Теперь рассмотрим среднее арифметическое частот истинно положительных и истинно отрицательных результатов, которое назовем *средней полнотой* и обозначим *avg-rec*¹. На прямой с угловым коэффициентом 1 имеем $avg-rec = (tpr + tnr)/2 = (tpr + 1 - fpr)/2 = (1 + y_0)/2$, то есть постоянна. *На нормированном графике покрытия отрезки прямых с угловым коэффициентом 1 соединяют классификаторы с одинаковой средней полнотой*. Если полнота на положительных и отрицательных результатах одинаково важна, то все равно, какой из классификаторов С1 и С3 выбрать; если положительная полнота важнее, то следует предпочесть С3; если же важнее отрицательная полнота, то лучше выбрать С1.

В литературе нормированные графики покрытия называются *графиками РХП*, и мы далее будем следовать этой терминологии². Графики РХП распространены гораздо шире, чем графики покрытия, но у тех и других есть свои специфические применения. Вообще говоря, график покрытия следует использовать, если требуется явно учесть распределение по классам, например при работе с одиночным набором данных. График РХП полезен, когда требуется объединить результаты обработки разных наборов данных с разными распределениями по классам. Очевидно, что оба понятия тесно связаны. Поскольку график РХП всегда квадратный, линии постоянной средней полноты (так называемые *изолинии* средней полноты) не просто имеют угловой коэффициент 1, а параллельны диагонали, идущей из левого нижнего в правый верхний угол. Последнее свойство переносится и на графики покрытия. Для иллюстрации на графике покрытия, изображенном на рис. 2.4, классификаторы С1 и С2 характеризуются одинаковой

¹ Напомним, что полнота – это просто другое название частоты истинно положительных результатов, а отрицательная полнота – то же самое, что частота истинно отрицательных результатов. Таким образом, средняя полнота – это среднее арифметическое положительной и отрицательной полноты. Иногда ее называют *макроусредненной верностью*.

² РХП расшифровывается как «*рабочая характеристика приемника*» (англ. ROC) – термин, заимствованный из *теории обнаружения сигналов*.

верностью (соединены отрезком прямой с угловым коэффициентом 1), а C1 и C3 – одинаковой средней полнотой (соединены отрезком прямой, параллельной диагонали). Можно заметить, что у C2 верность и средняя полнота больше, чем у C3 (почему?). На соответствующем графике РХП изолинии средней полноты имеют угловой коэффициент 1, а изолинии верности – угловой коэффициент $Neg/Pos = 1/clr$.

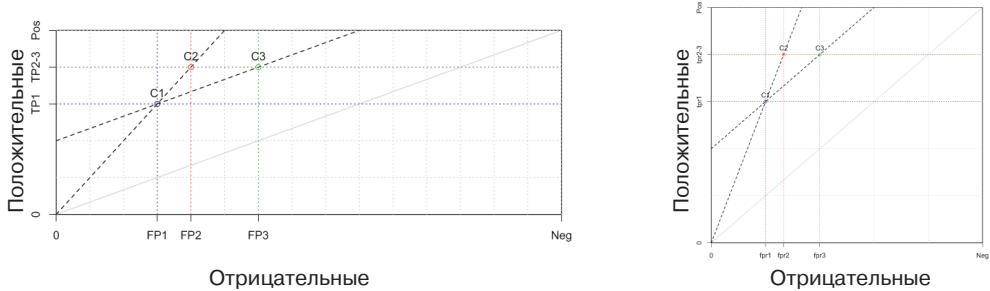


Рис. 2.4. (Слева) На графике покрытия изолинии верности имеют угловой коэффициент 1, а изолинии средней полноты параллельны диагонали. **(Справа)** На соответствующем графике РХП изолинии средней полноты имеют угловой коэффициент 1, изолинии верности в данном случае имеют угловой коэффициент 3, что соответствует отношению числа отрицательных примеров к числу положительных в наборе данных

2.2 Оценивание и ранжирование

Многие классификаторы вычисляют оценки, на основе которых предсказывают класс. Так, в прологе мы видели, что SpamAssassin вычисляет взвешенную сумму по правилам, «срабатывавшим» для конкретного сообщения. Такие оценки содержат дополнительную информацию, которая может быть полезна для разных целей, поэтому мы рассматриваем оценивание как самостоятельную задачу. Формально *оценивающим классификатором* называется отображение $\hat{s}: \mathcal{X} \rightarrow \mathbb{R}^k$, то есть отображение пространства объектов к k -мерное вещественное пространство. Полужирный шрифт призван подчеркнуть, что оценивающий классификатор порождает вектор $\hat{s}(x) = (\hat{s}_1(x), \dots, \hat{s}_k(x))$, а не одно число; $\hat{s}_i(x)$ – оценка, назначенная классу C_i для объекта x . Эта оценка показывает, насколько вероятна метка класса C_i . Если есть только два класса, то обычно достаточно вычислить оценку лишь для одного из них; в данном случае $\hat{s}(x)$ обозначает оценку положительного класса для экземпляра x .

На рис. 2.5 показано, как можно преобразовать дерево признаков в оценивающее дерево. Чтобы получить оценку для каждого листа, мы сначала вычисляем отношение спама к неспаму, оно равно 1/2 для левого листа, 2 для среднего и 4 для правого. Поскольку часто удобнее работать с аддитивной шкалой, то в ка-

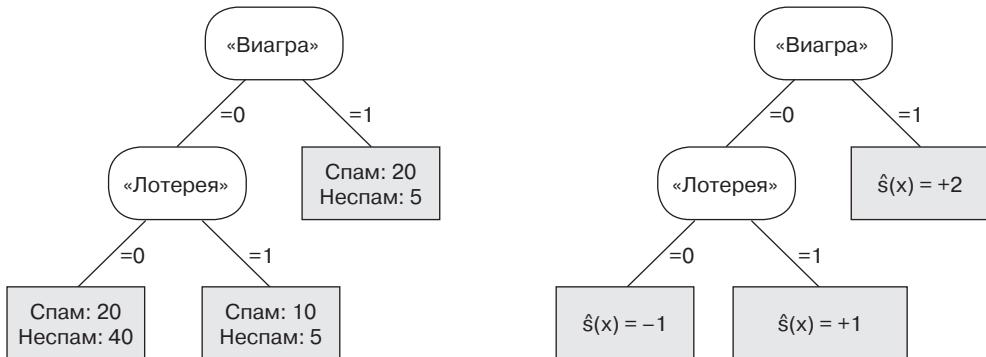


Рис. 2.5. (Слева) Дерево признаков, в листьях которого показано распределение обучающего набора по классам. **(Справа)** Оценивающее дерево, в котором в качестве оценок используются логарифмы отношения классов; спам считается положительным классом

чество оценок мы берем логарифм отношения классов (основание логарифма несущественно, мы взяли в качестве основания 2, чтобы получить красивые целые числа). Отметим, что решающее дерево с правилом мажоритарного класса соответствует выбору нулевого порогового значения $\hat{s}(x)$, то есть мы определяем сообщение как спам, если $\hat{s}(x) > 0$ и как неспам в противном случае.

Если в качестве истинного класса $c(x)$ взять $+1$ для положительных примеров и -1 для отрицательных, то величина $z(x) = c(x)\hat{s}(x)$ будет положительна для правильных предсказаний и отрицательна для неправильных; эта величина называется **зазором**, назначаемым оценивающим классификатором примеру¹. Мы хотели бы вознаграждать за большие положительные зазоры и штрафовать за большие отрицательные. Это достигается с помощью так называемой **функции потерь** $L: \mathbb{R} \mapsto [0, \infty)$, которая отображает зазор каждого примера $z(x)$ в ассоциированные с ним потери $L(z(x))$. Мы будем предполагать, что $L(0) = 1$, это потери, возникающие, когда пример находится на решающей границе. Кроме того, $L(z) \geq 1$ для $z < 0$ и обычно $0 \leq L(z) < 1$ для $z > 0$ (рис. 2.6). Средние потери по тестовому набору T_e определяются по формуле $\frac{1}{|T_e|} \sum_{x \in T_e} L(z(x))$.

Простейшая функция потерь, которую иногда называют **функцией потерь типа 0–1**, определяется как $L_{01}(z) = 1$, если $z \leq 0$, и $L(z) = 0$, если $z > 0$. Усреднение функций потерь типа 0–1 – это просто доля неправильно классифицированных тестовых примеров:

$$\frac{1}{|T_e|} \sum_{x \in T_e} L_{01}(z(x)) = \frac{1}{|T_e|} \sum_{x \in T_e} I[c(x)\hat{s}(x) \leq 0] = \frac{1}{|T_e|} \sum_{x \in T_e} I[c(x) \neq \hat{c}(x)] = err,$$

¹ Напомним, что в главе 1 мы называли зазором классификатора расстояние между границей решений и ближайшим примером. Здесь зазор используется в несколько более общем смысле: зазор есть у каждого примера, а не только у ближайшего. Дополнительные пояснения будут даны в разделе 7.3.

где $\hat{c}(x) = +1$, если $\hat{s}(x) > 0$, $\hat{c}(x) = 0$, если $\hat{s}(x) = 0$, и $\hat{c}(x) = -1$, если $\hat{s}(x) < 0$. (Иногда удобнее определять потери примеров на решающей границе как $1/2$.) Другими словами, функция потерь типа 0–1 игнорирует величину зазоров примеров, а принимает во внимание только их знак. Поэтому она не различает оценивающие классификаторы, коль скоро их предсказания согласуются. Это означает, что она не слишком полезна в качестве поисковой эвристики или целевой функции при обучении оценивающих классификаторов. На рис. 2.6 показано несколько функций потерь, применяемых на практике. За исключением функции потерь типа 0–1, все они *выпуклые*: отрезок, соединяющий любые две точки на кривой, целиком лежит выше кривой. С вычислительной точки зрения оптимизировать выпуклую функцию легче.

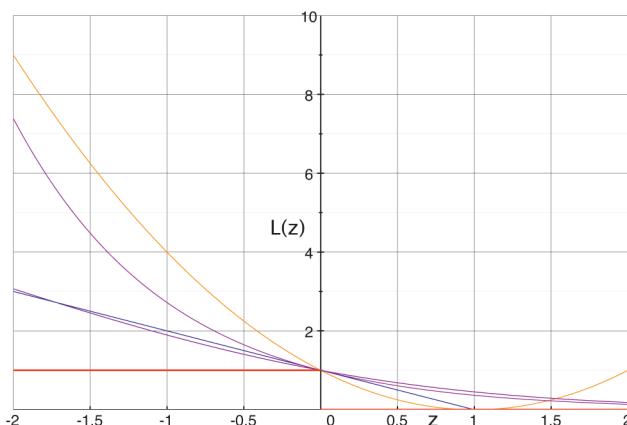


Рис. 2.6. Функции потерь. Начиная с нижней левой: (i) функция потерь типа 0–1 $L_{01}(z) = 1$, если $z \leq 0$, и $L_{01}(z) = 0$, если $z > 0$; (ii) кусочно-линейная функция потерь $L_h(z) = (1-z)$, если $z \leq 1$, и $L_h(z) = 0$, если $z > 1$; (iii) логарифмическая функция потерь $L_{\log}(z) = \log_2(1 + \exp(-z))$; (iv) экспоненциальная функция потерь $L_{\exp}(z) = \exp(-z)$; (v) квадратичная функция потерь $L_{\text{sq}}(z) = (1-z)^2$ (можно положить равно 1 при $z > 1$, как в случае кусочно-линейной функции)

Особый интерес для нас будет представлять *кусочно-линейная функция потерь*, определяемая в виде $L_h(z) = (1-z)$, если $z \leq 1$, и $L_h(z) = 0$, если $z > 1$. Своим английским названием (hinge loss) она обязана тому факту, что потери зависят (hinge) от того, больше зазор примера 1 или нет: если больше (то есть пример лежит по правильную сторону от решающей границы на расстоянии не меньше 1), то связанные с ним потери равны нулю, в противном случае потери тем больше, чем меньше зазор. По сути дела, эта функция потерь означает, что следует избегать примеров с зазором, намного меньшим единицы, но большие положительные зазоры при этом не вознаграждаются. Эта функция применяется при обучении [метода опорных векторов](#) (раздел 7.3). Ниже, при обсуждении [усиления](#) в разделе 11.2, нам встретится также *экспоненциальная функция потерь*.

Оценка и визуализация качества ранжирования

Следует понимать, что оценки назначаются классификатором, а не являются внутренним свойством объектов. Оценки не выводятся из «истинных оценок» – вместо этого оценивающий классификатор необходимо обучить на примерах в виде объектов x , помеченных классами $c(x)$, точно так же, как обычный классификатор. (Задача, в которой мы строим функцию \hat{f} в результате обучения на примерах, помеченных истинными значениями функции $(x, f(x))$, называется *регрессией* и рассматривается в разделе 3.2.) Часто удобнее ввести на множестве объектов порядок, индуцированный оценками, но игнорировать их величины; одним из преимуществ такого подхода является резкое уменьшение чувствительности к выбросам. Это также означает, что нам не нужно делать никаких предположений относительно масштабной шкалы оценок; в частности, классификатор не устанавливает какой-то пороговой оценки для различения положительных и отрицательных примеров. Под *ранжированием* понимается определение полного порядка на множестве объектов, возможно, с неопределенностями¹.

Пример 2.2 (пример ранжирования). Оценивающее дерево на рис. 2.5 порождает следующее ранжирование: $[20+, 5-][10+, 5-][20+, 40-]$. Здесь $20+$ обозначает последовательность из 20 положительных примеров, а между экземплярами в квадратных скобках [...] имеется неопределенность. Путем выбора точки разделения в ранжировании мы можем преобразовать ранжирование в классификацию. В данном случае есть четыре возможности: (А) поставить точку разделения до первого сегмента и тем самым отнести все сегменты к отрицательному классу; (Б) отнести первый сегмент к положительному классу, а остальные два – к отрицательному; (В) отнести первые два сегмента к положительному классу; (Г) отнести все сегменты к положительному классу. В терминах самих оценок получаем такое соответствие: (А) выбор любой оценки, большей 2, в качестве порогового значения; (Б) выбор порогового значения между 1 и 2; (В) пороговое значение выбирается между -1 и 1; (Г) пороговое значение меньше -1 .

Пусть x и x' – два объекта, причем оценка x меньше: $\hat{s}(x) < \hat{s}(x')$. Поскольку более высокая оценка означает большую уверенность в том, что рассматриваемый объект положительный, это нормально во всех случаях, кроме одного: если x на самом деле положителен, а x' отрицателен. Эту ситуацию мы называем *ошибкой ранжирования*. Тогда общее число ошибок ранжирования можно выразить в виде $\sum_{x \in Te^{\oplus}, x' \in Te^{\ominus}} I[\hat{s}(x) < \hat{s}(x')]$. Для всех положительных и отрицательных примеров, получивших одну и ту же оценку – *неопределенность*, ошибка ранжирования уменьшается наполовину. Максимальное число ошибок ранжирования равно $|Te^{\oplus}| \cdot |Te^{\ominus}| = Pos \cdot Neg$, поэтому *частота ошибок ранжирования* определяется следующим образом:

¹ Полный порядок с неопределенностями не следует путать с частичным порядком (см. замечание 2.1). При наличии полного порядка с неопределенностями (по существу, это полный порядок на классах эквивалентности) любые два элемента сравнимы, в каком-то одном или в обоих направлениях. В случае частичного порядка некоторые элементы несравнимы.

$$rank - err = \frac{\sum_{x \in T_e^{\oplus}, x' \in T_e^{\ominus}} I[\hat{s}(x) < \hat{s}(x')] + \frac{1}{2} I[\hat{s}(x) = \hat{s}(x')]}{Pos \cdot Neg}, \quad (2.4)$$

и аналогично для *верности ранжирования*:

$$rank - acc = \frac{\sum_{x \in T_e^{\oplus}, x' \in T_e^{\ominus}} I[\hat{s}(x) > \hat{s}(x')] + \frac{1}{2} I[\hat{s}(x) = \hat{s}(x')]}{Pos \cdot Neg} = 1 - rank - err. \quad (2.5)$$

Верность ранжирования можно рассматривать как оценку вероятности того, что произвольно взятая пара положительный–отрицательный ранжирована правильно.

Пример 2.3 (верность ранжирования). Продолжим рассмотрение оценивающего дерева на рис. 2.5, в котором левый лист включает 20 спамных и 40 хороших сообщений, средний лист – 10 спамных и 5 хороших сообщений, а правый лист – 20 спамных и 5 хороших. Пять отрицательных примеров в правом листе имеют оценку выше, чем 10 положительных в среднем листе и 20 положительных в левом листе, что дает $50 + 100 = 150$ ошибок ранжирования. Пять отрицательных примеров в среднем листе имеют оценку выше, чем 20 положительных в левом листе, что дает еще 100 ошибок ранжирования. Кроме того, левый лист привносит 800 половинных ошибок ранжирования (потому что 20 положительных и 40 отрицательных примеров получили одну и ту же оценку), средний лист – 50, и правый лист – 100. Итого мы получаем 725 ошибок ранжирования из $50 \cdot 50 = 2500$ возможных, поэтому частота ошибок ранжирования составляет 29%, а верность ранжирования – соответственно, 71%.

Графики покрытия и графики РХП, введенные в предыдущем разделе для наглядного представления о качестве классификатора, являются отличным инструментом также и для визуализации качества ранжирования. Если вдоль вертикальной и горизонтальной оси отложить соответственно *Pos* положительных и *Neg* отрицательных примеров, то каждой паре положительный–отрицательный будет соответствовать одна «клетка» на графике. Если упорядочить положительные и отрицательные примеры по убыванию оценки, то есть примеры с более высокими оценками расположить ближе к началу координат, то можно будет четко различить правильно ранжированные пары в правом нижнем углу, ошибки ранжирования в левом верхнем углу и неопределенности между ними (рис. 2.7). Количество клеток в каждой области дает нам количество правильно ранжированных пар, ошибок ранжирования и неопределенностей соответственно. Диагонали рассекают область неопределенности на две половины, так что площадь под диагоналями соответствует верности ранжирования, умноженной на *Pos* · *Neg*, а площадь над диагоналями – частоте ошибок ранжирования, умноженной на тот же коэффициент.

Оставив только диагонали, мы получим кусочно-линейную функцию, изображенную на рис. 2.7 справа, – *кривую покрытия*. Ее можно интерпретировать

следующим образом. Каждая из точек A, B, C, D определяет качество классификации в терминах истинно- и ложноположительных результатов, достигаемое в результате установки точек деления ранжирования или пороговых оценок из примера 2.2. В частности, точка С была бы получена при выборе пороговой оценки 0, что дает $TP_2 = 20 + 10 = 30$ истинно положительных результатов и $FP_2 = 5 + 5 = 10$ ложноположительных. Аналогично точка B была бы получена при более высоком пороге 1.5, что дает $TP_1 = 20$ истинно положительных результатов и $FP_1 = 5$ ложноположительных. Точка A получилась бы, если бы установили недостижимо высокий порог, а точка D – если бы порог был trivialно низким.

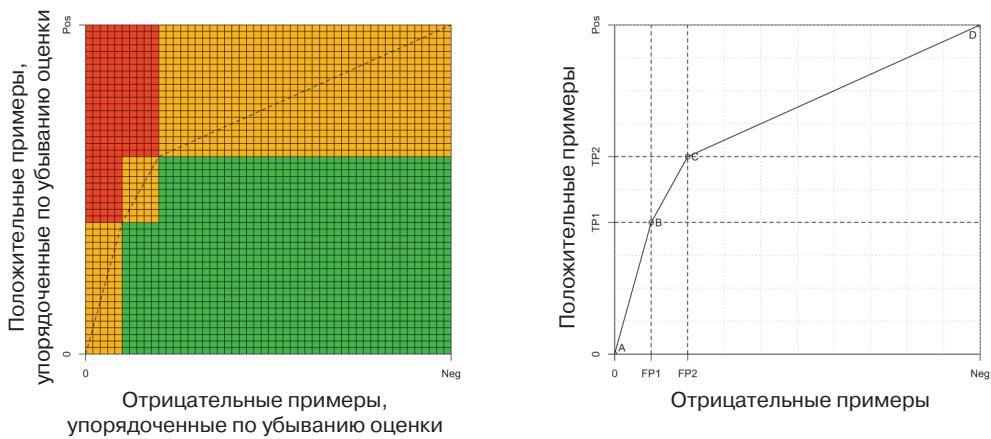


Рис. 2.7. (Слева) Каждая клетка обозначает одну пару, состоящую из положительного и отрицательного примера: зеленые клетки соответствуют парам, которые правильно ранжированы классификатором, красные – ошибкам ранжирования, а оранжевые – полушибкам из-за неопределенности. **(Справа)** Кривая покрытия древовидного оценивающего классификатора, в которой каждый отрезок прямой соответствует одному листу дерева, а каждая пара (FP , TP) – возможной пороговой оценке

Почему эти точки соединены отрезками прямых? Как произвести интерполяцию, например, между точками С и D? Предположим, что пороговое значение в точности равно -1 , то есть оценке, называемой левым листом дерева. Вопрос: какой класс следует предсказать для 20 положительных и 40 отрицательных примеров, оказавшихся в этом листе? Разумно было бы принять решение, подбросив правильную монету: тогда половина положительных примеров (в среднем) будет отнесена к положительному классу, а половина – к отрицательному, и точно так же для отрицательных примеров. Таким образом, общее число истинно положительных результатов равно $30 + 20/2 = 40$, а ложноположительных – $10 + 40/2 = 30$. Иными словами, мы оказываемся точно в середине отрезка CD. Такую же процедуру можно применить для достижения эффективности, соответствующей середине отрезка BC: установить порог равным 1 и подбросить правильную монету, чтобы получить равномерно распределенные предсказания для 10 положитель-

ных и 5 отрицательных примеров в среднем листе, что дает $20 + 10/2 = 25$ истинно положительных и $5 + 5/2 = 7.5$ ложноположительных результатов (разумеется, в реальном испытании невозможно получить нецелое число ложноположительных результатов, это лишь математическое ожидание количества ложноположительных результатов при большом числе испытаний). Более того, если взять неправильную монету, сместив вероятность выпадения положительного или отрицательного предсказания, то можно будет получить ожидаемую эффективность, соответствующую любой точке отрезка.

Вообще говоря, кривая покрытия – это произвольная кусочно-линейная функция, монотонно возрастающая от $(0,0)$ до (Neg, Pos) , то есть TP и FP не могут убывать при уменьшении порога принятия решения. Каждый отрезок кривой соответствует классу эквивалентности в разбиении пространства объектов, индуцированном рассматриваемой моделью (то есть листьями дерева признаков). Отметим, что количество отрезков не может быть больше количества тестовых примеров. Угловой коэффициент каждого отрезка равен отношению числа положительных примеров к числу отрицательных в соответствующем классе эквивалентности. Например, в нашем примере угловой коэффициент первого отрезка равен 4, второго – 2, а третьего – $1/2$; это не что иное, как оценки, назначенные каждым листом дерева! В общем случае это не так, потому что кривая покрытия зависит только от ранжирования, индуцированного оценками, а не от самих оценок. Но это и не случайное совпадение, как мы убедимся в следующем разделе, посвященном оцениванию вероятности класса.

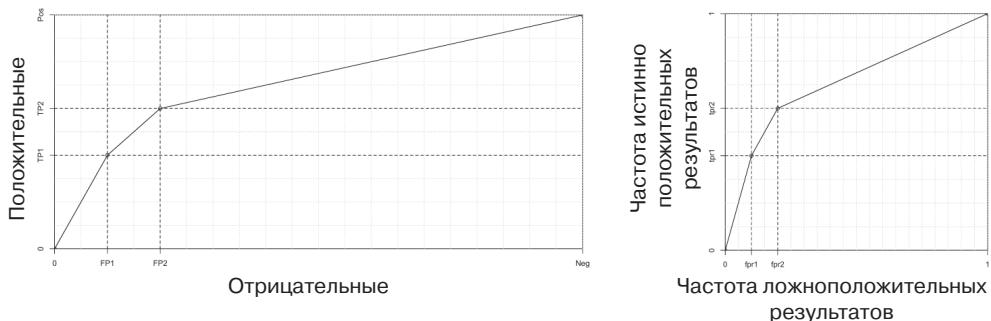


Рис. 2.8. (Слева) Кривая покрытия, полученная на основе тестового набора с отношением классов $clr = 1/2$. **(Справа)** Соответствующая ей кривая РХП – такая же, как для кривой покрытия на рис. 2.7 справа

Кривая РХП получается из кривой покрытия нормированием осей на $[0,1]$. В нашем примере особой разницы нет, но в общем случае кривые покрытия могут быть прямоугольными, тогда как кривые РХП всегда располагаются в единичном квадрате. Это, в частности, приводит к умножению угловых коэффициентов на $Neg/Pos = 1/clr$. Кроме того, если на графике покрытия площадь под кривой покрытия равна абсолютному числу правильно ранжированных пар, то

на графике РХП *площадь под кривой РХП равна верности ранжирования*, определенной по формуле 2.5. Поэтому часто встречается аббревиатура *AUC*, означающая «площадь под кривой (РХП)»; далее я тоже буду употреблять ее.

Пример 2.4 (дисбаланс классов). Предположим, что мы подали на вход оценивающему дереву на рис. 2.5 расширенный тестовый набор, содержащий еще 50 негативных примеров. Так случилось, что они совпали с исходными примерами, поэтому в результате количество отрицательных примеров в каждом листовом узле удвоилось. Это привело к изменению кривой покрытия (поскольку изменилось отношение классов), но кривая РХП осталась такой же (рис. 2.8). Отметим, что и AUC не изменилась: хотя классификатор делает вдвое больше ошибок ранжирования, но и количество пар положительный–отрицательный тоже удвоилось, так что частота ошибок ранжирования осталась такой же.

Рассмотрим теперь пример кривой покрытия для ранжирующего классификатора. На рис. 2.9 слева показан результат применения линейного классификатора (решающая граница обозначена В) к небольшому набору данных, содержащему пять положительных и пять отрицательных примеров. Он достигает верности 0.80. Мы можем вывести из этого линейного классификатора оценку, взяв расстояние от примера до решающей границы; если пример находится с отрицательной стороны, то расстояние берется со знаком минус. Это означает, что примеры будут ранжированы в следующем порядке: $p_1 - p_2 - p_3 - n_1 - p_4 - n_2 - n_3 - p_5 - n_4 - n_5$. При таком ранжировании возникают четыре ошибки ранжирования: n_1 раньше p_4 и n_1 , n_2 , n_3 раньше p_5 . На рис. 2.9 справа эти ошибки ранжирования представлены в левом верхнем углу. AUC для такого ранжирования равна $21/25 = 0.84$.

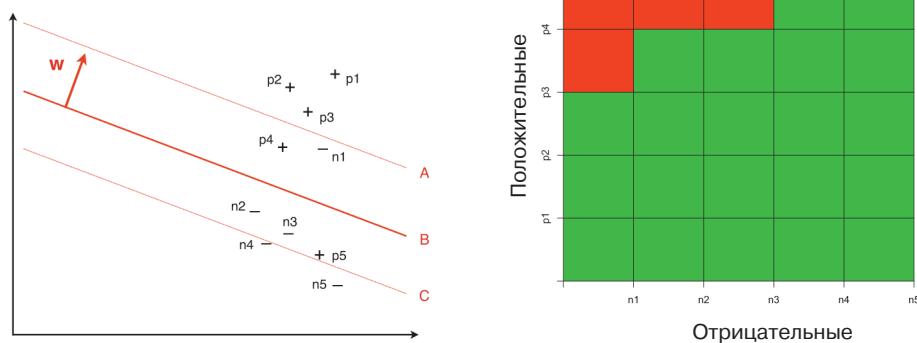


Рис. 2.9. (Слева) Линейный классификатор индуцирует ранжирование, если в качестве оценки взять расстояние со знаком до решающей границы. Это ранжирование зависит только от ориентации решающей границы: все три прямые приводят к одинаковому ранжированию. **(Справа)** Сетка правильно ранжированных пар положительный–отрицательный (зеленые клетки) и ошибок ранжирования (красные клетки)

На основании этой сетки получаем кривую покрытия, изображенную на рис. 2.10. Из-за ступенчатости она выглядит совсем не так, как кривые покрытия для оценивающих деревьев, рассмотренные выше в этом разделе. Основная причина – отсутствие неопределенностей, вследствие чего все отрезки кривой либо горизонтальны, либо вертикальны, а количество отрезков совпадает с количеством примеров. Такую ступенчатую кривую можно сгенерировать из ранжирования следующим образом: начиная с левого нижнего угла, мы сдвигаемся на один шаг вверх, если следующий пример в ранжировании положительный, и на один шаг вправо, если он отрицательный. В результате получается кривая, состоящая из трех шагов вверх (для $p1-3$), одного шага вправо (для $n1$), одного шага вверх ($p4$), двух шагов вправо ($n2-3$), одного шага вверх ($p5$) и, наконец, двух шагов вправо ($n4-5$).

Такую же процедуру можно использовать и для группирующих моделей, если обрабатывать неопределенностии следующим образом: в случае неопределенностей между p положительными и n отрицательными примерами идем на p шагов вверх и одновременно на n шагов вправо. Взглянув на рис. 2.7, вы увидите, что именно так и построены диагональные отрезки, проходящие внутри оранжевых прямоугольников, которые возникают из-за наличия неопределенностей в листьях решающего дерева. Таким образом, принципы, лежащие в основе кривых покрытия и кривых РХП, одинаковы для группирующих и ранжирующих моделей, но сами кривые выглядят по-разному. *В случае группирующей модели количество отрезков в кривой РХП совпадает с количеством сегментов пространства объектов в модели, а в случае ранжирующей модели имеется по одному отрезку для каждого примера в наборе данных.* Это конкретное проявление положения, упомянутого в прологе: «разрешающая способность» ранжирующих моделей гораздо выше, чем у группирующих; иногда это называют также *уточнением* модели.

Обратите внимание на три точки А, В и С на рис. 2.10. Они обозначают качество, достижаемое при решающих границах, помеченных соответствующими буквами на рис. 2.9. Так, средняя граница В неправильно классифицирует один из пяти положительных примеров ($tpr = 0.80$) и один из пяти отрицательных ($fpr = 0.80$). Граница А правильно классифицирует все отрицательные примеры, а граница С – все положительные. На самом деле все они должны иметь одинаковую ориентацию, но их точное положение несущественно при условии, что граница А расположена между $p3$ и $n1$, граница В – между $p4$ и $n2$, и граница С – между $p5$ и $n4$. Как мы скоро увидим, для такого выбора этих трех границ существуют веские причины. А пока посмотрим, что произойдет, если мы воспользуемся этими границами для преобразования линейной модели в группирующую модель с четырьмя сегментами: область над А, область между А и В, кусочек между В и С и все остальное под С. В результате мы больше не отличаем $n1$ от $p4$, а также $n2-3$ от $p5$. После внесенных таким образом неопределенностей кривая покрытия принимает вид, изображенный пунктирной линией на рис. 2.10. Отметим, что AUC в результате оказывается больше 0.90. Следовательно, *уменьшая уточнение модели, мы иногда можем достичь лучшего качества ранжирования.*

Обучение модели предполагает не только усиление значимых различий, но и уменьшение эффекта различий, вводящих в заблуждение.

Преобразование ранжировщика в классификатор

Выше я уже отмечал, что основное различие между ранжировщиками и оценивающими классификаторами заключается в том, что ранжировщик предполагает лишь, что более высокая оценка означает более сильное свидетельство в пользу положительного класса, но не делает никаких других предположений ни о шкале оценок, ни о том, какая пороговая оценка позволила бы лучше разделить положительные и отрицательные примеры. Теперь мы рассмотрим, как получить такое пороговое значение по кривой покрытия или кривой РХП.

Основным понятием здесь являются изолинии верности. Напомним, что на графике покрытия точки с одинаковой верностью лежат на прямых с угловым коэффициентом 1. Поэтому нам нужно лишь провести прямую с угловым коэффициентом 1 через левую верхнюю точку (иногда ее называют *вершиной РХП* (ROC heaven)) и опускать ее вниз, пока она не коснется кривой покрытия в одной или нескольких точках. В каждой из этих точек достигается максимально возможная для данной модели верность. На рис. 2.10 этот метод нашел бы точки А и В, которые и являются точками наибольшей верности (0.80). Но способы достижения максимума в них разные, например модель А более консервативна на положительных примерах.

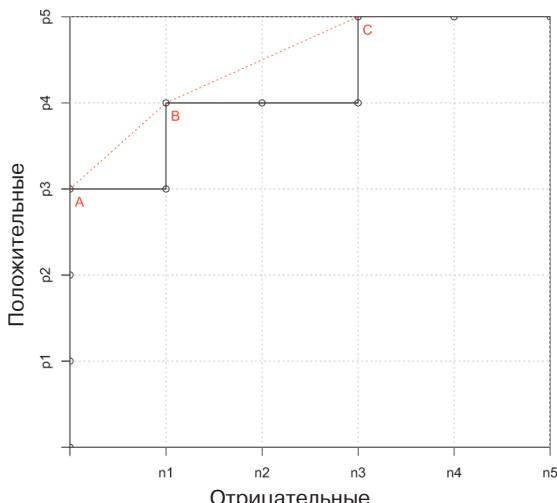


Рис. 2.10. Кривая покрытия линейного классификатора, изображенного на рис. 2.9. Точки А, В и С обозначают качество классификации на соответствующих решающих границах. Пунктирные линии показывают улучшение, которого можно достичь, преобразовав ранжирующий классификатор в группирующий с четырьмя сегментами

Аналогичную процедуру можно проделать для графиков РХП, нужно только помнить, что все угловые коэффициенты следует умножать на обратное отношение классов $1/clr = Neg/Pos$.

Пример 2.5 (настройка фильтра спама). Итак, вы тщательно обучили свой байесовский фильтр спама, и осталось только задать порог принятия решения. Вы делаете выборку из шести спамных и четырех неспамных сообщений и собираете оценки, вычисленные фильтром. Вот они в порядке убывания: 0.89 (спам), 0.80 (спам), 0.74 (неспам), 0.71 (спам), 0.63 (спам), 0.49 (неспам), 0.42 (спам), 0.32 (спам), 0.24 (неспам), 0.13 (неспам). Если отношение классов 3 (спам) к 2 (неспам)reprезентативно, то можно выбрать оптимальную точку на кривой РХП, воспользовавшись изолинией с угловым коэффициентом 2/3. Как видно из рис. 2.11, решающая граница при этом проходит между шестым спамным и третьим неспамным сообщениями, и в качестве порога принятия решения мы можем взять среднее арифметическое их оценок (0.28).

Альтернативный способ поиска оптимальной точки состоит в том, чтобы перебрать все возможные точки разделения, начиная с той, что предшествует сообщению с наибольшим рангом, и заканчивая следующей за сообщением с наименьшим рангом, и вычислить количество правильно классифицированных примеров в каждой такой точке: 4 – 5 – 6 – 5 – 6 – 7 – 6 – 7 – 8 – 7 – 6. Максимум достигается в той же точке разделения, что и выше, и дает верность 0.80. Существует полезный прием для определения того, какая верность соответствует изолинии на графике РХП, – найти точку пересечения изолинии с нисходящей диагональю. Поскольку верность – это взвешенное среднее частот истинно положительных и истинно отрицательных результатов и поскольку в точке на нисходящей диагонали эти частоты совпадают, то для получения верности можно просто взять ординату.

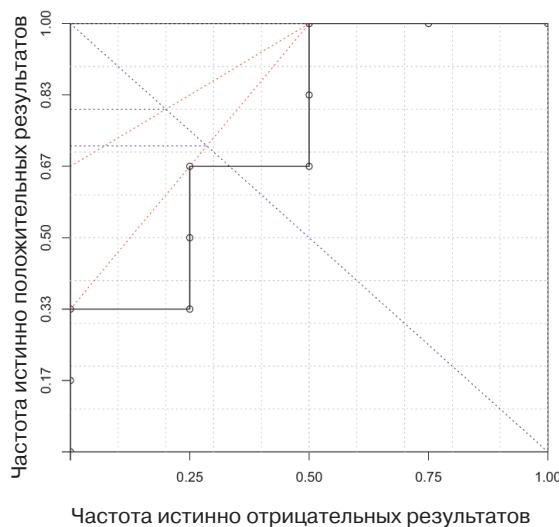


Рис. 2.11. Выбор оптимальной точки на кривой РХП. Верхняя пунктирная прямая – это изолиния верности с угловым коэффициентом 2/3. Нижняя изолиния удваивает ценность (или встречаемость) отрицательных примеров и допускает выбор пороговых значений. Пересекая изолинии с нисходящей диагональю, мы можем прочитать достигнутую верность с оси y .

Если распределение данных по классам *не* репрезентативно, то можно просто подкорректировать угловой коэффициент изолинии. Например, если неспам встречается в два раза чаще, то мы возьмем изолинию с угловым коэффициентом 4/3. В предыдущем примере это даст три оптимальные точки на кривой РХП¹. Даже если отношение классов в данных репрезентативно, могут быть иные причины назначить классам другие веса. Например, если наш фильтр спама отбрасывает ложноположительные результаты (хорошие сообщения, неправильно классифицированные как спам), то может возникнуть желание снизить частоту ложноположительных результатов, назначив более высокий вес отрицательным примерам (неспаму). Это часто выражают в виде *отношения затрат* $c = c_{FN}/c_{FP}$, то есть отношения стоимости ложноотрицательных результатов к стоимости ложноположительных; в данном случае ему нужно присвоить значение меньше 1. Тогда релевантные изолинии будут иметь угловой коэффициент $1/c$ на графике покрытия и коэффициент $1/(c \cdot clr)$ на графике РХП. Сочетание отношения затрат и отношения классов дает точный контекст, в котором будет развернут классификатор, и называется *режимом работы*.

Если отношение классов или отношение затрат очень асимметрично, то эта процедура может породить классификатор, который относит все примеры к одному классу. Так, если отрицательные примеры встречаются в 1000 раз чаще положительных, то изолинии верности почти вертикальны, что приводит к недостижимо высокому порогу принятия решения и классификатору, который считает все примеры отрицательными. Наоборот, если стоимость одного истинно положительного результата в 1000 раз превышает стоимость ложноположительного, то мы будем классифицировать все примеры как положительные – на самом деле именно этот принцип несет ответственность за пропуск спамной почты! Однако часто такое одинаковое поведение на все случаи жизни неприемлемо, а значит, верность – не тот параметр, который следует оптимизировать. В таких случаях лучше использовать изолинии средней полноты. Они параллельны восходящей диагонали на графиках покрытия и РХП и позволяют добиться схожего качества для обоих классов.

Только что описанная процедура находит порог принятия решения по размеченым данным с помощью кривой РХП и подходящей изолинии верности. Часто эта процедура предпочтительнее априорного задания порога, особенно если оценки выражены в произвольной шкале; например, это позволило бы точно настроить порог принятия решения для фильтра SpamAssassin с учетом конкретной ситуации и наших предпочтений. Даже если в роли оценок выступают вероятности, как в следующем разделе, они могут быть вычислены недостаточно хорошо, чтобы считать порог 0.5 правильным.

¹ Представляется разумным выбрать среднюю из трех точек, что дает пороговое значение 0.56. Альтернатива – рассматривать все сообщения с оценкой из интервала [0.28, 0.77] как лежащие на границе решений и назначать им класс случайным образом.

2.3 Оценивание вероятностей классов

Оценка вероятностей классов, или просто оценка вероятностей, – это оценивающий классификатор, который порождает вектор вероятностей классов, то есть отображение $\hat{p} : \mathcal{X} \rightarrow [0,1]^k$. Мы пишем $\hat{p}(x) = (\hat{p}_1(x), \dots, \hat{p}_k(x))$, где $\hat{p}_i(x)$ – вероятность, что объект x принадлежит классу C_i , и $\sum_{i=1}^k \hat{p}_i(x) = 1$. Если есть всего два класса, то вероятность, ассоциированная с одним из них, равна 1 минус вероятность, ассоциированная с другим; в таком случае мы обозначаем $\hat{p}(x)$ оценку вероятности положительного класса для объекта x . Как и в случае оценивающих классификаторов, истинные вероятности $p_i(x)$ обычно неизвестны. Одна из интерпретаций вероятностей $\hat{p}_i(x)$ – оценки вероятности $P_C(c(x') = C_i | x' \sim x)$, где $x' \sim x$ означает « x' похож на x ». Иначе говоря, как часто объекты этого класса встречаются среди объектов, похожих на x ? Интуиция подсказывает, что чем чаще (или реже) они встречаются, тем больше (или меньше) наша уверенность в том, что x также принадлежит этому классу. Что в этом контексте понимать под похожестью, зависит от рассматриваемой модели – здесь мы проиллюстрируем идею на нескольких примерах с двумя классами. Сначала рассмотрим ситуацию, когда любые два объекта похожи друг на друга. Тогда $P_C(c(x') = \oplus | x' \sim x) = P_C(c(x') = \oplus)$, а эта величина оценивается просто долей *pos* положительных примеров в нашем наборе данных (дальше я буду опускать нижний индекс C). Иными словами, в этом сценарии мы предсказываем, что $\hat{p}(x) = pos$ вне зависимости от того, знаем мы что-нибудь об истинном классе x или нет. В качестве другого крайнего случая рассмотрим ситуацию, когда никакие два объекта не похожи друг на друга, если только не являются одним и тем же объектом, то есть $x' \sim x$, если $x' = x$, иначе $x' \not\sim x$. В этом случае $P(c(x') = \oplus | x' \sim x) = P(c(x) = \oplus)$, а эта величина – поскольку x фиксирован – равна 1, если $c(x) = \oplus$, и 0 в противном случае. Иначе говоря, мы предсказываем $\hat{p}(x) = 1$ для всех известных положительных примеров и $\hat{p}(x) = 0$ для всех известных отрицательных, но обобщить предсказание на объекты, которых раньше не видели, не можем.

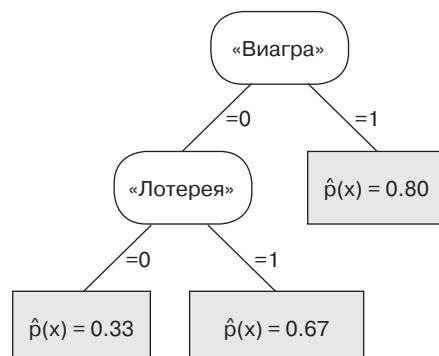


Рис. 2.12. Дерево оценивания вероятностей, построенное на основе дерева признаков на рис. 1.4

Дерево признаков позволяет найти баланс между этими крайними упрощенными случаями, взяв в качестве отношения похожести \sim_T следующее отображение, ассоциированное с деревом: $T: x' \sim_T x$ тогда и только тогда, когда x' и x приписаны одному и тому же листовому узлу. Тогда в каждом листе мы в качестве оценки вероятности берем долю положительных примеров, оказавшихся в этом листе. Например, в правом листе на рис. 1.4 доля положительных примеров равна $40/50 = 0.80$, поэтому мы предсказываем $\hat{p}(x) = 0.80$ для всех объектов x в этом листе и аналогично для двух других листьев (рис. 2.12). Если установить порог $\hat{p}(x)$ равным 0.5 (то есть предсказывать спам, если вероятность спама равна 0.5 или выше, а в противном случае – неспам), то получится тот же классификатор, что в результате предсказания по мажоритарному классу в каждом листе дерева признаков.

Качество оценивания вероятностей классов

Как и в случае классификаторов, мы теперь можем задаться вопросом, насколько хороши эти оценки вероятностей классов. Небольшое усложнение связано с тем, что, как уже отмечалось, истинные вероятности нам неизвестны. Один часто применяемый прием заключается в том, чтобы определить битовый вектор $(I[c(x) = C_1], \dots, I[c(x) = C_k])$, в котором i -й бит равен 1, если истинный класс x совпадает с C_i , а все остальные биты равны 0, и использовать эти векторы как «истинные» вероятности. После этого можно определить *квадратичную ошибку* (*SE*) предсказанного вектора вероятностей $\hat{\mathbf{p}}(x) = (\hat{p}_1(x), \dots, \hat{p}_k(x))$ в виде

$$\text{SE}(x) = \frac{1}{2} \sum_{i=1}^k (\hat{p}_i(x) - I[c(x) = C_i])^2 \quad (2.6)$$

и *среднеквадратичную ошибку* (*MSE*) как квадратичную ошибку, усредненную по всем объектам в тестовом наборе:

$$\text{MSE}(Te) = \frac{1}{|Te|} \sum_{x \in Te} \text{SE}(x). \quad (2.7)$$

Это определение ошибки при оценивании вероятности часто применяется в *теории прогнозирования*, где называется *оценкой Брайера*. Коэффициент $1/2$ в формуле 2.6 гарантирует, что квадратичная ошибка каждого объекта попадает в интервал от 0 до 1: худший из возможных случаев возникает, когда неверный класс предсказывается с вероятностью 1, то есть два «бита» ошибочны. Для двух классов сумма сводится к одному члену $(\hat{p}(x) - I[c(x) = \oplus])^2$, в котором встречается только положительный класс. Отметим, что если оценка вероятностей классов «категорическая», то есть назначает вероятность 1 одному классу и вероятность 0 другому, то это по существу классификатор, и MSE сводится к верности, определенной в разделе 2.1.

Пример 2.6 (квадратичная ошибка). Предположим, что одна модель предсказывает вероятности (0.70, 0.10, 0.20) для некоторого примера x в трехклассовой задаче, а другая ведет себя гораздо увереннее и предсказывает вероятности (0.99, 0, 0.01). Если первый класс является истинным, то второе предсказание, очевидно, лучше первого: квадратичная ошибка первого предсказания равна $((0.70 - 1)^2 + (0.10 - 0)^2 + (0.20 - 0)^2)/2 = 0.07$, а второго $- ((0.99 - 1)^2 + (0 - 0)^2 + (0.01 - 0)^2)/2 = 0.0001$. Первая модель штрафуется сильнее, поскольку, будучи в основном правильной, она меньше уверена в этом.

Однако если истинным является третий класс, то ситуация становится прямо противоположной: теперь квадратичная ошибка первого предсказания равна $((0.70 - 0)^2 + (0.10 - 0)^2 + (0.20 - 1)^2)/2 = 0.57$, а второго $- ((0.99 - 0)^2 + (0 - 0)^2 + (0.01 - 1)^2)/2 = 0.98$. Сильнее штрафуется вторая модель – не только за неверные предсказания, но и за самоуверенность.

Возвращаясь к дереву оценивания вероятностей на рис. 2.12, вычисляем квадратичную ошибку в каждом листе (слева направо):

$$\begin{aligned}\text{SE}_1 &= 20(0.33 - 1)^2 + 40(0.33 - 0)^2 = 13.33, \\ \text{SE}_2 &= 10(0.67 - 1)^2 + 5(0.67 - 0)^2 = 3.33, \\ \text{SE}_3 &= 20(0.80 - 1)^2 + 5(0.80 - 0)^2 = 4.00,\end{aligned}$$

что дает среднеквадратичную ошибку $\text{MSE} = 1/100(\text{SE}_1 + \text{SE}_2 + \text{SE}_3) = 0.21$. Интересный вопрос: можно ли изменить предсказанные вероятности в каждом листе, так чтобы получить меньшую среднеквадратичную ошибку? Оказывается, нет: предсказанные вероятности, полученные с помощью распределения по классам в каждом листе, оптимальны в смысле наименьшей **MSE**. Например, если изменить предсказанные вероятности в левом узле, задав для спама вероятность 0.40, а для неспама 0.60 или 0.20 для спама и 0.80 для неспама, то квадратичная ошибка окажется больше:

$$\begin{aligned}\text{SE}'_1 &= 20(0.40 - 1)^2 + 40(0.40 - 0)^2 = 13.6, \\ \text{SE}''_1 &= 20(0.20 - 1)^2 + 40(0.20 - 0)^2 = 14.4.\end{aligned}$$

Причина станет понятной, если переписать формулу квадратичной ошибки в листе в случае двух классов в следующем виде, где n^{\oplus} и n^{\ominus} обозначают количество положительных и отрицательных примеров в листе:

$$\begin{aligned}n^{\oplus}(\hat{p} - 1)^2 + n^{\ominus}\hat{p}^2 &= (n^{\oplus} + n^{\ominus})\hat{p}^2 - 2n^{\oplus}\hat{p} + n^{\oplus} = \\ &= (n^{\oplus} + n^{\ominus})[\hat{p}^2 - 2p\hat{p} + \dot{p}] = (n^{\oplus} + n^{\ominus})[(\hat{p} - \dot{p})^2 + \dot{p}(1 - \dot{p})].\end{aligned}$$

Здесь $\dot{p} = n^{\oplus}/(n^{\oplus} + n^{\ominus})$ – относительная частота положительного класса среди примеров, попадающих в данный лист; ее называют также *эмпирической вероятностью*. Из того, что член $\dot{p}(1 - \dot{p})$ не зависит от предсказанной вероятности \hat{p} , сразу следует, что наименьшая квадратичная ошибка в листе достигается, если положить $\hat{p} = \dot{p}$.

Эмпирические вероятности важны, потому что позволяют получить или точно настроить оценки вероятностей, вычисляемые классификаторами или ранжиров-

щиками. Если имеется множество S помеченных экземпляров и n_i – число встречающихся в S экземпляров из класса C_i , то вектор эмпирических вероятностей, ассоциированный с S , имеет вид $\hat{\mathbf{p}}(S) = (n_1/|S|, \dots, n_k/|S|)$. На практике почти всегда имеет смысл *сгладить* эти относительные вероятности, чтобы избежать проблем, вызванных экстремальными значениями (0 или 1). Чаще всего для этого кладут

$$\dot{p}_i(S) = \frac{n_i + 1}{|S| + k}. \quad (2.8)$$

Эта формула называется *поправкой Лапласа* в честь французского математика Пьера-Симона Лапласа, который вывел ее для случая $k = 2$ (также известна под названием «правило следования Лапласа»). По существу, мы добавляем равномерно распределенные *псевдосчетчики* к каждой из k альтернатив, это отражает нашу априорную веру в то, что эмпирические вероятности будут распределены равномерно¹. Можно также применить неравномерное сглаживание, положив

$$\dot{p}_i(S) = \frac{n_i + m \cdot \pi_i}{|S| + m}. \quad (2.9)$$

Эта техника сглаживания, известная под названием *m-оценка*, позволяет выбирать как количество псевдосчетчиков m , так и априорные вероятности π_i . Поправка Лапласа – частный случай *m-оценки* при $m = k$ и $\pi_i = 1/k$.

Если все элементы S получают один и тот же вектор предсказанных вероятностей $\hat{\mathbf{p}}(S)$ – что бывает в случае, когда S представляет собой сегмент группирующей модели, – то с помощью аналогичного рассуждения мы сможем выразить полную индуцированную квадратичную ошибку на S в терминах оцененных и эмпирических вероятностей:

$$\begin{aligned} SE(S) &= \sum_{x \in S} SE(x) = \sum_{x \in S} \frac{1}{2} \sum_{i=1}^k (\hat{p}_i(x) - I[c(x) = C_i])^2 = \\ &= \frac{1}{2} |S| \sum_{i=1}^k (\hat{p}_i(S) - \dot{p}_i(S))^2 + \frac{1}{2} |S| \sum_{i=1}^k (\dot{p}_i(S)(1 - \dot{p}_i(S))). \end{aligned}$$

Первый член окончательного выражения называется *потерями на калибровку* и измеряет квадратичную ошибку относительно эмпирических вероятностей. Его можно свести к нулю в группирующих моделях, где мы вправе самостоятельно задавать предсказанные вероятности в каждом сегменте, как в деревьях оценивания вероятностей. Говорят, что модели с низкими потерями на калибровку хорошо откалиброваны. **Второй член** называется *потерями на уточнение*; он зависит только от эмпирических вероятностей и тем меньше, чем менее равномерно они распределены.

¹ Это можно смоделировать математически с помощью априорного распределения вероятности, известного как *априорное распределение Дирихле*.

Из этого анализа следует, что получать оценки вероятностей лучше всего из эмпирических вероятностей, вычисленных на основе обучающего набора или иного набора помеченных примеров, специально зарезервированного для этой цели. Однако нужно рассмотреть два вопроса. Во-первых, для некоторых моделей необходимо гарантировать, что предсказанные вероятности согласованы с подразумеваемым моделью ранжированием. Во-вторых, для ранжирующих моделей эмпирические вероятности напрямую неизвестны, так как каждому примеру обычно назначается его собственный класс эквивалентности. Обсудим это подробнее.

Преобразование ранжировщиков в оценки вероятностей классов

Снова рассмотрим пример 2.5 и представим, что оценки – не вероятности, а величины с неизвестной масштабной шкалой, так что фильтр спама – это ранжировщик, а не инструмент оценивания вероятностей классов. Поскольку оценки всех тестовых примеров различны, то «эмпирические вероятности» равны либо 0 (для отрицательных примеров), либо 1 (для положительных), что приводит к последовательности значений $\dot{p} \ 1 - 1 - 0 - 1 - 1 - 0 - 1 - 1 - 0 - 0$ в порядке убывания оценок. Очевидная проблема заключается в том, что эти значения не согласованы с порядком, индуцированным оценками, и потому для получения оценок вероятности использовать их напрямую невозможно. Сглаживание эмпирических вероятностей с помощью поправки Лапласа не решает проблему, потому что оно лишь заменяет 0 на 1/3, а 1 – на 2/3. Нужна какая-то другая идея.

Глядя на рис. 2.11, мы замечаем, что $\dot{p} = 1$ соответствует вертикальному отрезку кривой РХП, а $\dot{p} = 0$ – горизонтальному. Наша проблема вызвана тем, что вертикальный отрезок идет за горизонтальным, или, более общо, тем, что более крутой отрезок следует за более пологим. Такую последовательность отрезков с возрастающими угловыми коэффициентами мы будем называть *вогнутостью*, поскольку она образует «впадину» на кривой РХП. Кривая без вогнутостей является *выпуклой* кривой РХП. У нашей кривой две вогнутости: одна образована третьим, четвертым и пятым примерами, другая – шестым, седьмым и восьмым.

Предположим теперь, что примеры с третьего по пятый получили одну и ту же оценку, скажем 0.7; а в примерах с шестого по восьмой также имеет место общая неопределенность, скажем 0.4. В таком случае в кривой РХП будет шесть отрезков с эмпирическими вероятностями $1 - 1 - 2/3 - 2/3 - 0 - 0$. Как видно, значения \dot{p} теперь убывают вместе с оценками; иными словами, вогнутости исчезли, и кривая РХП стала выпуклой.

В общем случае *вогнутости кривых РХП можно устраниТЬ, объединив отрезки с одинаковыми оценками*. Это достигается выявлением так называемых *нарушителей монотонности*. Например, в последовательности $1 - 1 - 0 - 1 - 1 - 0 - 1 - 1 - 0 - 0$ третий и четвертый члены – нарушители монотонности, так как нарушают правило, согласно которому оценки должны образовывать убывающую последовательность (точнее, невозрастающую). Для исправления ситуации нужно заменить оба члена их средним арифметическим, что приводит к последователь-

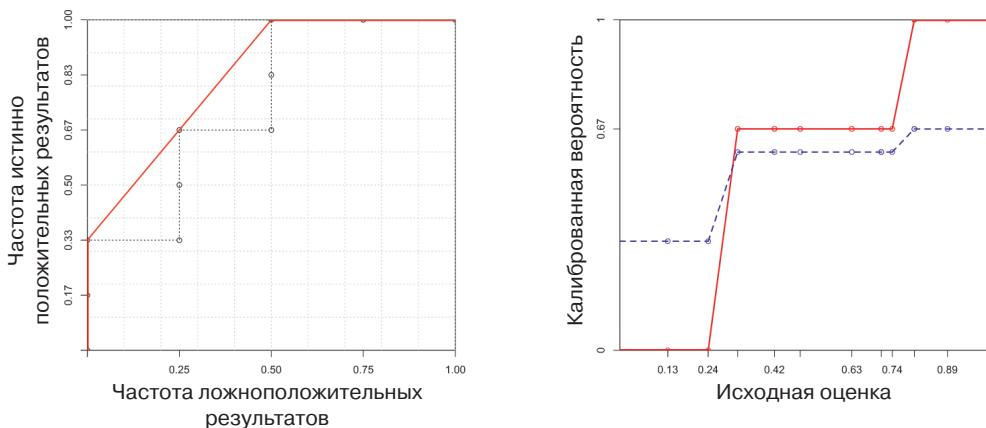


Рис. 2.13. (Слева) Сплошная красная линия – это выпуклая оболочка пунктирной кривой РХП. **(Справа)** Соответствующее калибровочное отображение показано **красным** цветом: горизонтальные участки соответствуют нескольким примерам, отображенными на один и тот же отрезок выпуклой оболочки, а при переходе от одного отрезка выпуклой оболочки к другому производится линейная интерполяция между оценками примеров. Калибровочное отображение с поправкой Лапласа показано **синей** пунктирной линией: сглаживание Лапласа сжимает диапазон откалиброванных вероятностей, но иногда отражается на ранжировании

ности $1 - 1 - [1/2 - 1/2] - 1 - 0 - 1 - 1 - 0 - 0$. Теперь появилось нарушение монотонности между новой парой и четвертым членом, поэтому мы заменяем все три члена средней оценкой и приходим к последовательности $1 - 1 - [2/3 - 2/3 - 2/3] - 0 - 1 - 1 - 0 - 0^1$. Вторая вогнутость $0 - 1 - 1$ обрабатывается аналогично, и в итоге получается последовательность $1 - 1 - [2/3 - 2/3 - 2/3] - [2/3 - 2/3 - 2/3] - 0 - 0$.

Результат изображен на рис. 2.13. Слева мы видим, как две вогнутости заменяются диагональными отрезками с одинаковыми угловыми коэффициентами. Эти диагональные отрезки совпадают с изолинией верности, которая придает трем «самым внешним» точкам вогнутостей одинаковую верность (рис. 2.11). Все вместе, красные отрезки образуют выпуклую оболочку кривой РХП – однозначно определенную выпуклую линию, проходящую через самые внешние точки исходной кривой РХП. У выпуклой оболочки площадь под кривой (AUC) больше, чем у исходной кривой, потому что она заменяет (некоторые) ошибки ранжирования исходной кривой полушибками – вследствие появления неопределенностей. В нашем примере исходное ранжирование индуцирует 6 из 24 ошибок ранжирования ($AUC = 0.75$), тогда как выпуклая оболочка превращает их все в полушибки ($AUC = 0.83$).

¹ Эти два шага можно объединить в один: как только найдена пара нарушителей монотонности, смотрим, нет ли слева и справа от нее примеров с такими же оценками, как у левого и правого нарушителей соответственно.

Построив выпуклую оболочку, мы можем использовать эмпирические вероятности на каждом ее отрезке как калиброванные вероятности. На рис. 2.13 справа показано получающееся *калибровочное отображение*, представляющее собой кусочно-линейную неубывающую функцию, отображающую исходные оценки на оси x в калиброванные вероятности на оси y . Показано также альтернативное калибровочное отображение, которое дает оценки вероятностей после применения поправки Лапласа: для данной последовательности получим $2/3 - 2/3 - [3/5 - 3/5 - 3/5] - [3/5 - 3/5 - 3/5] - 1/3 - 1/3$, то есть диапазон оценок вероятностей оказывается гораздо уже.

Теперь взглянем на этот процесс с точки зрения среднеквадратичной ошибки, калибровки и уточнения. Среднеквадратичная ошибка исходных оценок была равна $\frac{1}{10} [(0.89 - 1)^2 + (0.80 - 1)^2 + (0.74 - 0)^2 + (0.71 - 1)^2 + (0.63 - 1)^2 + (0.49 - 0)^2 + (0.42 - 1)^2 + (0.32 - 1)^2 + (0.24 - 0)^2 + (0.13 - 0)^2] = 0.19$. Отметим, что она целиком обусловлена потерями на калибровку, потому что все эмпирические вероятности равны 0 или 1, а значит, потери на уточнение нулевые. У калиброванных оценок среднеквадратичная ошибка равна $\frac{1}{10} [(1 - 1)^2 + (1 - 1)^2 + (0.67 - 0)^2 + (0.67 - 1)^2 + (0.67 - 1)^2 + (0.67 - 0)^2 + (0.67 - 1)^2 + (0.67 - 1)^2 + (0 - 0)^2 + (0 - 0)^2] = 0.13$. Теперь вся среднеквадратичная ошибка обусловлена потерями на уточнение, поскольку оцененные вероятности равны эмпирическим на каждом отрезке по построению. Мы обменяли увеличение потерь на уточнение на уменьшение потерь на калибровку; так как последние больше, общая ошибка уменьшилась. Увеличение потерь на калибровку вызвано появлением диагональных отрезков в результате построения выпуклой оболочки. Для процесса получения калиброванных оценок путем построения выпуклой оболочки кривой РХП есть специальный термин: *изотонная калибровка*, поскольку лежащая в его основе математическая проблема называется *изотонной регрессией*. Применяя изотонную калибровку, следует проявлять осторожность, чтобы не переобучить данные. В калибровочном отображении на рис. 2.13 справа все точки перехода – на горизонтальных участках и на границе с наклонными – получены непосредственно из имеющихся данных, на неизвестные данные они могут обобщаться плохо. Поэтому рекомендуется применять к эмпирическим вероятностям поправку Лапласа, несмотря даже на то, что она увеличивает потери на калибровку для предъявленных данных.

2.4 Бинарная классификация и родственные задачи: итоги и дополнительная литература

В этой главе мы рассмотрели бинарную классификацию – широко распространенную задачу, которая является отправной точкой для многих проблем машинного обучения. И хотя в данной главе мы почти не говорили об обучении, я по-

лагаю, что вы сможете лучше понять модели и алгоритмы машинного обучения, если сначала познакомитесь с задачами, которые они призваны решать.

- ☞ В разделе 2.1 мы определили задачу бинарной классификации и ввели важный инструмент для оценки качества ее решения: таблицу сопряженности 2×2 . Из элементов этой таблицы можно вывести самые разные показатели качества. Я рассказал о графике покрытия, который позволяет наглядно представить таблицу сопряженности в виде прямоугольника высотой Pos и шириной Neg и точки внутри этого прямоугольника с ординатой TP и абсциссой FP . Мы можем представить несколько моделей классификации над одним и тем же набором данных несколькими точками и, используя тот факт, что верность постоянна на отрезках прямых с угловым коэффициентом 1, визуально ранжировать эти классификаторы по верности. Вместо этого мы можем нормировать прямоугольник, преобразовав его в единичный квадрат, по осям которого отложены частоты истинно положительных и ложноположительных результатов. В этом РХП-пространстве отрезки прямых с угловым коэффициентом 1 (параллельные восходящей диагонали) являются геометрическим местом точек с одинаковой средней полнотой (иногда ее называют также макрополнотой). Впервые эти графики были применены в машинном обучении в работе Provost, Fawcett (2001). Ненормированные графики покрытия были введены в рассмотрение в работе Furnkranz, Flach (2003).
- ☞ В разделе 2.2 рассматривалась более общая задача вычисления оценки каждого примера (или вектора оценок в случае, когда классов больше двух). Хотя шкала оценок не указывается, принято в качестве порога принятия решения брать $\hat{s}(x) = 0$ и считать, что знак оценки определяет предсказание (положительное или отрицательное). Умножая оценку на истинный класс, мы получаем зазор, положительный для правильного предсказания и отрицательный для неправильного. Функция потерь определяет, насколько сильно штрафуется отрицательный зазор и вознаграждается положительный. Выпуклые и непрерывно дифференцируемые «суррогатные» функции потерь (вместо функции потерь типа 0–1, которую мы в конечном итоге хотим оптимизировать) часто порождают вычислительно более простые задачи оптимизации, и в этом их преимущество.
- ☞ Можно вместо этого вообще игнорировать шкалу измерения оценок и принимать во внимание только их упорядоченность. Такой ранжировщик визуализируется на графике покрытия или РХП кусочно-непрерывной кривой. Для группирующих моделей отрезки этой кривой соответствуют сегментам пространства объектов (например, листьям древовидной модели), а для ранжирующих существует отрезок для каждой уникальной оценки, вычисленной моделью. Площадь под кривой РХП дает верность ранжирования (оценку вероятности того, что случайно выбранный положительный пример будет ранжирован раньше случайно выбранного отрицательного примера) и известна в статистике как критерий Уилкоксона.

на-Манна-Уитни. Эти кривые можно также использовать для нахождения подходящей рабочей точки путем преобразования режима работы (сочетания отношения затрат и отношения классов) в изолинию на графике РХП или покрытия. Истоки кривых РХП следует искать в теории обнаружения сигналов (Egan, 1975); доступное введение см. в (Fawcett, 2006; Flach, 2010b).

- ☞ В разделе 2.3 мы рассматривали модели оценивания, порождаемые ими оценки можно интерпретировать как оценки вероятностей принадлежности объекта конкретному классу. Такие модели впервые были применены в теории прогнозирования, в частности в работах Brier (1950) и Murphy, Winkler (1984). Для получения представления о качестве оценок вероятностей классов мы можем сравнить их с «идеальными» вероятностями (1 для положительных примеров и 0 для отрицательных) и вычислить среднеквадратичную ошибку. Поскольку нет никаких причин, почему истинные вероятности должны быть категоричными, эта величина является весьма грубой оценкой, а ее разложение на потери на калибровку и потери на уточнение дает полезную дополнительную информацию. Мы также познакомились с очень полезным приемом — слгаживанием оценок вероятности в виде относительной частоты путем прибавления псевдосчетчиков, имеющих либо равномерное (поправка Лапласа), либо априорно выбранное распределение (m -оценка). Наконец, мы видели, как можно использовать выпуклую оболочку кривой РХП для получения калиброванных оценок вероятностей классов. Своими корнями этот подход уходит в изотонную регрессию (Best, Chakravarti, 1990) и к машинному обучению был впервые применен в работе Zadrozny, Elkan (2002). В работах Fawcett, Niculescu-Mizil (2007) и Flach, Matsubara (2007) показано, что этот подход эквивалентен калибровке с помощью выпуклой оболочки РХП. (Отметим, что в этой главе термин «выпуклый» встречался в двух контекстах: во-первых, применительно к функциям потерь, где выпуклость означает, что при линейной интерполяции любых двух точек на кривой функции потерь мы никогда не получим точку, расположенную под этой кривой; а во-вторых, применительно к выпуклой оболочке РХП, которая представляет собой состоящую из отрезков прямых границу выпуклого множества, охватывающего все точки набора.)



За пределами бинарной классификации

В предыдущей главе мы познакомились с бинарной классификацией и родственными задачами, в частности ранжированием и оцениванием вероятностей классов. В этой главе мы пойдем дальше в нескольких направлениях. В разделе 3.1 обсуждается, что делать, когда классов больше двух. В разделе 3.2 мы рассмотрим случай вещественнозначной целевой переменной. Раздел 3.3 посвящен различным формам обучения – без учителя и рассчитанного на обучение дескриптивных моделей.

3.1 Когда классов больше двух

Некоторые понятия принципиально бинарны. Например, понятие кривой покрытия так просто не обобщается на случай более двух классов. Сейчас мы рассмотрим общие вопросы классификации, оценки и оценивания вероятностей, возникающие, когда классов больше двух. Нас будут интересовать две проблемы: как оценить качество решения многоклассовой задачи и как построить многоклассовые модели из бинарных. Последнее необходимо для некоторых моделей, например линейных классификаторов, которые предназначены прежде всего для разделения двух классов. Другие модели, в том числе решающие деревья, вполне естественно работают с произвольным числом классов.

Многоклассовая классификация

Задачи классификации, в которых больше двух классов, чрезвычайно распространены. Например, после того как у пациента диагностирована ревматическая болезнь, врач захочет продолжить классификацию, определив один из нескольких вариантов. Если имеется k классов, то качество классификатора можно оценить по таблице сопряженности размером $k \times k$. Оценить качество легко, если нас интересует верность классификатора, которая по-прежнему равна сумме чисел на нисходящей диагонали таблицы сопряженности, поделенной на количество тестовых объектов. Однако, как и раньше, это может замаскировать различия в качестве работы на разных классах. Кроме того, другие показатели могут дать больше информации.

Пример 3.1 (качество работы многоклассовых классификаторов). Рассмотрим следующую матрицу неточностей для трех классов (дополненную маргиналами):

		Предсказано			
		15	2	3	20
Фактически	7	15	8	30	
	2	3	45	50	
	24	20	56	100	

Верность этого классификатора равна $(15 + 15 + 45)/100 = 0.75$. Можно посчитать точность и полноту для каждого класса: для первого класса – $15/20 = 0.75$ и $15/30 = 0.50$, для второго и третьего – $45/56 = 0.80$ и $45/50 = 0.90$. Мы можем усреднить эти числа, получив тем самым величины точности и полноты для классификатора в целом, или взять взвешенное среднее, учитывающее долю каждого класса. Например, средневзвешенная точность равна $0.20 \cdot 0.63 + 0.30 \cdot 0.75 + 0.50 \cdot 0.80 = 0.75$. Отметим, что верность – по-прежнему взвешенное среднее значений полноты отдельных классов, как и в двухклассовом случае (см. пример 2.1).

Другая возможность – провести более детальный анализ, посмотрев на точность и полноту для каждой пары классов; например, при попытке отличить первый класс от третьего точность равна $15/17 = 0.88$, а полнота – $15/18 = 0.83$, тогда как при попытке отличить третий класс от первого – $45/48 = 0.94$ и $45/47 = 0.96$ соответственно (можете ли вы объяснить, почему в обратном направлении получаются гораздо большие значения?).

Теперь представим, что нужно построить многоклассовый классификатор, но мы умеем обучать только двухклассовые модели, скажем, линейные классификаторы. Существуют различные способы объединить их в один k -классовый классификатор. Идея подхода «один против всех» состоит в том, чтобы обучить k бинарных классификаторов, первый из которых отделяет класс C_1 от C_2, \dots, C_n , второй отделяет C_2 от всех остальных классов и т. д. При обучении i -го классификатора мы рассматриваем объекты класса C_i как положительные примеры, а все остальные – как отрицательные. Иногда обучение ведется в фиксированном порядке, то есть мы обучаем $k - 1$ моделей, i -я из которых отделяет C_i от C_{i+1}, \dots, C_n , $1 \leq i < n$. Альтернативой схеме «один против всех» является схема «один против одного». В этой схеме мы обучаем $k(k - 1)/2$ бинарных классификаторов, по одному для каждой пары различных классов. Если бинарный классификатор рассматривает классы асимметрично, что характерно для некоторых моделей, то имеет смысл обучить по два классификатора для каждой пары – тогда всего получится $k(k - 1)$ классификаторов.

Для описания этих и других схем удобно разложить k -классовую задачу на l задач бинарной классификации с помощью так называемой матрицы *выходных кодов*. Это матрица размерности $k \times l$, состоящая из элементов $+1, 0, -1$. Ниже показаны выходные коды, описывающие два способа преобразования трехклассовой задачи с применением схемы «один против одного»:

$$\begin{pmatrix} +1 & +1 & 0 \\ -1 & 0 & +1 \\ 0 & -1 & -1 \end{pmatrix} \quad \begin{pmatrix} +1 & -1 & +1 & -1 & 0 & 0 \\ -1 & +1 & 0 & 0 & +1 & -1 \\ 0 & 0 & -1 & +1 & -1 & +1 \end{pmatrix}$$

Каждый столбец матрицы описывает задачу бинарной классификации, в которой положительным является класс, соответствующий строке, в которой стоит $+1$, а отрицательным – класс, соответствующий строке, содержащей -1 . Таким образом, в симметричной схеме слева мы обучаем три классификатора: один, умеющий различать классы C_1 (положительный) и C_2 (отрицательный), другой – для различия C_1 (положительный) и C_3 (отрицательный) и третий – для различия C_2 (положительный) и C_3 (отрицательный). В асимметричной схеме справа обучаются еще три классификатора, в которых роли положительного и отрицательного классов поменяны местами. Матрицы кодов для неупорядоченного и упорядоченного вариантов схемы «один против всех» выглядят следующим образом:

$$\begin{pmatrix} +1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \end{pmatrix} \quad \begin{pmatrix} +1 & 0 \\ -1 & +1 \\ -1 & -1 \end{pmatrix}$$

В схеме слева мы обучаем один классификатор, умеющий отличать C_1 (положительный) от C_2 и C_3 (отрицательные), другой – умеющий отличать C_2 (положительный) от C_1 и C_3 (отрицательные), и третий – умеющий отличать C_3 (положительный) от C_1 и C_2 (отрицательные). Справа мы зафиксировали порядок классов $C_1 - C_2 - C_3$, поэтому обучить нужно только два классификатора. Чтобы установить класс нового тестового объекта, мы собираем предсказания, выданые всеми бинарными классификаторами, которые могут принимать значения $+1$ (положительный класс), -1 (отрицательный класс) или 0 (отсутствие предсказания, или *отвержение*, это возможно, например, в случае классификатора, основанного на правилах). В совокупности эти предсказания образуют «слово», которое можно поискать в матрице кодов, этот процесс называется *декодированием*. Предположим, что получено слово $-1 +1 -1$, и использовалась неупорядоченная схема «один против всех». Тогда мы знаем, что результатом должен быть класс C_2 . Вопрос: что делать, если слово отсутствует в матрице кодов? Например, если использовалась симметричная схема «один против одного» (первая из приведенных выше матриц кодов) и получено слово $0 +1 0$. В этом случае можно было бы счесть, что ближайшим кодовым словом является первая строка матрицы и, значит, следует предсказать класс C_1 . Чтобы формализовать это рассуждение, определим расстояние между словом w и кодовым словом c как $d(w, c) = \sum_i (1-w_i c_i)/2$, где i пробегает множество «битов» слов (столбцов кодовой матрицы). Это означает, что биты, которые в обоих словах совпадают, не дают вклада в расстояние; каждый бит, который в одном слове равен $+1$, а в другом -1 , дает вклад 1; если же один из битов равен 0, то вклад будет $1/2$ вне зависимости от значения второго бита¹. Тогда предсказанный класс слова w равен $\min_j d(w, c_j)$,

где c_j – j -я строка кодовой матрицы. Таким образом, если $w = 0 + 1 \ 0$, то $d(w, c_1) = 1$ и $d(w, c_2) = d(w, c_3) = 1.5$, то есть мы предсказаем класс C_1 .

Однако ближайший код не всегда определен однозначно. Пусть, например, применяется четырехклассовая схема «один против всех», и два из бинарных классификаторов предсказывают положительный класс, а два других – отрицательный. Получающееся слово равноудалено от двух кодовых слов, поэтому мы не можем решить, какой из соответствующих этим словам классов выбрать. Ситуацию можно улучшить, добавив в кодовую матрицу дополнительные столбцы:

$$\begin{pmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{pmatrix} \quad \begin{pmatrix} +1 & -1 & -1 & -1 & +1 & +1 & +1 \\ -1 & +1 & -1 & -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 & -1 & -1 & +1 \end{pmatrix}$$

Слева находится стандартная кодовая матрица для четырехклассовой схемы «один против всех», а справа – она же, пополненная тремя столбцами (то есть бинарными проблемами обучения). В результате расстояние между любыми двумя кодовыми словами увеличилось с 2 до 4, и, следовательно, повысились шансы, что мы сможем однозначно декодировать слова, отсутствующие в кодовой матрице. Такую схему можно рассматривать как комбинацию схем «один против всех» и «один против одного». Отметим, однако, что дополнительные бинарные проблемы обучения могут оказаться трудными. Например, если четыре класса – это спамная почта, рабочая почта, хозяйственная почта (например, счета за коммунальные услуги или выписки по карточным счетам) и личная почта, то каждая задача бинарной классификации типа «один против всех» может оказаться гораздо проще, чем, скажем, задача различения спамной и рабочей почты, с одной стороны, и хозяйственной и личной – с другой.

Схемы «один против всех» и «один против одного» употребляются чаще всего для преобразования бинарных классификаторов в многоклассовые. Чтобы принять определенное решение в схеме «один против всех», мы можем установить фиксированный порядок классов до или после обучения. В схеме «один против одного» для принятия решения можно использовать голосование, что на самом деле эквивалентно декодированию на основе расстояния, как показано в следующем примере.

Пример 3.2 (голосование в схеме «один против одного»). Матрица кодов для схемы «один против одного» при $k = 4$ классах выглядит следующим образом:

$$\begin{pmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}.$$

¹ Это небольшое обобщение *расстояния Хэмминга* между двоичными строками, которое равно количеству позиций, в которых строки различаются.

Предположим, что шесть попарных классификаторов предсказывают слово $w = +1 -1 +1 -1 +1 +1$. Это слово можно интерпретировать как голоса $C_1 - C_3 - C_1 - C_3 - C_2 - C_3$, то есть три голоса за класс C_3 , два голоса за класс C_1 и один голос за класс C_2 . В общем случае голос i -го классификатора за j -ый класс можно записать в виде $(1 + w_i c_{ji})/2$, где c_{ji} – элемент матрицы кодов на пересечении j -ой строки и i -го столбца матрицы кодов. Однако при этом переоценивается значимость нулей в матрице кодов; так как каждый класс участвует в $k - 1$ попарных бинарных задачах и всего существует $l = k(k - 1)/2$ задач, то количество нулей в каждой строке равно $k(k - 1)/2 - (k - 1) = (k - 1)(k - 2)/2 = l(k - 2)/k$ (в нашем случае 3). Для каждого нуля мы должны вычесть половину голоса, поэтому общее число голосов за класс C_j равно

$$\begin{aligned} v_j &= \left(\sum_{i=1}^l \frac{1 + w_i c_{ji}}{2} \right) - l \frac{k-2}{2k} = \left(\sum_{i=1}^l \frac{w_i c_{ji} - 1}{2} \right) + l - l \frac{k-2}{2k} = \\ &= -d_j + l \frac{2k - k + 2}{2k} = \frac{(k-1)(k+2)}{4} - d_j, \end{aligned}$$

где $d_j = \sum_i (1 - w_i c_{ji})/2$ – битовое расстояние, рассмотренное выше. Иными словами, сумма расстояния и количества голосов, поданных за каждый класс, равна константе, зависящей только от количества классов, при трех классах она равна 4.5. Это легко проверить, заметив, что расстояние между w и первым кодовым словом равно 2.5 (два голоса за C_1), со вторым кодовым словом 3.5 (один голос за C_2), с третьим кодовым словом 1.5 (три голоса за C_3) и с четвертым кодовым словом 4.5 (нет голосов).

Если наши бинарные классификаторы порождают оценки, то их можно учесть следующим образом. Как и раньше, предположим, что знак оценки s_i обозначает класс. Тогда мы можем использовать подходящий элемент матрицы кодов c_{ji} для вычисления зазора $z_i = s_i c_{ji}$, который передается функции потерь L (зазоры и функции потерь обсуждались в разделе 2.2). Таким образом, мы определяем расстояние между вектором оценок s и j -ым кодовым словом c_j как $d(s, c_j) = \sum_i L(s_i c_{ji})$ и выбираем класс, который минимизирует это расстояние. Этот способ получения многоклассового решения из бинарных оценок называется *декодированием на основе потерь*.

Пример 3.3 (декодирование на основе потерь). Продолжая предыдущий пример, предположим, что оценки, порождаемые шестью попарными классификаторами, таковы: (+5, -0.5, +4, -0.5, +4, +0.5). Это приводит к следующей матрице зазоров:

$$\begin{pmatrix} +5 & -0.5 & +4 & 0 & 0 & 0 \\ -5 & 0 & 0 & -0.5 & +4 & 0 \\ 0 & +0.5 & 0 & +0.5 & 0 & +0.5 \\ 0 & 0 & -4 & 0 & -4 & -0.5 \end{pmatrix}.$$

Применяя функцию потерь типа 0–1, мы игнорируем абсолютные величины зазоров и, следовательно, предсказываем класс C_3 , как в схеме на основе голосования в примере 3.2. Применяя экспоненциальную функцию потерь $L(z) = \exp(-z)$, мы получаем расстояния (4.67, 153.08, 4.82, 113.85). Схема декодирования на основе потерь должна была бы предпочесть класс C_1 вследствие того, что он много выигрывает у C_2 и C_4 ; с другой стороны, C_3 хотя и выигрывает во всех трех случаях, но с небольшим преимуществом.

Следует отметить, что в схеме декодирования на основе потерь предполагается, что все бинарные классификаторы пользуются одной и той же шкалой оценок.

Многоклассовые оценки и вероятности

Для вычисления многоклассовых оценок и вероятностей по бинарным классификаторам есть несколько возможностей.

- ☞ Можно взять расстояния, полученные с помощью декодирования на основе потерь, и превратить их в оценки с помощью подходящего преобразования по аналогии с тем, как мы преобразовали битовые расстояния в голоса в примере 3.2. Этот метод применим, если бинарные классификаторы порождают калиброванные оценки по одной и той же шкале.
- ☞ Можно вместо этого использовать результаты каждого бинарного классификатора как признаки (вещественные, если используются оценки, или двоичные, если классификатор только предсказывает класс) и обучить модель, которая порождает многоклассовые оценки, например наивную байесовскую или древовидную. Этот метод применим всегда, но требует дополнительного обучения.
- ☞ Существует еще одна простая и также универсально применимая альтернатива, которая зачастую дает удовлетворительные результаты – выводить оценки из *счетчиков покрытия*: количества примеров каждого класса, классифицированных как положительные бинарным классификатором. Это показано в примере 3.4.

Пример 3.4 (счетчики покрытия в качестве оценок). Предположим, что есть три класса и три бинарных классификатора, каждый из которых предсказывает положительный или отрицательный класс (и не может отвергать примеры). Первый классификатор определяет как положительные 8 примеров из первого класса, ни одного примера из второго класса и 2 примера из третьего класса. Для второго классификатора счетчики равны 2, 17, 1, а для третьего – 4, 2, 8. Предположим, что тестовый объект определен как положительный первым и третьим классификаторами. Сложив счетчики покрытия этих двух классификаторов, мы получим вектор оценок (12, 2, 10). Аналогично если для некоторого тестового объекта «срабатывают» (то есть выдают положительное предсказание) все три классификатора, то вектор оценок будет равен (14, 19, 11). Эту схему удобно описать с помощью матричной нотации:

$$\begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 8 & 0 & 2 \\ 2 & 17 & 1 \\ 4 & 2 & 8 \end{pmatrix} = \begin{pmatrix} 12 & 2 & 10 \\ 14 & 19 & 11 \end{pmatrix}. \quad (3.1)$$

Средняя матрица содержит счетчики классов (по одной строке для каждого классификатора). Левая матрица 2×3 для каждого примера содержит строку, показывающую, какие классификаторы сработали на этом примере. Тогда матрица в правой части дает объединенные счетчики для каждого примера.

В случае l бинарных классификаторов эта схема разбивает пространство объектов на 2^l регионов. Каждому региону сопоставляется его собственный вектор оценок, поэтому для получения разнотипных оценок l должно быть достаточно велико.

Имея многоклассовые оценки, мы можем задать стандартный вопрос: насколько они хороши? Как мы видели в разделе 2.1, важным показателем качества бинарного оценивающего классификатора является площадь под кривой РХП (AUC), равная доле правильно ранжированных пар положительный–отрицательный. К сожалению, у ранжирования нет прямого многоклассового аналога, так что самый очевидный подход – вычислить среднюю AUC по задачам бинарной классификации, в варианте «один против всех» или «один против одного». Так, средняя AUC в варианте «один против всех» оценивает вероятность того, что если назначить положительным случайно (с равномерным распределением) выбранный класс, то случайно (опять же с равномерным распределением) выбранный пример из этого класса получит более высокую оценку, чем случайно выбранный пример из любого из прочих классов. Отметим, что «отрицательный» пример с большей вероятностью будет принадлежать какому-то из чаще встречающихся классов; поэтому при выборке положительного класса иногда предполагают неравномерное распределение, в котором каждому классу назначается вес, равный его встречаемости в тестовом наборе.

Пример 3.5 (многоклассовая AUC). Предположим, что имеется многоклассовый оценивающий классификатор, который порождает k -мерный вектор оценок $\hat{s}(x) = (\hat{s}_1(x), \dots, \hat{s}_k(x))$ для каждого тестового объекта x . Ограничившись только оценкой $\hat{s}_i(x)$, мы получим оценивающий классификатор для класса C_i против всех остальных классов и можем обычным образом вычислить AUC по схеме «один против всех» для C_i .

Для примера допустим, что имеется три класса и AUC по схеме «один против всех» для первого класса оказалась равной 1, для второго – 0.8, а для третьего – 0.6. Таким образом, для всех объектов класса 1 первый элемент в векторе оценок будет больше, чем для любого объекта из двух остальных классов. Среднее этих трех значений AUC равно 0.8, и это означает, что если случайно выбрать индекс i , объект x из класса C_i и другой объект x' из любого класса, кроме C_i (во всех случаях предполагается равномерное распределение вероятности случайного выбора), то математическое ожидание того, что $\hat{s}_i(x) > \hat{s}_i(x')$, равно 0.8.

Предположим теперь, что в классе C_1 10 объектов, в классе C_2 – 20, а в классе C_3 – 70. Тогда взвешенное среднее значений AUC по схеме «один против всех» равно 0.68, то есть если случайно (равномерно) выбрать x из любого класса, а затем случайно выбрать x' из всех объектов, не принадлежащих тому же классу, что и x , то математическое ожидание того, что $\hat{s}_i(x) > \hat{s}_i(x')$, равно 0.68. Это меньше, чем раньше, потому что теперь больше вероятность того, что случайно выбранный объект x окажется принадлежащим классу C_3 , оценки которого ранжируют хуже.

Мы можем аналогично вычислить средние для AUC по схеме «один против одного». Например, можно определить AUC_{ij} как AUC, полученную с использованием оценок \hat{s}_j , и с ее помощью ранжировать объекты из классов C_i и C_j . Отметим, что оценка \hat{s}_j может ранжировать эти объекты по-другому, поэтому

$\text{AUC}_{ji} \neq \text{AUC}_{ij}$. Если взять невзвешенное среднее по всем $i \neq j$, то мы получим оценку вероятности того, что для случайно (равномерно) выбранных классов i и $j \neq i$ и случайно выбранных объектов $x \in C_i$ и $x' \in C_j$ окажется, что $\hat{s}_i(x) > \hat{s}_j(x')$. А взвешенное среднее дает оценку вероятности того, что объекты правильно ранжированы, если класс заранее не выбирался.

Простейший способ превратить многоклассовые оценки в классификацию заключается в том, чтобы выбирать класс, на котором достигается максимальная оценка, то есть если $\hat{\mathbf{s}}(x) = (\hat{s}_1(x), \dots, \hat{s}_k(x))$ – вектор оценок, полученный для объекта x , и $m = \operatorname{argmax}_i \hat{s}_i(x)$, то объекту x назначается класс $\hat{c}(x) = C_m$. Однако, как и в случае двух классов, такое фиксированное решающее правило может оказаться неоптимальным, и лучше бы вместо его применения провести обучение на данных. Означает это, что мы хотим в результате обучения найти вектор весов $\mathbf{w} = (w_1, \dots, w_k)$, который позволит скорректировать оценки и назначать класс $\hat{c}(x) = C_m$, где $m' = \operatorname{argmax}_i w_i \hat{s}_i(x)$ ¹. Поскольку умножение вектора весов на константу не оказывает влияния на m , мы можем зафиксировать одну из степеней свободы, положив $w_1 = 1$. К сожалению, нахождение глобально оптимального вектора весов требует неподъемного объема вычислений. На практике хорошо работает эвристический подход: сначала обучить w_2 оптимально отличать C_2 от C_1 , как в случае двух классов; затем обучить w_3 отличать C_3 от $C_1 \cup C_2$ и т. д.

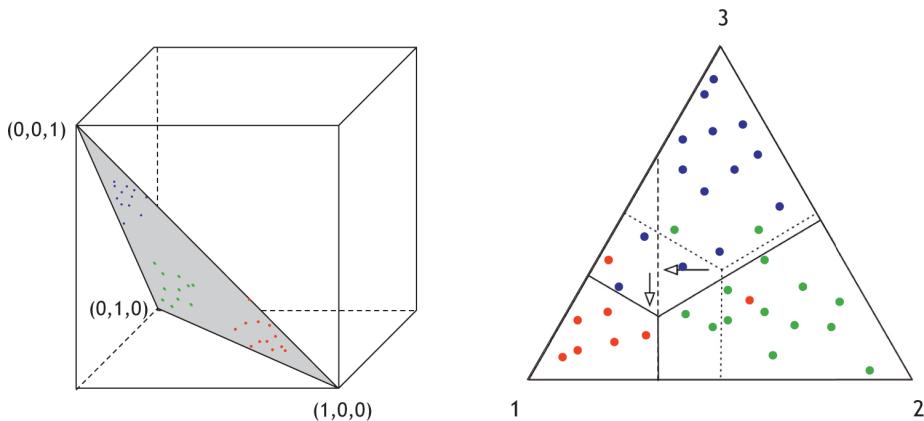


Рис. 3.1. (Слева) Тройки оценок вероятностей, представленные точками в равностороннем треугольнике, соединяющем три вершины единичного куба. **(Справа)** Стрелки показывают, как корректируются веса: сначала они равны (пунктирные линии), потом оптимизируется разделение C_2 и C_1 (штриховая линия), потом – разделение C_3 и двух других классов (сплошные линии). В конечном итоге вес C_1 значительно уменьшился – в пользу двух других классов

¹ Отметим, что при двух классах такое взвешенное решающее правило назначает класс C_1 , если $w_1 \hat{s}_1(x) > w_2 \hat{s}_2(x)$ или эквивалентно $\hat{s}_1(x)/\hat{s}_2(x) > w_2/w_1$. Это можно интерпретировать как пороговое значение для подходящим образом преобразованных оценок, так что описанное взвешенное решающее правило просто обобщает порог принятия решения в случае двух классов.

Пример 3.6 (повторное взвешивание многоклассовых оценок). Проиллюстрируем эту процедуру на примере вероятностного классификатора с тремя классами. Векторы вероятностей $\hat{p}(x) = (\hat{p}_1(x), \hat{p}_2(x), \hat{p}_3(x))$ можно представлять себе как точки внутри единичного куба. Поскольку сумма вероятностей равна 1, эти точки лежат на поверхности равностороннего треугольника, соединяющего три вершины куба (рис. 3.1 слева). Каждая вершина этого треугольника соответствует одному из классов; вероятность, сопоставленная какому-то классу в данной точке, пропорциональна расстоянию до противоположной стороны.

Любое решающее правило вида $\text{argmax}_{w_i} \hat{s}_i(x)$ разрезает треугольник на три области пряммыми, перпендикулярными его сторонам. В случае невзвешенного решающего правила эти линии пересекаются в центре масс треугольника (рис. 3.1 справа). Оптимизация разделения C_2 и C_1 означает, что эта точка сдвигается вдоль прямой, параллельной основанию треугольника, отдаляясь от класса, который получает больший вес. После того как оптимальная точка на этой прямой найдена, мы оптимизируем разделение C_3 и двух других классов, сдвигая точку в направлении, перпендикулярном предыдущей прямой.

Напоследок бегло рассмотрим вопрос о получении калиброванных вероятностей в случае нескольких классов. Это нерешенная проблема, но в литературе предлагалось несколько подходов к ней. Один из самых простых и надежных – вычислить нормированные счетчики покрытия. Точнее, берем сумму или среднее счетчиков покрытия по всем сработавшим классификаторам и нормируем для получения векторов вероятностей, так чтобы сумма элементов равнялась единице. Эквивалентно – можно получить векторы вероятностей для каждого классификатора отдельно и взять их взвешенное среднее с весами, определяемыми относительным покрытием каждого классификатора.

Пример 3.7 (получение многоклассовых вероятностей из счетчиков покрытия). В примере 3.4 мы можем поделить счетчики классов на общее число положительных предсказаний. Это даст такое распределение по классам: $(0.80, 0, 0.20)$ для первого классификатора, $(0.10, 0.85, 0.05)$ – для второго и $(0.29, 0.14, 0.57)$ – для третьего. Распределение вероятностей, ассоциированное с комбинацией первого и третьего классификаторов, таково:

$${}^{10}/_{24} (0.80, 0, 0.20) + {}^{14}/_{24} (0.29, 0.14, 0.57) = (0.50, 0.08, 0.42),$$

и это то же самое распределение, что получено нормировкой комбинированных счетчиков $(12, 2, 10)$. Аналогично распределение, ассоциированное со всеми тремя классификаторами:

$${}^{10}/_{44} (0.80, 0, 0.20) + {}^{20}/_{44} (0.10, 0.85, 0.05) + {}^{14}/_{44} (0.29, 0.14, 0.57) = (0.32, 0.43, 0.25).$$

В матричной нотации это записывается очень лаконично:

$$\begin{pmatrix} 10/24 & 0 & 14/24 \\ 14/44 & 20/44 & 14/44 \end{pmatrix} \begin{pmatrix} 0.80 & 0.00 & 0.20 \\ 0.10 & 0.85 & 0.05 \\ 0.29 & 0.14 & 0.57 \end{pmatrix} = \begin{pmatrix} 0.50 & 0.08 & 0.42 \\ 0.32 & 0.43 & 0.25 \end{pmatrix}.$$

Средняя матрица – это нормированная по строкам средняя матрица из уравнения 3.1. **Нормировка по строкам** – это деление каждого элемента матрицы на сумму элементов в его строке. В результате сумма всех элементов в каждой строке оказывается равна 1, а значит, каждую строку можно интерпретировать как распределение вероятностей. В левой матрице представлена информация двух видов: (i) какие классификаторы срабатывают для каждого примера (в частности, второй классификатор для первого примера не срабатывает) и (ii) покрытие каждого классификатора. Тогда матрица в правой части дает распределение по классам для каждого примера. Отметим, что произведение нормированных по строкам матриц также нормировано по строкам.

В этом разделе мы рассмотрели много интересных вопросов, возникающих, когда классов больше двух. Общий подход к k -классовой проблеме обучения с применением бинарных классификаторов состоит в том, чтобы (i) разбить проблему на l бинарных проблем обучения; (ii) обучить l бинарных классификаторов на двухклассовых версиях исходных данных; (iii) объединить предсказания этих l классификаторов в одно k -классовое предсказание. Для выполнения первого и третьего шагов чаще всего применяется схема «один против одного» или «один против всех», но матрицы кодов открывают возможность для реализации других схем. Мы также видели, как получить многоклассовые оценки и вероятности с помощью бинарных классификаторов, и обсудили эвристический метод калибровки многоклассового решающего правила путем повторного взвешивания. На этом мы завершаем обсуждение классификации – наверное, самой распространенной задачи машинного обучения. В оставшейся части этой главы мы сначала рассмотрим еще одну прогностическую задачу, решаемую путем обучения с учителем, а затем перейдем к обучению без учителя и дескриптивным моделям в разделе 3.3.

3.2 Регрессия

Во всех задачах, рассматривавшихся до сих пор, – классификация, оценка, ранжирование и оценивание вероятностей – пространство меток было дискретным набором классов. В этом разделе мы рассмотрим случай вещественнонозначной целевой переменной. *Оценочной функцией* называется отображение $\hat{f} : \mathcal{X} \rightarrow \mathbb{R}$. Проблема обучения регрессии заключается в построении оценочной функции по примерам $(x_i, f(x_i))$. Например, может стоять задача найти оценочную функцию для индекса Доу-Джонса или лондонского биржевого индекса FTSE 100, исходя из выбранных экономических показателей. Хотя поначалу это выглядит как естественное и не сулящее особых трудностей обобщение дискретной классификации, отличия есть – и существенные. Во-первых, мы переходим от целевой переменной с относительно низким разрешением к переменной, разрешение которой бесконечно. Попытка достичь такого разрешения при обучении оценочной функции почти наверняка приведет к переобучению, а кроме того, весьма вероятно, что какая-то часть значений целевой переменной появилась в результате флуктуаций, которые модель не в состоянии уловить. Поэтому разумно предположить, что примеры зашумлены и что оценочная функция должна улавливать лишь общий тренд, или форму функции.

Пример 3.8 (аппроксимация точек). Рассмотрим следующее множество из пяти точек:

x	y
1.0	1.2
2.5	2.0
4.1	3.7
6.1	4.6
7.9	7.0

Мы хотим построить оценку y в виде полиномиальной функции от x . На рис. 3.2 слева показаны результаты для полиномов степени от 1 до 5, полученных с помощью [линейной регрессии](#), о которой будет рассказано в главе 7. Полиномы степеней 4 и 5 точно проходят через все заданные точки (в общем случае для любого множества из n точек можно найти полином степени не более $n - 1$, график которого проходит через все точки), но заметно различаются на краях; так, полином степени 4 демонстрирует убывающий тренд на участке от $x = 0$ до $x = 1$, хотя из самих данных это не следует.

Чтобы избежать такого переобучения, как показано в примере 3.8, рекомендуется выбирать как можно меньшую степень полинома – часто предполагается простая линейная зависимость.

Регрессия – это задача, на которой отчетливо проявляется различие между группирующими и ранжирующими моделями. Идея группирующей модели состоит в том, чтобы разумным образом разбить пространство объектов на сегменты и в каждом сегменте обучить как можно более простую локальную модель. Например, в решающих деревьях локальная модель – это классификатор на основе мажоритарного класса. Действуя в том же духе, для получения дерева регрессии мы могли бы предсказать в каждом листовом узле постоянное значение. В одномерной проблеме из примера 3.8 это привело бы к кусочно-постоянной функции, изображенной на рис. 3.2 справа. Отметим, что такую группирующую модель можно точно подогнать под заданные точки. В этом смысле она похожа на полином достаточно высокой степени и страдает теми же недостатками, связанными с опасностью переобучения.

Чтобы лучше разобраться в феномене переобучения, рассмотрим количество параметров в каждой модели. У полинома степени n параметров $n + 1$: так, у прямой $y = ax + b$ два параметра, а у полинома степени 4, который проходит через пять точек, пять параметров. Кусочно-постоянная модель с n отрезками имеет $2n - 1$ параметров: n значений y и $n - 1$ значений x , в которых происходят «перескоки». Таким образом, чтобы модель могла точно интерполировать заданные точки, у нее должно быть больше параметров. Эвристическое правило таково: *чтобы избежать переобучения, количество параметров, оцениваемых на основе выборки данных, должно быть значительно меньше объема имеющейся выборки*.

Мы видели, что модели классификации можно рассчитать путем применения функции потерь к зазорам, штрафования отрицательных зазоров (за неправильную классификацию) и вознаграждения положительных (за правильную классификацию). Регрессионные модели рассчитываются путем применения функции потерь к *невязкам* $f(x) - \hat{f}(x)$. В отличие от функций потерь для классификации, функция потерь для регрессии обычно симметрична относительно 0 (хотя вполне допустимо задавать разные веса для положительной и отрицательной невязки). Чаще всего в качестве функции потерь берется квадрат невязки. Это удобно с математической точки зрения, и в качестве обоснования выдвигается гипотеза о том, что наблюдаемые значения функции – это истинные значения, загрязненные аддитивным шумом с нормальным распределением. Однако хоро-

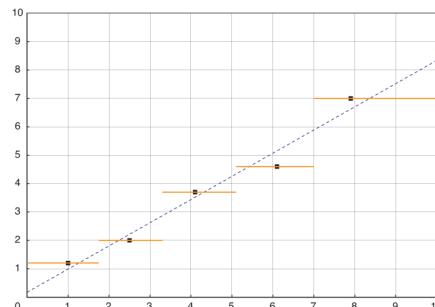
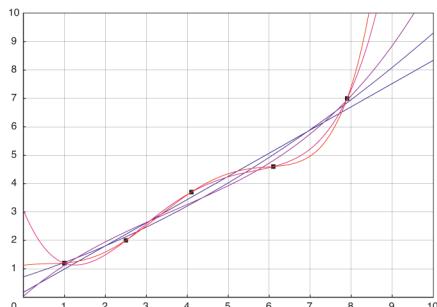


Рис. 3.2. (Слева) Аппроксимация пяти точек полиномами разных степеней. Снизу вверху в правом верхнем углу: степень 1 (прямая линия), степень 2 (парабола), степень 3, степень 4 (наименьшая степень, при которой все пять точек точно укладываются на кривую), степень 5. **(Справа)** Кусочно-постоянная функция, полученная в результате обучения группирующей модели; пунктирная линия – линейная функция с левого рисунка

шо известно, что квадратичная функция потерь чувствительна к выбросам: пример показан на рис. 7.2.

Недооценив количество параметров модели, мы не сможем свести потери к нулю, сколько бы ни предъявляли обучающих данных. С другой стороны, модель с большим числом параметров будет сильнее зависеть от обучающей выборки, и небольшие изменения выборки могут приводить к существенному изменению модели. Иногда это называют *дилеммой смещения-дисперсии*: менее сложная модель не так сильно страдает от изменчивости при случайных колебаниях обучающих данных, но может давать систематическую ошибку, не устранимую даже при увеличении объема обучающих данных; с другой стороны, более сложная модель устраняет такое смещение, но может быть подвержена несистематическим ошибкам из-за дисперсии.

Мы можем немного уточнить это рассуждение, заметив, что математическое ожидание квадрата потерь на обучающем примере x можно представить в следующем виде¹:

$$\mathbb{E}[(f(x) - \hat{f}(x))^2] = (f(x) - \mathbb{E}[\hat{f}(x)])^2 + \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]. \quad (3.2)$$

Важно отметить, что математическое ожидание вычисляется по различным обучающим наборам и, следовательно, по различным оценочным функциям, но алгоритм обучения и пример фиксированы. Первый член в правой части уравнения 3.2 равен нулю, если оценочные функции правильны в среднем; в противном случае алгоритм демонстрирует систематическую ошибку, или *смещение*. Второй член количественно выражает *дисперсию* оценочной функции $\hat{f}(x)$, являющуюся результатом вариаций обучающего набора. На рис. 3.3 это показано графично

¹ Здесь мы раскрываем скобки, воспользовавшись свойством линейности $\mathbb{E}[\cdot]$ и тем, что $\mathbb{E}[f(x)] = f(x)$, вследствие чего члены можно переупорядочить, как в уравнении 3.2.

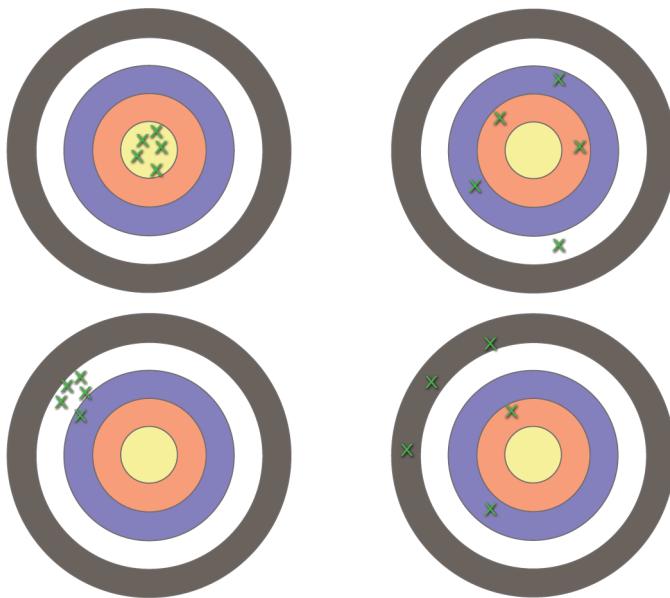


Рис. 3.3. Мишень для дартса иллюстрирует понятия смещения и дисперсии. Разные мишени соответствуют разным алгоритмам обучения, а дротики обозначают разные обучающие выборки. Для алгоритмов в верхнем ряду характерно малое смещение – дротики в среднем располагаются близко к центру мишени (истинному значению функции для конкретного x), а для алгоритмов в нижнем ряду – большое смещение. Для алгоритмов в левой колонке характерна малая дисперсия, а в правой колонке – большая

ски с использованием метафоры мишени для дартса. Очевидно, что наилучшим является случай в левом верхнем углу, но на практике такая удача встречается редко, и приходится выбирать либо низкое смещение и высокую дисперсию (например, аппроксимировать целевую функцию полиномом высокой степени), либо высокое смещение и низкую дисперсию (например, применять линейную аппроксимацию). Мы еще не раз вернемся к дилемме смещения-дисперсии: хотя показанное выше разложение на слагаемые не однозначно для большинства функций потерь (кроме квадратичной), оно служит полезным концептуальным средством для понимания переобучения и недообучения.

3.3 Обучение без учителя и дескриптивные модели

До сих пор нас интересовало исключительно обучение прогностических моделей с учителем. То есть в результате обучения мы находим отображение пространства объектов \mathcal{X} в пространство выходов \mathcal{Y} , используя помеченные примеры

$(x, l(x)) \in \mathcal{X} \times \mathcal{L}$ (возможно, зашумленные). Такой вид обучения называется «с учителем» из-за присутствия в обучающих данных целевой переменной $l(x)$, которая должна быть предоставлена «учителем», обладающим определенными знаниями об истинной помечающей функции l . Модели называются «прогностическими», потому что порождаемые ими результаты либо являются прямыми оценками целевой переменной, либо дают достаточно информации о ее наиболее вероятном значении. Таким образом, мы уделяли внимание только типу моделей в левой верхней ячейке табл. 3.1. В оставшейся части этой главы мы познакомимся с тремя другими видами моделей:

- ☞ обучение без учителя прогностической модели на примере прогностической кластеризации;
- ☞ обучение без учителя дескриптивной модели на примере дескриптивной кластеризации и выявления ассоциативных правил;
- ☞ обучение с учителем дескриптивной модели на примере выявления подгрупп.

	Прогностическая модель	Дескриптивная модель
Обучение с учителем	Классификация, регрессия	Выявление подгрупп
Обучение без учителя	Прогностическая кластеризация	Дескриптивная кластеризация, выявление ассоциативных правил

Таблица 3.1. В оставшейся части этой главы рассматриваются типы моделей, выделенные полужирным шрифтом

Давайте поразмыслим о природе дескриптивного обучения. Задача состоит в том, чтобы получить описание данных – породить дескриптивную модель. Отсюда следует, что выход задачи, будучи моделью, имеет тот же тип, что и вход. Получается, что для порождения дескриптивной модели не имеет смысла использовать отдельный обучающий набор, поскольку мы хотим, чтобы модель описывала наши фактические данные, а не какие-то тестовые. Иными словами, *при дескриптивном обучении задача и проблема обучения совпадают* (рис. 3.4). Это усложняет некоторые вещи: например, маловероятно существование некоей «непререкаемой истины» или «золотого стандарта», на котором можно было бы проверить модель, а следовательно, оценка алгоритмов дескриптивного обучения оказывается намного сложнее, чем в случае прогностических моделей. С другой стороны, можно сказать, что дескриптивное обучение *выявляет* по-настоящему новые знания, поэтому оно зачастую находится на пересечении машинного обучения и добычи данных.

Прогностическая и дескриптивная кластеризация

Различие между прогностической и дескриптивной моделями наглядно проявляется в задачах кластеризации. Одна из возможных интерпретаций кластеризации – это обучение новой помечающей функции по неразмеченным данным.



Рис. 3.4. В случае дескриптивного обучения задача и проблема обучения совпадают: нам не нужен отдельный обучающий набор, а задача состоит в том, чтобы породить дескриптивную модель данных

Следовательно, мы могли бы определить «кластеризатор» так же, как классификатор, а именно как отображение $\hat{q} : \mathcal{X} \rightarrow \mathcal{C}$, где $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ – множество новых меток. Эта интерпретация соответствует *прогностическому* взгляду на кластеризацию, поскольку область определения отображения – все пространство объектов, и, следовательно, оно обобщается на не предъявленные модели объекты. *Дескриптивная* модель кластеризации, обученная на данных $D \subseteq \mathcal{X}$, – это отображение $\hat{q} : D \rightarrow \mathcal{C}$ с областью определения D , а не \mathcal{X} . В любом случае у меток нет никакого внутреннего смысла, кроме того, что они выражают принадлежность к одному и тому же кластеру. Поэтому альтернативный способ определения кластеризатора – отношение эквивалентности $\hat{q} \subseteq \mathcal{X} \times \mathcal{X}$ или $\hat{q} \subseteq D \times D$ (определение отношения эквивалентности см. в замечании 2.1) либо, что то же самое, разбиение \mathcal{X} или D .

Различие между прогностической и дескриптивной кластеризацией тонкое и не всегда четко проводится в литературе. Некоторые хорошо известные алгоритмы кластеризации, в том числе метод \textcircled{K} средних (подробно рассматривается в главе 8), строят прогностическую модель. То есть на основе обучающих данных они порождают модель кластеризации, которую впоследствии можно использовать для отнесения новых данных к кластерам. Это согласуется с введенным нами различием между задачей (кластеризация произвольных данных) и проблемой обучения (построение модели кластеризации на обучающих данных). Однако это различие не применимо к методам дескриптивной кластеризации: здесь модель кластеризации, построенную по данным D , можно использовать только для кластеризации D . По существу, ставится задача обучить подходящую модель кластеризации для имеющихся данных.

Без дополнительной информации любая кластеризация не хуже и не лучше любой другой. Хорошую кластеризацию отличает от плохой тот факт, что данные разбиваются на *компактные* группы, или кластеры. Под «компактностью» здесь понимается, что в среднем два объекта из одного кластера имеют между собой больше общего (более похожи), чем два объекта из разных кластеров. Тем самым предполагается, что существует какой-то способ оценить схожесть, или, что обычно удобнее, *расхождение* произвольной пары объектов, то есть расстояние

между ними. Если все наши признаки числовые, то есть $\mathcal{X} = \mathbb{R}^d$, то самое очевидное расстояние – евклидова метрика, но возможны и другие варианты, некоторые из которых обобщаются и на нечисловые признаки. Большинство методов метрической кластеризации зависит от понятия «центра масс», или *центроида* произвольного множества объектов, – точки, в которой обращается в минимум некоторая зависящая от расстояния величина, вычисляемая по всем объектам множества. Эта величина называется *разбросом* множества. Таким образом, хорошим считается кластеризатор, в котором сумма разбросов по всем кластерам – она называется *внутрикластерным разбросом* – намного меньше разброса всего набора данных.

Проведенный анализ позволяет определить проблему кластеризации как нахождение такого разбиения $D = D_1 \cup \dots \cup D_K$, которое минимизирует внутрикластерный разброс. Однако у этого определения есть несколько недостатков:

- ☞ так, поставленная задача имеет тривиальное решение: положить $K = |D|$, так чтобы каждый «кластер» содержал только один объект из D , тогда его разброс будет нулевым;
- ☞ если зафиксировать число кластеров K заранее, то задачу невозможно решить эффективно для больших наборов данных (она NP-трудная).

Первая проблема – это аналог переобучения в задаче кластеризации. Ее можно решить, штрафуя за большое K . Впрочем, в большинстве практических подходов предполагается, что можно выдвинуть обоснованную гипотезу о величине K . Остается вторая проблема: вычислительная невозможность найти глобально оптимальное решение задачи для большого набора данных. Такая ситуация часто встречается в информатике и разрешается одним из двух способов:

- ☞ применить эвристический подход, который находит «достаточно хорошее» решение, пусть и не лучшее из возможных;
- ☞ ослабить условия задачи, в данном случае разрешить «мягкую» кластеризацию, когда объект может быть «неполным» членом нескольких кластеров.

В большинстве алгоритмов кластеризации, в том числе в алгоритме K средних, применяется эвристический подход. Алгоритмы мягкой кластеризации тоже существуют, к ним относятся, например, ☞ *EM-алгоритм* (раздел 9.4) и ☞ *разложение матрицы* (раздел 10.3). На рис. 3.5 показаны оба подхода: эвристический и мягкой кластеризации. Отметим, что мягкая кластеризация обобщает понятие разбиение – точно так же, как оценка вероятности обобщает классификатор.

Представление модели кластеризации зависит от того, является она прогностической, дескриптивной или мягкой. Дескриптивную кластеризацию n точек по c кластерам можно представить с помощью *матрицы разбиения*: двоичной матрицы размерности $n \times c$, в каждой строке которой имеется ровно одна единица, а остальные элементы равны нулю (и по меньшей мере, одна единица в каждом столбце, иначе некоторые кластеры оказались бы пустыми). Мягкой кластеризации соответствует нормированная по строкам матрица $n \times c$. Прогностическая кластеризация разбивает все пространство объектов, и потому матричное пред-

ставление для нее не подходит. Как правило, в методах прогностической кластеризации кластер представляется своим *центроидом*: в таком случае границы кластеров образованы прямыми, совокупность которых называется *диаграммой Вороного* (рис. 3.5 слева). В общем случае каждый кластер можно представить плотностью вероятности, а границы проходят там, где плотности соседних кластеров одинаковы; при этом возможно появление кластеров с нелинейными границами.

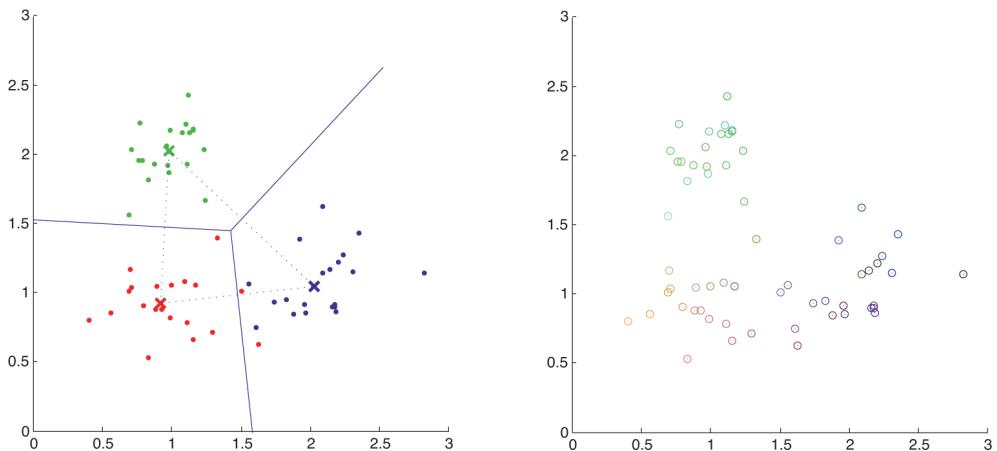


Рис. 3.5. (Слева) Пример прогностической кластеризации. Цветные точки – три выборки с двумерным гауссовым распределением с центрами $(1,1)$, $(1,2)$ и $(2,1)$. Крестики и сплошные линии – центроиды и границы кластеров, найденных методом З средних. **(Справа)** Мягкая кластеризация тех же данных, найденная методом разложения матрицы

Пример 3.9 (представления кластеризации). Центроиды кластеров, показанных на рис. 3.5 слева, можно описать матрицей 3×2 :

$$\begin{pmatrix} 0.92 & 0.93 \\ 0.98 & 2.02 \\ 2.03 & 1.04 \end{pmatrix}$$

Следующие матрицы $n \times c$ представляют дескриптивную кластеризацию (слева) и мягкую кластеризацию (справа) заданных точек:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ \dots & \dots & \dots \end{pmatrix} \quad \begin{pmatrix} 0.40 & 0.30 & 0.30 \\ 0.40 & 0.51 & 0.09 \\ 0.44 & 0.29 & 0.27 \\ 0.35 & 0.08 & 0.57 \\ \dots & \dots & \dots \end{pmatrix}$$

Интересный вопрос: как оценивать модели кластеризации? В отсутствие размеченных данных мы не можем воспользоваться тестовым набором так, как в задачах классификации или регрессии. В качестве меры качества кластеризации можно взять внутрикластерный разброс. Для прогностической кластеризации можно вычислить внутрикластерный разброс на зарезервированных данных, которые не использовались для построения кластеров. Другой способ оценки качества кластеризации появляется, если мы что-то знаем об объектах, которые должны или, наоборот, не должны попасть в один кластер.

Пример 3.10 (оценивание качества кластеризации). Предположим, что имеется пять объектов, которые, как нам кажется, должны быть кластеризованы следующим образом: $\{e1,e2\}, \{e3,e4,e5\}$. Следовательно, из $5 \cdot 4 = 20$ возможных пар четыре считаются «обязательно связанными», а остальные 16 – «обязательно разделенными». Требуется оценить модель, которая построила кластеры $\{e1,e2,e3\}, \{e4,e5\}$, – здесь две из «обязательно связанных» пар действительно оказались в одном кластере ($e1-e2, e4-e5$), а две другие – нет. Это можно представить в виде следующей таблицы:

	Связаны	Разделены	
Должны быть связаны	2	2	4
Должны быть разделены	2	14	16
	4	16	20

Эту таблицу можно рассматривать как таблицу сопряженности 2×2 и соответственно оценить. Например, можно взять долю пар на «хорошей» диагонали: $16/20 = 0.8$. В задаче классификации мы называли бы этот показатель верностью, а в контексте кластеризации он называется *индексом Рэнда*.

Отметим, что обычно «обязательно разделенных» пар гораздо больше, чем «обязательно связанных», и было бы неплохо это как-то компенсировать. Один из способов – вычислить среднее гармоническое точности и полноты (полнота – то же самое, что частота истинно положительных результатов, см. табл. 2.3) – величину, которая в литературе по информационному поиску называется *F-мерой*¹. Точность вычисляется по левому столбцу таблицы сопряженности, а полнота – по верхней строке; в результате правая нижняя клетка («обязательно разделенные пары, которые по ошибке попали в один кластер») игнорируется, а это именно то, что нам и надо. В примере и полнота, и точность равны $2/4 = 0.5$, как и F-мера. Это показывает, что относительно хороший индекс Рэнда учитывается главным образом обязательными разделенными парами, которые оказываются в разных кластерах.

¹ Среднее гармоническое точности и полноты равно $\frac{2}{1/prec + 1/rec} = \frac{2 \cdot prec \cdot rec}{prec + rec}$. Вообще, среднее гармоническое хорошо подходит для усреднения отношений, см. замечание 10.1 на стр. 311.

Другие дескриптивные модели

Чтобы довершить рассмотрение нашего каталога задач машинного обучения, познакомимся еще с двумя дескриптивными моделями, одна из которых обучена с учителем по размеченным данным, а другая – без учителя.

Модели подгрупп не пытаются аппроксимировать помечающую функцию, а ставят целью найти подмножества данных, в которых распределение по классам существенно отличается от распределения в генеральной совокупности в целом. Формально *подгруппой* называется отображение $\hat{g} : D \rightarrow \{\text{true}, \text{false}\}$, ищется она по результатам обучения на наборе помеченных примеров $(x_i, l(x_i))$, где $l : \mathcal{X} \rightarrow \mathcal{C}$ – истинная помечающая функция. Отметим, что \hat{g} – характеристическая функция множества $G = \{x \in D | \hat{g}(x) = \text{true}\}$, которое называется *расширением* подгруппы. Отметим также, что в качестве области определения подгруппы мы используем только предъявленные данные D , а не все пространство объектов \mathcal{X} , поскольку это дескриптивная модель.

Пример 3.11 (выявление подгрупп). Представьте, что вам нужно вывести на рынок новую версию успешного продукта. У вас имеется база данных людей, которым была разослана информация о предыдущей версии. В ней хранятся различные сведения демографического, экономического и социального характера, а также о том, купил человек продукт или нет. Если бы мы стали строить классификатор или ранжировщик для нахождения наиболее вероятных покупателей продукта, то вряд ли он оказался бы лучше классификатора по мажоритарному классу (как правило, продукт покупает сравнительно немного людей). Однако в действительности нас интересует нахождение достаточно больших групп людей, в которых доля покупателей значительно выше, чем в генеральной совокупности в целом. Именно эти люди могут стать целью маркетинговой кампании, а всех остальных можно игнорировать.

Подгруппа – это, по существу, бинарный классификатор, поэтому один из способов разработать систему выявления подгрупп состоит в том, чтобы адаптировать какой-нибудь существующий алгоритм обучения классификатора. Возможно, потребуется всего лишь выбрать поисковую эвристику, так чтобы она отражала конкретную цель подгруппы (выявить подмножества данных с существенно отличающимся распределением по классам).

Как отличить интересные группы от неинтересных? Для этого можно построить таблицу сопряженности, похожую на используемые в бинарной классификации. Для трех классов такая таблица выглядит следующим образом:

	<i>В подгруппе</i>	<i>Не в подгруппе</i>	
Помечен C_1	g_1	$C_1 - g_1$	C_1
Помечен C_2	g_2	$C_2 - g_2$	C_2
Помечен C_3	g_3	$C_3 - g_3$	C_3
	$ G $	$ D - G $	$ D $

Здесь $g_i = |\{x \in D | \hat{g}(x) = \text{true} \wedge l(x) = C_i\}|$, а C_i – сокращенное обозначение величины $|\{x \in D | l(x) = C_i\}|$. Далее есть несколько возможностей. Одна идея –

измерить, насколько распределение по классам в левом столбце отличается от распределения по классам в столбце маргиналов строк (самый правый столбец). Как мы увидим ниже (пример 6.6), это сводится к использованию разновидности средней полноты в качестве показателя оценки. Другая идея – рассматривать подгруппу как разделение решающего дерева и заимствовать критерий разделения из обучения решающих деревьев (раздел 5.1). Можно также использовать критерий χ^2 для оценки того, насколько каждое g_i отличается от величины, которую можно было бы ожидать, исходя из маргиналов C_i и $|G|$. У этих показателей есть общая особенность: они предпочитают, чтобы распределение по классам в подгруппе и ее дополнении отличалось от общего распределения в D , а кроме того, предпочитают большие подгруппы малым. Большинство таких метрик симметричны в том смысле, что одинаково оценивают подгруппу и ее дополнение, откуда следует, что они также предпочитают большие дополнения малым. Иными словами, предпочтительны подгруппы, размер которых примерно равен половине размера всего множества (при прочих равных условиях).

Теперь я приведу пример обучения дескриптивных моделей без учителя. Ассоциации – это вещи, которые обычно встречаются вместе. Например, при анализе корзины покупок нас интересует, какие товары часто покупают вместе. Пример ассоциативного правила – «**если пиво, то чипсы**»; это означает, что люди, покупающие пиво, часто покупают также картофельные чипсы. Выявление ассоциативных правил начинается с поиска значений признаков, которые часто встречаются вместе. Здесь есть поверхностное сходство с выявлением подгрупп, но эти так называемые «частые предметные наборы» выявляются без всякого вмешательства учителя, не нуждаясь в размеченных обучающих данных. Далее из предметных наборов выводятся правила, описывающие совместную встречаемость значений признаков. Эти ассоциативные правила имеют вид «если – то» и, стало быть, аналогичны правилам классификации, с тем отличием, что часть «то» не ограничивается переменной конкретного класса, а может содержать любой признак (и даже несколько признаков). Вместо того чтобы адаптировать существующий алгоритм обучения, нам необходим новый алгоритм, который сначала находит частые предметные наборы, а затем преобразует их в ассоциативные правила. В ходе этого процесса нужно принимать во внимание различные статистические критерии, чтобы избежать порождения тривиальных правил.

Пример 3.12 (выявление ассоциативного правила). На бензозаправке большинство клиентов покупают бензин. Это означает, что должно быть много частых предметных наборов, в которые входит бензин, например {газета, бензин}. Это наводит на мысль о построении ассоциативного правила «**если газета, то бензин**», однако оно легко предсказуемо, потому что {бензин} сам по себе является частым предметным набором (и, очевидно, не менее частым, чем {газета, бензин}). Более интересно обратное правило «**если бензин, то газета**», которое выражает тот факт, что значительная часть людей, покупающих бензин, покупает также и газету.

Прослеживается очевидная связь с выявлением подгрупп в том смысле, что ассоциативные правила также выявляют подмножества с иным распределением по классам, чем у всего множества данных, конкретно в отношении части «то» правила. Различие же в том, что часть «то» – не фиксированная заранее целевая переменная, а определяется в процессе выявления. Выявление подгрупп и ассоциативных правил будет впоследствии рассмотрено в контексте обучения правил в разделе 6.3.

3.4 За пределами бинарной классификации: итоги и литература для дальнейшего чтения

Хотя бинарная классификация – важная задача в машинном обучении, существует и много других, часть из них мы рассмотрели в этой главе.

- ☞ В разделе 3.1 мы обсудили задачи классификации с числом классов больше двух. В последующих главах мы увидим, что в некоторых моделях эта ситуация обрабатывается вполне естественно, но если модель принципиально двухклассовая (как, например, линейные модели), то для решения приходится комбинировать задачи двоичной классификации. Одна из основных идей – воспользоваться матрицей кодов для объединения результатов нескольких бинарных классификаторов – была предложена в работе Dietterich, Bakiri (1995) под названием «исправляющие ошибки выходные коды» и развита в работе Allwein et al. (2000). Мы также рассмотрели способы получения оценок в случае, когда классов больше двух, и определения их качества с помощью обобщения площади под кривой РХП на несколько классов. Одно из таких обобщений **AUC** было предложено и проанализировано в работе Hand, Till (2001). Эвристическая процедура повторного взвешивания многоклассовых оценок в примере 3.6 была предложена в работе Lachiche, Flach (2003), а в работе Bourke et al. (2008) продемонстрировано, что она достигает хорошего качества, по сравнению с рядом альтернативных подходов.
- ☞ Раздел 3.2 был посвящен регрессии – предсказанию вещественнонозначного целевого значения. Эту классическую проблему анализа данных изучал еще Карл Фридрих Гаусс в конце XVIII века. Естественно использовать в качестве функции потерь квадрат невязки, хотя это сопряжено с повышенной чувствительностью к выбросам. В этой области чаще всего применяются ранжирующие модели, хотя возможно также использование группирующей модели, которая разбивает пространство объектов на сегменты, допускающие простую локальную модель. Поскольку часто имеется возможность интерполировать множество точек точно (например, полиномом высокой степени), следует проявлять осторожность во избежание

переобучения. Отыскание правильного баланса между переобучением и недообучением иногда называют дилеммой смещения-дисперсии. Подробное обсуждение этого вопроса (включая и метафору мишени для дартса) можно найти в работе Rajnarayan, Wolpert (2010).

- ☞ В разделе 3.3 мы рассмотрели задачи обучения без учителя и дескриптивного обучения. Мы видели, что в случае дескриптивного обучения задача и проблема обучения совпадают. Модель кластеризации может быть прогностической или дескриптивной; в последнем случае она определяет классы вообще без учителя, после чего обученную модель можно применять к не предъявлявшимся ранее данным, как обычно. С другой стороны, дескриптивная кластеризация применяется только к имеющимся данным. Следует отметить, что различие между прогностической и дескриптивной кластеризациями не является универсально признанным в литературе; иногда термин «прогностическая кластеризация» используется в несколько ином смысле – как кластеризация одновременно по целевой переменной и признакам (Blockeel et al., 1998).
- ☞ Как и дескриптивная кластеризация, выявление ассоциативных правил – еще одна дескриптивная задача, решаемая без учителя. Впервые она была поставлена в работе Agrawal, Imielinski, Swami (1993) и породила многочисленные работы в области добычи данных. Выявление подгрупп – это разновидность обучения дескриптивных моделей с учителем, оно ставит целью отыскание подмножеств данных, в которых распределение целевой переменной по классам значительно отличается от распределения в генеральной совокупности. Впервые эта задача была изучена в работе Klosgen (1996), а затем распространена на более общее понятие добычи моделей исключений (exceptional model mining), позволяющее работать, в частности, с вещественными целевыми переменными, в работе Leman et al. (2008). Вопрос об обучении дескриптивных моделей без учителя – обширная тема, впервые поднятая в работе Tukey (1977).



Концептуальное обучение

Осудив разнообразные задачи в предыдущей главе, мы теперь обладаем достаточной подготовкой, чтобы приступить к рассмотрению моделей и алгоритмов машинного обучения. Эта и две следующие главы посвящены логическим моделям, фирменным знаком которых являются логические выражения для разбиения пространства объектов на сегменты и, следовательно, построения группирующих моделей. Цель заключается в том, чтобы найти такое разбиение, где данные в каждом сегменте наиболее однородны – с точки зрения решаемой задачи. Например, в случае классификации мы ищем разбиение, в каждом сегменте которого объект принадлежит преимущественно одному классу, а в задаче регрессии хорошим считается разбиение, для которого целевая переменная представляет собой простую функцию небольшого числа независимых переменных. Существуют два больших класса логических моделей: древовидные и на основе правил. Модель на основе правил состоит из набора импликаций, то есть правил вида «если – то», в которых часть «если» определяет сегмент, а часть «то» – поведение модели в этом сегменте. Древовидная модель – это специальный тип модели на основе правил, где части «если» всех правил организованы в виде дерева.

В этой главе мы рассмотрим методы обучения логических выражений, или **концептов**, на примерах. Эти методы лежат в основе как древовидных моделей, так и моделей на основе правил. В ходе концептуального обучения требуется только составить описание положительного класса, а все, что не отвечает этому описанию, помечать как принадлежащее отрицательному классу. Мы будем обращать особое внимание на упорядочение по степени общности, поскольку оно играет важную роль в логических моделях. В двух следующих главах мы будем рассматривать древовидные модели и модели на основе правил, которые выходят далеко за рамки концептуального обучения, поскольку могут работать с несколькими классами, оценивать вероятности, решать задачи регрессии и кластеризации.

Простейшими логическими выражениями являются равенства вида **Признак = Значение** и, в случае числовых признаков, неравенства вида **Признак < Значение**; они называются **литералами**. Составные булевые выражения строятся с помощью логических связок: **конъюнкция \wedge** , **дизъюнкция \vee** , **отрицания \neg** и **импликации \rightarrow** . Справедливы следующие эквивалентности (последние две называются **правилами де Моргана**):

$$\begin{aligned}\neg(A \wedge B) &\equiv \neg A \vee \neg B & \neg \neg A &\equiv A \\ \neg(A \vee B) &\equiv \neg A \wedge \neg B & A \rightarrow B &\equiv \neg A \vee B\end{aligned}$$

Если булево выражение A истинно для объекта x , мы будем говорить, что A *покрывает* x . Множество объектов, покрываемых выражением A , называется его *расширением* и обозначается $\mathcal{X}_A = \{x \in \mathcal{X} | A \text{ покрывает } x\}$, где \mathcal{X} – пространство объектов, которое играет роль универсального множества (см. замечание 2.1). Существует прямое соответствие между логическими связками и операциями над множествами, например: $\mathcal{X}_{A \wedge B} = \mathcal{X}_A \cap \mathcal{X}_B$, $\mathcal{X}_{A \vee B} = \mathcal{X}_A \cup \mathcal{X}_B$ и $\mathcal{X}_{\neg A} = \mathcal{X} \setminus \mathcal{X}_A$. Если $\mathcal{X}_A \supseteq \mathcal{X}_{A'}$, то говорят, что A *не менее общее, чем* A' , а если к тому же $\mathcal{X}_A \not\subseteq \mathcal{X}_{A'}$, то говорят, что A *более общее, чем* A' . Это *упорядочение по общности* вводит частичный порядок на множестве логических выражений в смысле, определенном в замечании 2.1. (Точнее, это частичный порядок на множестве классов эквивалентности по отношению к логической эквивалентности \equiv .) *Дизъюнктом* называется импликация $P \rightarrow Q$ – такая, что P – конъюнкция литералов, а Q – дизъюнкция литералов. С помощью приведенных выше эквивалентностей можно переписать такую импликацию в виде:

$$(A \wedge B) \rightarrow (C \vee D) \equiv \neg(A \wedge B) \vee (C \vee D) \equiv \neg A \vee \neg B \vee C \vee D,$$

а следовательно, дизъюнкт можно рассматривать также как дизъюнкцию литералов или их отрицаний. Любое логическое выражение можно представить в виде конъюнкции дизъюнктов, такое представление называется *конъюнктивной нормальной формой (КНФ)*. Можно также представить любое логическое выражение в виде дизъюнкции конъюнкций литералов или их отрицаний, такое представление называется *дизъюнктивной нормальной формой (ДНФ)*. *Правилом* называется дизъюнкт $A \rightarrow B$, в котором B – одиночный литерал; такие выражения называются также *дизъюнктами Хорна*, по имени американского логика Альфреда Хорна.

Замечание 4.1. Некоторые понятия и обозначения математической логики

4.1 Пространство гипотез

Простейший случай концептуального обучения возникает, когда при описании концептов мы ограничиваемся логическими выражениями, состоящими из конъюнкций литералов (см. обзор важных понятий и обозначений математической логики в замечании 4.1). Это иллюстрируется в следующем примере¹.

Пример 4.1 (обучение конъюнктивных концептов). Предположим, что имеется ряд морских животных, которые, как мы подозреваем, относятся к одному виду. Мы измеряем их длину в метрах, смотрим, есть ли у них жабры и клововидный выступ, много ли у них зубов. С помощью этих признаков первое животное можно описать такой конъюнкцией:

$$\text{Длина} = 3 \wedge \text{Жабры} = \text{нет} \wedge \text{Клововидный выступ} = \text{да} \wedge \text{Зубы} = \text{много}.$$

Следующее животное имеет такие же характеристики, но на метр длиннее, поэтому мы отбрасываем условие длины и обобщаем конъюнкцию:

$$\text{Жабры} = \text{нет} \wedge \text{Клововидный выступ} = \text{да} \wedge \text{Зубы} = \text{много}.$$

Третье животное также длиной 3 метра, имеет клововидный выступ и не имеет жабр, но зубов у него мало, поэтому наше описание принимает вид:

$$\text{Жабры} = \text{нет} \wedge \text{Клововидный выступ} = \text{да}.$$

Все остальные животные подходят под это описание, и мы, наконец, приходим к выводу, что перед нами какой-то дельфин.

¹ Идея заимствована с сайта www.cwtstrandings.org.

Хотя этот пример был совсем простым, пространство возможных концептов – обычно его называют *пространством гипотез* – уже довольно велико. Предположим, что есть три разные длины: 3, 4 и 5 метров, а остальные три признака имеют по два значения. Тогда всего будет $3 \cdot 2 \cdot 2 \cdot 2 = 24$ возможных объекта. Сколько существует конъюнктивных концептов, в которых встречаются эти признаки? Чтобы ответить на этот вопрос, мы будем рассматривать отсутствие признака как дополнительное «значение». Тогда получается $4 \cdot 3 \cdot 3 \cdot 3 = 108$ разных концептов. И хотя, на первый взгляд, это много, нужно понимать, что число возможных расширений – множеств объектов – гораздо больше: 2^{24} , больше 16 миллионов! Таким образом, если взять случайный набор объектов, то шансы на то, что мы не сможем найти конъюнктивный концепт, который в точности описывает эти экземпляры, превышают 100 000 к 1. На самом деле это хорошо, потому что вынуждает исследователя обобщать, выходя за рамки обучающих данных, и покрывать объекты, которых он раньше не видел. На рис. 4.1 показано это пространство гипотез с использованием упорядочения по общности (то есть отношения между расширениями концептов, см. замечание 4.1).

Наименьшее обобщение

Если исключить все концепты, которые не покрывают хотя бы один из объектов в примере 4.1, то пространство гипотез сократится до 32 конъюнктивных концептов (рис. 4.2). Настояв на том, чтобы любая гипотеза покрывала все три объекта, мы оставим всего четыре концепта, из которых наименее общий приведен в примере. Это называется *наименьшим обобщением* (*LGG*). Алгоритм 4.1 формализует эту процедуру, которая сводится просто к повторному применению попарной операции LGG (алгоритм 4.2) к объекту и текущей гипотезе, поскольку они имеют одну и ту же логическую форму. Строение пространства гипотез гарантирует, что результат не зависит от порядка обработки объектов.

Интуитивно наименьшее обобщение двух объектов – это ближайший концепт в пространстве гипотез, в котором пересекаются пути вверх от обоих объектов. То, что такая точка определена однозначно, – специальное свойство многих логических пространств гипотез, которому можно найти полезное применение в ходе обучения. Точнее, такое пространство гипотез образует *решетку*: частичный порядок, в котором у любых двух элементов есть *точная верхняя грань* (*sup*) и *точная нижняя грань* (*inf*). Таким образом, наименьшее обобщение множества объектов – это не что иное, как точная верхняя грань объектов в этой решетке. Кроме того, это точная нижняя грань множества всех обобщений объектов: любое возможное обобщение является не менее общим, чем LGG. В этом, очень точно определенном смысле *наименьшее обобщение – это самое консервативное обобщение, которое можно вывести из данных*.

Если мы готовы к авантюрам, то можем выбрать и менее общую гипотезу, например: **Жабры = нет** или **Клюковидный выступ = да**. Однако вряд ли нам захочется выбирать самую общую гипотезу, которая просто утверждает, что все жи-

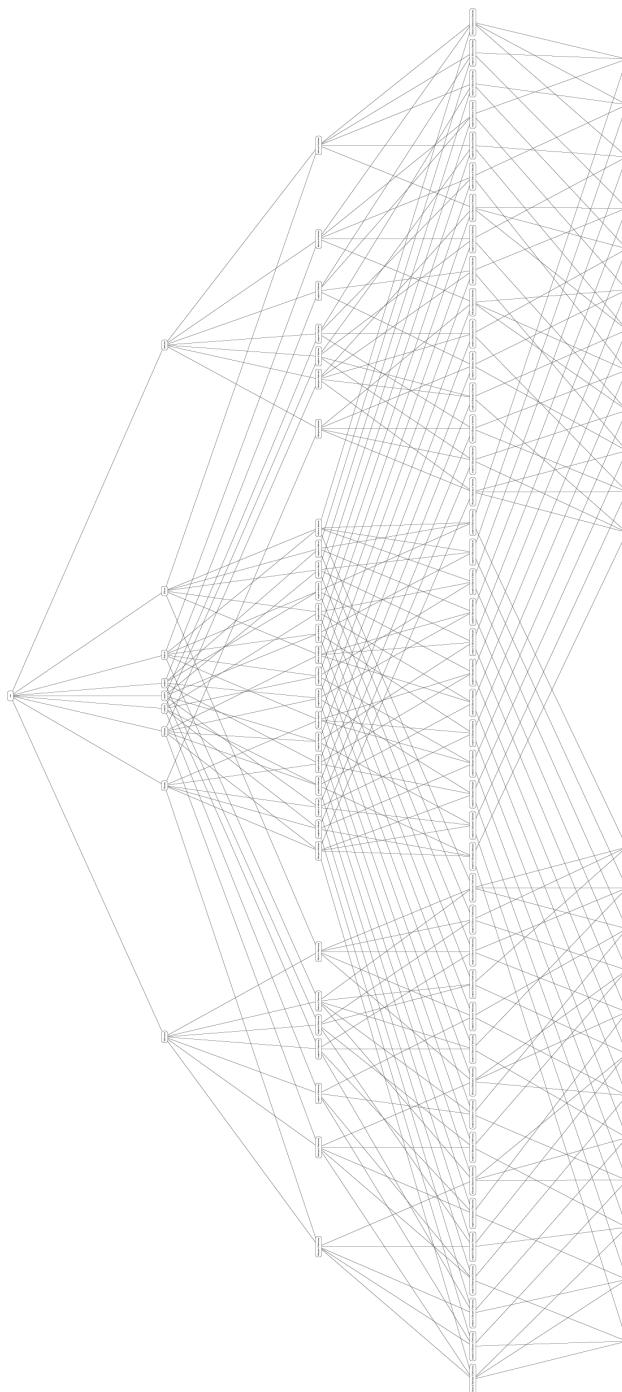


Рис. 4.1. Пространство гипотез, соответствующее примеру 4.1. Нижняя строка соответствует 24 возможным объектам, которые представлены полными конъюнкциями, содержащими по четырем литералам. В расположенной прямо над ней строке присутствуют все 44 концепта с тремя литералами в каждом; затем 30 концептов с двумя литералами, 9 концептов с одним литералом, и, наконец, верхний концепт – пустая конъюнкция, которая всегда истинна и потому покрывает все возможные объекты. Концепты на соседних уровнях соединены линией, если расположенный выше является более общим, чем расположенный ниже (то есть расширение верхнего концепта содержит в себе расширение нижнего)

вотные – дельфины, поскольку это было бы очевидным переобобщением. Для предотвращения переобобщения очень полезны отрицательные примеры.

Алгоритм 4.1. $\text{LGG-Set}(D)$ – найти наименьшее обобщение множества объектов

```

Вход : данные  $D$ .
Выход : логическое выражение  $H$ .
1  $x \leftarrow$  первый объект из  $D$ ;
2  $H \leftarrow x$ ;
3 while остаются объекты do
4    $x \leftarrow$  следующий объект из  $D$ ;
5    $| H \leftarrow \text{LGG}(H, x)$ ; // например, LGG-Conj (алг. 4.2) или LGG-Conj-ID (алг. 4.3)
6 end
7 return  $H$ 
```

Пример 4.2 (отрицательные примеры). В примере 4.1 мы наблюдали таких дельфинов:

- p1: **Длина** = 3 \wedge **Жабры** = нет \wedge **Клювовидный выступ** = да \wedge **Зубы** = много;
- p2: **Длина** = 4 \wedge **Жабры** = нет \wedge **Клювовидный выступ** = да \wedge **Зубы** = много;
- p3: **Длина** = 3 \wedge **Жабры** = нет \wedge **Клювовидный выступ** = да \wedge **Зубы** = мало.

Предположим, что следующим мы наблюдаем животное, которое явно не относится к этому виду, – отрицательный пример. Оно описывается такой конъюнкцией:

- n1: **Длина** = 5 \wedge **Жабры** = да \wedge **Клювовидный выступ** = да \wedge **Зубы** = много.

Этот отрицательный пример позволяет исключить некоторые обобщения, которые до сих пор были возможны; в частности, исключается концепт **Клювовидный выступ** = да, а также пустой концепт, который утверждает, что все животные являются дельфинами.

Данный процесс показан на рис. 4.3. Теперь остались только две гипотезы, из которых одна – наименее общая, а другая – наиболее общая.

Внутренняя дизъюнкция

Возможно, глядя на этот и предыдущий примеры, вы пришли к выводу, что всегда существует и единственная наиболее общая гипотеза, но в общем случае это не так. Для демонстрации данного факта мы немного обогатим наш логический язык, допустим ограниченную форму дизъюнкции, которая называется **внутренней дизъюнкцией**. Идея очень проста: если мы наблюдаем дельфина длиной 3 метра и другого дельфина длиной 4 метра, то может возникнуть желание добавить к концепту условие вида «длина равна 3 или 4 метрам». Мы будем записывать это в виде **Длина** = [3,4], что логически означает **Длина** = 3 \vee **Длина** = 4. Разумеется, это имеет смысл, лишь если значений больше двух; внутренняя дизъюнкция **Зубы** = [много, мало] всегда истинна и потому может быть опущена.

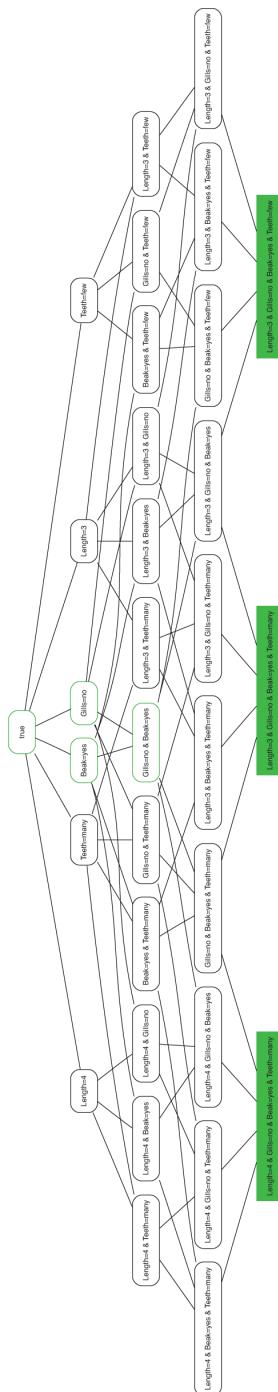


Рис. 4.2. Часть пространства гипотез на рис. 4.1, содержащая только концепты, более общие, чем хотя бы один из трех объектов в нижней строке. Лишь четыре конъюнкции, обведенные зеленой рамкой в верхней части рисунка, являются более общими, чем все три объекта, а наименее общая из них – **Жабры = нет** & **Клюковидный выступ = да** (*Gills = no* & *Beak = yes*). Можно заметить, что для вывода этого концепта было бы достаточно самого левого и самого правого объектов

Алгоритм 4.2. $\text{LGG-Conj}(x, y)$ – найти наименьшее конъюнктивное обобщение двух конъюнкций

Вход : конъюнкции x, y .

Выход : конъюнкция z .

- 1 $z \leftarrow$ конъюнкция всех литералов, общих для x и y ;
 - 2 **return** z
-

Пример 4.3 (внутренняя дизъюнкция). При тех же трех положительных примерах, что в примере 4.1, вторую и третью гипотезы теперь можно записать в виде:

Длина = [3,4] \wedge Жабры = нет \wedge Клювовидный выступ = да \wedge Зубы = много

и

Длина = [3,4] \wedge Жабры = нет \wedge Клювовидный выступ = да.

Мы можем опустить любое из трех условий в последнем наименьшем обобщении, не покрыв при этом отрицательный пример из примера 4.2. Обобщая и дальше до одиночных условий, мы видим, что гипотезы **Длина** = [3,4] и **Жабры** = нет по-прежнему годятся, а **Клювовидный выступ** = да не годится, потому что покрывает и отрицательный пример.

Алгоритм 4.3 уточняет, как можно вычислить LGG двух конъюнкций с использованием внутренней дизъюнкции. Функция **Combine-ID**(v_x, v_y) возвращает $[v_x, v_y]$, если v_x и v_y – константы, и их объединение, если v_x или v_y уже являются множествами значений, например: **Combine-ID**([3,4], [4,5]) = [3,4,5].

Алгоритм 4.3. $\text{LGG-Conj-ID}(x, y)$ – найти наименьшее конъюнктивное обобщение двух конъюнкций, применяя внутреннюю дизъюнкцию

Вход : конъюнкции x, y .

Выход : конъюнкция z .

- 1 $z \leftarrow \text{true}$;
 - 2 **for** каждого признака f **do**
 - 3 | **if** $f = v_x$ – конъюнкт от x и $f = v_y$ – конъюнкт от y **then**
 - 4 | | прибавить $f = \text{Combine-ID}(v_x, v_y)$ к z ; // **Combine-ID**: см. в тексте
 - 5 | **end**
 - 6 **end**
 - 7 **return** z
-

4.2 Пути в пространстве гипотез

По рис. 4.4 очевидно, что в данном примере мы имеем не одну, а две наиболее общие гипотезы. Еще можно заметить, что *каждый концепт между наименее общим и одним из наиболее общих также является допустимой гипотезой*, то есть

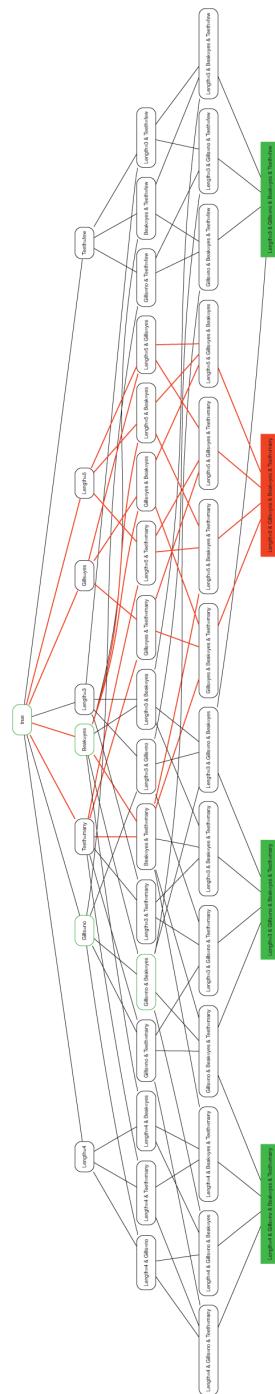


Рис. 4.3. Отрицательный пример может исключить некоторые обобщения LGG положительных примеров. Каждый концепт, соединенный **красным** путем с отрицательным примером, покрывает этот пример и потому не может быть гипотезой. Только две конъюнкции покрывают все положительные и ни одного отрицательного примера: **Жабры = нет \wedge Клевовидный выступ = да (Gills = no \wedge Weak = yes)** и **Жабры = нет (Gills = no)**

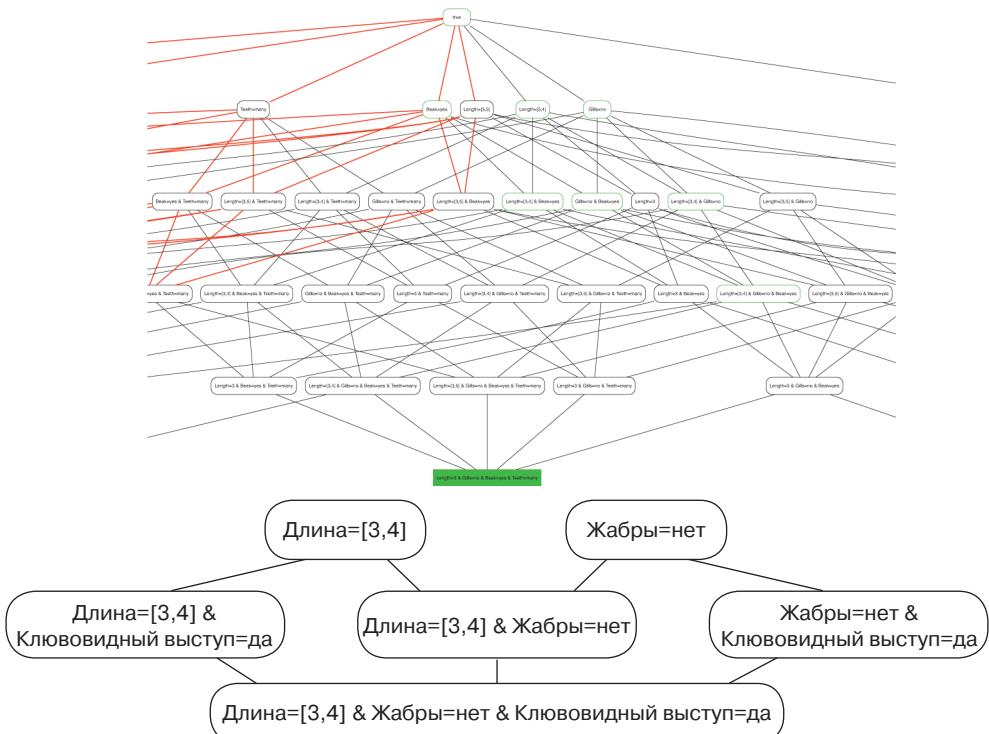


Рис. 4.4. (Сверху) Картина расширенного пространства гипотез, получающегося, когда для признака «Длина» (*Length*) используется внутренняя дизъюнкция. Нам нужно на один шаг обобщения больше, чтобы пройти вверх по пути от полностью специфицированного примера до пустой конъюнкции. **(Снизу)** Пространство версий состоит из одной наименее общей гипотезы, двух наиболее общих и трех, лежащих между ними

покрывает все положительные примеры и ни одного отрицательного. В математической терминологии это означает, что множество согласующихся с данными гипотез **выпукло**, а это значит, что можно интерполировать любые два его элемента, и если мы найдем концепт, менее общий, чем один из них, и более общий, чем другой, то этот концепт также будет являться элементом множества. Это, в свою очередь, означает, что можно описать множество всех возможных гипотез с помощью его наименее общего и наиболее общего элемента. Эти наблюдения суммированы в следующем определении.

Определение 4.1 (пространство версий). Концепт называется **полным**, если он покрывает все положительные примеры. Концепт называется **непротиворечивым**, если он не покрывает ни одного отрицательного примера. **Пространством версий** называется множество всех полных и непротиворечивых концептов. Это множество выпукло и однозначно определяется двумя своими элементами: **наименее и наиболее общим**.

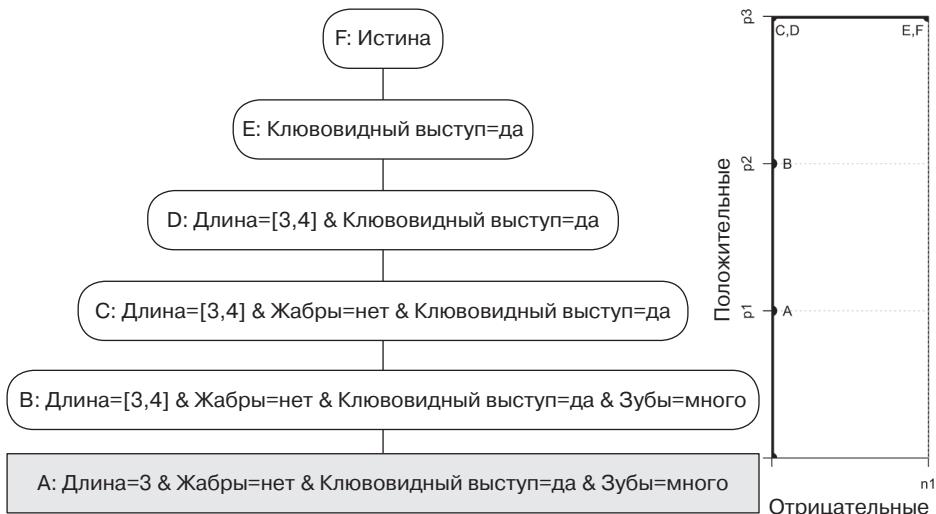


Рис. 4.5. (Слева) Путь в пространстве гипотез на рис. 4.3 от одного из положительных примеров (p_1 , см. пример 4.2) вверх до пустого концепта. Концепт А покрывает только один пример, концепт В – еще один пример, С и D принадлежат пространству версий и потому покрывают все три положительных примера, Е и F покрывают также отрицательный пример. **(Справа)** Соответствующая кривая покрытия с ранжированием $p_1 - p_2 - p_3 - n_1$

Мы можем установить полезную связь между пространствами логических гипотез и графиками покрытия, описанными в главе 2. Предположим, что мы идем вверх по пути в пространстве гипотез, ведущем от положительного примера через ряд обобщений к пустому концепту. Последний по построению покрывает все положительные и отрицательные примеры и потому занимает правую верхнюю точку (*Neg*, *Pos*) на графике покрытия. Начальная точка, будучи одиночным положительным примером, занимает точку (0,1) на графике покрытия. На практике принято пополнять пространство гипотез нижним элементом, который не покрывает ни одного примера и потому является менее общим, чем любой другой концепт. Если взять эту точку в качестве начала пути, то окажется, что мы начинаем путешествие из левого нижнего угла (0,0) на графике покрытия.

Перемещение вверх в пространстве гипотез через обобщения означает, что количество покрытых положительных примеров остается неизменным или увеличивается, но уменьшаться не может. Иными словами, *путь вверх в пространстве гипотез соответствует кривой покрытия* и, следовательно, некоторому ранжированию. На рис. 4.5 это показано для нашего сквозного примера. Выбранный путь – один из многих возможных; отметим, однако, что если подобный путь включает элементы пространства версий, то соответствующая кривая покрытия проходит через «вершину РХП» (0, *Pos*) и $AUC = 1$. Иначе говоря, такие пути оптимальны. Концептуальное обучение можно рассматривать как поиск оптимального пути в пространстве гипотез.

Вы можете спросить, что произойдет, если LGG положительных примеров покрывает один или несколько отрицательных? В таком случае любое обобщение LGG также будет противоречивым. Обратно, любая непротиворечивая гипотеза будет неполной. Отсюда следует, что пространство версий в этом случае пусто; мы говорим, что данные не являются *конъюнктивно отделыми*. Это иллюстрирует следующий пример.

Пример 4.4 (конъюнктивно неотделимые данные). Пусть имеются следующие пять положительных примеров (первые три такие же, как в примере 4.1):

- p1: *Длина = 3* \wedge *Жабры = нет* \wedge *Клювовидный выступ = да* \wedge *Зубы = много*;
- p2: *Длина = 4* \wedge *Жабры = нет* \wedge *Клювовидный выступ = да* \wedge *Зубы = много*;
- p3: *Длина = 3* \wedge *Жабры = нет* \wedge *Клювовидный выступ = да* \wedge *Зубы = мало*;
- p4: *Длина = 5* \wedge *Жабры = нет* \wedge *Клювовидный выступ = да* \wedge *Зубы = много*;
- p5: *Длина = 5* \wedge *Жабры = нет* \wedge *Клювовидный выступ = да* \wedge *Зубы = мало*

и следующие отрицательные (первый такой же, как в примере 4.2):

- n1: *Длина = 5* \wedge *Жабры = да* \wedge *Клювовидный выступ = да* \wedge *Зубы = много*;
- n2: *Длина = 4* \wedge *Жабры = да* \wedge *Клювовидный выступ = да* \wedge *Зубы = много*;
- n3: *Длина = 5* \wedge *Жабры = да* \wedge *Клювовидный выступ = нет* \wedge *Зубы = много*;
- n4: *Длина = 4* \wedge *Жабры = да* \wedge *Клювовидный выступ = нет* \wedge *Зубы = много*;
- n5: *Длина = 4* \wedge *Жабры = нет* \wedge *Клювовидный выступ = да* \wedge *Зубы = мало*.

Наименее общая полная гипотеза, как и раньше, *Жабры = нет* \wedge *Клювовидный выступ = да*, но она покрывает n5 и, значит, противоречива. Существуют также семь более общих непротиворечивых гипотез, но ни одна из них не является полной:

- | | |
|---|---|
| <i>Длина = 3</i> | (покрывает p1 и p3) |
| <i>Длина = [3, 5] \wedge Жабры = нет</i> | (покрывает все положительные примеры, кроме p2) |
| <i>Длина = [3, 5] \wedge Зубы = мало</i> | (покрывает p3 и p5) |
| <i>Жабры = нет \wedge Зубы = много</i> | (покрывает p1, p2 и p4) |
| <i>Жабры = нет \wedge Клювовидный выступ = нет</i> | |
| <i>Жабры = да \wedge Зубы = мало</i> | |
| <i>Клювовидный выступ = нет \wedge Зубы = мало</i> | |

Последние три из них не покрывают ни одного положительного примера.

Наиболее общие непротиворечивые гипотезы

Как следует из этого примера, нахождение наиболее общих непротиворечивых гипотез – задача, существенно более сложная, чем нахождение наименее общих полных гипотез. По существу, процесс сводится к перебору. Алгоритм 4.4 возвращает все наиболее общие непротиворечивые специализации заданного концепта, причем под минимальной специализацией понимается та, которой можно достичь за один шаг вниз в решетке гипотез (например, путем добавления конъюнкта или удаления значения из внутренней дизъюнкции). Если вызвать этот алгоритм с C = *true*, то он вернет наиболее общие непротиворечивые гипотезы.

Алгоритм 4.4. MGConsistent(C, N) – найти наиболее общие непротиворечивые специализации концепта

Вход	: концепт C , отрицательные примеры N .
Выход	: множество концептов S .

- 1 **if** C не покрывает ни одного элемента N **then return** $\{C\}$;
- 2 $S \leftarrow \emptyset$
- 3 **for** каждой минимальной специализации C' концепта C **do**
- 4 | $S \leftarrow S \cup \text{MGConsistent}(C', N)$;
- 5 **end**
- 6 **return** S

На рис. 4.6 показаны путь в пространстве гипотез из примера 4.4 и соответствующая кривая покрытия. Мы видим, что путь проходит через три непротиворечивые гипотезы, которые представлены последовательными точками на оси y графика покрытия. Остальные три гипотезы полные и потому оказываются в верхней части графика; одна из них является наименьшим обобщением положительных примеров (D). Этой кривой покрытия соответствует ранжирование $p_3 - p_5 - [p_1, p_4] - [p_2, n_5] - [n_1-4]$, в котором имеется одна полушибка ранжирования из 25, так что $AUC = 0.98$. Мы можем выбрать концепт из ранжирования, применив технику, описанную в разделе 2.2. Например, предположим, что оптимизируемым критерием является верность классификации. В пространстве покрытия угловой коэффициент изолиний верности равен 1, поэтому мы сразу видим, что на рис. 4.6 оба концепта C и D (или E) достигают наилучшей вер-

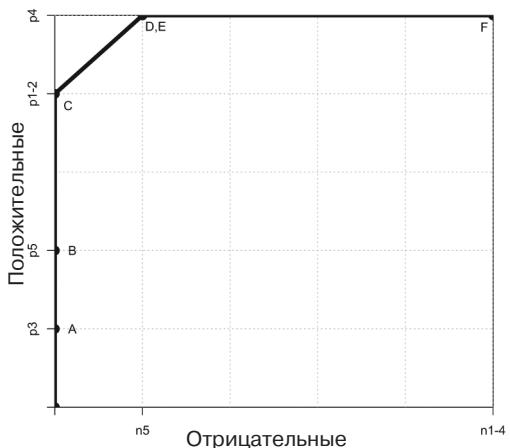
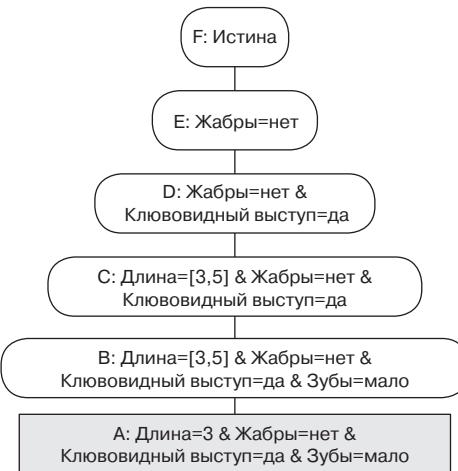


Рис. 4.6. (Слева) Путь в пространстве гипотез из примера 4.4. Концепт А покрывает один положительный пример (p_3); В покрывает еще один положительный пример (p_5); С покрывает все положительные примеры, кроме p_4 ; D – наименьшее обобщение всех пяти положительных примеров, но покрывает также отрицательный пример (n_5), как и Е. **(Справа)** Соответствующая кривая покрытия

ности. Если важнее качество классификации положительных примеров, то мы предпочтем полный, но противоречивый концепт D; если большую ценность представляет качество классификации отрицательных примеров, то выбираем неполный, но непротиворечивый концепт C.

Замкнутые концепты

Полезно поразмыслить над тем фактом, что концепты D и E оказались в одной и той же точке пространства покрытия. Это означает, что обобщение D до E путем исключения условия «**Клювовидный выступ = да**» не изменяет покрытие в терминах положительных и отрицательных примеров. Можно сказать, что из данных следует, что в контексте концепта E условие «**Клювовидный выступ = да**» подразумевается неявно. Концепт, который включает все неявно подразумеваемые условия, называется **замкнутым**. По существу, замкнутый концепт – это наименьшее обобщение всех примеров, которые он покрывает. Например, D и E покрывают все положительные примеры и п5; наименьшее обобщение этих шести примеров – «**Жабры = нет \wedge Клювовидный выступ = да**», то есть D. Математики говорят, что замыканием E является D, которое также является своим собственным замыканием, – отсюда и термин «замкнутый концепт». Это не означает, что D и E логически эквивалентны; напротив, так как $\mathcal{X}_D \subset \mathcal{X}_E$ (расширение D является собственным подмножеством E), в \mathcal{X} существуют объекты, покрываемые E, но не покрываемые D. Однако ни один из этих «свидетелей» не представлен в данных, и потому с точки зрения имеющихся данных D и E неразличимы. Как видно по рис. 4.7, рассмотрение одних лишь замкнутых концептов может существенно скратить пространство гипотез.

В этом разделе мы рассмотрели проблему обучения одного логического выражения, которое покрывало бы большую часть или все положительные примеры и небольшое число отрицательных (или вообще ни одного). Мы видели, что такие концепты являются элементами пространства гипотез с отношением порядка, индуцированным общностью, а обучение концепта можно интерпретировать как поиск хорошего пути в этом пространстве. Такой путь имеет естественную интерпретацию в качестве ранжировщика, что позволяет установить связь с кривыми покрытия или кривыми РХП. С другой стороны, требование о том, чтобы условие содержало только конъюнкцию литералов признак–значение, – слишком сильное ограничение; в следующем разделе мы поговорим о том, как можно его ослабить.

4.3 За пределами конъюнктивных концептов

Напомним (см. замечание 4.1), что конъюнктивная нормальная форма (КНФ) – это конъюнкция дизъюнкций литералов, или, эквивалентно, конъюнкция дизъюнктов. Конъюнкции литералов, которые мы рассматривали до сих пор, представляют собой тривиальные примеры КНФ, в которых каждая дизъюнкция состоит

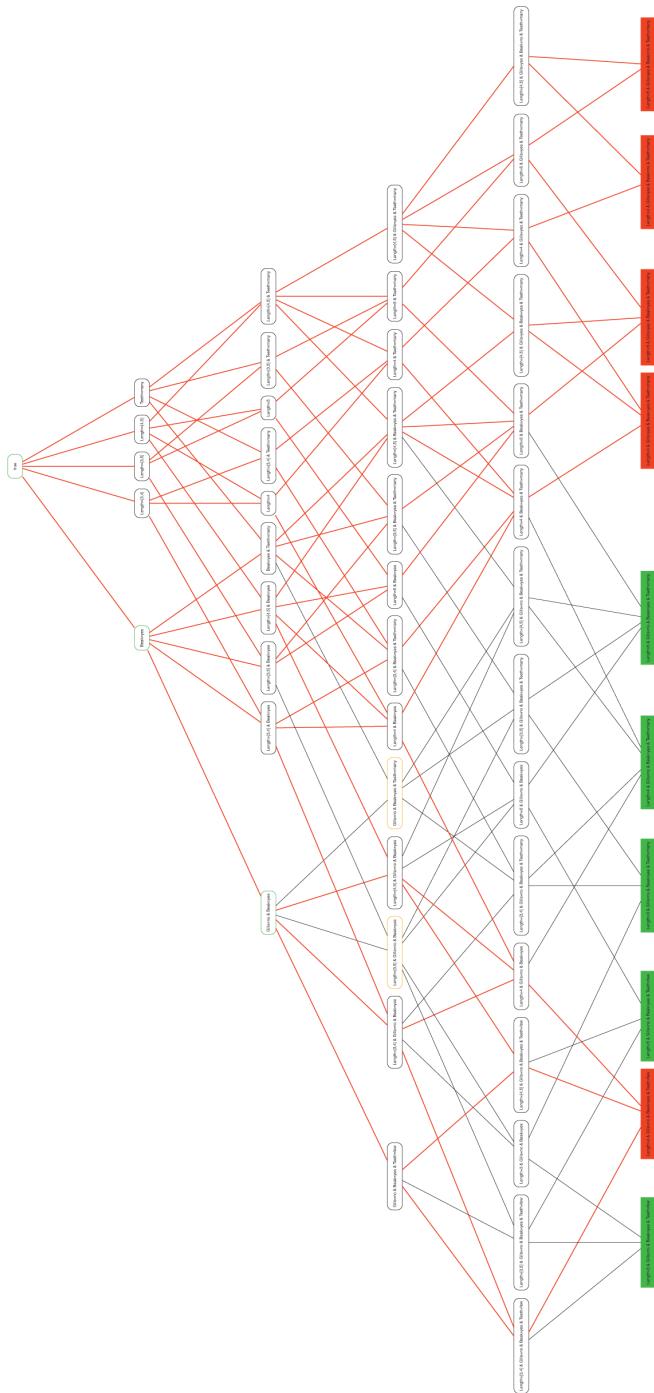


Рис. 4.7. Пространство гипотез значительно сокращается, если ограничиться только замкнутыми концептами. Существует три, а не четыре полных концепта (показаны **зеленым**), два, а не семь, наиболее общих непротиворечивых замкнутых концепта (**оранжевые**). Отметим, что два последних одновременно являются специализациями LGG положительных примеров, и потому можно выбрать путь, который включает как LGG, так и наиболее общие непротиворечивые гипотезы

из одного литерала. Выражения КНФ обладают гораздо большей выразительной силой, особенно если учесть, что литералы могут встречаться в нескольких дизъюнктах. Мы рассмотрим алгоритм обучения теорий Хорна, в которых каждый дизъюнкт $A \rightarrow B$ является дизъюнктом Хорна, то есть A – конъюнкция литералов, а B – одиночный литерал. Для простоты обозначений мы ограничимся булевыми признаками и будем писать F вместо $F = \text{true}$ и $\neg F$ вместо $F = \text{false}$. Ниже мы модифицируем пример с дельфинами, перейдя к булевым переменным **ManyTeeth** (то же, что **Зубы = много**), **Gills** (**Жабры**), **Short** (то же, что **Длина = 3**) и **Beak** (**Клювовидный выступ**).

При изучении конъюнктивных концептов основным интуитивно очевидным соображением было то, что наличие непокрытых положительных примеров позволяет сделать обобщение за счет исключения литералов из конъюнкции, тогда как покрытые отрицательные примеры требуют специализации путем добавления литералов. Это соображение остается справедливым и при изучении теорий Хорна, но теперь следует оперировать «дизъюнктами», а не «литералами». Таким образом, если теория Хорна не покрывает некоего положительного примера, то мы должны исключить все дизъюнкты, которые этот пример нарушают, при этом считается, что дизъюнкт $A \rightarrow B$ нарушает положительный пример, если все литералы в конъюнкции A в примере истинны, а литерал B ложен.

Ситуация становится интереснее, если рассматривать покрытые отрицательные примеры, потому что тогда нам нужно найти один или более дизъюнктов, которые следует добавить в теорию, чтобы исключить отрицательный пример. Предположим, к примеру, что наша текущая гипотеза покрывает отрицательный пример

$$\text{ManyTeeth} \wedge \text{Gills} \wedge \text{Short} \wedge \neg \text{Beak}.$$

Чтобы его исключить, мы можем добавить следующий дизъюнкт Хорна в теорию:

$$\text{ManyTeeth} \wedge \text{Gills} \wedge \text{Short} \rightarrow \text{Beak}.$$

Хотя существуют и другие дизъюнкты, позволяющие исключить отрицательный пример (например, **ManyTeeth** \rightarrow **Beak**), этот является наиболее специфичным, а потому опасность исключить еще и покрытые положительные примеры для него минимальна. Однако наиболее специфичный дизъюнкт, исключающий отрицательный пример, определен однозначно, только если в отрицательном примере ровно один литерал равен **false**. Например, если покрытый отрицательный пример имеет вид

$$\text{ManyTeeth} \wedge \text{Gills} \wedge \neg \text{Short} \wedge \neg \text{Beak},$$

то имеется выбор между следующими двумя дизъюнктами Хорна:

$$\begin{aligned} &\text{ManyTeeth} \wedge \text{Gills} \rightarrow \text{Short}, \\ &\text{ManyTeeth} \wedge \text{Gills} \rightarrow \text{Beak}. \end{aligned}$$

Отметим, что чем меньше литералов равны **true** в отрицательном примере, тем более общими являются дизъюнкты, которые их исключают.

Подход, примененный в алгоритме 4.5, заключается в том, чтобы добавить в гипотезу *все* такие дизъюнкты. Однако в нем есть две хитрости. Первая – введение списка S отрицательных примеров, по которому алгоритм периодически перестраивает гипотезу. Вторая состоит в том, что он не просто добавляет новые отрицательные примеры в список, а пытается найти отрицательные примеры, в которых меньше литералов равно **true**, поскольку это приводит к более общим дизъюнктам. Это возможно, если предположить, что у нас есть доступ к *оракулу членства* Mb , который может сказать, является конкретный пример членом концепта, который мы обучаем, или нет. Таким образом, в строке 7 алгоритма мы строим *пересечение* нового отрицательного примера x с существующим примером $s \in S$ – то есть пример, в котором в **true** установлены только те литералы, которые равны **true** как в x , так и в s , – и передаем результат z оракулу членства, чтобы он проверил его на принадлежность целевому концепту. В алгоритме также предполагается наличие доступа к *оракулу эквивалентности* Eq , который либо говорит нам, что текущая гипотеза h логически эквивалентна целевой формуле f , либо порождает *контрпример*, который может быть как ложноположительным (покрыт h , но не f), так и ложноотрицательным (покрыт f , но не h).

Алгоритм 4.5. *Horn(Mb, Eq)* – обучить конъюнкцию хорновских дизъюнктов, пользуясь оракулами членства и эквивалентности

Вход : оракул эквивалентности Eq , оракул членства Mb .

Выход : теория Хорна h , эквивалентная целевой формуле f .

```

1  $h \leftarrow \text{true};$  // конъюнкция хорновских дизъюнктов, первоначально пустая
2  $S \leftarrow \emptyset;$  // список отрицательных примеров, первоначально пустой
3 while  $Eq(h)$  возвращает контрпример  $x$  do
4   if  $x$  нарушает хотя бы один дизъюнкт  $h$  then //  $x$  ложноотрицательный
5     | специализировать  $h$ , удалив все дизъюнкты, которые  $x$  нарушает
6   else //  $x$  ложноположительный
7     | найти первый отрицательный пример  $s \in S$  – такой, что (i)  $z = s \cap x$  имеет
      | меньше литералов, равных true, чем  $s$ , и (ii)  $Mb(z)$  помечает его как
      | отрицательный;
8     | если такой пример существует, заменить  $s$  в  $S$  на  $z$ , иначе добавить
      |  $x$  в конец  $S$ ;
9    $h \leftarrow \text{true};$ 
10  for всех  $s \in S$  do // перестроить  $h$  по  $S$ 
11     $p \leftarrow$  конъюнкция литералов, равных true, в  $s$ ;
12     $Q \leftarrow$  множество литералов, равных false, в  $s$ ;
13    for всех  $q \in Q$  do  $h \leftarrow h \wedge (p \rightarrow q);$ 
14  end
15 end
16 end
17 return  $h$ 
```

Пример 4.5 (обучение теории Хорна). Пусть целевая теория такова:

$$(\text{ManyTeeth} \wedge \text{Short} \rightarrow \text{Beak}) \wedge (\text{ManyTeeth} \wedge \text{Gills} \rightarrow \text{Short}).$$

У этой теории 12 положительных примеров: в восьми `ManyTeeth` равно `false`, еще в двух `ManyTeeth` равно `true`, но `Gills` и `Short` равны `false` и еще в двух `ManyTeeth`, `Short` и `Beak` равны `true`. В таком случае отрицательные примеры таковы:

$$\begin{aligned} n1: & \text{ManyTeeth} \wedge \text{Gills} \wedge \text{Short} \wedge \neg\text{Beak}; \\ n2: & \text{ManyTeeth} \wedge \text{Gills} \wedge \neg\text{Short} \wedge \text{Beak}; \\ n3: & \text{ManyTeeth} \wedge \text{Gills} \wedge \neg\text{Short} \wedge \neg\text{Beak}; \\ n4: & \text{ManyTeeth} \wedge \neg\text{Gills} \wedge \text{Short} \wedge \neg\text{Beak}. \end{aligned}$$

S инициализируется пустым списком, а h – пустой конъюнкцией. Мы вызываем оракул эквивалентности, который возвращает контрпример, обязанный быть ложноположительным (потому что все примеры удовлетворяют начальной гипотезе), допустим $n1$, нарушающий первый дизъюнкт в f . До сих пор в S не было отрицательных примеров, поэтому мы добавляем $n1$ в S (шаг 8 алгоритма 4.5). Затем мы порождаем новую гипотезу по S (шаги 9–13): p равно `ManyTeeth` \wedge `Gills` \wedge `Short` и Q равно `{Beak}`, так что h становится равным (`ManyTeeth` \wedge `Gills` \wedge `Short` \rightarrow `Beak`). Отметим, что этот дизъюнкт вытекает из нашей целевой теории: если `ManyTeeth` и `Gills` равны `true`, то и `Short` тоже равно `true` в силу второго дизъюнкта f ; но тогда и `Beak` равно `true` в силу первого дизъюнкта f . Однако нам нужны дополнительные дизъюнкты, чтобы исключить все отрицательные примеры.

Предположим теперь, что следующий контрпример – ложноположительный пример $n2$. Мы строим пересечение с $n1$, уже присутствующим в S , и смотрим, удастся ли получить отрицательный пример, в котором `true` равно меньше литералов (шаг 7). Результат равен $n3$, поэтому оракул членства подтвердит, что это отрицательный пример, и мы заменяем $n1$ в S на $n3$. Затем мы перестраиваем h по S , что дает гипотезу (вспомним, что p равно `ManyTeeth` \wedge `Gills` и Q равно `{Short, Beak}`)

$$(\text{ManyTeeth} \wedge \text{Gills} \rightarrow \text{Short}) \wedge (\text{ManyTeeth} \wedge \text{Gills} \rightarrow \text{Beak}).$$

Наконец, предположим, что следующий ложноположительный пример, возвращаемый оракулом эквивалентности, – $n4$. Его пересечение с $n3$ в S является положительным примером, поэтому вместо пересечения с $n3$ мы добавляем $n4$ в S и перестраиваем h . Это дает предыдущие два дизъюнкта из $n3$ плюс следующие два из $n4$:

$$(\text{ManyTeeth} \wedge \text{Short} \rightarrow \text{Gills}) \wedge (\text{ManyTeeth} \wedge \text{Short} \rightarrow \text{Beak}).$$

Первый дизъюнкт из второй пары будет затем заменен ложноотрицательным примером от оракула эквивалентности, что приведет к окончательной теории:

$$\begin{aligned} & (\text{ManyTeeth} \wedge \text{Gills} \rightarrow \text{Short}) \wedge \\ & (\text{ManyTeeth} \wedge \text{Gills} \rightarrow \text{Beak}) \wedge \\ & (\text{ManyTeeth} \wedge \text{Short} \rightarrow \text{Beak}), \end{aligned}$$

которая логически эквивалентна f (хотя и не совпадает с ней).

В алгоритме Хорна соединилось несколько интересных новых идей. Во-первых, это алгоритм *активного обучения*: он не просто обучается на предложенном наборе данных, но и конструирует новые обучающие примеры и просит оракул членства пометить их. Во-вторых, в основе алгоритма лежит список специально подобранных отрицательных примеров, с учетом которых гипотеза периодически перестраивается. Важнейшим шагом здесь является пересечение: если бы алгоритм просто запоминал отрицательные примеры, то гипотеза состояла бы из многих специфичных дизъюнктов. Можно показать, что для обучения теории, состоящей из m дизъюнктов и n булевых переменных, алгоритм должен сделать

$O(mn)$ запросов к оракулу эквивалентности и $O(m^2n)$ запросов к оракулу членства. Кроме того, время работы алгоритма имеет квадратичную зависимость от m и n . И хотя из-за этого его практическое применение вряд ли возможно, можно показать, что алгоритм Хорна всегда успешно выводит теорию Хорна, эквивалентную целевой. И даже если у нас нет доступа к оракулу эквивалентности, все равно гарантируется, что алгоритм «почти всегда» найдет теорию Хорна, которая «правильна в большинстве случаев». Что это значит, мы уточним в разделе 4.4.

Применение логики первого порядка

Еще один способ выйти за пределы конъюнктивных концептов, определяемых простыми признаками, состоит в использовании более богатого логического языка. До сих пор мы работали с *пропозициональными* языками: все литералы являются высказываниями (пропозицией) – например, «Жабры = да» означает, что «у дельфина есть жабры», – из которых строятся более крупные выражения с помощью логических связок. *Предикатная логика первого порядка*, или, для краткости, просто логика первого порядка, обобщает эту конструкцию, позволяя строить более сложные литералы из предикатов и термов. Например, литерал первого порядка может иметь вид `BodyPart(Dolphin42,PairOf(Gill))`. Здесь **Dolphin42** и **PairOf(Gill)** – термы, ссылающиеся на объекты: **Dolphin42** – константа, а **PairOf(Gill)** – составной объект, состоящий из функционального символа **PairOf** и терма **Gills**. **BodyPart** (часть тела) – бинарный предикат, образующий высказывание (нечто, способное принимать значения `true` и `false`) из термов. Этот более богатый язык несет с собой ряд преимуществ:

- ☞ мы можем использовать термы типа **Dolphin42** для ссылки на интересующие нас отдельные объекты;
- ☞ структуру объектов можно явно описать;
- ☞ мы можем вводить переменные для ссылки на неспецифицированные объекты и применять к ним кванторы.

Проиллюстрируем последнее положение: литерал первого порядка `BodyPart(x,PairOf(Gill))` можно использовать для ссылки на множество всех объектов, имеющих парные жабры, а в следующем выражении квантор всеобщности применяется для формулировки утверждения о том, что всякое существо, имеющее парные жабры, является рыбой:

$$\forall x : \text{BodyPart}(x, \text{PairOf}(\text{Gill})) \rightarrow \text{Fish}(x).$$

Поскольку мы изменили структуру литералов, придется пересмотреть понятия обобщения и наименьшего обобщения (LGG). Напомним, что для пропозициональных литералов с внутренней дизъюнкцией мы использовали функцию **Combine-ID** для объединения двух внутренних дизъюнкций. Например, `LGG-Conj-ID(Length = [3,4], Length = [4,5])` возвращает `Length = [3,4,5]`. Для обобщения литералов первого порядка используются переменные. Рассмотрим, к примеру, два литерала первого порядка `BodyPart(Dolphin42,PairOf(Gill))` и

`BodyPart(Human123,PairOf(Leg))`; они обобщаются в литерал `BodyPart(x,PairOf(y))`, обозначающий множество объектов, имеющих парную неспецифицированную часть тела. Существует алгоритм для вычисления наименьшего обобщения литералов первого порядка, он называется *антиунификация*, поскольку является математически двойственным к дедуктивной операции *унификации*.

Пример 4.6 (унификация и антиунификация). Рассмотрим следующие термы:

`BodyPart(x,PairOf(Gill))` описывает объекты, имеющие парные жабры;

`BodyPart(Dolphin42,PairOf(y))` описывает части тела, которые у `Dolphin42` являются парными.

Следующие два терма являются их унификацией и антиунификацией соответственно:

`BodyPart(Dolphin42,PairOf(Gill))` описывает `Dolphin42` как имеющий пару жабр;

`BodyPart(x,PairOf(y))` описывает объекты, имеющие неспецифицированные парные части тела.

Таким образом, мы видим, что литералы в логике первого порядка имеют весьма развитую структуру благодаря использованию переменных. Мы еще вернемся к ним в разделе 6.4, когда будем обучать правила классификации в логике первого порядка.

4.4 Обучаемость

В этой главе мы видели несколько языков гипотез для концептуального обучения, в том числе конъюнкции литералов (возможно, с внутренней дизъюнкцией), конъюнкции дизъюнктов Хорна и дизъюнктов в логике первого порядка. Интуитивно очевидно, что эти языки отличаются с точки зрения выразительности: например, конъюнкция литералов одновременно является конъюнкцией дизъюнктов Хорна с пустой частью «если», поэтому теории Хорна строго выразительнее, чем конъюнктивные концепты. Но у более выразительного языка концептов есть и недостаток: ему труднее обучать. Раздел теории вычислительного обучения, в котором изучается именно этот вопрос, называется *обучаемостью*.

Для начала нам понадобится *модель обучения*: четкая формулировка того, что мы имеем в виду, когда говорим, что язык концептов обучаем. Одной из самых распространенных моделей обучения является модель *почти корректного* (*probably approximately correct* – PAC) обучения. PAC-обучаемость означает, что существует алгоритм обучения, который дает почти правильный результат в большинстве случаев. Модель вводит поправку на ошибки в случае нетипичных примеров, отсюда выражения «почти правильный» или «*approximately correct*» в английском названии. Модель также признает возможность того, что иногда она дает совершенно неверные результаты, например если обучающие данные содержат очень много нетипичных примеров, отсюда выражение «в большинстве случаев» и «*probably*» в английском названии. Мы предполагаем, что типичность примеров определяется некоторым неизвестным распределением вероятности D ,

и оцениваем частоту ошибок err_D гипотезы относительно этого распределения D . Более формально: для любой допустимой частоты ошибок $\epsilon < 1/2$ и частоты отказов $\delta < 1/2$ мы требуем, чтобы алгоритм РАС-обучения порождал с вероятностью не меньше $1 - \delta$ гипотезу h – такую, что $err_D < \epsilon$.

Временно предположим, что данные не содержат шума и что целевая гипотеза выбрана из нашего языка гипотез. Еще предположим, что обучаемый всегда порождает гипотезу, которая полна и непротиворечива на обучающей выборке. Тем не менее существует возможность, что такая нулевая ошибка обучения обманчива и на самом деле гипотеза «плохая» в том смысле, что на всем пространстве объектов будет давать ошибку, большую ϵ . Мы лишь хотим быть уверены, что такое происходит с вероятностью меньше δ . Теперь я покажу, что это можно гарантировать, взяв достаточно большую обучающую выборку. Предположим, что наше пространство гипотез H содержит одну плохую гипотезу, тогда вероятность, что она полна и непротиворечива на m независимо выбранных обучающих примерах, не больше $(1-\epsilon)^m$. Поскольку $1-\epsilon \leq e^{-\epsilon}$ для любого $0 \leq \epsilon \leq 1$, то эта вероятность никак не больше $e^{-m\epsilon}$. Мы хотим, чтобы она была не больше δ , для этого нужно взять $m \geq 1/\epsilon \ln 1/\delta$. На самом деле H может содержать несколько плохих гипотез, скажем $k \leq |H|$; тогда вероятность того, что хотя бы одна из них полна и непротиворечива на m независимо выбранных обучающих примерах, не превышает $k(1-\epsilon)^m \leq |H|(1-\epsilon)^m \leq |H|e^{-m\epsilon}$, а эта величина не превышает δ , если

$$m \geq \frac{1}{\epsilon} \left(\ln |H| + \ln \frac{1}{\delta} \right). \quad (4.1)$$

Эта величина называется *выборочной сложностью* полного и непротиворечивого обучаемого. Хорошая новость состоит в том, что она линейно зависит от $1/\epsilon$ и логарифмически от $1/\delta$. Отсюда следует, что экспоненциально дешевле уменьшить частоту отказов, чем частоту ошибок. Любой алгоритм обучения, в котором время обработки одного обучающего примера полиномиально зависит от $1/\epsilon$ и $1/\delta$, будет поэтому требовать полиномиального времени на обучение – это еще одно требование к РАС-обучаемости. Однако нахождение полной и непротиворечивой гипотезы вычислительно нереализуемо для большинства языков гипотез.

Отметим, что член $\ln |H|$ возник из-за того, что в худшем случае все гипотезы в H плохие. Однако на практике это означает, что оценка (4.1) излишне пессимистична. Тем не менее она показывает, что языки концептов, размер которых экспоненциально зависит от некоторого параметра n , являются РАС-обучаемыми. Например, число конъюнкций с n булевыми переменными равно 3^n , поскольку каждая переменная может встречаться без отрицания, с отрицанием или вообще не встречаться. Следовательно, выборочная сложность равна $(1/\epsilon)(n \ln 3 + \ln(1/\delta))$. Если положить $\delta = 0.05$ и $\epsilon = 0.1$, то выборочная сложность будет приблизительно равна $10(n \cdot 1.1 + 3) = 11n + 30$. В нашем примере с дельфином при $n = 4$ это, очевидно, пессимистическая оценка, потому что всего существует $2^4 = 16$ различных объектов! Для больших n оценка более реалистична. Отметим также, что модель РАС не зависит от распределения: обучаемый не располагает

никакой информацией о распределении объектов D . Это еще один источник пессимизма в оценке выборочной сложности.

Мы не всегда в состоянии вывести полную и непротиворечивую гипотезу: это может оказаться вычислительно невозможным, или целевая гипотеза непредставима на языке гипотез, или примеры слишком зашумлены. Разумная стратегия – выбрать гипотезу с наименьшей ошибкой обучения. Тогда «плохой» будет считаться гипотеза, для которой истинная ошибка превосходит ошибку обучения по меньшей мере на ϵ . Применяя некоторые результаты из теории вероятности, можно найти, что вероятность такого события составляет не более e^{-2me^2} . Поэтому множитель $1/\epsilon$ в уравнении 4.1 заменяется на $1/2\epsilon^2$: при $\epsilon = 0.1$ нам, следовательно, понадобится в 5 раз больше обучающих примеров, чем в предыдущем случае.

Мы уже упоминали, что член $|H|$ является слабым местом в приведенном выше анализе. Нам необходима мера, которая не просто равна размеру пространства гипотез, а характеризует его выразительность или емкость с точки зрения классификации. Такая мера существует и называется размерностью *Вапника-Червоненкиса*, или *VC-размерностью*, по имени придумавших ее Владимира Вапника и Алексея Червоненкиса. Основную идею мы проиллюстрируем на примере.

Пример 4.7 (разделение множества объектов). Рассмотрим следующие объекты:

```
m = ManyTeeth ∧ ¬Gills ∧ ¬Short ∧ ¬Beak;
g = ¬ManyTeeth ∧ Gills ∧ ¬Short ∧ ¬Beak;
s = ¬ManyTeeth ∧ ¬Gills ∧ Short ∧ ¬Beak;
b = ¬ManyTeeth ∧ ¬Gills ∧ ¬Short ∧ Beak.
```

Существует всего 16 различных подмножеств множества $\{m, g, s, b\}$. Можно ли каждый из них представить своим конъюнктивным концептом? Ответ – да: для каждого объекта, который мы хотим исключить, добавляем в конъюнкцию соответствующий литерал с отрицанием. Таким образом, подмножество $\{m, s\}$ будет представлено конъюнкцией $\neg Gills \wedge \neg Beak$, $\{g, s, b\}$ – конъюнкцией $\neg ManyTeeth \wedge \{s\}$ – конъюнкцией $\neg ManyTeeth \wedge \neg Gills \wedge \neg Beak$ и т. д. Мы говорим, что это множество из четырех объектов *разделяется* языком гипотез конъюнктивных концептов.

VC-размерность – это размер самого большого множества объектов, которое можно разделить заданным языком гипотез или классом модели. Предыдущий пример показывает, что VC-размерность конъюнктивных концептов над d булевыми литералами не меньше d . На самом деле она в точности равна d , но доказать это труднее (потому что необходимо доказать, что никакое множество, содержащее $d + 1$ экземпляров, разделить невозможно). Эта величина измеряет способность класса модели к представлению концептов или бинарных классификаторов. В качестве еще одного примера отметим, что VC-размерность линейных классификаторов в d -мерном пространстве равна $d + 1$: пороговое значение на вещественной прямой может разделить две точки, но не три (поскольку среднюю точку нельзя отделить от двух других одним пороговым значением); прямая линия на плоскости может разделить три точки, но не четыре, и т. д.

VC-размерность можно использовать для оценки разности между выборочной и истинной ошибкой гипотезы (это тот шаг, на котором в наших предшествующих рассуждениях появилась величина $|H|$). Следовательно, ее можно также использовать для вывода оценки выборочной сложности полного и непротиворечивого обучаемого в терминах VC-размерности D , а не $|H|$:

$$m \geq \frac{1}{\epsilon} \max \left(8D \log_2 \frac{13}{\epsilon}, 4 \log_2 \frac{2}{\delta} \right). \quad (4.2)$$

Как видим, оценка линейно зависит от D , тогда как раньше имела место логарифмическая зависимость от $|H|$. Это естественно, потому что для разделения D точек требуется по меньшей мере 2^D гипотез, так что $\log_2 |H| \geq D$. Оценка все еще логарифмически зависит от $1/\delta$, но линейно от логарифма $1/\epsilon$. Подставляя прежние значения $\delta = 0.05$ и $\epsilon = 0.1$, получаем оценку выборочной сложности $\max(562 \cdot D, 213)$.

Мы можем сделать вывод, что VC-размерность позволяет оценить выборочную сложность бесконечных классов концептов при условии, что их VC-размерность конечна. Следует также упомянуть классический результат теории вычислительного обучения, согласно которому класс концептов PAC-обучаем тогда и только тогда, когда его VC-размерность конечна.

4.5 Концептуальное обучение: итоги и литература для дальнейшего чтения

В этой главе мы рассмотрели методы индуктивного концептуального обучения: процесс построения логического выражения, определяющего множество объектов, на основе примеров. Эта задача привлекала большое внимание в ранних работах по искусственному интеллекту (Winston, 1970; Vere, 1975; Vanetj, 1980), за которыми последовали основополагающие работы психологов Bruner, Goodnow, Austin (1956) и Hunt, Marin, Stone (1966).

- ☞ В разделе 4.1 мы рассмотрели строение пространства гипотез: множества возможных концептов. У каждой гипотезы есть расширение (множество покрываемых ей объектов), а потому связи между расширениями, например отношение «быть подмножеством», переносятся на пространство гипотез. Это вводит в пространстве гипотез структуру решетки: частичный порядок с точной нижней и верхней гранями. В частности, наименьшее обобщение (LGG) является точной верхней гранью множества объектов и наиболее консервативной оценкой, которую можно вывести из примеров. Это понятие было определено в контексте логики первого порядка в работе Plotkin (1971), где показано, что имеется математическая двойственность с дедуктивной операцией унификации. Мы можем обобщить язык гипотез, допустив внутреннюю дизъюнкцию значений признака, при этом по-

лучится большее пространство гипотез, сохраняющее тем не менее структуру решетки. Внутренняя дизъюнкция – характерная черта применения языков атрибут–значение для обучения, согласно работе Michalski (1973). За дальнейшими ссылками на работу по языкам гипотез и пространствам гипотез отсылаем читателя к обзору Blockeel, 2010 а, б.

- ☞ В разделе 4.2 определены полные и непротиворечивые гипотезы как концепты, покрывающие все положительные примеры и ни одного отрицательного. Множество полных и непротиворечивых гипотез называется пространством версий, это понятие было введено в работе Mitchell (1977). Пространство версий можно полностью описать наименее и наиболее общей гипотезой, поскольку любой концепт, расположенный между наименее общим и наиболее общим, также является полным и непротиворечивым. Альтернативно пространство версий можно описать всеми путями, идущими от наименее общей гипотезы к наиболее общей. Таким восходящим путем соответствует кривая покрытия, которая описывает расширение каждого лежащего на пути концепта в терминах покрываемых им положительных и отрицательных примеров. Тогда концептуальное обучение можно рассматривать как нахождение восходящего пути, проходящего через вершину РХП. Синтаксически различные концепты могут иметь одно и то же расширение в конкретном множестве данных: замкнутым называется самый специфичный из таких концептов (технически наименьшее обобщение объектов в его расширении). Это понятие изучалось в рамках формального анализа концептов (Ganter, Wille, 1999) и было включено в контекст добычи данных в работе Pasquier, Bastide, Taouil, Lakhal (1999); Garriga, Kralj, Lavrač (2008) исследовали его полезность для помеченных данных.
- ☞ В разделе Section 4.3 мы обсудили алгоритм обучения концептов, описываемых конъюнкциями правил Хорна; впервые он был опубликован в работе Angluin et al. (1992). Алгоритм пользуется оракулом членства, который можно рассматривать как раннюю форму активного обучения (Cohn, 2010; Dasgupta, 2010). Теории Хорна обладают поверхностным сходством с моделями классификации на основе правил, которые будут рассматриваться в главе 6. Однако имеется и важное различие, поскольку в правилах классификации в части «то» стоит целевая переменная, тогда как рассмотренные здесь дизъюнкты Хорна могут содержать в части «то» произвольный литерал. На самом деле в этой главе целевая переменная вообще не является частью логического языка. Схему Хорна иногда называют *обучением по интерпретациям*, потому что примеры – это значения истинности в нашей теории. Схему с правилом классификации называют *обучением по импликации* (learning from entailment), потому что, чтобы узнать, покрывает ли конкретное правило пример, мы должны применить логический вывод. В работе De Raedt (1997) объясняются и исследуются различия между этими схемами. Более полное введение в логику перво-

го порядка и ее применение к обучению см. в работах Flach (2010a) и De Raedt (2010).

- ☞ В разделе 4.4 дан краткий обзор основных понятий и результатов теории обучаемости. В своем изложении я отчасти следовал работе Mitchell (1997, глава 7); еще одно великолепное введение содержится в работе Zeugmann (2010). PAC-обучаемость, допускающая частоту ошибки ϵ и частоту отказов δ , была введена в основополагающей работе Valiant (1984). Haussler (1988) получил оценку выборочной сложности полных и непротиворечивых обучаемых (уравнение 4.1), линейно зависящую от $1/\epsilon$ и логарифмически от $1/\delta$ и размера пространства гипотез. VC-размерность – это мера емкости языка гипотез, введенная в работе Vapnik (1971), для количественного измерения различия между ошибкой обучения и истинной ошибкой. Это позволяет выразить выборочную сложность в терминах VC-размерности (уравнение 4.2), как сделано в работе Blumer, Ehrenfeucht, Haussler, Warmuth (1989). В той же работе доказано, что класс модели PAC-обучаем тогда и только тогда, когда его VC-размерность конечна.



Древовидные модели

Древовидные модели относятся к числу наиболее популярных в машинном обучении. Например, в основе алгоритма распознавания поз в датчике движения Kinect для игровой приставки Xbox лежит решающее дерево (на самом деле ансамбль решающих деревьев, называемый случайным лесом, но об этом мы поговорим в главе 11). Деревья – выразительный и простой для понимания механизм, а интерес, который они вызывают у специалистов по информатике, связан с рекурсивной природой, допускающей алгоритмы типа «разделяй и властвуй».

Пути в пространстве логических гипотез, обсуждавшиеся в предыдущей главе, тоже представляют собой деревья очень простого вида. Например, дерево признаков на рис. 5.1 слева эквивалентно пути на рис. 4.6 слева. Убедиться в их эквивалентности проще всего, пройдя по пути и по дереву снизу вверх.

1. Самый левый лист дерева признаков представляет концепт в нижней точке пути, покрывающий один положительный пример.
2. Следующий концепт на пути вверх обобщает литерал **Длина = 3** до **Длина = [3,5]** посредством внутренней дизъюнкции; дополнительное покрытие (еще один положительный пример) представлено вторым слева листом в дереве признаков.
3. Исключив условие **Зубы = мало**, мы добавляем еще два покрытых положительных примера.
4. Вообще исключив условие «**Длина**» (или расширив внутреннюю дизъюнкцию путем добавления последнего оставшегося значения 4), мы добавляем последний положительный пример, а также один отрицательный.
5. Исключение литерала **Клюковидный выступ = да** не покрывает дополнительных примеров (вспомните обсуждение замкнутых концептов в предыдущей главе).
6. Наконец, исключение литерала **Жабры = нет** покрывает четыре оставшихся отрицательных примера.

Мы видим, что путь в пространстве гипотез можно преобразовать в эквивалентное дерево признаков. Чтобы получить дерево, эквивалентное i -му концепту, считая от нижней точки пути, мы можем либо обрезать дерево, объединив i самых левых листьев в один лист, представляющий концепт, либо пометить i самых левых листьев как положительные, а остальные – как отрицательные, превратив дерево признаков в решающее дерево.

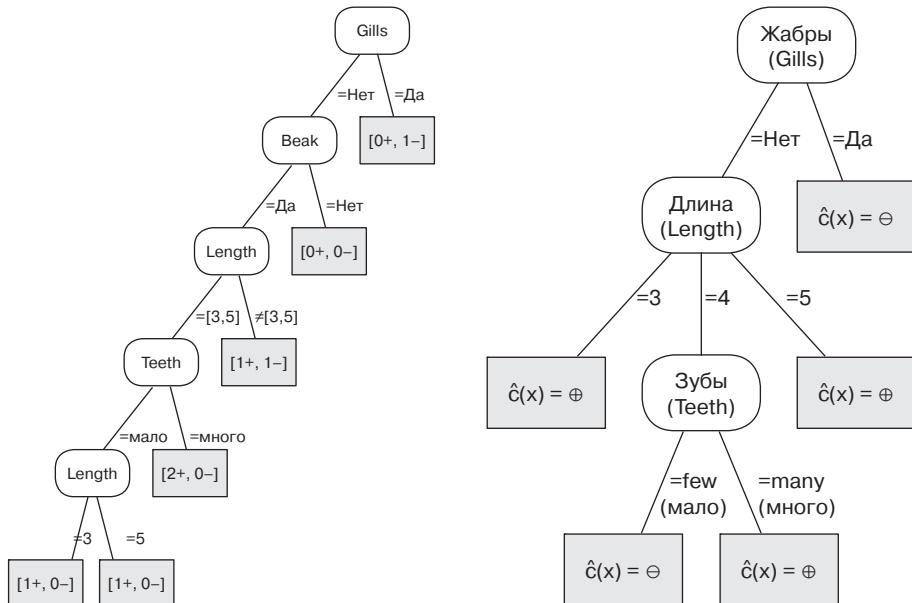


Рис. 5.1. (Слева) Путь с рис. 4.6, изображенный в виде дерева. Числа покрытия в листах получены на основе данных из примера 4.4. **(Справа)** Решающее дерево, обученное на тех же данных. Это дерево идеально разделяет положительные и отрицательные примеры

В решающих деревьях не используется внутренняя дизъюнкция для признаков, имеющих больше двух значений, вместо этого по каждому значению производится ветвление. Разрешаются также пометки, не обязательно следующие порядку обхода листьев слева направо. Такое дерево показано на рис. 5.1 справа. Это дерево можно преобразовать в логическое выражение многими способами, в том числе:

$$\begin{aligned}
 & (\text{Жабры} = \text{нет} \wedge \text{Длина} = 3) \vee (\text{Жабры} = \text{нет} \wedge \text{Длина} = 4 \wedge \text{Зубы} = \text{много}) \\
 & \quad \vee (\text{Жабры} = \text{нет} \wedge \text{Длина} = 5); \\
 & \text{Жабры} = \text{нет} \wedge [\text{Длина} = 3 \vee (\text{Длина} = 4 \wedge \text{Зубы} = \text{много}) \vee \text{Длина} = 5] \\
 & \quad \neg[(\text{Жабры} = \text{нет} \wedge \text{Длина} = 4 \wedge \text{Зубы} = \text{мало}) \vee \text{Жабры} = \text{да}]; \\
 & (\text{Жабры} = \text{да} \vee \text{Длина} = [3,5] \vee \text{Зубы} = \text{много}) \wedge \text{Жабры} = \text{нет}.
 \end{aligned}$$

Первое выражение – это дизъюнктивная нормальная форма (ДНФ, см. замечание 4.1), оно получено дизъюнкцией всех путей от корня дерева к листьям с положительными пометками, причем каждый путь представляет собой конъюнкцию литералов. Второе выражение получено упрощением первого за счет применения дистрибутивных правил $(A \wedge B) \vee (A \wedge C) \equiv A \wedge (B \vee C)$. Для получения третьего выражения мы сначала сконструировали ДНФ, представляющую отрицательный класс, а затем применили к ней логическое отрицание. Четвертое выражение – КНФ, полученная из третьего выражения применением правил де Моргана $\neg(A \wedge B) \equiv \neg A \vee \neg B$ и $\neg(A \vee B) \equiv \neg A \wedge \neg B$.

Существует и много других логических выражений, эквивалентных концепту, определенному решающим деревом. А возможно ли получить эквивалентный конъюнктивный концепт? Интересно, что ответ на этот вопрос отрицательный: некоторые деревья представляют конъюнктивные концепты, но большинство – нет, и показанное на рисунке – одно из таких¹. *Решающие деревья строго более выразительны, чем конъюнктивные концепты.* На самом деле из того, что решающие деревья соответствуют ДНФ-выражениям, и того, что каждое логическое выражение можно записать в виде эквивалентной ДНФ, следует, что решающие деревья максимально выразительны: они не могут разделить лишь данные с противоречивыми метками, когда один и тот же объект помечен разными метками. Это объясняет, почему данные, не являющиеся конъюнктивно отделыми, как в нашем примере, тем не менее разделяются решающим деревом.

С использованием столь выразительного языка гипотез связана потенциальная проблема. Пусть Δ – дизъюнкция всех положительных примеров, тогда Δ представлено в виде дизъюнктивной нормальной формы. Очевидно, что Δ покрывает все положительные примеры, – на самом деле расширение Δ и есть в точности множество положительных примеров. Иными словами, в пространстве гипотез, состоящем из ДНФ-выражений (или решающих деревьев), Δ – это наименьшее обобщение положительных примеров, но никаких других объектов оно не покрывает. Следовательно, Δ не обобщается за пределы множества положительных примеров, а просто запоминает их – вот и толкую тут о переобучении! Если перевернуть это рассуждение, то получается, что *один из способов избежать переобучения и способствовать обучению состоит в том, чтобы сознательно выбирать ограничительный язык гипотез*, такой как конъюнктивные концепты: в таком языке даже операция LGG, как правило, обобщается за рамки положительных примеров. А если наш язык достаточно выразителен, чтобы представить произвольное множество положительных примеров, то мы должны позаботиться о том, чтобы в алгоритме обучения применялись какие-то другие механизмы, гарантирующие обобщение за рамки примеров и отсутствие переобучения, – это называется *индуктивным смещением* алгоритма. Как мы увидим ниже, алгоритмы обучения, которые работают в выразительных пространствах гипотез, имеют индуктивное смещение в сторону менее сложных гипотез: либо неявно – за счет способа поиска в пространстве гипотез, либо явно – путем штрафования за сложность в целевой функции.

Древовидные модели не ограничиваются классификацией, а могут применяться для решения почти всех задач машинного обучения, в том числе ранжирования и оценивания вероятностей, регрессии и кластеризации. Древовидная структура, общая для всех этих моделей, может быть определена следующим образом.

¹ Если бы мы разрешили создавать новые конъюнктивные признаки, то могли бы представить это дерево в виде конъюнктивного концепта *Жабры = нет \wedge F = false*, где $F \equiv$ *Длина = 4 \wedge Зубы = мало* – новый конъюнктивный признак. Создание новых признаков во время обучения называется *конструктивной индукцией* и, как здесь показано, может расширить выразительность логического языка.

Определение 5.1 (дерево признаков). Деревом признаков называется такое дерево, в котором каждый внутренний узел (не являющийся листовым) помечен признаком и каждое ребро, исходящее из внутреннего узла, помечено литералом. Множество литералов в узле называется разделением. Каждый лист дерева представляет логическое выражение, являющееся конъюнкцией литералов, встречающихся на пути от корня дерева к этому листу. Расширение этой конъюнкции (множество покрываемых ей объектов) называется сегментом пространства объектов, ассоциированным с листом.

По существу, дерево признаков – это компактный способ представления ряда конъюнктивных концептов в пространстве гипотез. Тогда проблема обучения состоит в том, чтобы определить, какие из возможных концептов лучше всего решают имеющуюся задачу. При обучении правилам обучаемый (см. обсуждение в следующей главе) узнает об этих концептах по одному, тогда как при обучении дерева производится поиск сверху вниз сразу всех концептов.

Алгоритм 5.1 описывает процедуру, общую для большинства способов обучения дерева. Предполагается, что определены следующие три функции:

Homogeneous(D) возвращает true, если объекты в D достаточно однородны, чтобы их можно было пометить одной меткой, и false в противном случае;

Label(D) возвращает метку, наиболее подходящую для множества объектов D ;

BestSplit(D, F) возвращает наилучшее множество литералов для помещения в корень дерева.

Эти функции зависят от решаемой задачи. Например, для задач классификации множество объектов однородно, если большинство из них принадлежит одному классу, а самой подходящей меткой является мажоритарный класс. Для задач кластеризации множество объектов однородно, если они находятся рядом, а самой подходящей меткой будет некоторый эталон, например среднее (подробнее об эталонах см. главу 8).

Алгоритм 5.1. $\text{GrowTree}(D, F)$ – построить дерево признаков по обучающим данным

Вход: данные D ; множество признаков F .

Выход: дерево признаков T с помеченными листьями.

```

1 if Homogeneous( $D$ ) then return Label( $D$ );           // однородны, помечаем: см. текст
2  $S \leftarrow \text{BestSplit}(D, F)$ ;                   // например, BestSplit-Class (алгоритм 5.2)
3 разделить  $D$  на подмножества  $D_i$  в соответствии с литералами в  $S$ ;
4 for каждого  $i$  do
5   | if  $D_i \neq \emptyset$  then  $T_i \leftarrow \text{GrowTree}(D_i, F)$  else  $T_i$  – лист, помеченный Label( $D$ );
6 end
7 return дерево, корень которого помечен  $S$ , а потомками являются  $T_i$ 
```

Алгоритм 5.1 относится к типу «разделяй и властвуй»: он разбивает данные на подмножества, строит для каждого из них дерево, а затем объединяет эти поддеревья в единое дерево. Такие алгоритмы давно и часто используются в информатике. Обычно они реализуются рекурсивно, потому что каждая подзадача

(построение дерева для части данных) имеет такой же вид, как исходная задача. Но, чтобы этот способ работал, необходимо какое-то условие окончания рекурсии, и оно присутствует в первой же строке алгоритма. Однако следует отметить, что такие алгоритмы *жадные*: если нужно сделать выбор (например, выбор наилучшего разделения), то он делается на основе уже имеющейся информации и никогда не пересматривается. В результате найденное решение может оказаться неоптимальным. Альтернатива – использовать алгоритм *перебора с возвратом*, который может вернуть оптимальное решение ценой увеличения времени работы и потребляемой памяти, но в этой книге мы такие алгоритмы рассматривать не будем. Далее в этой главе мы применим алгоритм 5.1 к задачам классификации, ранжирования и оценивания вероятностей, кластеризации и регрессии.

5.1 Решающие деревья

Как уже отмечалось, для задачи классификации можно просто определить, что множество объектов D однородно, если все объекты принадлежат одному и тому же классу, а функция $\text{Label}(D)$ тогда просто возвращает этот класс. Отметим, что в строке 5 алгоритма 5.1 функция $\text{Label}(D)$ может вызываться для неоднородного множества объектов, если одно из D_i пусто, поэтому в общем случае $\text{Label}(D)$ должна возвращать мажоритарный класс объектов, входящих в D^1 . Теперь остается решить, как определить функцию $\text{BestSplit}(D, F)$.

Временно предположим, что признаки булевые, так что D разбивается на два множества D_1 и D_2 . Предположим также, что имеются два класса, и обозначим D^\oplus и D^\ominus множества положительных и отрицательных примеров в D (и аналогично для D_1^\oplus и т. д.). Вопрос в том, как оценить полезность признака с точки зрения разделения примеров на положительные и отрицательные. Очевидно, наилучшей была бы ситуация, когда $D_1^\oplus = D^\oplus$ и $D_1^\ominus = \emptyset$ или $D_1^\oplus = \emptyset$ и $D_1^\ominus = D^\ominus$. В таком случае оба подмножества, являющихся результатом разделения, называются *чистыми*. Таким образом, нам нужно определить меру нечистоты множества из n^\oplus положительных и n^\ominus отрицательных примеров. Мы будем придерживаться важного принципа: нечистота должна зависеть только от относительной величины n^\oplus и n^\ominus и не должна изменяться при умножении обоих чисел на одно и то же число. Это, в свою очередь, означает, что нечистоту можно определить в терминах отношения $\hat{p} = n^\oplus / (n^\oplus + n^\ominus)$, которое, как мы помним из раздела 2.2, называется *эмпирической вероятностью* положительного класса. Кроме того, нечистота не должна изменяться, если поменять положительный и отрицательный классы местами, то есть при замене \hat{p} на $1 - \hat{p}$. Мы также хотим, чтобы функция обращалась в 0, когда $\hat{p} = 0$ или $\hat{p} = 1$, и чтобы она достигала максимума при $\hat{p} = 1/2$. Всем этим условиям отвечают следующие функции.

¹ Если существует более одного наибольшего класса, то выбор производится случайно, обычно в предположении равномерного распределения.

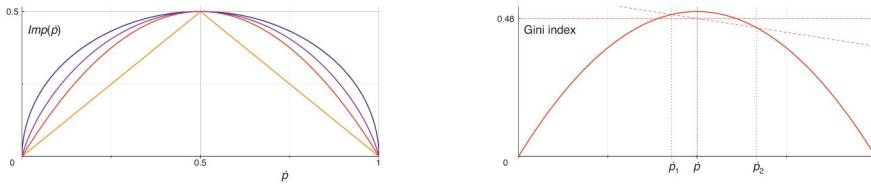


Рис. 5.2. (Слева) Графики зависимости нечистоты от эмпирической вероятности положительного класса. Снизу вверх: относительный размер миноритарного класса $\min(\dot{p}, 1 - \dot{p})$, индекс Джини $2\dot{p}(1 - \dot{p})$, энтропия $-\dot{p}\log_2 \dot{p} - (1 - \dot{p})\log_2(1 - \dot{p})$ (поделенная на 2, чтобы величина максимума была такой же, как в других функциях) и (масштабированный) квадратный корень из индекса Джини $\sqrt{\dot{p}(1 - \dot{p})}$; обратите внимание, что график последней функции – полуокружность. **(Справа)** Геометрическое построение для определения нечистоты разделения (**Зубы = [много, мало]** из примера 5.1): \dot{p} – эмпирическая вероятность родителя, а \dot{p}_1 и \dot{p}_2 – эмпирические вероятности потомков

Миноритарный класс $\min(\dot{p}, 1 - \dot{p})$ – иногда называют также частотой ошибок, поскольку эта функция измеряет долю неправильно классифицированных примеров в случае, если бы листовой узел был помечен мажоритарным классом; чем чище множество примеров, тем меньше ошибок при этом возникнет. Этую меру нечистоты можно записать в эквивалентном виде $1/2 - |\dot{p} - 1/2|$.

Индекс Джини $2\dot{p}(1 - \dot{p})$ – математическое ожидание ошибки в случае, когда примеры в листе помечаются случайным образом: отрицательные с вероятностью \dot{p} , а положительные с вероятностью $1 - \dot{p}$. Тогда вероятность ложноположительного результата равна $\dot{p}(1 - \dot{p})$, а ложноотрицательного – $(1 - \dot{p})\dot{p}$ ¹.

Энтропия $-\dot{p}\log_2 \dot{p} - (1 - \dot{p})\log_2(1 - \dot{p})$ – ожидаемый объем информации в битах, получаемой, когда кто-то сообщает вам класс выбранного наугад примера; чем чище множество примеров, тем более предсказуемо это сообщение и тем меньше ожидаемый объем информации.

На рис. 5.2 слева представлены графики всех трех мер нечистоты, причем некоторые масштабированы так, чтобы достигали максимума в одной и той же точке (0.5, 0.5). Я добавил еще одну меру: квадратный корень из индекса Джини, которую буду обозначать $\sqrt{\text{Gini}}$ и у которой, как мы впоследствии увидим, есть одно преимущество над другими мерами. Если обозначить $\text{Imp}(D_j)$ нечистоту одиночного листа D_j , то нечистота множества взаимно исключающих листьев $\{D_1, \dots, D_l\}$ определяется как взвешенное среднее:

$$\text{Imp}(\{D_1, \dots, D_l\}) = \sum_{j=1}^l \frac{|D_j|}{|D|} \text{Imp}(D_j), \quad (5.1)$$

¹ Когда я искал «индекс Джини» в Википедии, то меня отослали на страницу с описанием **коэффициента Джини**, какой – в контексте машинного обучения – является линейно масштабированной площадью под кривой (**AUC**), приведенной к отрезку $[-1, 1]$. Это совершенно другое понятие, и единственное, что родит индекс Джини и коэффициент Джини, – тот факт, что оба были предложены итальянским статистиком Коррадо Джини. Поэтому опасайтесь путаницы.

где $D = D_1 \cup \dots \cup D_r$. В случае бинарного разделения существует изящное геометрическое построение для нахождения величины $\text{Imp}(\{D_1, D_2\})$, определяемой эмпирическими вероятностями родителя и потомков; оно показано на рис. 5.2 справа.

1. Сначала находим значения нечистоты $\text{Imp}(D_1)$ и $\text{Imp}(D_2)$ обоих потомков на графике кривой нечистоты (в данном случае используется индекс Джини).
2. Затем соединяем эти значения прямой линией, так как любое их взвешенное среднее должно находиться на этой прямой.
3. Поскольку эмпирическая вероятность родителя также является взвешенным средним эмпирических вероятностей потомков с теми же весами (то есть $\dot{p} = \frac{|D_1|}{|D|} \dot{p}_1 + \frac{|D_2|}{|D|} \dot{p}_2$ – вывод приведен в уравнении (5.2) на стр. 152), то \dot{p} дает правильную точку интерполяции.

Это построение применимо к любой кривой нечистоты, изображенной на рис. 5.2 слева. Отметим, что если распределение по классам в родителе сильно асимметрично, то эмпирическая вероятность обоих потомков может сдвинуться влево или справо от вертикали $\dot{p} = 0.5$. Это не проблема – за исключением случая, когда мерой нечистоты является миноритарный класс, поскольку из геометрического построения видно, что для всех таких разделений взвешенная средняя нечистота будет одинакова. По этой причине миноритарный класс в качестве меры нечистоты обычно не применяют.

Пример 5.1 (вычисление нечистоты). Снова рассмотрим данные из примера 4.4. Мы хотим найти наилучший признак для помещения в корень решающего дерева. Четыре имеющихся признака дают такие разделения:

Длина = [3,4,5]	[2+,0-][1+,3-][2+,2-]
Жабры = [да,нет]	[0+,4-][5+,1-]
Клювовидный выступ = [да,нет]	[5+,3-][0+,2-]
Зубы = [много, мало]	[3+,4-][2+,1-]

Вычислим нечистоту первого разделения. Имеются три сегмента: первый чистый, и, следовательно, его энтропия равна 0; энтропия второго равна $-(1/4)\log_2(1/4) - (3/4)\log_2(3/4) = 0.5 + 0.31 = 0.81$; энтропия третьего равна 1. Тогда полная энтропия равна взвешенному среднему: $2/10 \cdot 0 + 4/10 \cdot 0.81 + 4/10 \cdot 1 = 0.72$.

Аналогичные вычисления для остальных трех признаков дают следующие величины энтропии:

Жабры = [да,нет]	$4/10 \cdot 0 + 6/10 \cdot (-(5/6)\log_2(5/6) - (1/6)\log_2(1/6)) = 0.39$
Клювовидный выступ = [да,нет]	$8/10 \cdot (-(5/8)\log_2(5/8) - (3/8)\log_2(3/8)) + 2/10 \cdot 0 = 0.76$
Зубы = [много, мало]	$7/10 \cdot (-(3/7)\log_2(3/7) - (4/7)\log_2(4/7)) + 3/10 \cdot (-(2/3)\log_2(2/3) - (1/3)\log_2(1/3)) = 0.97$

Таким образом, очевидно, что признак «Жабры» очень хорошо подходит для разделения, признак «Зубы» – плохо, а остальные – где-то между ними.

Вычисление индекса Джини дает следующие результаты (по шкале от 0 до 0.5):

Длина	$2/10 \cdot 2 \cdot (2/2 \cdot 0/2) + 4/10 \cdot 2 \cdot (1/4 \cdot 3/4) + 4/10 \cdot 2 \cdot (2/4 \cdot 2/4) = 0.35$
Жабры	$4/10 \cdot 0 + 6/10 \cdot 2 \cdot (5/6 \cdot 1/6) = 0.17$
Клювовидный выступ	$8/10 \cdot 2 \cdot (5/8 \cdot 3/8) + 2/10 \cdot 0 = 0.38$
Зубы	$7/10 \cdot 2 \cdot (3/7 \cdot 4/7) + 3/10 \cdot 2 \cdot (2/3 \cdot 1/3) = 0.48$

Как и следовало ожидать, обе меры нечистоты хорошо согласуются друг с другом. На рис. 5.2 справа приведена геометрическая иллюстрация последнего вычисления, относящегося к признаку «Зубы».

Обобщение этих мер нечистоты на случай $k > 2$ классов производится путем суммирования величин нечистоты каждого класса по методу один против остальных. В частности, k -классовая энтропия определяется по формуле $\sum_{i=1}^k -\dot{p}_i \log_2 \dot{p}_i$, а k -классовый индекс Джини – по формуле $\sum_{i=1}^k \dot{p}_i(1 - \dot{p}_i)$. Для оценки качества признака для разделения родительского узла D на листья D_1, \dots, D_l обычно рассматривают величину $\text{Imp}(D) = \text{Imp}(\{D_1, \dots, D_l\})$. Если чистота измеряется с помощью энтропии, то она называется *приростом информации*, поскольку измеряет увеличение информации о классе, полученное в результате включения данного признака. Отметим, однако, что в алгоритме 5.1 сравниваются только разделения с одним и тем же родителем, поэтому мы можем игнорировать нечистоту родителя и искать признак, который дает наименьшее взвешенное среднее нечистоты потомков (алгоритм 5.2).

Итак, мы имеем полностью описанный алгоритм обучения решающих деревьев и можем посмотреть, какое дерево он даст для наших данных о дельфинах. Мы уже видели, что наилучшим признаком для разделения в корне дерева является «Жабры»: условие Жабры = да приводит к чистому листу $[0+, 4-]$, помеченному как отрицательный, и к преимущественно положительному потомку $[5+, 1-]$. Для следующего разделения у нас есть выбор между признаками «Длина» и «Зубы», поскольку разделение по признаку «Клювовидный выступ» не уменьшает нечистоту. Выбор «Длины» дает разделение $[2+, 0-][1+, 1-][2+, 0-]$, а «Зубов» – разделение $[3+, 0-][2+, 1-]$; как энтропия, так и индекс Джини приводят к выводу, что первое чище второго. И для разделения последнего оставшегося нечистого узла мы используем признак «Зубы». Получившееся дерево было показано ранее на рис. 5.1 и еще раз воспроизведено на рис. 5.3 слева. Мы обучили свое первое решающее дерево!

Дерево описывает разбиение пространства объектов и потому сопоставляет класс и тем 14 объектам, которые не входили в обучающий набор, – именно поэтому мы можем сказать, что дерево обобщает обучающие данные. Лист С оставляет незаданными значения трех признаков, что дает $3 \cdot 2 \cdot 2 = 12$ возможных комбинаций; четыре из них входили в состав обучающего набора, так что лист С покрывает восемь непомеченных объектов и классифицирует их как отрицательные. Аналогично два непомеченных объекта классифицируются листом D как положительные, а еще два – листом F; один классифицируется как отрицательный листом G, и последний – как положительный листом H. То, что непомеченных объектов, классифицированных как положительные (9), больше, чем классифицированных как отрицательные (5), объясняется в основном влиянием листа С: будучи расположен выше в дереве, он покрывает много объектов. Можно предположить, что тот факт, что четыре из пяти отрицательных примеров имеют жабры, свидетельствует о сильной регулярности, найденной в данных.

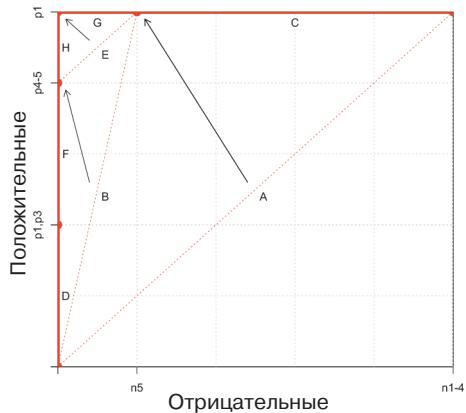
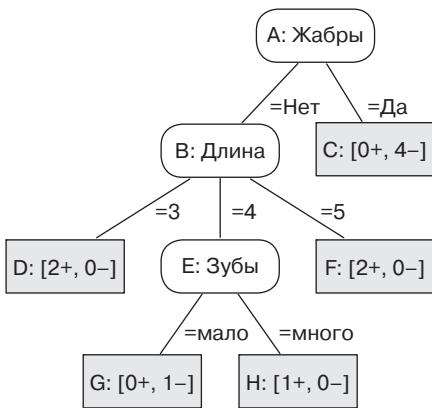


Рис. 5.2. (Слева) Решающее дерево, обученное на данных из примера 4.4. **(Справа)** Каждый внутренний и листовой узел дерева соответствует отрезку прямой в пространстве покрытия: вертикальным отрезкам соответствуют чисто положительные узлы, горизонтальным – чисто отрицательные, а диагональным – нечистые

Алгоритм 5.2. BestSplit-Class(D, F) – найти наилучшее разделение решающего дерева

Вход: данные D ; множество признаков F .

Выход: признак f , по которому производить разделение.

```

1    $I_{\min} \leftarrow 1;$ 
2   for каждого  $f \in F$  do
3       | разделить  $D$  на подмножества  $D_1, \dots, D_l$  согласно значениям  $v_j$  признака  $f$ ;
4       | if  $\text{Imp}(\{D_1, \dots, D_l\}) < I_{\min}$  then
5           |   |  $I_{\min} \leftarrow \text{Imp}(\{D_1, \dots, D_l\});$ 
6           |   |  $f_{\text{best}} \leftarrow f;$ 
7       | end
8   end
9   return  $f_{\text{best}}$ 
  
```

Стоит также проследить за построением этого дерева в пространстве покрытия (рис. 5.3 справа). Каждый узел дерева, внутренний или листовой, покрывает сколько-то положительных и отрицательных примеров и потому может быть представлен отрезком прямой в пространстве покрытия. Например, корень дерева покрывает все положительные и все отрицательные примеры и, значит, представляется восходящей диагональю А. После первого разделения отрезок А заменяется отрезком В (узел нечистый, поэтому отрезок диагональный) и отрезком С – чистым и потому далее не разделяемым. Отрезок В затем разделяется на D (чистый и положительный), Е (нечистый) и F (чистый и положительный). Наконец, Е разделяется на чистые узлы.

Идея о том, что кривая покрытия решающего дерева «сама подтягивает себя» от восходящей диагонали в духе «разделяй и властвуй», кажется красивой, но,

к сожалению, в общем случае неверна. Упорядочение отрезков кривой покрытия основано исключительно на распределении по классам в листьях и напрямую не связано со структурой дерева. Чтобы лучше уяснить это, рассмотрим, как древовидные модели можно преобразовать в ранжировщики и оценки вероятностей.

5.2 Деревья ранжирования и оценивания вероятностей

Группирующие классификаторы, в частности решающие деревья, разбивают пространство объектов на сегменты, а потому могут быть преобразованы в ранжировщики, если обучить их упорядочению сегментов. В отличие от некоторых других группирующих моделей, решающие деревья имеют доступ к локальному распределению по классам в сегментах, или листьях, и этим можно воспользоваться для задания порядка на листьях, оптимального для обучающих данных. Так, например, на рис. 5.3 получается порядок [D – F] – H – G – C, который дает идеальное ранжирование ($AUC = 1$). Этот порядок можно легко получить из эмпирических вероятностей \hat{p} , разрешая неопределенности так, что приоритет отдается листьям, которые покрывают наибольшее число положительных примеров¹. Почему этот порядок оптимален? Дело в том, что угловой коэффициент отрезка кривой покрытия с эмпирической вероятностью \hat{p} равен $\hat{p}/(1 - \hat{p})$; поскольку $\hat{p} \mapsto \frac{\hat{p}}{1 - \hat{p}}$ – монотонное преобразование (если $\hat{p} > \hat{p}'$, то $\frac{\hat{p}}{1 - \hat{p}} > \frac{\hat{p}'}{1 - \hat{p}'}$), то

сортировка отрезков по невозрастанию эмпирических вероятностей гарантирует, что они также будут отсортированы в порядке невозрастания углового коэффициента, а значит, кривая выпуклая. Это важный момент, и я подчеркну его еще раз: *ранжирование, полученное из эмпирических вероятностей в листьях решающего дерева, дает выпуклую кривую РХП на обучающих данных*. Впоследствии мы увидим, что некоторые другие группирующие модели, в том числе *справки правил* (раздел 6.1), также обладают этим свойством, но ранжирующие модели – никогда.

Как уже отмечалось, упорядочение отрезков нельзя вывести из древовидной структуры. Основная причина состоит в том, что даже если мы знаем эмпирическую вероятность, ассоциированную с родительским узлом разделения, это не накладывает никаких ограничений на эмпирические вероятности потомков. Например, пусть $[n^{\oplus}, n^{\ominus}]$ – распределение по классам в родительском узле, $n = n^{\oplus} + n^{\ominus}$, и пусть $[n_1^{\oplus}, n_1^{\ominus}]$ и $[n_2^{\oplus}, n_2^{\ominus}]$ – распределение по классам в дочернем узле и $n_1 = n_1^{\oplus} + n_1^{\ominus}$, $n_2 = n_2^{\oplus} + n_2^{\ominus}$. Тогда

¹ Разрешения неопределенностей – хотя оно не изменяет форму кривой покрытий и в этом смысле не является существенным – можно добиться и путем вычитания $\epsilon \ll 1$ из числа покрытых положительных примеров. Поправка Лапласа также разрешает неопределенности в пользу больших листьев, но это немонотонное преобразование и потому может изменить форму кривой покрытия.

$$\dot{p} = \frac{n^\oplus}{n} = \frac{n_1}{n} \frac{n_1^\oplus}{n_1} + \frac{n_2}{n} \frac{n_2^\oplus}{n_2} = \frac{n_1}{n} \dot{p}_1 + \frac{n_2}{n} \dot{p}_2. \quad (5.2)$$

Иными словами, эмпирическая вероятность родителя равна взвешенному среднему эмпирических вероятностей его потомков; но это лишь позволяет сделать вывод, что $\dot{p}_1 \leq \dot{p} \leq \dot{p}_2$ или $\dot{p}_2 \leq \dot{p} \leq \dot{p}_1$. Даже если известно положение родительского отрезка в кривой покрытия, его потомки могут оказаться значительно раньше или значительно позже с точки зрения данного порядка.

Пример 5.2 (рост дерева). Рассмотрим дерево на рис. 5.4. Каждый его узел помечен количеством покрытых им положительных и отрицательных примеров. Так, корень дерева помечен общим распределением по классам (50 положительных и 100 отрицательных), что дает тривиальное ранжирование $[50+, 100-]$. Соответствующая кривая покрытия состоит из одного отрезка – восходящей диагонали (рис. 5.4 снизу). После добавления разделения (1) это ранжирование уточняется до $[30+, 35-][20+, 65-]$, что дает кривую из двух отрезков. Добавление разделений (2) и (3) снова разбивает отрезок, соответствующий родителю, на два отрезка, соответствующих потомкам. Однако ранжирование, порожденное полным деревом, – $[15+, 3-][29+, 10-][5+, 62-][1+, 25-]$ – отличается от упорядочения листьев слева направо, поэтому мы должны переупорядочить отрезки кривой покрытия, что приводит к верхней кривой, изображенной сплошной линией.

Таким образом, разделение решающего дерева можно интерпретировать в терминах кривых покрытия как двухшаговый процесс:

- ☞ разбить соответствующий отрезок кривой на два или более отрезков;
- ☞ переупорядочить отрезки по убыванию углового коэффициента.

Весь процесс роста решающего дерева можно представлять себе как повторение этих двух шагов или – альтернативно – как последовательность шагов разделения, за которыми следует один общий шаг переупорядочения. Именно этот последний шаг гарантирует выпуклость кривой покрытия (на обучающих данных).

Было бы поучительно сделать еще один шаг и рассмотреть все возможные ранжирования, которые можно построить по заданному дереву. Один из подходов – рассматривать дерево как дерево признаков без меток классов и задаться вопросом, сколько существует способов разметить это дерево и какое при этом получится качество, если известно количество положительных и отрицательных примеров, покрываемых каждым листом. В общем случае, если в дереве признаков l листьев и всего имеется c классов, то число возможных способов пометки листьев классами равно c^l ; в примере на рис. 5.4 получается $2^4 = 16$. На рис. 5.5 эти 16 вариантов разметки представлены в пространстве покрытия. Как и следовало ожидать, график обладает выраженной симметрией. Например, варианты разметки встречаются парами (скажем, $+--$ и $-+-$), которым соответствуют противоположные точки на графике (подумайте, сможете ли вы определить, что в данном контексте означает «противоположная»). Ранжирование мы получим, если начнем с варианта $----$ в левом нижнем углу и будем заменять в узле один – на + в некотором порядке. Например, для оптимальной кривой покрытия по-

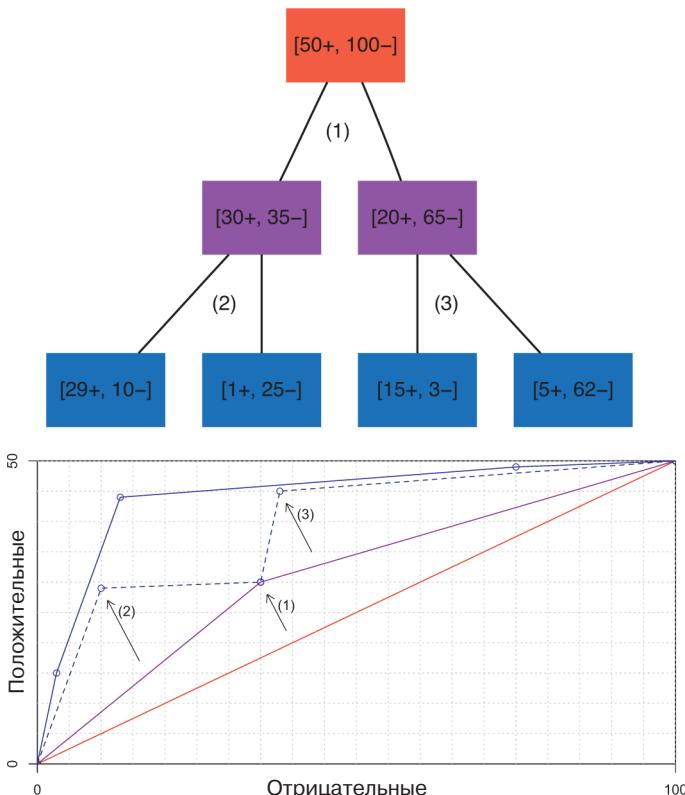


Рис. 5.4. (Сверху) Абстрактное представление дерева, в каждом узле которого указано количество покрытых этим узлом положительных и отрицательных примеров. Бинарные разделения дерева производятся в указанном порядке. **(Снизу)** Разделение дерева добавляет новые отрезки к кривую покрытия, как показывают стрелки. После разделения отрезки, возможно, придется переупорядочить, поэтому фактическими кривыми покрытия являются только те, что изображены сплошной линией

рядок таков: $----$, $--+-$, $+--+$, $+---$, $++-$, $+++$. Для дерева с l листьями существует $l!$ перестановок листьев и потому $l!$ различных кривых покрытия (в нашем примере 24).

Если бы мне нужно было выбрать один-единственный образ, передающий существование древовидных моделей, то это был бы рис. 5.5. На нем наглядно представлены распределения по классам в листьях непомеченного дерева признаков, которые можно использовать для превращения одного и того же дерева в решающее, ранжирующее или оценивающее вероятности.

☞ Чтобы превратить дерево признаков в ранжировщик, мы упорядочиваем его листья по неубыванию эмпирических вероятностей; можно доказать, что полученное таким образом ранжирование оптимально на обучающем наборе.

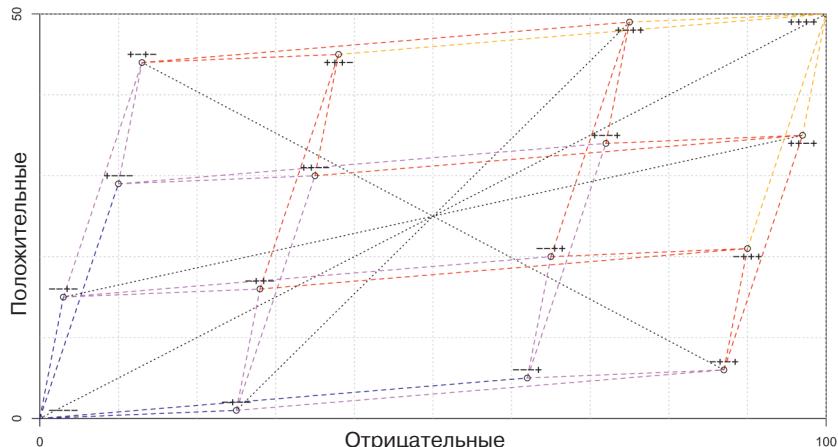


Рис. 5.5. Графическое представление всех возможных способов пометки и всех возможных ранжирований, которые можно получить с помощью решающего дерева с четырьмя листьями, изображенного на рис. 5.4. Существует $2^4 = 16$ возможных способов пометки листьев, например: '+--+' означает, что первый и третий слева листья имеют метку +, а второй и четвертый имеют метку -. Обозначены также попарные симметрии (пунктирные линии), например: +--+ и -+-+ взаимно обратны и оканчиваются в противоположных углах графика. Существует $4! = 24$ путей синий-фиолетовый-красный-оранжевый, проходящих через эти точки, которые начинаются в узле ---- и переключают все узлы в + в каком-то порядке; они представляют все возможные кривые покрытия или ранжирования, состоящие из четырех отрезков.

- ☞ Чтобы получить на основе дерева оценки вероятностей, мы предсказываем эмпирические вероятности в каждом листе, применяя сглаживание Лапласа или на основе m -оценок, чтобы повысить надежность оценок в малых листах.
- ☞ Чтобы превратить дерево в классификатор, мы выбираем рабочие условия и находим рабочую точку, оптимальную при этих условиях.

Последняя процедура объяснена в разделе 2.2. Мы проиллюстрируем ее здесь в предположении, что отношение классов $clr = 50/100$ в обучающем наборе презентативно. У нас есть на выбор пять способов разметки, зависящих от ожидаемого отношения затрат $c = c_{FN}/c_{FP}$ — отношения стоимости неправильной классификации положительных примеров к стоимости неправильной классификации отрицательных:

+--+ выбирается, если $c = 1$, или, более общо, если $10/29 < c < 62/5$;

+--+ выбирается, если $62/5 < c < 25/1$;

+++- выбирается, если $25/1 < c$, то есть мы бы всегда предсказывали положительный класс, если ложноотрицательное предсказание обходится дороже ложноположительного более чем в 25 раз, потому что в таком случае даже предсказание положительного класса во втором листе снизило бы затраты;

--+- выбирается, если $3/15 < c < 10/29$;

— выбирается, если $c < 3/15$, то есть мы бы всегда предсказывали отрицательный класс, если ложноположительное предсказание обходится дороже ложноотрицательного более чем в 5 раз, потому что в таком случае даже предсказание отрицательного класса в третьем листе снизило бы затраты.

Первый вариант соответствует пометке на основе мажоритарного класса, и именно он рекомендуется в большинстве учебников при рассмотрении решающих деревьев. И я также рекомендовал его при обсуждении функции $\text{Label}(D)$ в контексте алгоритма 5.1. Во многих случаях это самый практический подход. Однако важно понимать, какие предположения лежат в основе такой пометки: требуется, чтобы распределение по классам в обучающем наборе было репрезентативно, а затраты равномерны, или, более общо, чтобы произведение ожидаемых затрат и отношений классов было равно отношению классов, наблюдаемому в обучающем наборе. (Отсюда следует полезный способ манипулирования обучающим набором с целью отразить ожидаемое отношение классов: чтобы сымитировать ожидаемое отношение классов c , мы должны включить положительных примеров в c раз больше, чем отрицательных, если $c > 1$, и включить отрицательных примеров в $1/c$ раз больше, чем положительных, если $c < 1$. Ниже мы еще вернемся к этому совету.)

Итак, предположим, что распределение по классам репрезентативно и что ложноотрицательное предсказание (например, недиагностированное у пациента заболевания) обходится примерно в 20 раз дороже ложноположительного. Как мы только что видели, оптимальной при таких рабочих условиях является разметка $+--+$, которая означает, что для фильтрации отрицательных примеров используется только второй лист. Другими словами, два правых листа можно объединить в один — их общий родительский узел. Операция объединения всех листьев поддерева называется *редукцией* поддерева. Этот процесс иллюстрируется на рис. 5.6. Преимущество редукции заключается в том, что она позволяет упростить дерево, не оказывая влияния на выбранную рабочую точку, — иногда это полезно, если мы хотим передать древовидную модель кому-то другому. Недостаток же в том, что мы теряем в качестве ранжирования, как видно по рис. 5.6 снизу. Поэтому редукция не рекомендуется, если (i) вы намереваетесь использовать дерево не только для классификации, но и для ранжирования и оценивания вероятностей, и (ii) если вы не можете определить ожидаемые рабочие условия с достаточной точностью. Один из популярных алгоритмов редукции решающих деревьев называется *редукцией с уменьшением ошибки* (алгоритм 5.3). В этом алгоритме используется отдельный редуцирующий набор помеченных данных, который не предъявляется во время обучения, поскольку редукция никогда не повышает точность на обучающих данных. Однако если простота дерева не является важным аргументом, то я рекомендую не сокращать его и выбирать рабочую точку только за счет пометки листьев; это также можно сделать с помощью зарезервированного набора данных.

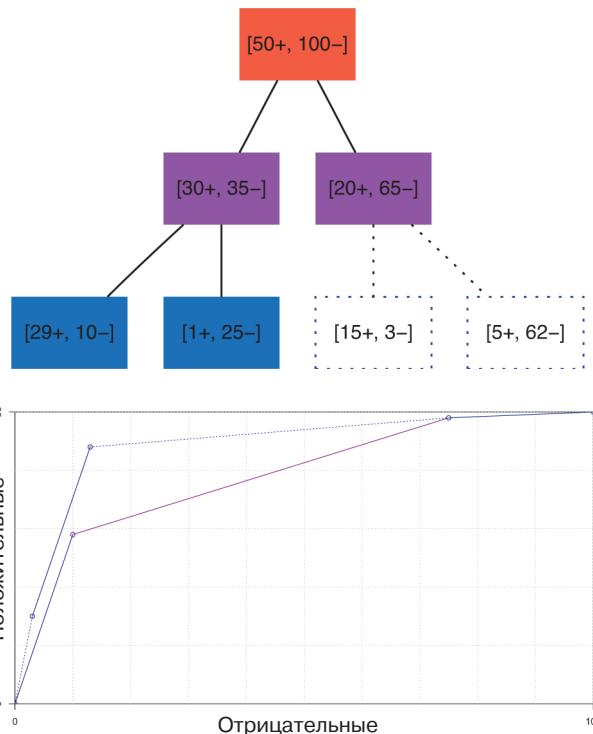


Рис. 5.6. (Сверху) Чтобы достичь разметки $+--$, нам не нужно самое правое разделение, которое, следовательно, можно редуцировать. (**Снизу**) Редукция не оказывает влияния на выбранную рабочую точку, но уменьшает качество ранжирования данным деревом

Чувствительность к асимметричному распределению по классам

Я только что мимоходом упомянул, что один из способов гарантировать, что обучающий набор отражает правильные рабочие условия, состоит в том, чтобы продублировать положительные или отрицательные примеры, так чтобы отношение классов в обучающем наборе было равно произведению ожидаемых затрат на отношения классов при развертывании модели. По сути дела, это изменяет отношение сторон прямоугольника, представляющего пространство покрытия. Достоинство этого метода – в его непосредственной применимости к любой модели, без необходимости колдовать с эвристикой поиска или мерами оценки. Недостаток же – в увеличении времени обучения, и, кроме того, он может никак не оказаться на обучаемой модели. Я проиллюстрирую это на примере.

Пример 5.3 (чувствительность критерия разделения к затратам). Предположим, что имеются 10 положительных и 10 отрицательных примеров и что требуется выбрать между двумя вариантами разделения: $[8+,2-][2+,8-]$ и $[10+,6-][0+,4-]$. Вы честно вычисляете средневзвешенную энтропию обоих вариантов и приходите к выводу, что первый лучше. На всякий случай вы вычисляете также средний индекс Джини, и снова первое разделение оказывается лучше. Затем вы припоминаете, что где-то слышали, будто квадратный корень из индекса Джини – более точная мера нечистоты, поэтому решаете проверить и на ней тоже. И подумать только – она отдает предпочтение второму варианту разделения! И что теперь делать?

Тогда вы вспоминаете, что ошибки на положительных примерах обходятся примерно в 10 раз дороже ошибок на отрицательных. Вы не вполне уверены, как это выразить математически, поэтому решаете просто включить 10 копий каждого положительного примера: теперь варианты разделения примут вид $[80+,2-][20+,8-]$ и $[100+,6-][0+,4-]$. Вы пересчитываете все три критерия разделения, и теперь все дружно предпочитают второй вариант. И хотя вы слегка озадачены таким положением дел, все же останавливаитесь на втором разделении, поскольку все три критерия столь единодушны в своей рекомендации.

Алгоритм 5.3. $\text{PruneTree}(T,D)$ – редукция решающего дерева с уменьшением ошибки

Вход: решающее дерево T ; размеченные данные D .

Выход: редуцированное дерево T .

```

1 for каждого внутреннего узла  $N$  дерева  $T$ , начиная снизу do
2    $T_N \leftarrow$  поддерево  $T$  с корнем в  $N$ ;
3    $D_N \leftarrow \{x \in D \mid x$  покрывается  $N\}$ ;
4   if верность  $T_N$  над  $D_N$  хуже той, что дает мажоритарный класс в  $D_N$  then
5     | заменить  $T_N$  в  $T$  листом, помеченным мажоритарным классом в  $D_N$ ;
6   end
7 end
8 return редуцированный вариант  $T$ 
```

Так что же здесь происходит? Сначала рассмотрим ситуацию с искусственно «раздутым» количеством положительных примеров. Интуитивно очевидно, что в этом случае второй вариант разделения предпочтительнее, потому что один из потомков чистый, а другой тоже вполне приличный, хотя, быть может, и не настолько хорош, как $[80+,2-]$. Но ситуация меняется, есть уменьшить количество положительных примеров в десять раз, по крайней мере, если верить энтропии и индексу Джини. В обозначениях, введенных ранее, это можно объяснить следующим образом. Индекс Джини родителя равен $2 \frac{n^{\oplus}}{n} \frac{n^{\ominus}}{n}$, а взвешенный индекс Джини одного из потомков равен $\frac{n_1}{n} 2 \frac{n_1^{\oplus}}{n_1} \frac{n_1^{\ominus}}{n_1}$. Следовательно, взвешенная нечистота потомка в пропорции к нечистоте родителя равна $\frac{n_1^{\oplus} n_1^{\ominus}}{n^{\oplus} n^{\ominus}} / n_1$; назовем эту величину *относительной нечистотой*. Те же вычисления для квадратного корня из индекса Джини дают:

- ☞ нечистота родителя: $\sqrt{\frac{n^{\oplus}}{n} \frac{n^{\ominus}}{n}};$
- ☞ взвешенная нечистота потомка: $\frac{n_1}{n} \sqrt{\frac{n_1^{\oplus}}{n_1} \frac{n_1^{\ominus}}{n_1}};$
- ☞ относительная нечистота: $\sqrt{\frac{n_1^{\oplus} n_1^{\ominus}}{n^{\oplus} n^{\ominus}}}.$

Важно отметить, что последнее отношение не изменится, если мы умножим все числа, имеющие отношение к количеству положительных примеров, на одно и то же число c . Это означает, что корень из индекса Джини предназначен для минимизации относительной нечистоты, а потому нечувствителен к изменениям в распределении по классам. Напротив, относительная частота в случае индекса Джини включает дробь n_1/n , которая изменяется, если увеличить число положительных примеров. Нечто подобное происходит и с энтропией. В результате эти два критерия разделения отдают предпочтение потомкам, покрывающим большие примеров.

Еще лучше прояснить ситуацию поможет картинка. Критерии разделения также, как верность и средняя полнота, имеют изолинии в пространствах покрытия и РХП. Из-за нелинейной природы эти изолинии криволинейны. Они также проходят по обе стороны от диагонали, поскольку мы можем поменять местами левого и правого потомка, не изменяя качества разделения. Ландшафт нечистоты можно представить себе как гору, на которую смотрят с высоты, – вершиной является хребет вдоль восходящей диагонали, представляющей разделения, при которых потомки имеют такую же нечистоту, как родитель. Эта гора понижается по обе стороны от хребта и достигает уровня земли в вершине РХП и в противоположной ей точке (которую можно было бы назвать «впадиной РХП»), поскольку именно здесь нечистота нулевая. Изолинии – контурные горизонтали горы – проходят через точки с равной высотой.

Взгляните на рис. 5.7 сверху. Два варианта разделения, между которыми нужно было сделать выбор в примере 5.3 (перед добавлением положительных примеров), показаны точками на графике. Я нарисовал шесть изолиний в левом верхнем углу графика: два варианта разделения, помноженные на три критерия разделения. Любой критерий предпочитает разделение, изолиния которого проходит *выше* (максимально близко к вершине РХП): как видно, только один из трех (корень из индекса Джини) предпочитает правое верхнее разделение. На рис. 5.7 снизу показано, как изменяется картина после увеличения количества положительных примеров в 10 раз (график покрытия не поместился бы на странице, поэтому я изобразил, как это выглядит в пространстве РХП, а линии сетки показывают, как изменилось распределение по классам). Теперь все три критерия разделения предпочитают правый верхний вариант, потому что «горы» энтропии и индекса Джини повернулись по часовой стрелке (индекс Джини в большей степени, чем энтропия), тогда как корень из индекса Джини вообще не сдвинулся с места.

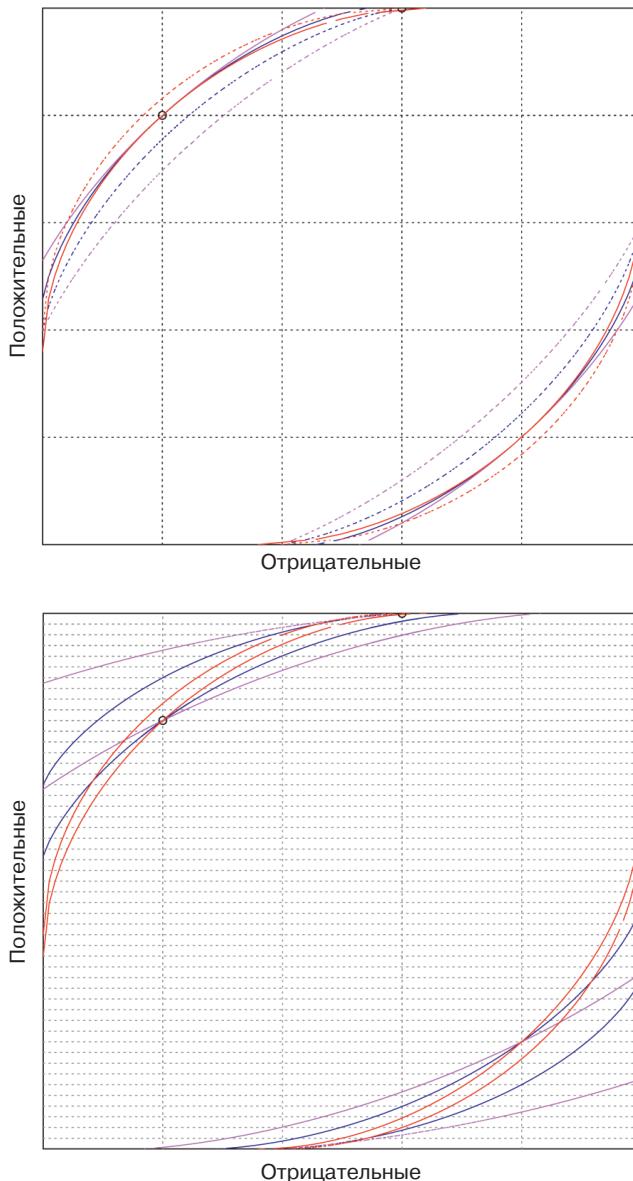


Рис. 5.7. (Сверху) Изолинии РХП для критерия энтропии показаны **синим** цветом, для индекса Джини – **фиолетовым**, а для корня из индекса Джини – **красным**. Изолинии построены для разделений $[8+, 2-][2+, 8-]$ (сплошные линии) и $[10+, 6-][0+, 4-]$ (пунктирные линии). Только корень из индекса Джини предпочитает второе разделение. (**Снизу**) Те же изолинии после увеличения количества положительных примеров в 10 раз. Теперь все критерии предпочитают второе разделение; лишь изолинии корня из индекса Джини не поменяли положения

Мораль сей басни состоит в том, что если при обучении решающего дерева или дерева оценивания вероятностей вы пользуетесь энтропией или индексом Джини как мерой нечистоты – а именно так делается практически во всех пакетах обучения деревьев, – то будьте готовы к тому, что модель изменится, если вы измените распределение по классам за счет добавления лишних примеров. В то же время если в качестве меры взять квадратный корень из индекса Джини, то каждый раз будет получаться одно и то же дерево. Вообще, *энтропия и индекс Джини чувствительны к флуктуациям распределения по классам, корень из индекса Джини – нет*. Так что же выбрать? Моя рекомендация будет в том же русле, что для пометки мажоритарным классом и редукции: используйте нечувствительную к распределению меру нечистоты (например, корень из индекса Джини), если рабочие условия, присутствующие в обучающем наборе, не являются репрезентативными¹.

Подведем итог предшествующему обсуждению древовидных моделей. «А как вы сами стали бы обучать решающее дерево на имеющемся наборе данных?» – спросите вы. Вот перечень шагов, которые я бы предпринял.

1. Прежде всего я поставил бы основной целью добиться хорошего поведения ранжирования, потому что, имея хороший ранжировщик, я могу получить хорошую классификацию и хорошее оценивание вероятностей, а вот обратное может быть и неверно.
2. Поэтому я попытался бы использовать меру нечистоты, не чувствительную к распределению, например квадратный корень из индекса Джини; если ее не предлагают, и я не могу влезть в код программы, то я бы прибег к добавлению примеров мажоритарного класса, чтобы сбалансировать распределение по классам.
3. Я бы отключил редукцию и сгладил оценки вероятностей с помощью поправки Лапласа (или t -оценки).
4. Если бы я знал рабочие условия в месте развертывания, то воспользовался бы ими для выбора наилучшей рабочей точки на кривой РХП (то есть пороговое значение для предсказания вероятностей или разметки дерева).
5. (Факультативно) Наконец, я бы редуцировал поддеревья, все листья которых имеют одну и ту же метку.

Хотя в обсуждении в фокусе нашего внимания были преимущественно задачи бинарной классификации, следует отметить, что решающие деревья без всяких усилий справляются с большим числом классов – как, впрочем, любая группирующая модель. Мы уже отмечали, что для вычисления многоклассовой меры нечистоты нужно просто просуммировать величины нечистоты для каждого класса по схеме «один против остальных». Единственный шаг в списке выше, который не вполне очевиден, когда имеется больше двух классов, – это шаг 4: в этом случае я бы применил обучение, чтобы найти вес каждого класса,

¹ Следует отметить, что не слишком трудно сделать такие меры, как энтропия и индекс Джини, тоже нечувствительными к распределению; по существу, для этого нужно ввесить поправку на наблюдаемое отношение классов $clr = 1$, поделив все счетчики положительных примеров или положительные эмпирические вероятности на clr .

как вкратце объяснялось в разделе 3.1, и, возможно, объединил бы это с шагом 5 и прибег к редукции с уменьшением ошибки (алгоритм 5.3), если бы она была реализована в пакете программ, которым я пользуюсь.

5.3 Обучение деревьев как уменьшение дисперсии

Теперь мы поговорим о том, как применить решающие деревья к задачам регрессии и кластеризации. Как выясняется, это на удивление просто и основано на следующей идеи. Выше мы определили двухклассовый индекс Джини $2\dot{p}(1 - \dot{p})$ листового узла как математическое ожидание ошибки, возникающей при случайной пометке объектов в этом узле: как положительных с вероятностью \dot{p} и как отрицательных с вероятностью $1 - \dot{p}$. Можно рассматривать это как классификацию примеров путем подбрасывания такой монеты, которая выпадает орлом с вероятностью \dot{p} . Если представить это случайной величиной, принимающей значение 1 в случае выпадения орла и значение 0 в случае решки, то ее математическое ожидание равно \dot{p} , а дисперсия $\dot{p}(1 - \dot{p})$ (поиските в сети «испытания Бернулли», если хотите почитать об этом). Это ведет к другой интерпретации индекса Джини – как дисперсии: чем чище лист, тем более несимметричной будет монета и тем меньше дисперсия. Для k классов мы просто суммируем дисперсии всех случайных величин «один против остальных»¹.

Точнее, рассмотрим бинарное разделение на n_1 и $n_2 = n - n_1$ примеров с эмпирическими вероятностями \dot{p}_1 и \dot{p}_2 , тогда средневзвешенная нечистота этих потомков в терминах индекса Джини равна

$$\frac{n_1}{n} 2\dot{p}_1(1 - \dot{p}_1) + \frac{n_2}{n} 2\dot{p}_2(1 - \dot{p}_2) = 2 \left(\frac{n_1}{n} \sigma_1^2 + \frac{n_2}{n} \sigma_2^2 \right),$$

где σ_j^2 – дисперсия распределения Бернулли с вероятностью успеха \dot{p}_j . Итак, нахождение разделения с минимальным средневзвешенным индексом Джини эквивалентно минимизации средневзвешенной дисперсии (множитель 2 присутствует для любого разделения, поэтому его можно опустить), а обучение решающего дерева сводится к такому разбиению пространства объектов, в котором дисперсия в каждом сегменте мала.

Деревья регрессии

В задачах регрессии целевая переменная является непрерывной, а не двоичной, и в таком случае дисперсия множества Y целевых значений определяется как среднеквадратичное расстояние до среднего:

¹ Здесь неявно предполагается, что случайные величины «один против остальных» не коррелируют, что, строго говоря, неверно.

$$\text{Var}(Y) = \frac{1}{|Y|} \sum_{y \in Y} (y - \bar{y})^2,$$

где $\bar{y} = \frac{1}{|Y|} \sum_{y \in Y} y$ – среднее арифметическое целевых значений, содержащихся в Y ; см. замечание 5.1 о некоторых полезных свойствах дисперсии. Если разделение разбивает множество целевых значений Y на взаимно непересекающиеся множества $\{Y_1, \dots, Y_l\}$, то средневзвешенная дисперсия равна

$$\begin{aligned} \text{Var}(\{Y_1, \dots, Y_l\}) &= \sum_{j=1}^l \frac{|Y_j|}{|Y|} \text{Var}(Y_j) = \sum_{j=1}^l \frac{|Y_j|}{|Y|} \left(\frac{1}{|Y_j|} \sum_{y \in Y_j} y^2 - \bar{y}_j^2 \right) = \\ &= \frac{1}{|Y|} \sum_{y \in Y} y^2 - \sum_{j=1}^l \frac{|Y_j|}{|Y|} \bar{y}_j^2. \end{aligned} \quad (5.4)$$

Дисперсия множества чисел $X \subseteq \mathbb{R}$ определяется как среднеквадратичное расстояние до среднего:

$$\text{Var}(X) = \frac{1}{|X|} \sum_{x \in X} (x - \bar{x})^2,$$

где $\bar{x} = \frac{1}{|X|} \sum_{x \in X} x$ – среднее арифметическое по множеству X . Раскрывая скобки $(x - \bar{x})^2 = x^2 - 2\bar{x}x + \bar{x}^2$, мы можем переписать это выражение в виде:

$$\text{Var}(X) = \frac{1}{|X|} \left(\sum_{x \in X} x^2 - 2x \sum_{x \in X} x + \sum_{x \in X} \bar{x}^2 \right) = \frac{1}{|X|} \left(\sum_{x \in X} x^2 - 2\bar{x}|X|\bar{x} + |X|\bar{x}^2 \right) = \frac{1}{|X|} \sum_{x \in X} x^2 - \bar{x}^2. \quad (5.3)$$

Таким образом, дисперсия – это разность между средним арифметическим квадратом среднего арифметического.

Иногда полезно рассматривать среднеквадратичное отклонение от другого значения $x \in \mathbb{R}$, в котором также можно раскрыть скобки:

$$\frac{1}{|X|} \sum_{x \in X} (x - x')^2 = \frac{1}{|X|} \left(\sum_{x \in X} x^2 - 2x'|X|\bar{x} + |X|x'^2 \right) = \text{Var}(X) + (x' - \bar{x})^2.$$

Последнее тождество справедливо потому, что из (5.3) имеем $\frac{1}{|X|} \sum_{x \in X} x^2 = \text{Var}(X) + \bar{x}^2$.

Еще одно полезное свойство заключается в том, что среднеквадратичное отклонение любых двух элементов X равно удвоенной дисперсии:

$$\frac{1}{|X|^2} \sum_{x' \in X} \sum_{x \in X} (x - x')^2 = \frac{1}{|X|} \sum_{x' \in X} (\text{Var}(X) + (x' - \bar{x})^2) = \text{Var}(X) + \frac{1}{|X|} \sum_{x' \in X} (x' - \bar{x})^2 = 2\text{Var}(X).$$

Если $X \subseteq \mathbb{R}^d$ – множество вещественных d -мерных векторов, то мы можем определить дисперсию $\text{Var}_i(X)$ для каждой координаты. А затем интерпретировать сумму дисперсий $\sum_{i=1}^d \text{Var}_i(X)$ как среднеквадратичное евклидово расстояние векторов из X до среднего вектора $\bar{\mathbf{x}} = \frac{1}{|X|} \sum_{x \in X} \mathbf{x}$.

(Иногда встречается определение выборочной дисперсии в виде $\frac{1}{|X|-1} \sum_{x \in X} (x - \bar{x})^2$, при этом получается несколько большее значение. Такая необходимость возникает, если мы оцениваем дисперсию генеральной совокупности, случайной выборкой из которой является X : нормировка на $|X|$ дала бы заниженную оценку дисперсии генеральной совокупности из-за разницы между выборочным средним и средним генеральной совокупности. Здесь нас интересует только оценка разброса заданных значений X безотносительно к некоторой неизвестной генеральной совокупности, так что этот нюанс можно игнорировать.)

Замечание 5.1. К вопросу о дисперсии

Итак, чтобы получить алгоритм обучения дерева регрессии, мы заменим меру нечистоты Imp в алгоритме 5.2 функцией Var. Отметим, что $\frac{1}{|Y|} \sum_{y \in Y} y^2 -$ это константа для заданного множества Y , поэтому минимизация дисперсии по всем возможным разделениям данного родителя – то же самое, что максимизация средневзвешенного квадратов средних по потомкам. Функция Label(Y) аналогично обобщается, так чтобы возвращать среднее значение по Y , а функция Homogeneous(Y) возвращает true, если дисперсия множества целевых значений Y равна нулю (или меньше низкого порога).

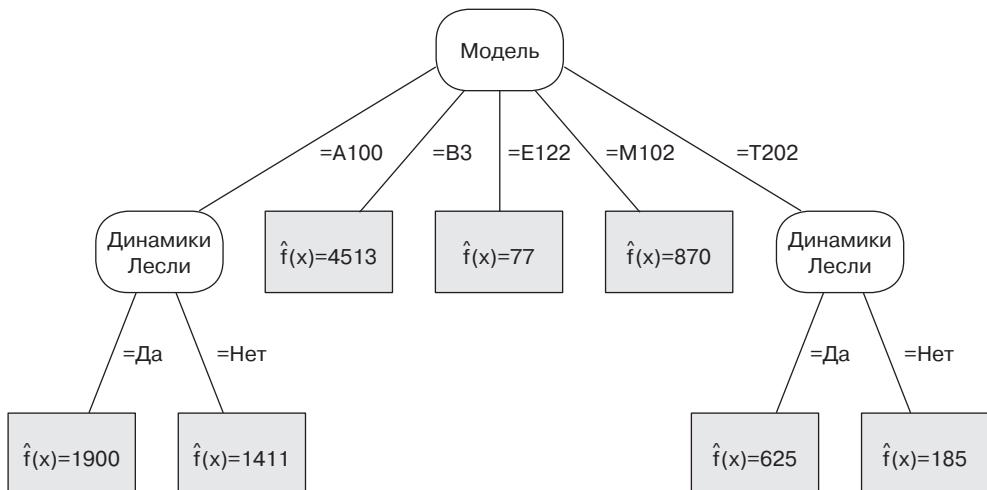


Рис. 5.8. Дерево регрессии, обученное по данным из примера 5.4

Пример 5.4 (обучение дерева регрессии). Представьте, что вы коллекционируете винтажные органы Хаммонда с фоническим колесом. Вы постоянно отслеживаете онлайновый аукционный сайт, с которого получили данные о некоторых интересных сделках.

#	Модель	Состояние	Динамики Лесли	Цена
1.	B3	Отличное	нет	4513
2.	T202	Удовлетворительное	да	625
3.	A100	Хорошее	нет	1051
4.	T202	Хорошее	нет	270
5.	M102	Хорошее	да	870
6.	A100	Отличное	нет	1770
7.	T202	Удовлетворительное	нет	99
8.	A100	Хорошее	да	1900
9.	E112	Удовлетворительное	нет	77

По этим данным вы хотите построить дерево регрессии, которое поможет определить разумную цену вашей следующей покупки.

Всего существует три признака, а значит, и три возможных разделения:

Model = [A100,B3,E112,M102,T202] [1051,1770,1900][4513][77][870][99,270,625]
 Состояние = [отличное,хорошее,удовлетворительное] [1770,4513][270,870,1051,1900][77,99,625]
 Динамики Лесли = [да,нет] [625,870,1900][77,99,270,1051,1770,4513]

Средние первого разделения равны 1574, 4513, 77, 870 и 331, а средневзвешенное квадратов средних равно $3.21 \cdot 10^6$. Средние второго разделения равны 3142, 1023 и 267, а средневзвешенное их квадратов равно $2.68 \cdot 10^6$. Для третьего разделения средние равны 1132 и 1297, а средневзвешенное их квадратов – $1.55 \cdot 10^6$. Таким образом, на верхнем уровне разделение происходит по модели. Это дает нам три листа с одним объектом, а также три модели A100 и три модели T202. Для A100 мы имеем такие разделения:

Состояние = [отличное,хорошее,удовлетворительное] [1770][1051,1900][]
 Динамики Лесли = [да,нет] [1900][1051,1770]

Даже не производя вычислений, видно, что второе разделение дает меньшую дисперсию (для пустого дочернего узла обычно принимается дисперсия его родителя). Для модели T202 разделения такие:

Состояние = [отличное,хорошее,удовлетворительное] []|[270][99,625]
 Динамики Лесли = [да,нет] [625][99,270]

И в этом случае разделение по динамикам Лесли порождает более компактные кластеры значений. Обученное дерево регрессии показано на рис. 5.8.

Деревья регрессии восприимчивы к переобучению. Например, если имеется ровно один пример каждой модели органа Хаммонда, то ветвление по признаку Модель уменьшит среднюю дисперсию по потомкам до нуля. Данные в примере 5.4 на самом деле слишком разрежены, чтобы обучить хорошее дерево регрессии. Кроме того, было бы неплохо зарезервировать редукционный набор и применить редукцию с уменьшением ошибки, редуцируя поддерево, если средняя дисперсия по редукционному набору меньше без поддерева, чем с ним (см. алгоритм 5.3). Следует также отметить, что предсказание константного значения в листе – очень простая стратегия, и существуют методы обучения так называемых *модельных деревьев*, то есть деревьев с линейными регрессионными моделями в листьях (*линейная регрессия* рассматривается в главе 7). В таком случае

критерий разделения был бы основан на корреляции целевой переменной с переменными регрессии, а не просто на дисперсии.

Кластеризующие деревья

Тот простой вид деревьев регрессии, который здесь рассматривается, позволяет также обучать кластеризующие деревья. Это может показаться удивительным, потому что регрессия – проблема обучения с учителем, а кластеризация – без учителя. Однако важно то, что деревья регрессии находят в пространстве объектов сегменты, в которых целевые значения компактно сгруппированы вокруг среднего значения сегмента, – действительно, дисперсия множества целевых значений – не что иное, как (одномерное) среднеквадратичное расстояние до среднего. Это наблюдение сразу же обобщается на вектор целевых значений, поскольку математика при этом не претерпевает существенных изменений. Но можно пойти еще дальше и ввести абстрактную функцию $\text{Dis}: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, которая измеряет расстояние или *расхождение* двух произвольных объектов $x, x' \in X$, такую, что чем больше $\text{Dis}(x, x')$, тем меньше сходство между x и x' . *Кластерное расхождение* множества объектов D вычисляется тогда следующим образом:

$$\text{Dis}(D) = \frac{1}{|D|^2} \sum_{x \in D} \sum_{x' \in D} \text{Dis}(x, x'). \quad (5.5)$$

Средневзвешенное кластерное расхождение по всем потомкам разделения дает *расхождение разделения*, которое можно использовать для информирования функции *BestSplit(D,F)* в \mathcal{F} алгоритме *GrowTree* (алгоритм 5.1 на стр. 145).

Пример 5.5 (обучение кластеризующего дерева с помощью матрицы расхождений). В ходе оценки девяти сделок на аукционном сайте (пример 5.4) с применением таких дополнительных признаков, как отправная цена и количество предложений (сейчас эти признаки несущественны, но показаны в примере 5.6), мы приходим к следующей матрице расхождений:

0	11	6	13	10	3	13	3	12
11	0	1	1	1	3	0	4	0
6	1	0	2	1	1	2	2	1
13	1	2	0	0	4	0	4	0
10	1	1	0	0	3	0	2	0
3	3	1	4	3	0	4	1	3
13	0	2	0	0	4	0	4	0
3	4	2	4	2	1	4	0	4
12	0	1	0	0	3	0	4	0

Отсюда видно, что первая сделка сильно отличается от восьми других. Среднее попарное расхождение по всем девяти сделкам равно 2.94. При тех же признаках, что в примере 5.4, возможны такие три разделения (теперь вместо цены указан номер сделки):

Model = [A100,B3,E112,M102,T202]

[3,6,8][1][9][5][2,4,7]

Состояние = [отличное, хорошее, удовлетворительное]

[1,6][3,4,5,8][2,7,9]

Динамики Лесли = [да, нет]

[2,5,8][1,3,4,6,7,9]

Кластерное расхождение для сделок 3, 6 и 8 равно $(1/3^2)(\mathbf{0} + \mathbf{1} + \mathbf{2} + \mathbf{1} + \mathbf{0} + \mathbf{1} + \mathbf{2} + \mathbf{1} + \mathbf{0}) = 0.89$, а для сделок 2, 4 и 7 – $(1/3^2)(\mathbf{0} + \mathbf{1} + \mathbf{0} + \mathbf{1} + \mathbf{0} + \mathbf{0} + \mathbf{0} + \mathbf{0} + \mathbf{0}) = 0.22$. Остальные три потомка первого разделения содержат по одному элементу, поэтому кластерное расхождение для них равно нулю. Таким образом, средневзвешенное кластерное расхождение разделения равно $3/9 \cdot 0.89 + 1/9 \cdot 0 + 1/9 \cdot 0 + 1/9 \cdot 0 + 3/9 \cdot 0.22 = 0.37$. Для второго разделения аналогичные вычисления дают расхождение $2/9 \cdot 1.5 + 4/9 \cdot 1.19 + 3/9 \cdot 0 = 0.86$, а для третьего – $3/9 \cdot 1.56 + 6/9 \cdot 3.56 = 2.89$. Следовательно, признак «Модель» улавливает большинство имеющихся расхождений, тогда как признак «Динамики Лесли» практически не связан с ними.

Большинство подводных камней, характерных для деревьев регрессии, применимо и к кластеризующим деревьям: у мелких кластеров расхождение обычно невелико, поэтому они подвержены опасности переобучения. Рекомендуется резервировать редукционный набор для удаления нижних разделений, если они не улучшают связности кластеров на редукционном наборе. Одиночные примеры могут доминировать: в приведенном выше примере удаление первой сделки уменьшает попарное расхождение с 2.94 до 1.5, и потому трудно будет найти что-то лучше разделения, которое помещает эту сделку в отдельный кластер.

Возникает интересный вопрос: как следует пометить листья кластеризующего дерева? Интуитивно кажется, что разумно будет пометить кластер наиболее представительным для него объектом. Наиболее представительный объект можно определить как такой, для которого суммарное расхождение со всеми другими объектами минимально, – в главе 8 такой объект назван медоидом. Например, в кластере A100 наиболее представительной является сделка 6, потому что ее расхождение со сделками 3 и 8 равно 1, тогда как расхождение между сделками 3 и 8 равно 2. Аналогично в кластере T202 наиболее представительна сделка 7. Однако совершенно необязательно, что наиболее представительный объект единственный.

Типичная ситуация, в которой удается упростить вычисления, необходимые для нахождения наилучшего разделения, и получить уникальную метку кластера, возникает, когда расхождения определены как евклидовы расстояния, посчитанные по числовым признакам. В замечании 5.1 показано, что если $\text{Dis}(x, x')$ – квадрат евклидова расстояния, то $\text{Dis}(D)$ – удвоенное среднеквадратичное евклидово расстояние до среднего. Это позволяет упростить вычисления, потому что как среднее, так и среднеквадратичное расстояние до среднего можно вычислить за $O(|D|)$ шагов (за один проход по данным), а не за $O(|D|^2)$ шагов, необходимых, если у нас нет ничего, кроме матрицы расхождений. На самом деле среднеквадратичное евклидово расстояние – это просто сумма дисперсий отдельных признаков.

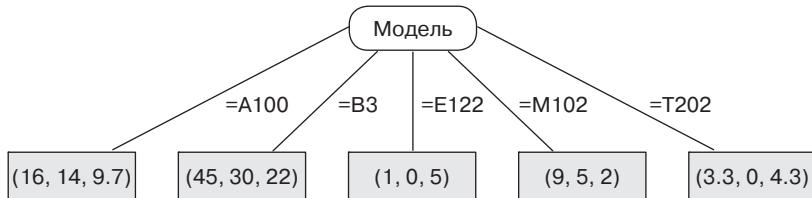


Рис. 5.9. Кластеризующее дерево, обученное на данных из примера 5.6 с использованием евклидова расстояния для числовых признаков

Пример 5.6 (обучение кластеризующего дерева с евклидовым расстоянием). Пополним данные о сделках с органами Хаммонда еще двумя числовыми признаками: отправная цена и количество предложений во время аукциона. Продажная и отправная цена выражена в сотнях фунтов стерлингов, чтобы у всех трех числовых признаков был примерно одинаковый вес при вычислении расстояний.

Модель	Состояние	Динамики Лесли	Цена	Отправная	Предложений
B3	Отличное	Нет	45	30	22
T202	Удовлетворительное	Да	6	0	9
A100	Хорошее	Нет	11	8	13
T202	Хорошее	Нет	3	0	1
M102	Хорошее	Да	9	5	2
A100	Отличное	Нет	18	15	15
T202	Удовлетворительное	Нет	1	0	3
A100	Хорошее	Да	19	19	1
E112	Удовлетворительное	Нет	1	0	5

Средние трех числовых признаков равны (13.3, 8.6, 7.9), а их дисперсии – (158, 101.8, 48.8). Среднеквадратичное евклидово расстояние до среднего равно сумме дисперсий, то есть 308.6 (при желании мы могли бы умножить эту величину на два и получить кластерное расхождение, определенное в уравнении 5.5). Для кластера A100 мы имеем векторы (16, 14, 9.7) и (12.7, 20.7, 38.2), для которых среднеквадратичное расстояние до среднего равно 71.6; для кластера T202 с векторами (3.3, 0, 4.3) и (4.2, 0, 11.6) среднеквадратичное расстояние равно 15.8. Используя такое разделение, мы можем построить кластеризующее дерево, листья которого помечены средними векторами (рис. 5.9).

В этом примере мы использовали категориальные признаки для разделения и числовые признаки – для вычисления расстояний. Действительно, во всех рассмотренных до сих пор примерах деревьев для разделения применялись только категориальные признаки¹. На практике числовые признаки часто используются

¹ Категориальными называются признаки, принимающие значения из сравнительного небольшого дискретного множества. Технически они отличаются от числовых тем, что не имеют масштабной шкалы и не упорядочены. Эта тема рассматривается ниже в главе 10.

для разделения, нужно лишь найти подходящее пороговое значение t – такое, что признаку F можно сопоставить бинарное разделение с условиями $F \geq t$ и $F < t$. Нахождение оптимальной точки разделения тесно связано с *дискретизацией* числовых признаков – вопросом, который будет подробно рассматриваться в главе 10. А пока приведем несколько наблюдений, которые дают представление о том, как можно определить порог числового признака в ходе обучения.

- ☞ Хотя теоретически существует бесконечно много возможных порогов, на практике нужно рассматривать только значения, которые разделяют два примера, оказавшиеся по соседству после сортировки обучающих данных по возрастанию (или по убыванию) значения признака.
- ☞ Если решается задача классификации, то нужно рассматривать только соседние примеры, принадлежащие разным классам; если это задача регрессии, то примеры, для которых целевые значения достаточно сильно различаются, а в случае задачи кластеризации – примеры с достаточно большим расхождением.
- ☞ Каждое потенциальное пороговое значение следует оценивать так, будто это отдельный бинарный признак.

5.4 Древовидные модели: итоги и литература для дальнейшего чтения

Древовидные структуры данных встречаются в информатике повсеместно, и машинное обучение – не исключение. Древовидные модели лаконичны, легко поддаются интерпретации и обучению и могут применяться к широкому кругу задач, включая классификацию, ранжирование, оценивание вероятностей, регрессию и кластеризацию. Древовидный классификатор для распознавания позы человека в датчике движения Microsoft Kinect описан в работе Shotton et al. (2011).

- ☞ Я ввел дерево признаков как общую основу всех древовидных моделей и рекурсивный алгоритм *GrowTree* как общий алгоритм типа «разделяй и властвуй», который можно специализировать для задач каждого вида путем подходящего выбора функций, которые оценивают однородность набора данных и находят подходящую метку, если набор однороден, или наилучший признак для разделения, если набор таковым не является.
- ☞ Применение дерева признаков для предсказания меток классов превращает их в решающие деревья – тема раздела 5.1. В машинном обучении существуют два классических подхода к решающим деревьям, которые очень похожи алгоритмически, но различаются в таких деталях, как эвристики и стратегии редукции. Подход Квинлана основан на использовании энтропии в качестве меры нечистоты, он эволюционировал от алгоритма ID3 (Quinlan, 1986), который, в свою очередь, был основан на работе Hunt, Marin, Stone (1966) до изощренного алгоритма C4.5 (Quinlan, 1993). Подход CART (Classification and Regression Trees – деревья классификации

и регрессии) описан в работе Breiman, Friedman, Olshen, Stone (1984); в нем в качестве меры нечистоты используется индекс Джини. Применение квадратного корня из индекса Джини как меры нечистоты было впервые предложено в работе Dietterich, Kearns, Mansour (1996) и потому иногда называется методом **DKM**. Геометрическое построение для нахождения $\text{Imp}([D1, D2])$, показанное на рис. 5.2 справа, также основано на идеях из этой работы.

- ☞ Применение эмпирических распределений в листьях дерева признаков для построения ранжировщиков и оценок вероятностей, описанное в разделе 5.2, – сравнительно недавняя разработка (Ferri et al., 2002; Provost, Domingos, 2003). Экспериментальные результаты, демонстрирующие, что более точные оценки вероятностей получаются, если отказаться от редукции дерева и сглаживать эмпирические вероятности с помощью поправки Лапласа, приведены в последней работе и подтверждены в работе Ferri et al. (2003). Степень нечувствительности критерия разделения решающего дерева к несбалансированности классов или стоимости неправильной классификации была изучена и объяснена в работах Drummond, Holte (2000) и Flach (2003). Из трех вышеупомянутых критериев разделения только корень из индекса Джини нечувствителен к такому дисбалансу классов и стоимости.
- ☞ Древовидные модели – это группирующие модели, нацеленные на минимизацию разнообразия в листьях, причем что именно понимать под разнообразием, зависит от задачи. Очень часто разнообразие можно интерпретировать как тот или иной вариант дисперсии; эта идея присутствовала уже в работе Breiman et al., (1984), а затем к ней неоднократно возвращались в работах Langley (1994), Kramer (1996), Blockeel, De Raedt, Ramon (1998) и ряде других. В разделе 5.3 мы видели, как эту идею можно применить к обучению деревьев регрессии и кластеризующих деревьев (при этом за кадром остались многие важные детали, например когда прекращать разделение узлов).

Следует помнить, что высокая выразительность древовидных моделей, по сравнению, например, с конъюнктивными концептами, означает, что нужно страховаться от переобучения. Кроме того, жадным алгоритмам типа «разделяй и властвуй» свойствен тот недостаток, что малые изменения в обучающих данных могут привести к выбору другого признака в корне дерева, что повлияет на выбор признаков в последующих разделениях. В главе 11 мы увидим, как с помощью таких методов, как баггинг, можно уменьшить подобную неустойчивость модели.



Модели на основе правил

Модели на основе правил – второй из основных типов логических моделей в машинном обучении. Вообще говоря, они обладают большей гибкостью, по сравнению с древовидными моделями, например если ветви решающего дерева являются взаимно исключающими, то правила могут перекрываться, и это иногда дает дополнительную информацию. Однако за эту гибкость приходится платить: хотя возникает сильный соблазн рассматривать правило как независимую информационную единицу, зачастую так поступать нельзя из-за способа обучения правил. И особенно это относится к обучению с учителем, где модель на основе правил – не просто набор правил, а еще – и это очень важная часть модели – спецификация того, как правила должны объединяться для образования предсказаний.

Существуют два основных подхода к обучению правил с учителем. Один основан на идее обучения решающих деревьев: найти комбинацию литералов – *тело* правила, которое мы раньше называли концептом, – которая покрывает достаточно однородное множество примеров, и метку, которую следует поместить в *заголовок* правила. Второй подход противоположен: сначала выбирается класс, который мы хотим обучить, а затем ищутся тела правил, которые покрывают большие подмножества примеров этого класса. Первый подход естественно приводит к модели, состоящей из упорядоченной последовательности правил – *списка правил*, он обсуждается в разделе 6.1. Во втором подходе наборы правил рассматриваются как неупорядоченные *множества правил*, это тема раздела 6.2. Как мы увидим, в этих моделях перекрытие правил обрабатывается по-разному. В третьем разделе этой главы будет рассмотрено выявление подгрупп и ассоциативных правил.

6.1 Обучение упорядоченных списков правил

Основная идея такого алгоритма обучения правил состоит в том, чтобы наращивать тело конъюнктивного правила, добавляя на каждом шаге литерал, который максимально улучшает его однородность. Таким образом, мы строим нисходящий путь в пространстве гипотез, похожий на обсуждавшийся в разделе 4.2, и останавливаемся, как только будет удовлетворен критерий однородности. Однородность естественно измерять в терминах чистоты, как мы делали в случае решающих деревьев. Можете считать, что добавление литерала в тело правила –

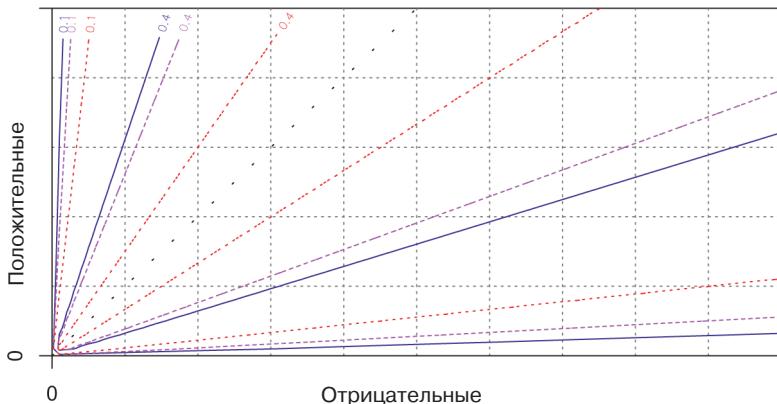


Рис. 6.1. Изолинии РХП для энтропии (масштабированы так, что максимум равен 1/2), индекса Джини и миноритарного класса. Серая пунктирная ось симметрии определена уравнением $\hat{p} = 1/2$; каждая изолиния состоит из двух частей: над осью симметрии (где нечистота убывает с ростом эмпирической вероятности \hat{p}) и ее отражение относительно оси симметрии (где нечистота пропорциональна \hat{p}). Если эти меры нечистоты используются в качестве поисковых эвристик, как при обучении правил, то существенна только форма изолинии, а не сами значения нечистоты, а потому все эти меры эквивалентны

полный аналог добавления бинарного разделения в решающее дерево, поскольку добавленный литерал разбивает множество объектов, покрытых исходным телом правила, на две группы: объекты, для которых новый литерал равен `true`, и объекты, для которых он равен `false`. А ключевое отличие состоит в том, что при обучении решающего дерева нас интересует чистота *обоих* потомков, и именно поэтому при его построении мы в качестве поисковой эвристики используем средневзвешенную нечистоту. С другой стороны, при обучении правила нас интересует только чистота одного из потомков: того, для которого добавленный литерал равен `true`. Отсюда следует, что мы можем напрямую использовать любую из мер нечистоты, рассмотренных в предыдущей главе (см. рис. 5.2, если подзабыли, как они выглядят), не прибегая к усреднению.

На самом деле даже не важно, какую именно меру нечистоты использовать для управления поиском, потому что все они дают одинаковый результат. Чтобы убедиться в этом, заметим, что нечистота концепта убывает вместе с эмпирической вероятностью \hat{p} (относительной частотой покрытых положительных примеров), если $\hat{p} > 1/2$, и возрастает, если $\hat{p} < 1/2$; см. рис. 6.1. Линейно это возрастание или убывание или нет, существенно, если мы усредняем нечистоту нескольких концептов, как при обучении решающего дерева, но несущественно, если вычисляются одиночные концепты. Другими словами, различие между этими мерами нечистоты при обучении правил исчезает, и мы могли бы с тем же успехом взять в качестве меры долю миноритарного класса $\min(\hat{p}, 1 - \hat{p})$ (или, если угодно, $1/2 - |\hat{p} - 1/2|$) — пожалуй, самую простую из всех. Только помните, что если другие авторы используют энтропию или индекс Джини для сравнения нечистоты

ты литералов или тел правил, то результат от этого не изменится (если говорить не о конкретных значениях нечистоты, а о том, какая мера лучше).

Мы проиллюстрируем основной алгоритм обучения списков правил на примере.

Пример 6.1 (обучение списка правил). Снова возьмем наш небольшой набор данных о дельфиах с такими положительными примерами:

p1: *Длина = 3* \wedge *Жабры = нет* \wedge *Клювовидный выступ = да* \wedge *Зубы = много*;

p2: *Длина = 4* \wedge *Жабры = нет* \wedge *Клювовидный выступ = да* \wedge *Зубы = много*;

p3: *Длина = 3* \wedge *Жабры = нет* \wedge *Клювовидный выступ = да* \wedge *Зубы = мало*;

p4: *Длина = 5* \wedge *Жабры = нет* \wedge *Клювовидный выступ = да* \wedge *Зубы = много*;

p5: *Длина = 5* \wedge *Жабры = нет* \wedge *Клювовидный выступ = да* \wedge *Зубы = мало*

и такими отрицательными:

n1: *Длина = 5* \wedge *Жабры = да* \wedge *Клювовидный выступ = да* \wedge *Зубы = много*;

n2: *Длина = 4* \wedge *Жабры = да* \wedge *Клювовидный выступ = да* \wedge *Зубы = много*;

n3: *Длина = 5* \wedge *Жабры = да* \wedge *Клювовидный выступ = нет* \wedge *Зубы = много*;

n4: *Длина = 4* \wedge *Жабры = да* \wedge *Клювовидный выступ = нет* \wedge *Зубы = много*;

n5: *Длина = 4* \wedge *Жабры = нет* \wedge *Клювовидный выступ = да* \wedge *Зубы = мало*.

Девять возможных литералов вместе со счетчиками покрытия показаны на рис. 6.2 сверху. Три из них чистые; на графике изолиний нечистоты (рис. 6.2 снизу) они оказываются на осиях *x* и *y*. Один литерал покрывает два положительных и два отрицательных примера и потому имеет такую же нечистоту, как весь набор данных; на графике покрытия этот литерал оказывается на восходящей диагонали.

Хотя сама по себе нечистота не различает чистых литералов (мы еще вернемся к этому моменту), можно счесть, что *Жабры = да* – наилучший литерал из трех, потому что он покрывает больше примеров, поэтому сформулируем наше первое правило в виде:

if Жабры = да then Class = \ominus .

Соответствующая точка покрытия обозначена на рис. 6.2 снизу стрелкой. Можете считать эту стрелку самым правым концом кривой покрытия, которая получится, если продолжить построение исходящего пути в пространстве гипотез путем добавления литералов. В данном случае нас не интересует продолжение этого пути, потому что найденный концепт уже чист (ниже мы увидим примеры, когда для выхода на какую-либо ось требуется добавить несколько литералов). А вот с чем мы раньше не сталкивались, так это с тем, что данная кривая покрытия лежит целиком под диагональю, – это следствие того факта, что мы не фиксировали класс заранее, а потому ныряние под восходящую диагональ устраивает нас не меньше, чем воспарение над ней. Можно осмыслить это и по-другому: перестановка меток местами отражается только на заголовках, но не на телах обученных правил.

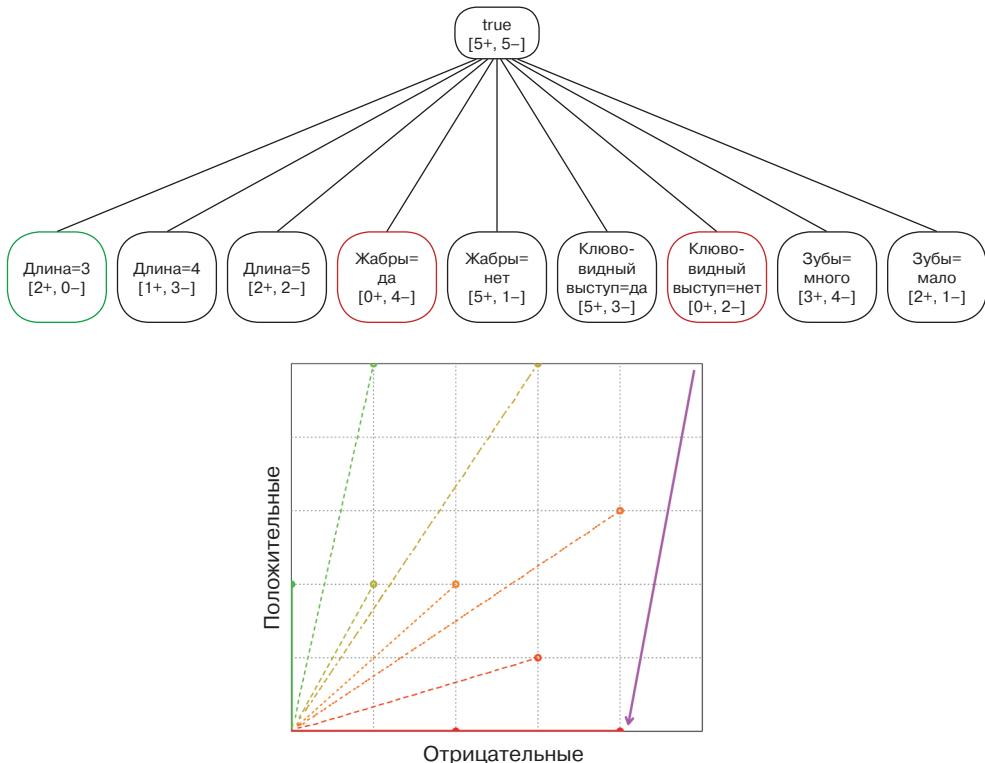


Рис. 6.2. (Сверху) Все литералы и их счетчики покрытия для данных из примера 6.1. **Зеленым (красным)** цветом показаны чистые литералы для положительного (отрицательного) класса. **(Снизу)** Все девять литералов изображены точками в пространстве покрытия, а значения их нечистоты представлены изолиниями нечистоты (чем дальше от восходящей диагонали, тем лучше). Значения нечистоты обозначены цветом: ближе к **зеленому**, если $\hat{p} > 1/2$; ближе к **красному**, если $\hat{p} < 1/2$; и **оранжевым**, если $\hat{p} = 1/2$ (на изолинии, наклоненной под углом 45°). **Фиолетовая** стрелка обозначает выбранный литерал, который исключает все пять положительных примеров и один отрицательный

Большинство алгоритмов обучения правил далее работают следующим образом: исключают из рассмотрения примеры, покрытые только что обученным правилом, и продолжают обрабатывать оставшиеся примеры. Эта стратегия называется «*отделяй и властвуй*» по аналогии со стратегией «разделяй и властвуй», применяемой при обучении решающих деревьев (разница в том, что в случае стратегии «отделяй и властвуй» у нас остается одна подпроблема, а не несколько, как в случае стратегии «разделяй и властвуй»). Таким образом, у нас осталось пять положительных примеров и один отрицательный, и мы снова ищем литералы с минимальной нечистотой. Как показано на рис. 6.3, мы можем интерпретировать это как работу в меньшем пространстве покрытия. Произведя вычисления, мы находим, что следующее обученное правило имеет вид:

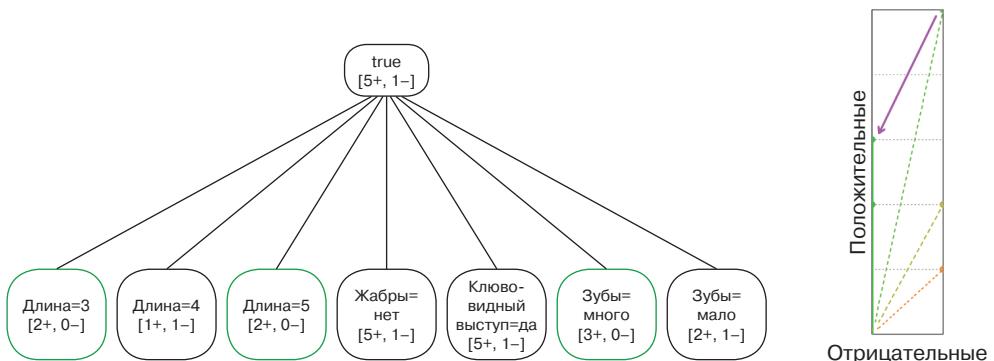


Рис. 6.3. (Слева) Пересмотренные счетчики покрытия после исключения четырех отрицательных примеров, покрытых первым найденным правилом (литералы, не покрывающие ни одного примера, опущены). **(Справа)** Теперь мы работаем в самом правом «срезе» рис. 6.2

if Зубы = много **then** Class = \oplus .

Как я уже говорил, следует проявлять осторожность при интерпретации в отрыве от контекста: вот и это правило в применении к исходному набору данных покрывает больше отрицательных примеров, чем положительных! Иными словами, это правило неявно предполагает, что предыдущее правило не «срабатывает»; в окончательной модели этому правилу будет предшествовать ‘else’.

Теперь у нас осталось два положительных примера и один отрицательный (рис. 6.4). На этот раз имеет смысл выбрать правило, которое покрывает единственный оставшийся отрицательный пример, а именно:

if Длина = 4 **then** Class = \ominus .

Поскольку все остальные примеры положительны, мы можем применить *правило по умолчанию*, которое покрывает примеры, на которых все прочие правила не сработали. Если собрать все вместе, то получится следующая обученная модель на основе правил:

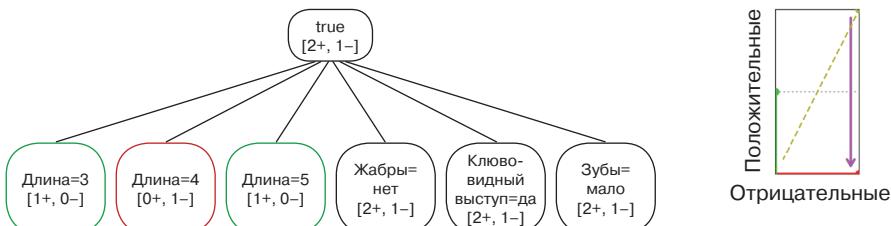


Рис. 6.4. (Слева) Третье правило покрывает один оставшийся отрицательный пример, а все оставшиеся положительные можно охватить правилом по умолчанию. **(Справа)** Пространство покрытия схлопывается

```

if Жабры = да then Class = ⊖;
else if Зубы = много then Class = ⊕;
else if Длина = 4 then Class = ⊖;
else Class = ⊕.

```

Организация правил в виде списка – один из способов обработки перекрытия правил. Например, мы знаем, что есть несколько примеров, в которых одновременно **Жабры = да** и **Зубы = много**, но список правил говорит, что в таких случаях приоритет отдается первому правилу. Однако можно было бы переписать список правил так, чтобы все правила были взаимно исключающими. Это полезно, поскольку тогда каждое правило можно будет использовать независимо от остальных и применять правила в любом порядке. Единственное, но не слишком серьезное осложнение состоит в том, что потребуется отрицание литералов (или внутренняя дизъюнкция) для тех признаков, у которых более двух значений, например «Длина»:

```

if Жабры = да then Class = ⊖;
if Жабры = нет ∧ Зубы = много then Class = ⊕;
if Жабры = нет ∧ Зубы = мало ∧ Длина = 4 then Class = ⊖;
if Жабры = нет ∧ Зубы = мало ∧ Длина = 4 then Class = ⊕.

```

В данном примере мы опираемся на тот факт, что в этом конкретном наборе правил в каждом правиле встречается всего один литерал. В общем случае понадобились бы неконъюнктивные тела правил. Рассмотрим, к примеру, следующий список правил:

```

if P ∧ Q then Class = ⊕;
else if R then Class = ⊖.

```

Чтобы сделать эти правила взаимно исключающими, второе пришлось бы переписать в виде:

if $\neg(P \wedge Q) \wedge R$ **then** Class = ⊖,

или эквивалентно:

if $(\neg P \vee \neg Q) \wedge R$ **then** Class = ⊖.

Понятно, что взаимно исключающие правила не столь компактны, и это объясняет, почему списки правил так эффективны и широко распространены.

В алгоритме 6.1 стратегия обучения правил «отделяй и властуй» описана более детально. Пока еще остаются обучающие примеры, алгоритм обучает очередное правило и исключает из набора данных все примеры, покрытые этим правилом. Этот алгоритм лежит в основе большинства систем обучения правил и называется также **алгоритмом построения покрытия**. Алгоритм обучения одного правила приведен в алгоритме 6.2. По аналогии с решающими деревьями в нем используются функции **Homogeneous(D)** и **Label(D)**, которые решают, нужна ли

дальнейшая специализация и какой класс поместить в заголовок правила соответственно. Используется также функция $\text{BestLiteral}(D, L)$, которая выбирает наилучший литерал для добавления в правило из множества кандидатов L при данных D ; в примере выше этот литерал выбирался бы, исходя из чистоты.

В литературе описано много вариантов этих алгоритмов. Условия в циклах while часто ослабляются за счет применения другого *критерия остановки*, чтобы можно было работать с зашумленными данными. Например, в алгоритме 6.1 можно было бы остановиться, когда ни в одном классе не останется больше определенного количества примеров, а для всех оставшихся примеров включить правило по умолчанию. Аналогично в алгоритме 6.2 можно было бы останавливаться, когда размер D окажется меньше определенного порога.

Списки правил имеют много общего с решающими деревьями. Поэтому мы можем проанализировать построение списка правил так же, как сделали это на рис. 5.3. Для нашего сквозного примера результат показан на рис. 6.5. Например, добавление первого правила изображено в пространстве покрытия как проведение восходящей диагонали A , которая отщепляет горизонтальный отрезок B , представляющий новое правило, и другой диагональный отрезок C , представляющий новое пространство покрытия. Добавление второго правила приводит к расщеплению отрезка C на вертикальный отрезок D (второе правило) и диагональный отрезок E (третье пространство покрытия). Наконец, E расщепляется на горизонтальный и вертикальный отрезки (третье правило и правило по умолчанию соответственно). Оставшиеся отрезки B , D , F и G горизонтальные или вертикальные, это говорит о том, что обученные нами правила чистые.

Алгоритм 6.1. $\text{LearnRuleList}(D)$ – обучить упорядоченный список правил

Вход: размеченные обучающие данные D .
Выход: список правил R .

```

1  $R \leftarrow \emptyset;$ 
2 while  $D \neq \emptyset$  do
3    $r \leftarrow \text{LearnRule}(D);$                                 // LearnRule: см. алгоритм 6.2
4   добавить  $r$  в конец  $R;$ 
5    $D \leftarrow D \setminus \{x \in D \mid x$  покрыт  $r\};$ 
6 end
7 return  $R$ 
```

Списки правил для ранжирования и оценивания вероятностей

Превращение списка правил в ранжировщик или средство оценивания вероятностей так же просто, как для решающих деревьев. Благодаря алгоритму построения покрытия мы имеем доступ к локальным распределениям по классам, ассоциированным с каждым правилом. Поэтому в основу оценок можно положить эмпирические вероятности. В случае двух классов объекты можно ранжировать по убыванию эмпирической вероятности положительного класса, что дает

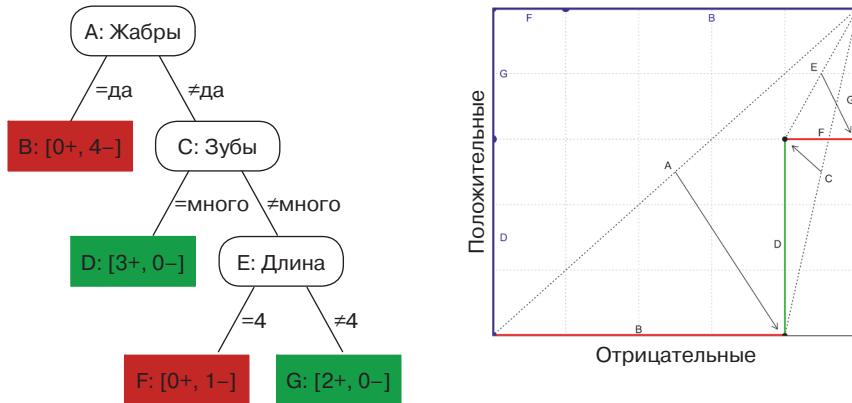


Рис. 6.5. (Слева) Ветвящееся вправо дерево признаков, соответствующее списку правил с одним литералом. **(Справа)** Построение этого дерева в пространстве покрытия. Листья дерева либо чисто положительные (зеленые), либо чисто отрицательные (красные). Переупорядочение листьев по эмпирической вероятности приводит к синей кривой покрытия. Поскольку список правил отделяет классы, то эта кривая покрытия идеальна

кривую покрытия, в которой имеется по одному отрезку для каждого правила. Важно отметить, что порядок ранжирования правил отличается от их порядка в списке – точно так же, как порядок ранжирования листьев дерева отличается от их порядка при обходе слева направо.

Алгоритм 6.2. *LearnRule(D)* – обучение одного правила

Вход: размеченные обучающие данные D .
Выход: правило r .

- 1 $b \leftarrow \text{true};$
- 2 $L \leftarrow \text{множество имеющихся литералов};$
- 3 **while** $\text{not Homogeneous}(D)$ **do**
- 4 $l \leftarrow \text{BestLiteral}(D, L);$ // например, максимальная частота, см. текст
- 5 $b \leftarrow b \wedge l;$
- 6 $D \leftarrow \{x \in D \mid x \text{ покрыт } b\};$
- 7 $L \leftarrow L \setminus \{l' \in L \mid l' \text{ использует тот же признак, что } l\};$
- 8 **end**
- 9 $C \leftarrow \text{Label}(D);$ // например, мажоритарный класс
- 10 $r \leftarrow \text{if } b \text{ then Class } = C;$
- 11 **return** r

Пример 6.2 (списки правил как ранжировщики). Рассмотрим следующие два концепта:

- | | | |
|-----------------------------|------|---------|
| (A) Длина = 4 | p2 | n2,n4–5 |
| (B) Клювовидный выступ = да | p1–5 | n1–2,n5 |

Справа указано покрытие каждым концептом всего обучающего набора. Используя эти концепты как тела правил, мы можем построить список правил **AB**:

if Длина = 4 then Class = \ominus	[1+,3-]
else if Клювовидный выступ = да then Class = \oplus	[4+,1-]
else Class = \ominus	[0+,1-]

Кривая покрытия этого списка правил показана на рис. 6.6. Первый отрезок кривой соответствует всем объектам, покрытым **B**, но не **A**, поэтому мы воспользовались теоретико-множественной нотацией **B\A**. Отметим, что хотя этот отрезок соответствует второму правилу в списке, в кривой покрытия он первый, потому что покрывает максимальное количество положительных примеров. Второй отрезок кривой покрытия соответствует правилу **A**, а третий – обозначенный ‘–’ – правилу по умолчанию. Этот отрезок последний, но не потому, что представляет последнее правило, а потому, что не покрывает ни одного положительного примера. Мы можем построить список правил в обратном порядке, **BA**:

if Клювовидный выступ = да then Class = \oplus	[5+,3-]
else if Длина = 4 then Class = \ominus	[0+,1-]
else Class = \ominus	[0+,1-]

Кривая покрытия для этого списка правил также изображена на рис. 6.6. На этот раз ее первый отрезок соответствует первому в списке правилу (**B**), а второй и третий образуют неопределенность между правилом **A** (после того как исключены объекты, покрытые правилом **B**: **A\B**) и правилом по умолчанию.

Какой из этих списков правил является лучшим ранжировщиком? Мы видим, что **AB** делает меньше ошибок ранжирования, чем **BA** (4.5 против 7.5), и потому имеет лучший показатель AUC (0.82 против 0.70). Мы также видим, что если критерием качества является верность, то **AB** будет оптимальен, поскольку достигает верности 0.80 ($tpr = 0.80$ и $tnr = 0.80$), тогда как **BA** – только 0.70 ($tpr = 1$ и $tnr = 0.40$). Однако если качество на положительных примерах в 3 раза важнее качества на отрицательных примерах, то оптимальная рабочая точка **BA** оказывается лучше, чем у **AB**. Таким образом, каждый список правил содержит информацию, отсутствующую в другом, и потому нельзя сказать, что один во всех отношениях лучше другого.

Главная причина этого состоит в том, что отрезок **A \wedge B** – перекрытие обоих правил – не доступен ни одному из списков правил. На рис. 6.6 он обозначен пунктирной линией, соединяющей отрезок **B** из списка правил **BA** и отрезок **B\A** из списка правил **AB**. Отсюда следует, что этот отрезок содержит те и только те примеры, которые входят в **B**, но не входят в **B\A**, то есть принадлежащие **A \wedge B**. Для доступа к перекрытию правил нужно либо объединить оба списка, либо выйти за пределы возможностей списков правил. Мы исследуем этот вопрос в конце следующего раздела.

Таким образом, списки правил и решающие деревья родственны в нескольких отношениях. Ко всему прочему *списки правил подобны решающим деревьям в том смысле, что эмпирические вероятности, ассоциированные с каждым правилом, порождают выпуклые кривые RXP и покрытия на обучающих данных*. Мы можем получить эмпирические вероятности в силу алгоритма построения покрытия,

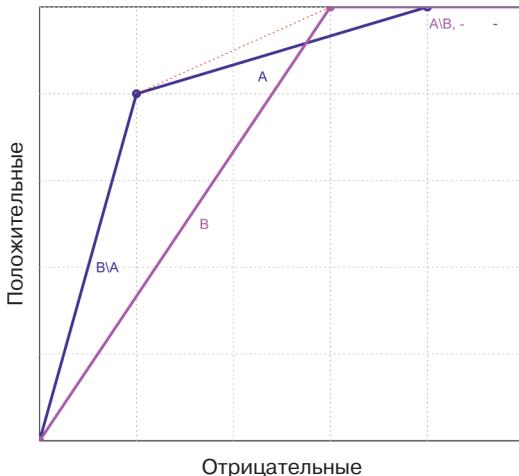


Рис. 6.6. Кривые покрытия двух списков правил из примера 6.2, взятых в разном порядке (**AB синим цветом, BA фиолетовым**). $B \setminus A$ соответствует покрытию правила **B** после исключения объектов, покрытых правилом **A**, а ' $-$ ' обозначает правило по умолчанию. Ни одна кривая не доминирует над другой, то есть для каждой существуют рабочие условия, при которых она лучше. Отрезок, соединяющий две кривые и изображенный **красной** пунктирной линией, соответствует перекрытию двух правил **A \wedge B**, не доступному ни одному из списков правил

которое исключает из обучающего набора все объекты, покрытые уже обученным правилом, перед тем как переходить к обучению следующего (алгоритм 6.1). В результате списки правил порождают вероятности, хорошо откалиброванные на обучающем наборе. Некоторые алгоритмы обучения, описанные в литературе, переупорядочивают список правил после того, как все правила построены. В таком случае выпуклость не гарантируется, если только не пересчитать покрытие каждого правила в переупорядоченном списке.

6.2 Обучение неупорядоченных множеств правил

Перейдем к альтернативному подходу к обучению правил, при котором обучение производится по одному классу за раз. Это означает, что поисковую эвристику можно еще упростить: вместо того чтобы искать минимум $\min(p, 1 - p)$, мы можем искать максимум p – эмпирической вероятности обучаемого класса. Этую поисковую эвристику традиционно называют **точностью** – по аналогии с оценочным показателем качества классификатора (см. табл. 2.3 на стр. 71).

Пример 6.3 (обучение множества правил для одного класса). Продолжим пример с дельфинами. На рис. 6.7 показано первое правило, обученное для положительного класса:

if Длина = 3 then Class = \oplus .

Оба покрытых этим правилом примера исключены, и мы приступаем к обучению нового правила. Здесь мы сталкиваемся с новой ситуацией, поскольку ни один из кандидатов не является чистым (рис. 6.8). Поэтому мы начинаем поиск второго уровня, результатом которого является следующее чистое правило:

if Жабры = нет \wedge Длина = 5 then Class = \oplus .

Чтобы покрыть оставшийся положительный пример, нам снова нужно правило с двумя условиями (рис. 6.9):

if Жабры = нет \wedge Зубы = много then Class = \oplus .

Отметим, что хотя эти правила и перекрываются, их перекрытие покрывает только положительные примеры (поскольку оба они чистые) и, значит, нет необходимости организовывать их в виде списка if-then-else.

Теперь мы имеем множество правил для положительного класса. При двух классах это можно считать достаточным, поскольку все объекты, не покрытые положительными правилами, можно классифицировать как отрицательные. Однако при этом может появиться смещение в сторону отрицательного класса, поскольку все трудные случаи, в которых мы не уверены, автоматически классифицируются как отрицательные. Поэтому обучим несколько правил для отрицательного класса. Применяя ту же процедуру, что в примере 6.3, мы находим следующие правила (можете проверить): сначала **if Жабры = да then Class = \ominus** , а потом **if Длина = 4 \wedge Зубы = мало then Class = \ominus** . Таким образом, окончательное множество, содержащее правила для обоих классов, имеет вид:

- (R1) **if Длина = 3 then Class = \oplus ;**
- (R2) **if Жабры = нет \wedge Длина = 5 then Class = \oplus ;**
- (R3) **if Жабры = нет \wedge Зубы = много then Class = \oplus ;**
- (R4) **if Жабры = да then Class = \ominus ;**
- (R5) **if Длина = 4 \wedge Зубы = мало then Class = \ominus .**

Обучение множества правил описывается алгоритмом 6.3. Основные отличия от алгоритма **LearnRuleList** (алгоритм 6.1) заключаются в том, что теперь мы перебираем классы по очереди, а кроме того, после нахождения правила исключаются только покрытые им примеры, принадлежащие классу, рассматриваемому в данный момент. Причина второго изменения в том, что правила, входящие во множество, исполняются не в каком-то определенном порядке, и потому покрытые отрицательные примеры не отфильтровываются другими правилами. Алгоритм 6.4 описывает обучение одного правила для одного класса и очень похож на алгоритм **LearnRule** (алгоритм 6.2) с двумя различиями: (i) наилучший лiteral выбирается теперь относительно рассматриваемого в данный момент класса C_i ; (ii) заголовок правила всегда содержит метку C_i . В литературе встречается

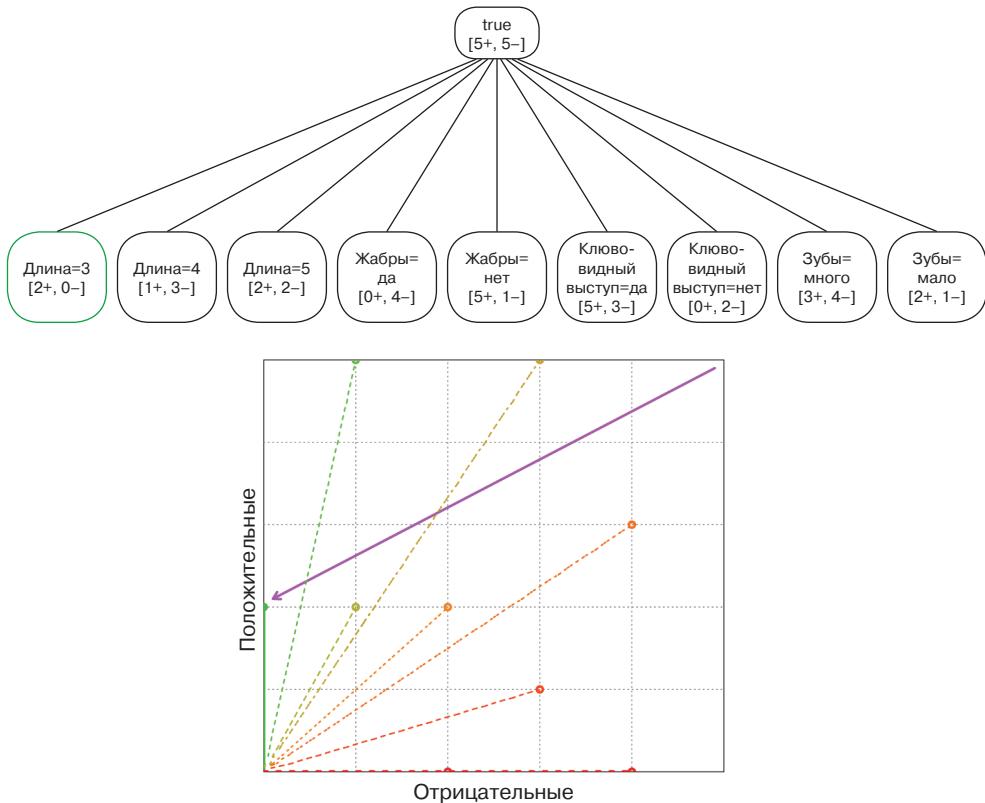


Рис. 6.7. (Сверху) Первое правило обучено для положительного класса. **(Снизу)** Изолинии точности выглядят так же, как изолинии нечистоты (рис. 6.2), однако различие состоит в том, что точность достигает минимума на оси x и максимума на оси y , тогда как чистота – минимума на восходящей диагонали и максимума на осях x и y

интересная вариация: инициализировать множество доступных литералов L , поместив в него те, что встречаются в заданном *начальном примере*, принадлежащем обучаемому классу; преимущество в том, что таким образом сокращается пространство поиска, а потенциальный недостаток в том, что начальный пример может быть выбран не оптимально.

С выбором точности в качестве поисковой эвристики связана одна проблема: она стремится находить чистые правила, а потому иногда пропускает почти чистые, которые можно было бы специализировать и получить более общее чистое правило. Рассмотрим рис. 6.10 сверху: критерий точности отдает предпочтение правилу **if Длина = 3 then Class = \oplus** , хотя почти чистый литерал **Жабры = нет** приводит к чистому правилу **if Жабры = нет \wedge Зубы = много then Class = \oplus** . Удобный способ борьбы с такой «близорукостью» точности дает поправка Лапласа, которая гарантирует, что пара $[5+, 1-]$ будет «скорректирована» в $[6+, 2-]$ и, значит,

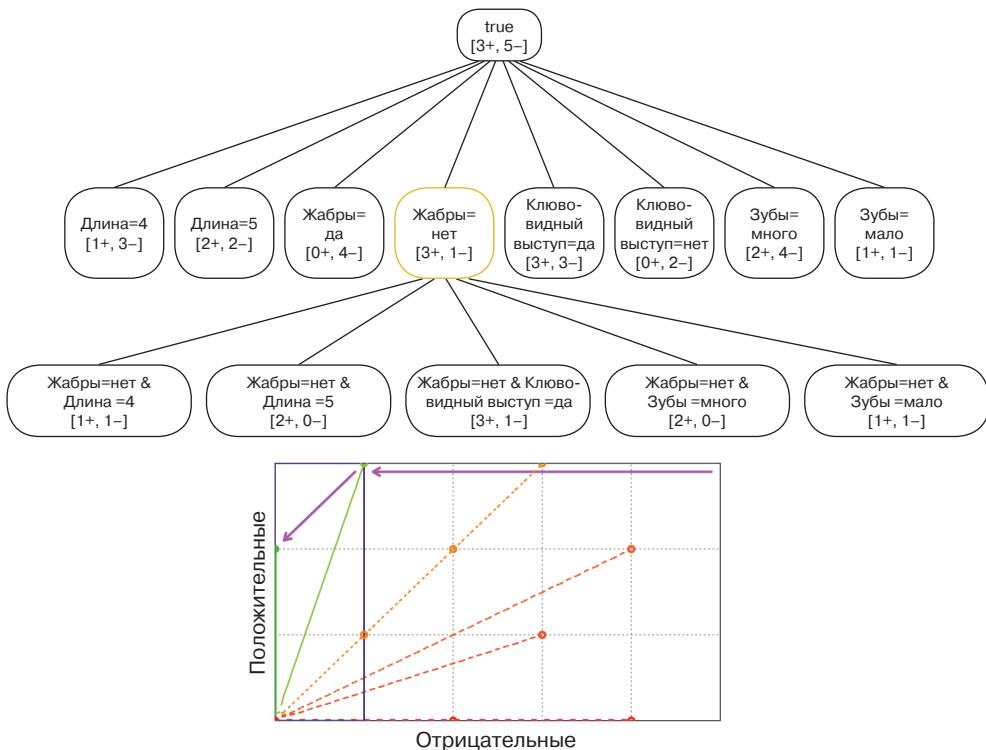


Рис. 6.8. (Сверху) Для второго правила нужны два литерала: при выборе обоих мы используем в качестве критерия максимальную точность. **(Снизу)** Пространство покрытия меньше, потому что два положительных примера, покрытых первым правилом, исключены. Синий прямоугольник слева обозначает еще меньшее пространство покрытия, в котором производится поиск второго литерала, после того как условие **Жабры = нет** отфильтровало четыре отрицательных примера. Внутри синего прямоугольника изолинии точности перекрываются с изолиниями в объемлющем прямоугольнике (так может и не быть, если в качестве поисковой эвристики выбрана не точность)

будет считаться имеющей такое же качество, как пара $[2+, 0-]$, она же $[3+, 1-]$ (рис. 6.10 снизу). Еще один способ уменьшить близорукость и разрешить такие неопределенности состоит в использовании **лучевого поиска**: вместо того чтобы жадно искать лучшего кандидата, мы храним фиксированное количество альтернативных кандидатов. В данном примере даже луч небольшого размера позволил бы найти более общее правило:

- ☞ первый луч включает тела-кандидаты **Длина = 3** и **Жабры = нет**;
- ☞ затем мы добавляем все возможные специализации нечистых элементов луча;
- ☞ из оставшегося множества, включающего элементы исходного луча плюс все добавленные специализации, оставляем только несколько лучших,

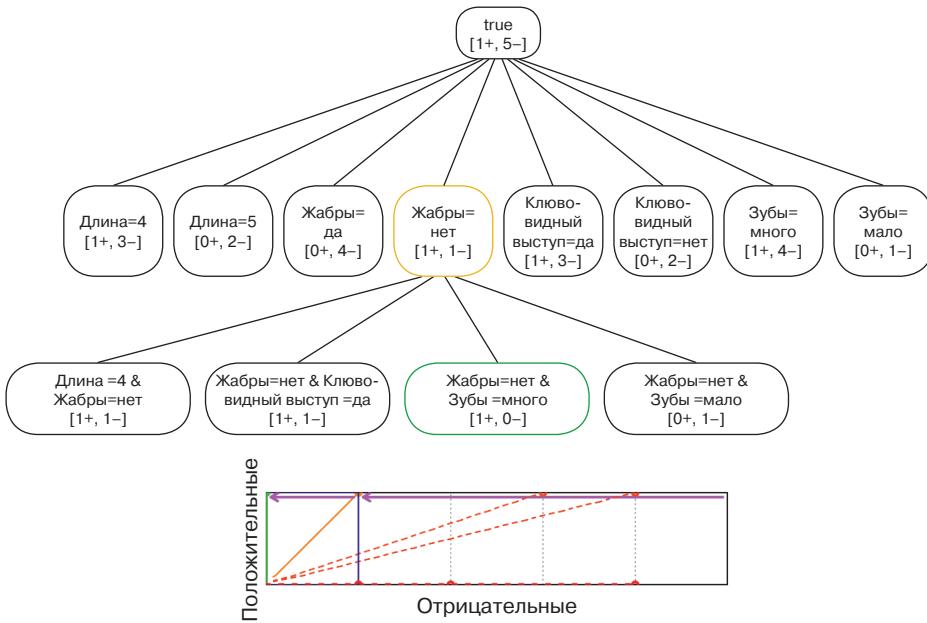


Рис. 6.9. (Сверху) Для третьего и последнего правила снова нужны два литерала. (**Снизу**) Первый литерал исключает четыре отрицательных примера, второй – последний оставшийся отрицательный пример

в случае неопределенности предпочитая те, которые уже были в личе, поскольку они короче;

- ☞ останавливаемся, когда все элементы луча чистые, и выбираем из них наилучший.

Поняв, как обучать множество правил, вернемся к вопросу о том, как использовать модель на основе множества правил в качестве классификатора. Предположим, что предъявляется новый объект, например **Длина = 3 \wedge Жабры = да \wedge Клювовидный выступ = да \wedge Зубы = много**. В случае списка правил, приведенного на стр. 175 сработало бы первое правило, и потому объект был бы классифицирован как отрицательный. В случае же множества правил на стр. 180 срабатывают правила **R1** и **R4**, дающие противоречивые предсказания. Как разрешить данное противоречие? Чтобы ответить на этот вопрос, проще рассмотреть сначала более общий: как использовать множество правил для ранжирования и оценивания вероятностей?

Применение множеств правил для ранжирования и оценивания вероятностей

В общем случае если множество правил состоит из r правил, то они могут перекрываться 2^r разными способами, отсюда и 2^r отрезков в пространстве объектов.

Алгоритм 6.3. $\text{LearnRuleSet}(D)$ – обучить неупорядоченное множество правил

Вход: размеченные обучающие данные D .
Выход: множество правил R .

```

1  $R \leftarrow \emptyset;$ 
2 for каждого класса  $C_i$  do
3    $D_i \leftarrow D;$ 
4   while  $D_i$  содержит примеры из класса  $C_i$  do
5      $r \leftarrow \text{LearnRuleForClass}(D_i, C_i);$            // LearnRuleForClass: см. алгоритм 6.4
6      $R \leftarrow R \cup \{r\};$ 
7      $D_i \leftarrow D_i \setminus \{x \in C_i \mid x \text{ покрыт } r\};$  // исключить только положительные
8   end
9 end
10 return  $R$ 
```

Алгоритм 6.4. $\text{LearnRuleForClass}(D, C_i)$ – обучить одно правило для данного класса

Вход: размеченные обучающие данные D , класс C_i
Выход: правило r .

```

1  $b \leftarrow \text{true};$ 
2  $L \leftarrow$  множество имеющихся литералов;           // можно инициализировать с помощью
                                                       // начального примера
3 while not  $\text{Homogeneous}(D)$  do
4    $l \leftarrow \text{BestLiteral}(D, L, C_i);$            // например, имеющий максимальную
                                                       // точность на классе  $C_i$ 
5    $b \leftarrow b \wedge l;$ 
6    $D \leftarrow \{x \in D \mid x \text{ покрыт } b\};$ 
7    $L \leftarrow L \setminus \{l' \in L \mid l' \text{ использует тот же признак, что } l\};$ 
8 end
9  $r \leftarrow \text{if } b \text{ then Class} = C_i;$ 
10 return  $r$ 
```

И хотя многие из них будут пустыми, потому что правила взаимно исключающие, вообще говоря, отрезков будет больше, чем правил. Следовательно, мы должны оценить покрытие некоторых из этих отрезков.

Пример 6.4 (множества правил как ранжировщики). Рассмотрим следующее множество правил (первые два использовались также в примере 6.2):

- | | |
|--|---|
| (A) if Длина = 4 then Class = \ominus
(B) if Клювовидный выступ = да then Class = \oplus
(C) if Длина = 5 then Class = \ominus | [1+,3-]
[5+,3-]
[2+,2-] |
|--|---|

Числа справа показывают покрытие каждого правила на всем обучающем наборе. Для объектов, покрытых только одним правилом, мы можем использовать эти счетчики покрытия для вычисления оценок вероятностей, например объект, покрытый только правилом A, получит вероятность $\hat{p}(A) = 1/4 = 0.25$ и аналогично $\hat{p}(B) = 5/8 = 0.63$, $\hat{p}(C) = 2/4 = 0.50$.

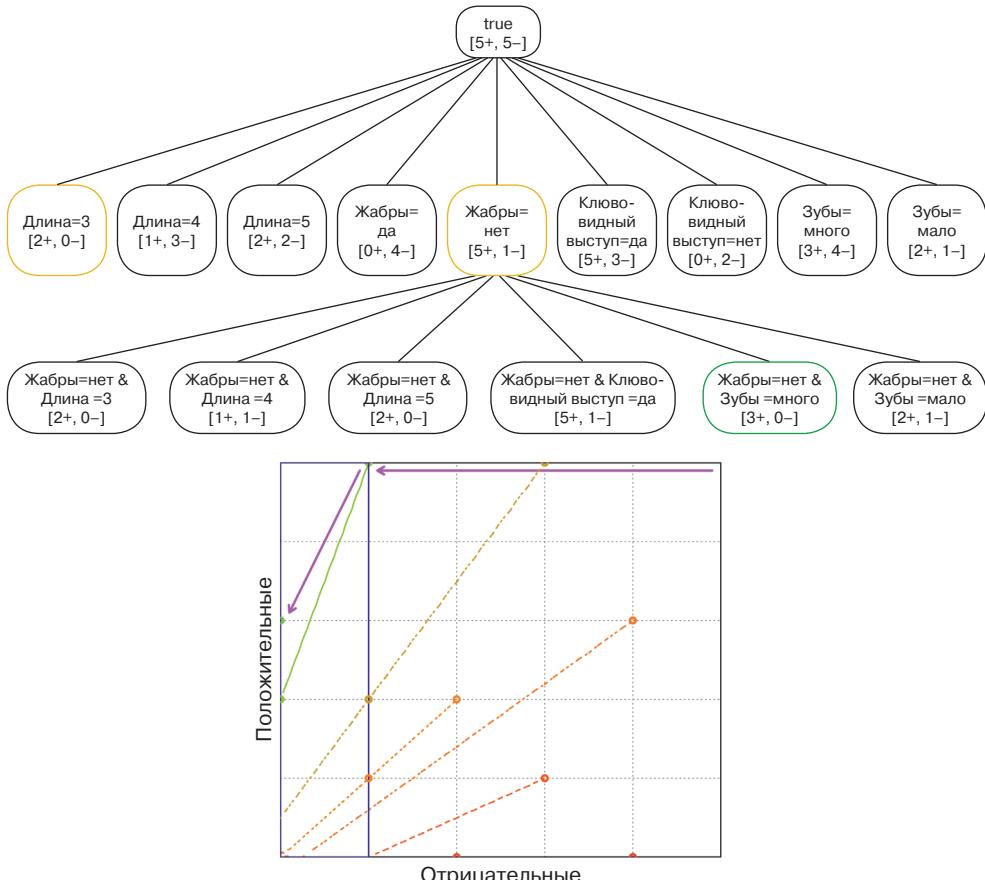


Рис. 6.10. (Сверху) Применение поправки Лапласа к точности позволяет обучить лучшее правило на первой итерации. **(Снизу)** Поправка Лапласа добавляет один положительный и один отрицательный псевдосчетчик, это означает, что изолинии теперь поворачиваются вокруг точки $(-1, -1)$ в пространстве покрытия, и в результате предпочтение отдается более общим правилам

Очевидно, что **A** и **C** взаимно исключающие, поэтому нужно принять во внимание только перекрытия **AB** и **BC**. Часто применяется простой прием – усреднить покрытия участвующих правил, например покрытие **AB** оценивается как $[3+, 3-]$, что дает $\hat{p}(\text{AB}) = 3/6 = 0.50$. Аналогично $\hat{p}(\text{BC}) = 3.5/6 = 0.58$. Соответствующим ранжированием будет **B** – **BC** – **[AB, C]** – **A**, что дает оранжевую кривую покрытия на обучающем наборе на рис. 6.11.

Теперь сравним это множество правил со списком правил **ABC**:

```

if Длина = 4 then Class = ⊖ [1+, 3-]
else if Клювовидный выступ = да then Class = ⊕ [4+, 1-]
else if Длина = 5 then Class = ⊖ [0+, 1-]

```

Кривая покрытия этого списка правил показана на рис. 6.11 синим цветом. Мы видим, что множество правил оказывается лучше списка благодаря тому, что может различить примеры, покрытые только правилом **B**, и примеры, покрытые обоими правилами **B** и **C**.

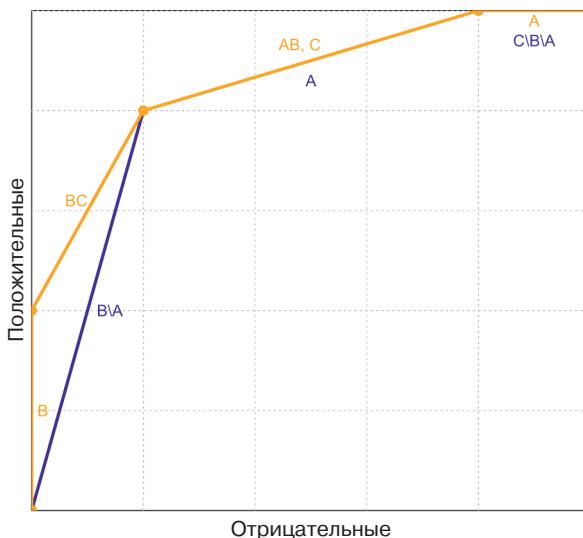


Рис. 6.11. Кривые покрытия множества правил из примера 6.4 (оранжевая) и списка правил ABC (синяя). Множество правил разбивает пространство объектов на меньшие сегменты, что в данном случае приводит к лучшему качеству ранжирования.

В этом примере множество правил оказалось лучше списка правил, но в общем случае это не гарантируется. Вследствие того, что счетчики покрытия некоторых отрезков приходится оценивать, кривая покрытия множества правил неизбежно является выпуклой на обучающем наборе. Предположим, к примеру, что правило **C** также покрывает пример p_1 , это не повлияет на качество списка правил (так как p_1 уже покрыт правилом **B**), но разрешит неопределенность между **AB** и **C** в пользу последнего, из-за чего появится вогнутость.

Чтобы превратить такой ранжировщик в классификатор, мы должны будем найти наилучшую рабочую точку на кривой покрытия. В предположении, что критерием качества является верность, оптимальна точка ($fpr = 0.2$, $tpr = 0.8$), и для ее достижения нужно классифицировать объекты с $p > 0.5$ как положительные, а остальные – как отрицательные. Если такая калибровка порога принятия решения проблематична (например, в случае, когда классов больше двух), то мы можем просто выбрать класс с наибольшим средним покрытием, а возникающие неопределенностям разрешать случайным образом.

Более пристальный взгляд на перекрытие правил

Мы видели, что списки правил всегда дают выпуклые кривые покрытия на обучающем наборе, но для заданного множества правил не существует глобально оптимального упорядочения. Основная причина состоит в том, что списки правил не дают доступа к перекрытию двух правил $A \wedge B$: либо есть доступ к $A = (A \wedge B) \vee (A \wedge \neg B)$, если правила следуют в порядке AB , либо к $B = (A \wedge B) \vee (\neg A \wedge B)$ в случае порядка BA . Более общо, список из r правил порождает только r отрезков в пространстве объектов (или $r + 1$, если добавить еще правило по умолчанию). Это означает, что нам не удастся извлечь преимущество из большинства из 2^r возможных способов перекрытия правил. С другой стороны, множества правил потенциально могут дать доступ к таким перекрытиям, но необходимость в оценке счетчиков покрытия перекрывающихся отрезков означает, что придется пожертвовать выпуклостью. Чтобы лучше понять это, введем в рассмотрение понятие *дерева правил*, это полное дерево признаков, в котором в качестве признаков фигурируют правила.

Пример 6.5 (дерево правил). Из правил в примере 6.4 мы можем построить дерево правил, показанное на рис. 6.12. Использование дерева, а не списка позволяет продолжить разделение отрезков, соответствующих списку правил. Например, узел A разделяется на AB ($A \wedge B$) и $A-$ ($A \wedge \neg B$). Так как последний узел чистый, то кривая покрытия оказывается лучше (красная линия на рис. 6.13).

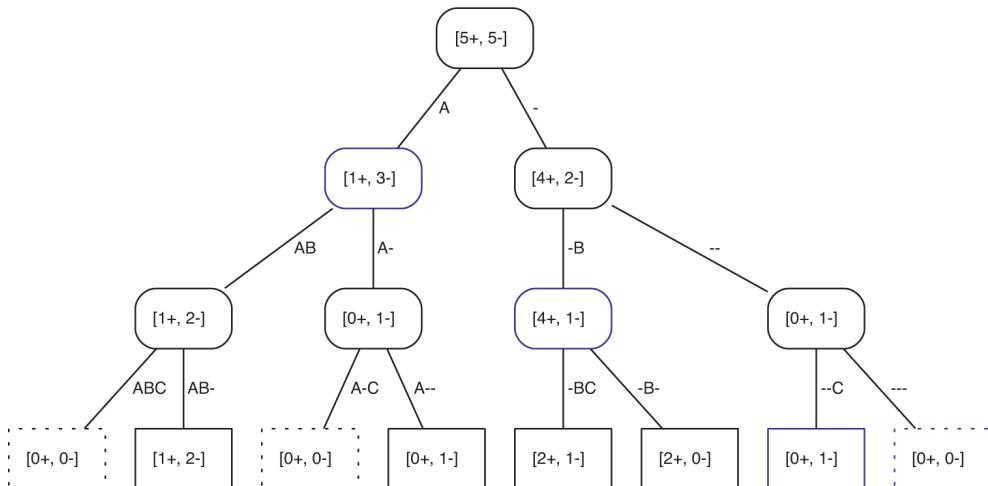


Рис. 6.12. Дерево правил, построенное из правил в примере 6.5. Узлы помечены покрытием (для узлов с пунктирной рамкой покрытие нулевое), а метки на ветвях соответствуют областям в пространстве объектов (например, $A-C$ обозначает $A \wedge \neg B \wedge C$). Синей рамкой обведены отрезки пространства объектов, соответствующие списку правил ABC : дерево правил дает более высокое качество, потому что может разделить их

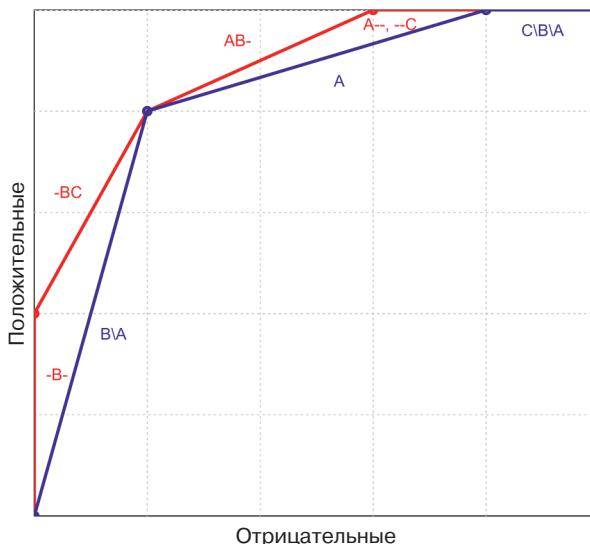


Рис. 6.13. Синяя линия – это кривая покрытия списка правил **ABC** в примере 6.4. Над ней доминирует **красная** кривая покрытия, соответствующая дереву правил на рис. 6.12. Дерево правил также улучшает множество правил (**оранжевая** кривая на рис. 6.11), поскольку имеет доступ к точным счетчикам покрытия во всех отрезках и потому распознает, что **AB-** предшествует **--C**

Как мы видим, в этом примере кривая покрытия, порождаемая деревом правил, доминирует над кривой покрытия списка правил. Это верно и в общем случае: не существует никакой другой информации о перекрытии правил, помимо содержащейся в дереве правил, а любой список правил обычно несет в себе лишь часть этой информации. Обратно, можно задаться вопросом, всякая ли рабочая точка на кривой дерева правил достижима списком правил. Ответ отрицательный, как следует из простого контрпримера на рис. 6.14.

Подводя итоги, можно сказать, что из трех рассмотренных моделей на основе правил только деревья правил в полной мере раскрывают потенциал перекрытия правил, поскольку обладают способностью представить все 2^r областей перекрытия r правил и предоставляют доступ к точным счетчикам покрытия в каждой области. Списки правил также несут в себе точные счетчики покрытия, но для меньшего числа отрезков; множества правил различают те же отрезки, что деревья правил, но нуждаются в оценивании счетчиков покрытия для областей перекрытия. С другой стороны, деревья правил обходятся дорого, потому что их размер экспоненциально зависит от количества правил. Еще один недостаток – тот факт, что для получения счетчиков покрытия нужен отдельный шаг, выполняемый после того, как правила уже обучены. Я включил деревья правил в основном из концептуальных соображений: для лучшего понимания более распространенных списков и множеств правил.

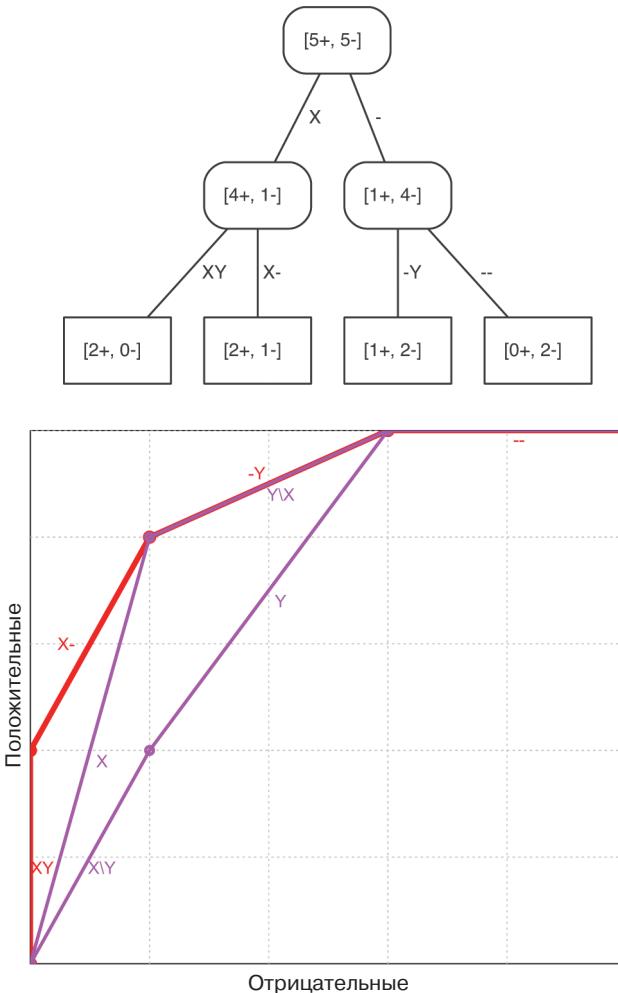


Рис. 6.14. (Сверху) Дерево правил, построенное из двух правил X и Y . **(Снизу)** Кривая покрытия дерева правил строго доминирует над выпуклой оболочкой двух кривых покрытия списков правил. Это означает, что существует рабочая точка $[2+, 0-]$, не достижимая ни одним списком правил

6.3 Обучение дескриптивных моделей на основе правил

Как мы видели, формат правил естественно тяготеет к прогностическим моделям, построенным из правил с целевой переменной в заголовке. Нетрудно обобщить модели на основе правил на задачи регрессии и кластеризации подобно

тому, как мы поступили для древовидных моделей в конце главы 5, но я не стану развивать эту тему. Вместо этого я покажу, что формат правил легко можно применить и к построению дескриптивных моделей. Как было сказано в разделе 1.1, дескриптивные модели можно обучать с учителем и без учителя. В качестве примера обучения с учителем мы обсудим адаптацию приведенных выше алгоритмов обучению правил к выявлению подгрупп. А в качестве примера обучения дескриптивной модели без учителя мы рассмотрим частые предметные наборы и выявление ассоциативных правил.

Обучение правил для выявления подгрупп

При обучении моделей классификации естественно искать правила, которые выявляют чистые подмножества обучающего набора, то есть такие множества примеров, все элементы которых принадлежат одному и тому же классу и удовлетворяют одному конъюнктивному концепту. Однако, как было показано в разделе 3.3, иногда нас интересует не столько предсказание класса, сколько нахождение интересных закономерностей. Мы определили подгруппу как отображение $\hat{g}: \mathcal{X} \rightarrow \{\text{true}, \text{false}\}$ – или, что то же самое, подмножество пространства объектов – обученное на основе набора помеченных примеров $(x_i, l(x_i))$, где $l: \mathcal{X} \rightarrow \mathcal{C}$ – помечающая функция. Хорошой считается подгруппа, для которой распределение по классам существенно отличается от распределения по классам в генеральной совокупности. Это по определению верно для чистых подгрупп, но интерес представляют не только они. Так, можно предположить, что дополнение подгруппы столь же интересно, сколь и сама подгруппа: в нашем примере с дельфинами концепт **Жабры = да**, покрывающий четыре положительных примера и ни одного отрицательного, можно считать таким же интересным, как его дополнение **Жабры = нет**, которое покрывает один отрицательный и все положительные примеры. Это означает, что нужно отказаться от критериев оценки, основанных на нечистоте.

Как и концепты, подгруппы можно изобразить точками в пространстве покрытия, откладывая положительные примеры по оси y , а отрицательные – по оси x . В любой подгруппе, оказавшейся на восходящей диагонали, доля положительных примеров такая же, как в генеральной совокупности, это наименее интересные подгруппы, потому что статистически они не отличаются от случайной выборки. В подгруппах выше (ниже) диагонали доля положительных примеров больше (меньше), чем в генеральной совокупности. Таким образом, один из способов оценить качество подгруппы состоит в том, чтобы взять одну из эвристик, использованных при обучении правил, и измерить абсолютное отклонение от значения по умолчанию на диагонали. Например, точность любой подгруппы на диагонали равна доле положительных примеров, поэтому в качестве одной из мер точности можно взять величину $|prec - pos|$. По причинам, которые уже обсуждались, часто результат можно улучшить, взяв точность с поправкой Лапласа $prec^L$, что дает альтернативную меру $|prec^L - pos|$. Как видно по рис. 6.15 слева, введение псевдосчетчиков означает, что точка $[5+, 1-]$ оценивается как $[6+, 2-]$ и потому так же интересна, как чистый концепт $[2+, 0-]$, который оценивается как $[3+, 1-]$.

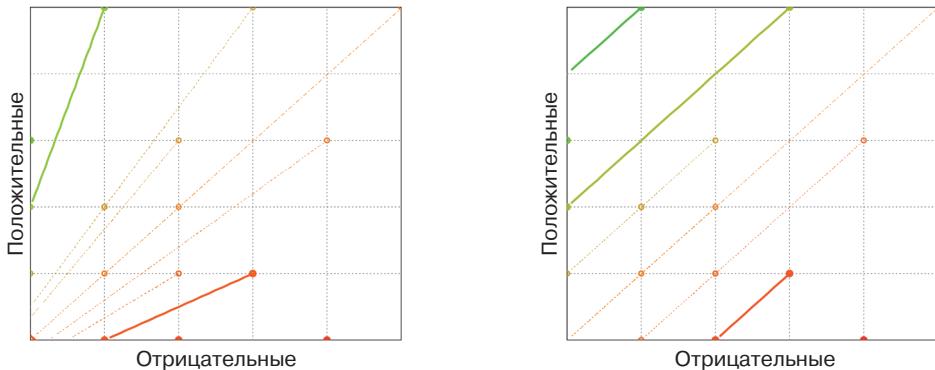


Рис. 6.15. (Слева) Подгруппы и их изолинии согласно точности с поправкой Лапласа. Сплошные, самые внешние изолинии показывают наилучшие подгруппы. **(Справа)** Ранжирование изменяется, если упорядочивать подгруппы по средней полноте. Например, [5+,1–] теперь лучше, чем [3+,0–], и так же хороша, как [0+,4–]

Однако это не ставит дополнительные подгруппы в совсем уж равные условия, поскольку качество [5+,1–] все же считается меньшим, чем качество [0+,4–]. Чтобы достичь такой дополнительности, нам нужен критерий оценки, в котором все изолинии параллельны восходящей диагонали. Но мы уже видели такой критерий в разделе 2.1, где называли его *средней полнотой* (см., например, рис. 2.4 на стр. 75). Отметим, что для групп на диагонали средняя полнота всегда равна 0.5 безотносительно к распределению по классам. Поэтому хорошей мерой является $|avg-rec - 0.5|$. Среднюю полноту можно записать в виде $(1 + tpr - fpr)/2$ и, следовательно, $|avg-rec - 0.5| = |tpr - fpr|/2$. Иногда нежелательно брать абсолютную величину, потому что знак разности говорит, где находится подгруппа: выше или ниже диагонали. С этим критерием оценки тесно связана *взвешенная относительная верность*, которую можно записать в виде $pos \cdot neg(tpr - fpr)$.

Как следует из сравнения двух графиков изолиний на рис. 6.15, использование средней полноты вместо точности с поправкой Лапласа оказывает влияние на ранжирование некоторых подгрупп. Подробный расчет приведен в табл. 6.1.

Пример 6.6 (сравнение точности с поправкой Лапласа и средней полноты). В табл. 6.1 приведены результаты ранжирования десяти подгрупп из примера с дельфинами в терминах точности с поправкой Лапласа и средней полноты. Одно из различий состоит в том, что подгруппа **Жабры = нет \wedge Зубы = много** с покрытием [3+,0–] лучше подгруппы **Жабры = нет** с точки зрения точности с поправкой Лапласа, но хуже с точки зрения средней полноты, поскольку последняя ранжирует эту подгруппу одинаково с ее дополнением **Жабры = да**.

Второе различие между обучением правил классификации и выявлением подгрупп заключается в том, что во втором случае нас, естественно, интересуют перекрывающиеся правила, тогда как стандартный алгоритм построения покрытия

Подгруппы	Покрытие	$prec^L$	Ранг	$avg-rec$	Ранг
Жабры = да	[0+,4-]	0.17	1	0.10	1–2
Жабры = нет \wedge Зубы = много	[3+,0-]	0.80	2	0.80	3
Жабры = нет	[5+,1-]	0.75	3–9	0.90	1–2
Клювовидный выступ = нет	[0+,2-]	0.25	3–9	0.30	4–11
Жабры = да \wedge Клювовидный выступ = да	[0+,2-]	0.25	3–9	0.30	4–11
Длина = 3	[2+,0-]	0.75	3–9	0.70	4–11
Длина = 4 \wedge Жабры = да	[0+,2-]	0.25	3–9	0.30	4–11
Длина = 5 \wedge Жабры = нет	[2+,0-]	0.75	3–9	0.70	4–11
Длина = 5 \wedge Жабры = да	[0+,2-]	0.25	3–9	0.30	4–11
Длина = 4	[1+,3-]	0.33	10	0.30	4–11
Клювовидный выступ = да	[5+,3-]	0.60	11	0.70	4–11

Таблица 6.1. Расчет оценки наилучших подгрупп. Применяя точность с поправкой Лапласа, мы можем оценить качество подгруппы как $|prec^L - pos|$. Вместо этого можно использовать среднюю полноту и определить качество подгруппы как $|avg-rec - 0.5|$. Эти два критерия качества дают несколько отличающиеся ранжирования

не поощряет этого, потому что уже покрытые примеры исключаются из обучающего набора. Один из способов решить эту проблему – назначить примерам веса, которые уменьшаются всякий раз, как пример оказывается покрыт вновь обученным правилом. На практике хорошо работает схема, в которой все веса инициализируются значением 1 и уменьшаются вдвое при каждом покрытии примера новым правилом. Поисковые эвристики затем вычисляются в терминах суммарного веса покрытых примеров, а не просто их количества.

Пример 6.7 (влияние взвешенного покрытия). Предположим, что первой была найдена подгруппа **Длина = 4**, в результате чего веса одного положительного и трех отрицательных примеров, покрытых ей, уменьшились вдвое. Результаты, показывающие, как это отразилось на взвешенном покрытии подгрупп, приведены в табл. 6.2. Мы видим, что пространство покрытия сократилось до обведенного синей рамкой прямоугольника на рис. 6.16. Это повлияло также на взвешенное покрытие подгрупп, перекрывающихся с подгруппой **Длина = 4**, на что указывают стрелки. Некоторые подгруппы оказались ближе к диагонали и потому потеряли важность, например сама подгруппа **Длина = 4**, которая переместилась из точки [3+,1-] в точку [1.5+,0.5-]. Другие, наоборот, отодвинулись от диагонали и, значит, стали важнее, например подгруппа **Длина = 5 \wedge Жабры = да** в точке [0+,2-].

Ниже приведен алгоритм *взвешенного покрытия* (алгоритм 6.5). Отметим, что этот алгоритм можно применить к выявлению подгрупп над $k > 2$ классами при условии, что критерий оценки, использованный для обучения одиночных правил, способен работать, когда число классов больше двух. Очевидно, это так в случае, когда такой мерой является средняя полнота. Есть и другие меры, в частности на основе критерия χ^2 и взаимной информации.

Подгруппы	Покрытие	<i>avg-rec</i>	Взвеш. покрытие	<i>W-avg-rec</i>	Ранг
Жабры = да	[0+,4-]	0.10	[0+, 3-]	0.07	1–2
Жабры = нет	[5+,1-]	0.90	[4.5+ , 0.5-]	0.93	1–2
Жабры = нет \wedge Зубы = много	[3+,0-]	0.80	[2.5+ ,0-]	0.78	3
Длина = 5 \wedge Жабры = да	[0+,2-]	0.30	[0+,2-]	0.31	4
Длина = 3	[2+,0-]	0.70	[2+,0-]	0.72	5–6
Длина = 5 \wedge Жабры = нет	[2+,0-]	0.70	[2+,0-]	0.72	5–6
Клювовидный выступ = нет	[0+,2-]	0.30	[0+, 1.5-]	0.29	7–9
Жабры = да \wedge Клювовидный выступ = да	[0+,2-]	0.30	[0+, 1.5-]	0.29	7–9
Клювовидный выступ = да	[5+,3-]	0.70	[4.5+ , 2-]	0.71	7–9
Длина = 4	[1+,3-]	0.30	[0.5+ , 1.5-]	0.34	10
Длина = 4 \wedge Жабры = да	[0+,2-]	0.30	[0+, 1-]	0.36	11

Таблица 6.2. В столбце «Взвеш. покрытие» показано, как изменяется взвешенное покрытие подгрупп, если наполовину уменьшить веса примеров, покрытых концептом **Длина = 4**. В столбце «W-avg-ges» показано, как на значения avg-ges, представленные в табл. 6.1, влияет взвешивание, что ведет к дифференциации групп, ранее считавшихся эквивалентными

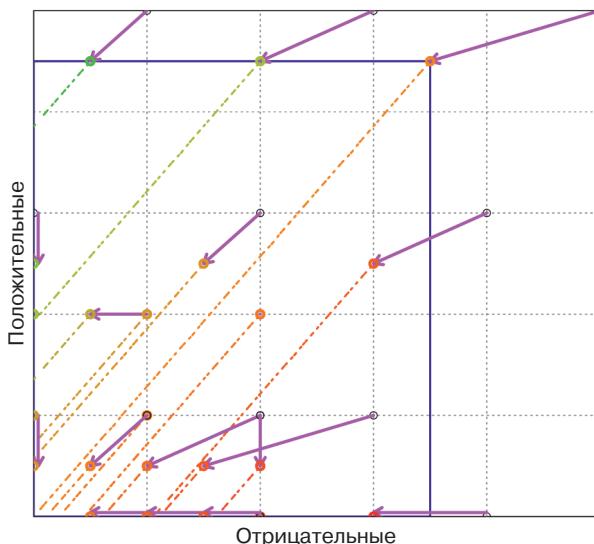


Рис. 6.16. Наглядное представление влияния взвешенного покрытия. Если первой была найдена подгруппа **Длина = 4**, то наполовину уменьшается вес одного положительного и трех отрицательных примеров, вследствие чего пространство покрытия сократилось до обведенного синей рамкой прямоугольника. Стрелки показывают, как это отразилось на взвешенном покрытии других подгрупп, в зависимости от того, какие примеры с уменьшенным весом они покрывают

Алгоритм 6.5. WeightedCovering(D) – обучить перекрывающиеся правила путем взвешивания примеров

Вход: размеченные обучающие данные D , в которых веса примеров инициализированы значением 1.

Выход: список правил R .

```

1  $R \leftarrow \emptyset;$ 
2 while в  $D$  есть примеры с весом 1 do                                // LearnRule: см. алгоритм 6.2
3    $r \leftarrow \text{LearnRule}(D);$ 
4   добавить  $r$  в конец  $R;$ 
5   уменьшить веса примеров, покрытых  $r;$ 
6 end
7 return  $R$ 
```

Добыча ассоциативных правил

Теперь я познакомлю вас с новым видом правил, которые можно обучить вообще без учителя и которые играют важную роль в приложениях для добычи данных. Допустим, что мы наблюдали за восьмью покупателями, приобретшими яблоки, пиво, чипсы и подгузники.

Сделка	Предметы
1	подгузники
2	пиво, чипсы
3	яблоки, подгузники
4	пиво, чипсы, подгузники
5	яблоки
6	яблоки, пиво, чипсы, подгузники
7	яблоки, чипсы
8	чипсы

Каждая *сделка* в этой таблице содержит набор *предметов*; обратно, для каждого предмета мы можем перечислить сделки, в которых он участвует: сделки 1, 3, 4 и 6 для подгузников, 3, 5, 6 и 7 – для яблок и т. д. Это можно сделать и для наборов предметов; например, пиво и чипсы покупали вместе в сделках 2, 4 и 6. Мы говорим, что набор $\{\text{beer}, \text{crisps}\}$ покрывает множество сделок $\{2, 4, 6\}$. Всего существует 16 таких предметных наборов (включая пустой, который покрывает все сделки). Если в качестве отношения частичного порядка между ними взять отношение «является подмножеством», то они образуют решетку (рис. 6.17).

Назовем количество сделок, покрытых предметным набором I , его *опорой* и будем обозначать $\text{Supp}(I)$ (иногда эту величину называют частотой). Нас интересуют *частые предметные наборы*, опора которых превосходит заданный порог f_0 . Опора – монотонная функция: при спуске вдоль пути в решетке предметных наборов она никогда не увеличивается. Это означает, что множество частых предметных наборов *выступило* и полностью определяется своей нижней границей,

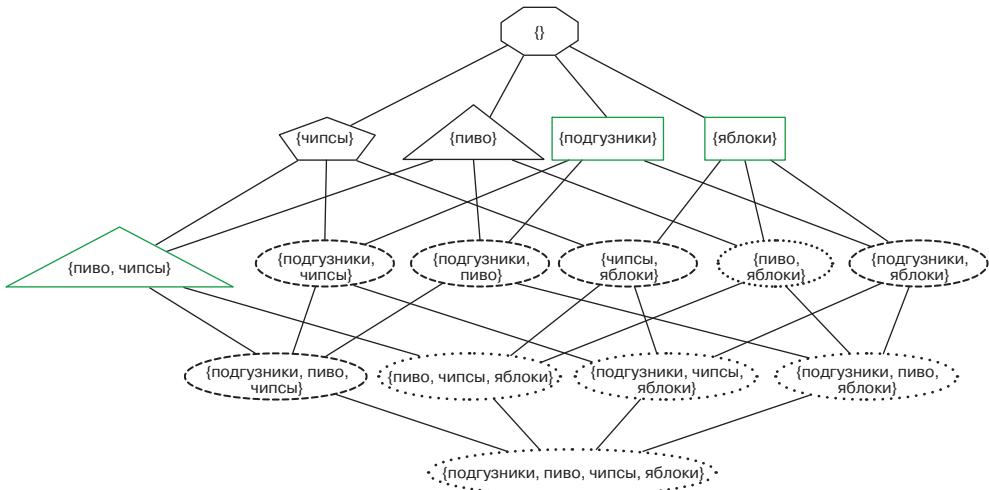


Рис. 6.17. Решетка предметных наборов. Наборы в овалах с пунктирной границей встречаются в одной сделке, со штриховой границей – в двух сделках, в треугольниках – в трех сделках, в n -угольниках – в n сделках. Максимальные предметные наборы с опорой 3 и больше обозначены фигурами с зеленой границей

состоящей из самых крупных предметных наборов. В нашем примере максимальными¹ частыми предметными наборами для $f_0 = 3$ являются **{яблоки}**, **{пиво, чипсы}** и **{подгузники}**. Таким образом, яблоки, подгузники и пиво вместе с чипсами участвуют по меньшей мере в трех сделках, любое другое сочетание предметов покупали реже.

Благодаря свойству монотонности опоры частые предметные наборы можно найти простым перебором с помощью алгоритма поиска в ширину – поурожневого обхода дерева (алгоритм 6.6). В этом алгоритме ведется очередь с приоритетом, которая в начальный момент содержит только пустой предметный набор, покрывающий все сделки. Извлекая очередной предметный набор-кандидат I из очереди, алгоритм генерирует все возможные расширения (объемлющие наборы, содержащие на один предмет больше, соседствующие с данным и находящиеся на следующем по глубине уровне решетки) и добавляет их в очередь, если их опора превышает порог (в конец, чтобы поиск производился именно в ширину). Если хотя бы одно расширение набора I является частым, то I не максимальен и может быть отброшен, в противном случае I добавляется во множество максимальных частых предметных наборов.

Вычисления можно ускорить, сосредоточившись только на **замкнутых предметных наборах**. Это полный аналог замкнутых концептов, обсуждавшихся в конце раздела 4.2: замкнутый предметный набор содержит все предметы, участ-

¹ «Максимальный» здесь означает, что не никакой включающий набор не является частым.

Алгоритм 6.6. $\text{FrequentItems}(D, f_0)$ – найти все максимальные предметные наборы, опора которых превосходит порог

Вход: данные $D \subseteq \mathcal{X}$; пороговая опора f_0

Выход: множество максимальных частых предметных наборов M .

```

1  $M \leftarrow \emptyset$ ;
2 инициализировать очередь с приоритетом  $Q$ , включив в нее пустой предметный набор;
3 while  $Q$  не пуста do
4    $I \leftarrow$  предметный набор, находящийся в начале очереди  $Q$ ;
5    $max \leftarrow \text{true}$ ; // флаг, показывающий, что  $I$  максимальен
6   for каждого возможного расширения  $I'$  набора  $I$  do
7     if  $\text{Supp}(I') \geq f_0$  then
8        $max \leftarrow \text{false}$ ; // найдено частое расширение, поэтому  $I$  не максимальен
9       поместить  $I'$  в конец  $Q$ ;
10      end
11    end
12    if  $max = \text{true}$  then  $M \leftarrow M \cup \{I\}$ ;
13  end
14 return  $M$ 

```

вующие в каждой сделке, которую он покрывает. Например, набор **{beer, crisps}** покрывает сделки 2, 4 и 6; единственные предметы, участвующие в каждой из этих сделок, – пиво и чипсы, поэтому данный предметный набор замкнут. С другой стороны, набор **{пиво}** не замкнут, так как он покрывает те же самые сделки, и, следовательно, его замыкание равно **{пиво, чипсы}**. Если два предметных набора, соединенных в решетке, имеют одинаковое покрытие, то меньший набор не может быть замкнут. Решетка, состоящая из замкнутых предметных наборов, показана на рис. 6.18. Отметим, что максимальные частые предметные наборы обязательно замкнуты (поскольку при их расширении покрытие становится меньше пороговой опоры, иначе набор не был бы максимальным). Поэтому описанное ограничение их не затрагивает, но эффективность поиска оно повышает.

Так в чем же смысл частых предметных наборов? В том, что они используются для построения *ассоциативных правил*, то есть правил вида **if B then H** , где тело B и заголовок H – предметные наборы, которые часто встречаются в сделках вместе. Возьмем любое ребро графа на рис. 6.17, скажем, соединяющее наборы **{пиво}** и **{подгузники, пиво}**. Мы знаем, что опора первого равна 3, а второго – 2, то есть пиво участвует в трех сделках, из которых в двух участвуют также подгузники. Мы говорим, что *уровень доверия* ассоциативного правила **if $пиво$ then $подгузники$** равен $2/3$. Аналогично наличие ребра между наборами **{подгузники}** и **{подгузники, пиво}** означает, что уровень доверия правила **if $подгузники$ then $пиво$** равен $2/4$. Существуют также правила с уровнем доверия 1, например **if $пиво$ then $чицы$** , и правила с пустыми телами, например **if true then $чицы$** с уровнем доверия $5/8$ (то есть чипсы участвовали в пяти из восьми сделок).

Но мы хотим строить только ассоциативные правила, в которых участвуют частные предметы. Уровень доверия правила **if** пиво \wedge яблоки **then** чипсы равен 1,

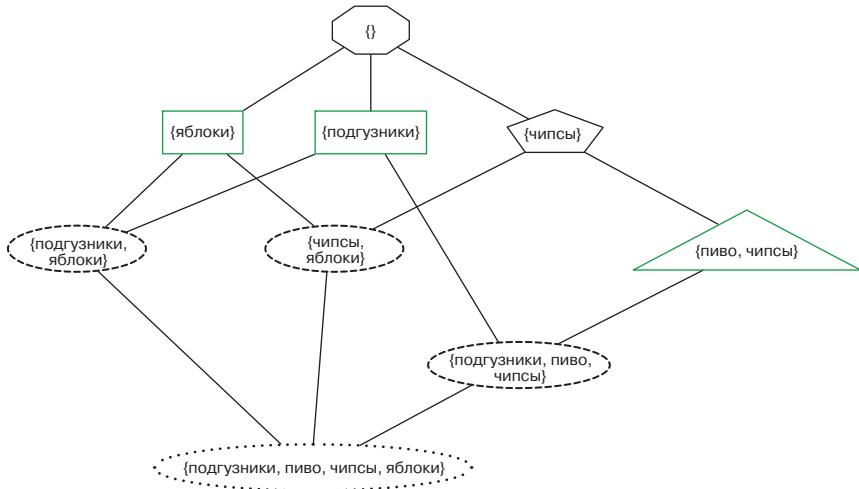


Рис. 6.18. Решетка замкнутых предметных наборов, соответствующая предметным наборам на рис. 6.17. Эта решетка обладает тем свойством, что ни у каких двух смежных предметных наборов нет одинакового покрытия

но существует всего одна сделка, в которой участвуют все три предмета, так что у этого правила нет основательной опоры в данных. Поэтому мы сначала применяем алгоритм 6.6, чтобы найти частые предметные наборы, а затем выбираем тела B и заголовки H из частых наборов m , отбрасывая правила с уровнем доверия меньше порогового. В алгоритме 6.7 описан базовый алгоритм. Отметим, что мы вправе отбрасывать некоторые предметы, входящие в максимальные частые наборы (то есть $H \cup B$ может быть меньше m), потому что любой поднабор частого предметного набора также является частым.

Прогон этого алгоритма с пороговой опорой 3 и порогом уровня доверия 0.6 порождает следующие ассоциативные правила:

if пиво then чипсы	опора 3, уровень доверия 3/3;
if чипсы then пиво	опора 3, уровень доверия 3/5;
if true then чипсы	опора 5, уровень доверия 5/8.

Алгоритм добычи ассоциативных правил часто включает также этап *постобработки*, на котором отфильтровываются лишние правила, например частные случаи, уровень доверия которых не выше, чем у общего случая. На этапе постобработки нередко используется понятие *подъема*, определяемое следующим образом:

$$\text{Lift}(\cdot \text{if } B \text{ then } H \cdot) = \frac{n \cdot \text{Supp}(B \cup H)}{\text{Supp}(B) \cdot \text{Supp}(H)},$$

где n – количество сделок. Например, для первых двух найденных выше ассоциативных правил подъем оказывается равным $(8 \cdot 3) / (3 \cdot 5) = 1/6$, потому что

Алгоритм 6.7. *AssociationRules(D, f_0, c_0)* – найти все ассоциативные правила, для которых опора и уровень доверия превышают заданные пороговые значения

Вход: данные $D \subseteq \mathcal{X}$; пороговая опора f_0 , порог уровня доверия c_0 .

Выход: множество ассоциативных правил R .

```

1  $R \leftarrow \emptyset;$ 
2  $M \leftarrow \text{FrequentItems}(D, f_0);$  // FrequentItems: см. алгоритм 6.6
3 for каждого  $m \in M$  do
4   for каждого  $H \subseteq m$  и  $B \subseteq m$  таких, что  $H \cap B = \emptyset$  do
5     if  $\text{Supp}(B \cup H)/\text{Supp}(B) \geq c_0$  then  $R \leftarrow R \cup \{ \text{if } B \text{ then } H \}$ 
6   end
7 end
8 return  $R$ 
```

$\text{Lift}(\text{if } B \text{ then } H) = \text{Lift}(\text{if } H \text{ then } B)$. Для третьего правила имеем $\text{Lift}(\text{if true then чипсы}) = (8 \cdot 5)/(8 \cdot 5) = 1$. Это справедливо для любого правила, в котором $B = \emptyset$, потому что

$$\text{Lift}(\text{if } \emptyset \text{ then } H) = \frac{n \cdot \text{Supp}(\emptyset \cup H)}{\text{Supp}(\emptyset) \cdot \text{Supp}(H)} = \frac{n \cdot \text{Supp}(H)}{n \cdot \text{Supp}(H)} = 1.$$

В общем случае подъем 1 означает, что $\text{Supp}(B \cup H)$ полностью определяется *маргинальными* частотами $\text{Supp}(B)$ и $\text{Supp}(H)$, а не является результатом значимого взаимодействия между B и H . Интерес представляют только ассоциативные правила с подъемом больше 1.

Такие величины, как уровень доверия и подъем, можно интерпретировать и в вероятностном контексте. Пусть $\text{Supp}(I)/n$ – оценка вероятности $p(I)$ того, что в сделке участвуют все предметы из набора I ; тогда уровень доверия оценивается условной вероятностью $p(H|B)$. В контексте классификации, где H обозначает фактический, а B – предсказанный класс, мы назвали бы эту величину точностью (см. табл. 2.3 на стр. 71), и в этой главе мы уже использовали ее в качестве поисковой эвристики при обучении правил. В таком случае подъем – это мера того, что события «случайно выбранная сделка включает все предметы из B » и «случайно выбранная сделка включает все предметы из H » статистически независимы.

Стоит отметить, что заголовки ассоциативных правил могут содержать несколько предметов. Например, предположим, что нас интересует правило **if подгузники then пиво** с опорой 2 и уровнем доверия 2/4. Однако **{подгузники, пиво}** – незамкнутый предметный набор, его замыкание равно **{подгузники, пиво, чипсы}**. Поэтому правило **if подгузники then пиво** на самом деле является частным случаем правила **if подгузники then пиво \wedge чипсы**, которое имеет такие же опору и уровень доверия, но включает только замкнутые предметные наборы.

Мы можем применить анализ частых предметных наборов и к набору данных в примере с дельфинами, если будем рассматривать каждый литерал вида **Признак = Значение** как предмет, помня о том, что различные значения одного

признака являются взаимно исключающими. Тогда предметные наборы соответствуют концептам, сделки – объектам, а расширение концепта – это в точности множество сделок, покрытых предметным набором. Поэтому решетка предметных наборов – то же самое, что раньше мы называли пространством гипотез, с той оговоркой, что в данном случае отрицательные примеры не рассматриваются (рис. 6.19). Сведение к замкнутым концептам (предметным наборам) показано на рис. 6.20. Например, мы видим, что правило

if Жабры = нет \wedge Клювовидный выступ = да then Зубы = много

имеет опору 3 и уровень доверия 3/5 (проверьте, имеет ли оно подъем!).

6.4 Обучение правил первого порядка

В разделе 4.3 мы вскользь упомянули об использовании логики первого порядка в качестве языка концептов. Основное отличие заключается в том, что литералами теперь могут быть не только простые пары признак–значение, но и гораздо более сложные структуры. Все подходы к обучению правил, рассмотренные в этой главе, в литературе были обобщены на обучение правил, выраженных в логике первого порядка. В этом разделе мы вкратце познакомимся с тем, как это работает.

Многие подходы к обучению логике первого порядка основаны на языке логического программирования **Пролог** (Prolog), а обучение правил первого порядка часто называют *индуктивным логическим программированием* (ILP). Логические правила в Прологе являются *дизъюнктами Хорна* с единственным литералом в заголовке – с дизъюнктами Хорна мы уже сталкивались в разделе 4.3. Нотация в Прологе несколько отличается от нотации логики первого порядка. Так, вместо

$\forall x : \text{BodyPart}(x, \text{PairOf}(\text{Gill})) \rightarrow \text{Fish}(x)$

мы пишем

`fish(X) :- bodyPart(X, pairOf(gills)).`

Основные отличия таковы:

- ☞ правила записываются «задом наперед» в виде «заголовок-if-тело»;
- ☞ имена переменных начинаются с заглавной буквы, имена констант, предикатов и функциональных символов (в Прологе они называются *функциями*) – со строчной буквы;
- ☞ неявно предполагается, что переменной предшествует квантор всеобщности.

Что касается третьего пункта, следует подчеркнуть различие между переменными, встречающимися одновременно в заголовке и теле, и переменными, встречающимися только в теле. Рассмотрим такое предложение Пролога:

`someAnimal(X) :- bodyPart(X, pairOf(Y)).`

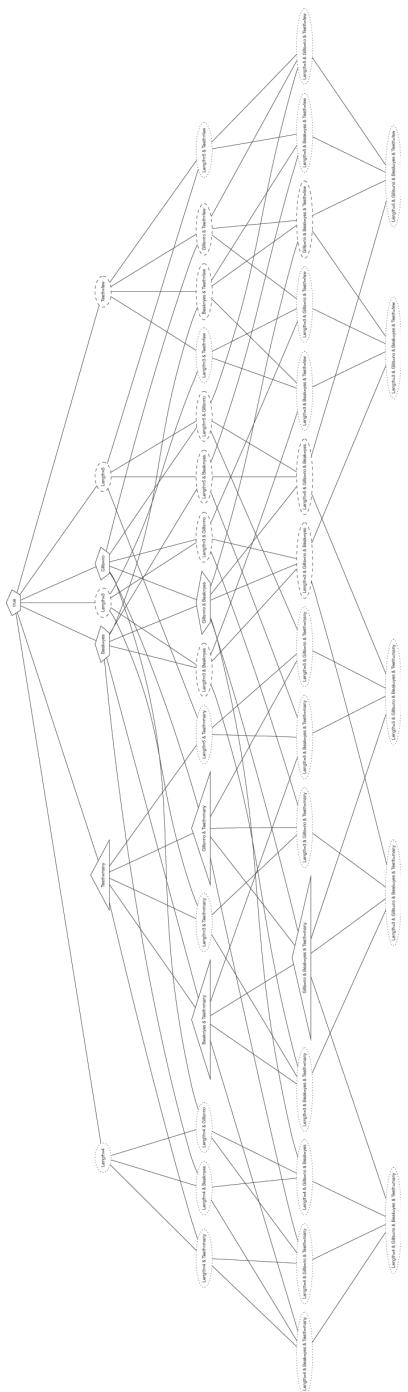


Рис. 6.19. Решетка предметных наборов, соответствующая положительному примером в задаче о дельфинах (см. пример 4.4 на стр. 128). Каждый «предмет» – это литерал вида **Признак = Значение**; каждый признак может встречаться в предметном наборе не более одного раза. Получающаяся структура в точности совпадает с той, что мы называли пространством гипотез в главе 4

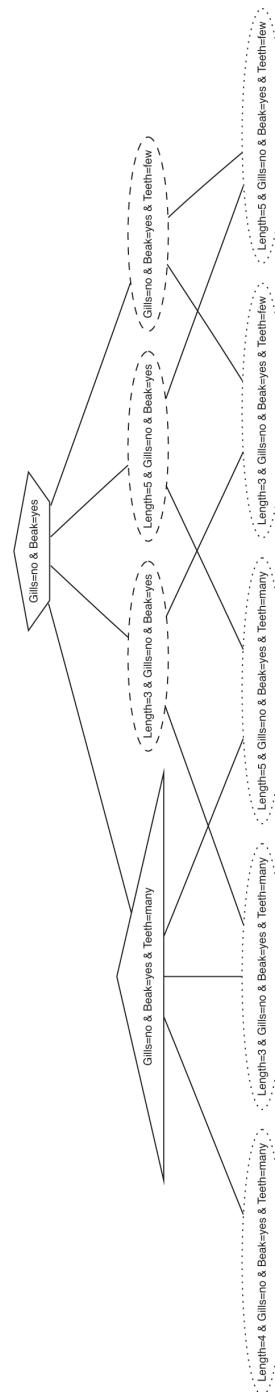


Рис. 6.20. Решетка замкнутых предметных наборов, соответствующая предметным наборам на рис. 6.19

Существуют два эквивалентных способа записать его в виде правила логики первого порядка:

$$\begin{aligned} \forall x : \forall y : \text{BodyPart}(x, \text{PairOf}(y)) &\rightarrow \text{SomeAnimal}(x); \\ \forall x : (\exists y : \text{BodyPart}(x, \text{PairOf}(y))) &\rightarrow \text{SomeAnimal}(x). \end{aligned}$$

Первое логическое предложение читается «для любых x и y , если x имеет пару y в качестве частей тела, то x есть какое-то животное», а второе – «для любого x , если существует y такое, что x имеет пару y в качестве частей тела, то x есть какое-то животное». Важно, что во второй форме областью видимости квантора существования является часть if правила, тогда как кванторы всеобщности всегда распространяются на все предложение. Переменные, встречающиеся в теле, но не в заголовке предложения [Пролога](#), называются *локальными переменными*; они являются источником дополнительной сложности обучения правил первого порядка, по сравнению с пропозициональными правилами.

Если мы хотим обучить упорядоченный список предложений [Пролога](#), то можем воспользоваться алгоритмом [LearnRuleList](#) (алгоритм 6.1) во всей его полноте и большей частью алгоритма [LearnRule](#) (алгоритм 6.2). В корректировке нуждается выбор литерала, добавляемого в предложение. Возможные литералы можно перебрать, перечислив предикаты, функторы и константы, которые можно использовать для построения нового литерала. Например, если имеются бинарный предикат `bodyPart`, унарный функтор `pairOf` и константы `gill` и `tail`, то можно построить различные литералы вида:

```
bodyPart(X,Y)
bodyPart(X,gill)
bodyPart(X,tail)
bodyPart(X,pairOf(Y))
bodyPart(X,pairOf(gill))
bodyPart(X,pairOf(tail))
bodyPart(X,pairOf(pairOf(Y)))
bodyPart(X,pairOf(pairOf(gill)))
bodyPart(X,pairOf(pairOf(tail)))
```

и т. д. Отметим, что наличие функторов означает, что язык гипотез становится бесконечным! Кроме того, я выписал только литералы, которые «имеют смысл», существует и много менее осмысленных возможностей, например `bodyPart(pairOf(gill),tail)` или `bodyPart(X,X)`. Хотя [Пролог](#) – нетипизированный язык, многие из таких нежелательных литералов можно исключить, добавив информацию о типе (в логическом программировании и ILP это часто делается с помощью « объявлений режима », которые также описывают, какие аргументы предиката являются входными, а какие – выходными).

Из этих примеров ясно, что существуют отношения между литералами, а значит, и между содержащими их предложениями. Рассмотрим, к примеру, следующие три предложения:

```
fish(X):-bodyPart(X,Y).
fish(X):-bodyPart(X,pairOf(Z)).
fish(X):-bodyPart(X,pairOf(gill)).
```

Первое говорит, что всё, имеющее какую-то часть тела, является рыбой. Второй специализирует это правило, говоря, что должна быть пара неконкретизированных частей тела. Третье специализирует еще больше, говоря, что рыбой является всё, имеющее пару жабр. Разумная стратегия поиска состоит в том, чтобы проверять гипотезы в этом порядке и переходить к частной лишь тогда, когда более общая исключена отрицательными примерами. Именно так устроены ILP-системы, работающие сверху вниз. С помощью простого приема – явного представления подстановок термов вместо переменных путем добавления литералов «равно» – показанную выше последовательность предложений можно переписать в виде:

```
fish(X):-bodyPart(X,Y).
fish(X):-bodyPart(X,Y),Y=pairOf(Z).
fish(X):-bodyPart(X,Y),Y=pairOf(Z),Z=gill.
```

В качестве альтернативы перебору литералов, рассматриваемых как кандидаты на включение в тело предложения, мы можем вывести их из данных, действуя снизу вверх. Допустим, что имеется следующая информация о дельфине:

```
bodyPart(dolphin42,tail).
bodyPart(dolphin42,pairOf(gills)).
bodyPart(dolphin42,pairOf(eyes)).
```

и о тунце:

```
bodyPart(tuna123,pairOf(gills)).
```

Образуя наименьшее обобщение каждого из литералов в первом примере с литералом из второго примера, мы получаем все обобщенные литералы, рассмотренные выше.

В этом кратком обсуждении обучения правил в логике первого порядка мы оставили за скобками много важных деталей, и потому взгляд на проблему оказался чрезмерно упрощенным. Хотя задачу обучения предложений **Пролога** можно сформулировать очень кратко, наивные подходы к ее решению вычислительно нереализуемы, а «дьявол кроется в деталях». Основные описанные выше подходы можно дополнить, включив фоновые знания, влияющие на упорядочение пространства гипотез по общности. Например, если в состав фоновых знаний входит предложение

```
bodyPart(X,scales):-bodyPart(X,pairOf(gill)).
```

то первая из следующих гипотез оказывается более общей, чем вторая:

```
fish(X):-bodyPart(X,scales).
fish(X):-bodyPart(X,pairOf(gill)).
```

Однако это невозможно определить чисто синтаксическими средствами, требуется логический вывод.

Еще одна интригующая возможность, предлагаемая логикой первого порядка, – обучение рекурсивных предложений. Например, частью нашей гипотезы могло бы быть такое предложение:

```
fish(X) :- relatedSpecies(X, Y), fish(Y).
```

Это размывает различие между фоновыми предикатами, которые можно использовать в теле гипотезы, и целевыми предикатами, которые требуется обучить. Кроме того, появляются вычислительные проблемы, например проблема незавершения. Но это не означает, что такое вообще нельзя сделать. К этой тематике примыкают также методы одновременного обучения нескольких взаимосвязанных предикатов и порождение новых фоновых предикатов, которые никогда не наблюдались.

6.5 Модели на основе правил: итоги и литература для дальнейшего чтения

В решающем дереве путь от корня до листа можно интерпретировать как конъюнктивное правило классификации. Модели на основе правил обобщают это наблюдение, предлагая большую гибкость объединения нескольких правил в модель. Типичным алгоритмом обучения правил является алгоритм построения покрытия, который итеративно обучает по одному правилу, а затем исключает из рассмотрения покрытые этим правилом примеры. Этот подход впервые был предложен в работе Michalski (1975) в составе его системы AQ, которая затем интенсивно разрабатывалась на протяжении более тридцати лет (Wojtusiak et al., 2006). Обзоры состояния дел приведены в работах Furnkranz (1999, 2010) и Furnkranz, Gamberger, Lavrač (2012). Графики покрытия впервые были использованы в работе Furnkranz, Flach (2005) с целью лучше понять алгоритмы обучения правил и продемонстрировать тесную связь (а во многих случаях и эквивалентность) распространенных поисковых эвристик.

- ☞ Правила могут перекрываться, поэтому необходима стратегия разрешения потенциальных конфликтов между правилами. Одна такая стратегия заключается в построении упорядоченного списка правил, о чем шла речь в разделе 6.1. В работе Rivest (1987) этот подход сравнивается с решающими деревьями, а модель на основе правил называется решающим списком (я предпочитаю термин «список правил», поскольку в нем отсутствует неявное предположение, будто элементами списка являются одиночные литералы). Из хорошо известных систем обучения правил отметим CN2 (Clark, Niblett, 1989) и Ripper (Cohen, 1995), причем последняя особенно эффективно предотвращает переобучение за счет инкрементной редукции

с уменьшением ошибки (Furnkranz, Widmer, 1994). Отметим также систему *Opus* (Webb, 1995), отличительной особенностью которой служит выполнение полного поиска в пространстве всех возможных правил.

- ☞ В разделе 6.2 мы рассмотрели неупорядоченные множества правил как альтернативу упорядоченным спискам правил. Алгоритм построения покрытия был модифицирован с целью обучения правил для одного класса за раз и исключения только тех покрытых примеров, которые относятся к рассматриваемому классу. Система **CN2** может работать в неупорядоченном режиме для обучения множеств правил (Clark, 1991). Концептуально и списки правил, и множества правил – частные случаи деревьев правил, которые способны различить все возможные булевы комбинации заданного набора правил. Это замечание позволяет понять, что списки правил приводят к меньшему количеству отрезков в пространстве объектов, чем множества правил (над набором правил). С другой стороны, кривые покрытия списков правил можно сделать выпуклыми над обучающим набором, тогда как в случае множеств правил необходимо оценивать распределение по классам в областях, где правила перекрываются.
- ☞ Модели на основе правил можно использовать для решения дескриптивных задач, и в разделе 6.3 мы рассмотрели применение обучения правил к выявлению подгрупп. Алгоритм взвешенного покрытия был сформулирован как адаптация системы **CN2** в работе Lavrač, Kavšek, Flach, Todorovski (2004); в работе Abudawood, Flach (2009) он был обобщен на число классов больше двух. Алгоритм 6.7 обучает ассоциативные правила и является модификацией хорошо известного алгоритма *Apriori*, описанного в работе Agrawal, Mannila, Srikant, Toivonen, Verkamo (1996). Существует широчайший выбор альтернативных алгоритмов, их обзор приведен в работе Han et al. (2007). Ассоциативные правила можно использовать и для построения эффективных классификаторов (Liu et al., 1998; Li et al., 2001).
- ☞ Тема обучения правил первого порядка, кратко рассмотренная в разделе 6.4, изучается на протяжении последних 40 лет и имеет чрезвычайно богатую историю. В работе De Raedt (2008) приведено отличное введение, включающее недавно полученные результаты, а обзор последних достижений и нерешенных проблем имеется в работе Muggleton et al. (2012). В работе Flach (1994) содержится введение в язык **Пролог** и предлагаются высокоуровневые реализации некоторых важнейших приемов индуктивного логического программирования. В системе **FOIL**, описанной в работе Quinlan (1990), реализован нисходящий алгоритм обучения, аналогичный рассмотренному здесь. Восходящий метод впервые был применен в системе **Golem** (Muggleton, Feng, 1990) и затем улучшен в системах **Progol** (Muggleton, 1995) и **Aleph** (Srinivasan, 2007) – двух самых распространенных системах ILP. Правила первого порядка можно обучать и без учителя, как, например, в системе **Tertius**, умеющей обучать дизъюнкты первого порядка (не обязательно хорновские) (Flach, Lachiche, 2001), и в системе

Warmr, которая обучает ассоциативные правила первого порядка (King et al., 2001). В логике высших порядков предлагаются более мощные типы данных, что открывает чрезвычайно благоприятные возможности для обучения (Lloyd, 2003). Из недавних направлений исследований можно упомянуть вероятностное моделирование логики первого порядка, которое ведет в область статистического реляционного обучения (De Raedt, Kersting, 2010).



Линейные модели

В предыдущих главах мы имели дело с логическими моделями, а теперь перейдем к моделям совершенно другого вида. Модели, рассматриваемые в этой и в следующей главах, определены в терминах геометрии пространства объектов. Чаще всего в геометрических моделях предполагается, что объекты описываются d вещественными признаками и, следовательно, $X = \mathbb{R}^d$. Например, мы могли бы описать объекты их координатами на карте: широтой и долготой ($d = 2$) или широтой, долготой и высотой над уровнем моря ($d = 3$). И хотя большинство вещественных признаков по природе своей не имеют отношения к геометрии – взять, к примеру, возраст человека или температуру предмета, – мы все равно можем изобразить их точками в d -мерной декартовой системе координат. А тогда можно воспользоваться такими геометрическими понятиями, как прямые и плоскости, чтобы ввести в этом пространстве структуру, например для построения модели классификации. Или использовать геометрическое понятие расстояния для представления сходства, считая, что если две точки близки, то их признаки похожи, а значит, можно ожидать, что они сходно ведут себя в интересующем нас отношении. Такие метрические модели, основанные на расстоянии, – тема следующей главы. А здесь мы рассмотрим модели, которые можно интерпретировать в терминах прямых и плоскостей, обычно их называют *линейными моделями*.

Линейность играет фундаментальную роль в математике и смежных дисциплинах, и математика линейных моделей хорошо разработана (см. описание важнейших понятий в замечании 7.1). В машинном обучении линейные модели представляют особый интерес в силу своей простоты (вспомните правило «все должно быть настолько просто, насколько возможно, но не проще»). Перечислим несколько проявлений этой простоты.

Если x_1 и x_2 – два скаляра или вектора одинаковой размерности, а α и β – произвольные скаляры, то $\alpha x_1 + \beta x_2$ называется *линейной комбинацией* x_1 и x_2 . Если f – *линейная функция* от x , то $f(\alpha x_1 + \beta x_2) = \alpha f(x_1) + \beta f(x_2)$.

Говоря словами: значение функции от линейной комбинации объектов равно линейной комбинации значений функции от этих объектов. В частном случае, когда $\beta = 1 - \alpha$, линейная комбинация называется взвешенным средним x_1 и x_2 , и линейность f тогда означает, что значение функции от взвешенного среднего равно взвешенному среднему значений функций.

Линейные функции принимают разные формы в зависимости от области определения и области значений f . Если x и $f(x)$ – скаляры, то f имеет вид $f(x) = a + bx$ для некоторых констант a и

b ; в этом случае a называется *свободным членом*, а b – *угловым коэффициентом*. Если $\mathbf{x} = (x_1, \dots, x_d)$ – вектор, а $f(\mathbf{x})$ – скаляр, то f принимает вид

$$f(\mathbf{x}) = a + b_1 x_1 + \dots + b_d x_d = a + \mathbf{b} \cdot \mathbf{x}, \quad (7.1)$$

где $\mathbf{b} = (b_1, \dots, b_d)$. Уравнение $f(\mathbf{x}) = 0$ определяет плоскость в пространстве \mathbb{R}^d , перпендикулярную *нормальному вектору* \mathbf{b} .

В самом общем случае $f(\mathbf{x})$ – d -мерный вектор, и тогда f принимает вид $f(\mathbf{x}) = \mathbf{M}\mathbf{x} + \mathbf{t}$, где \mathbf{M} – матрица размерности $d \times d$, представляющая *линейное преобразование*, например поворот или масштабирование, а \mathbf{t} – d -мерный вектор, представляющий параллельный перенос. В таком случае f называется *аффинным преобразованием* (разница между линейным и аффинным преобразованиями заключается в том, что первое оставляет начало координат на месте; отметим, что линейная функция вида (7.1) является линейным преобразованием, только если свободный член равен 0).

В любом из этих вариантов мы можем избавиться от свободного члена a или параллельного переноса \mathbf{t} , перейдя к однородным координатам. Например, положив в уравнении 7.1 $\mathbf{b}^\circ = (a, b_1, \dots, b_d)$ и $\mathbf{x}^\circ = (1, x_1, \dots, x_d)$, получим $f(\mathbf{x}) = \mathbf{b} \cdot \mathbf{x}^\circ$ (см. также замечание 1.2 на стр. 36).

Примерами нелинейных функций являются полиномы от x степени $p > 1$: $g(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_p x^p = \sum_{i=0}^p a_i x^i$. Другие нелинейные функции могут быть аппроксимированы полиномами путем разложения в ряд Тейлора. *Линейной аппроксимацией* функции g в точке x_0 называется функция $g(x_0) + g'(x_0)(x - x_0)$, где $g'(x)$ – производная от x . *Кусочно-линейная аппроксимация* получается путем объединения нескольких линейных аппроксимаций в разных точках.

Замечание 7.1. Линейные модели

- ☞ Линейные модели являются *параметрическими* в том смысле, что имеют фиксированную форму с небольшим количеством числовых параметров, которые должны быть найдены путем обучения по данным. Этим они отличаются от древовидных моделей и моделей на основе правил, структура которых (например, какие признаки использовать в дереве и в каком мес-те) заранее неизвестна.
- ☞ Линейные модели устойчивы, то есть небольшое изменение обучающих данных оказывает небольшое влияние на обученную модель. Древовидные модели обычно сильнее изменяются при изменении обучающих данных, поскольку выбор другого разделения в корне дерева, как правило, приводит к изменению и всего дерева в целом.
- ☞ Линейные модели менее склонны к переобучению, чем некоторые другие, в первую очередь потому, что имеют сравнительно немного параметров. Обратной стороной является *недообучение*: представьте, например, что на основе помеченных образцов требуется определить, где проходит граница между странами, – маловероятно, что линейная модель даст хорошую аппроксимацию.

Последние два положения можно сформулировать иначе, сказав, что у линейных моделей низкая дисперсия, но высокое смещение. Такие модели зачастую предпочтительны, когда объем данных ограничен и требуется избежать переобучения. Модели с высокой дисперсией и низким смещением, как, например, решающие деревья, предпочтительнее, если данных много, а недообучение соз-

дает трудности. Обычно рекомендуется начинать с простой модели с высоким смещением, например линейной, и переходить к более сложным, только если простая модель оказывается недообученной.

Существуют линейные модели для всех прогностических задач, в том числе классификации, оценивания вероятностей и регрессии. Особенно хорошо изучена задача линейной регрессии, которую можно решать методом наименьших квадратов, описанным в следующем разделе. В этой главе мы рассмотрим и ряд других линейных моделей, включая классификацию по методу наименьших квадратов (также в разделе 7.1), перцептрон (раздел 7.2) и метод опорных векторов (раздел 7.3). Мы также увидим, как эти модели можно превратить в методы оценивания вероятностей (раздел 7.4). Наконец, в разделе 7.5 мы вкратце обсудим, как все эти методы можно применить к обучению нелинейных моделей с помощью так называемых ядерных функций.

7.1 Метод наименьших квадратов

Начнем с метода, который можно применить к обучению линейных моделей для классификации и регрессии. Напомним, что задача регрессии заключается в том, чтобы обучить оценочную функцию $\hat{f}: \mathcal{X} \rightarrow \mathbb{R}$ на примерах $(x_i, f(x_i))$, причем в этой главе мы будем считать, что $\mathcal{X} = \mathbb{R}^d$. Разности между фактическими и оценочными значениями функции на обучающих примерах называются *невязками* $\epsilon_i = f(x_i) - \hat{f}(x_i)$. Метод наименьших квадратов, изобретенный Карлом Фридрихом Гауссом в конце XVIII века, заключается в отыскании такой функции \hat{f} , что величина $\sum_{i=1}^n \epsilon_i^2$ достигает минимума. Ниже этот метод демонстрируется в простом случае, когда существует всего один признак, это так называемая *одномерная регрессия*.

Пример 7.1 (одномерная линейная регрессия). Предположим, что изучается связь между ростом и весом человека. Мы собираем n измерений роста и веса (h_i, w_i) , $1 \leq i \leq n$. Одномерная линейная регрессия предполагает наличие линейного уравнения $w = a + bh$, в котором параметры a и b выбраны так, чтобы сумма квадратов невязок $\sum_{i=1}^n (w_i - (a + bh_i))^2$ обращалась в минимум. Чтобы найти эти параметры, мы приравниваем к нулю частные производные этого выражения и решаем получившиеся уравнения относительно a и b :

$$\begin{aligned} \frac{\partial}{\partial a} \sum_{i=1}^n (w_i - (a + bh_i))^2 &= -2 \sum_{i=1}^n (w_i - (a + bh_i)) = 0 & \Rightarrow \hat{a} = \bar{w} - \hat{b}\bar{h}; \\ \frac{\partial}{\partial b} \sum_{i=1}^n (w_i - (a + bh_i))^2 &= -2 \sum_{i=1}^n (w_i - (a + bh_i))h_i = 0 & \Rightarrow \hat{b} = \frac{\sum_{i=1}^n (h_i - \bar{h})(w_i - \bar{w})}{\sum_{i=1}^n (h_i - \bar{h})^2}. \end{aligned}$$

Таким образом, решение, найденное с помощью линейной регрессии, имеет вид $w = \hat{a} + \hat{b}h = \bar{w} + \hat{b}(\bar{h} - \bar{h})$; см. рис. 7.1.

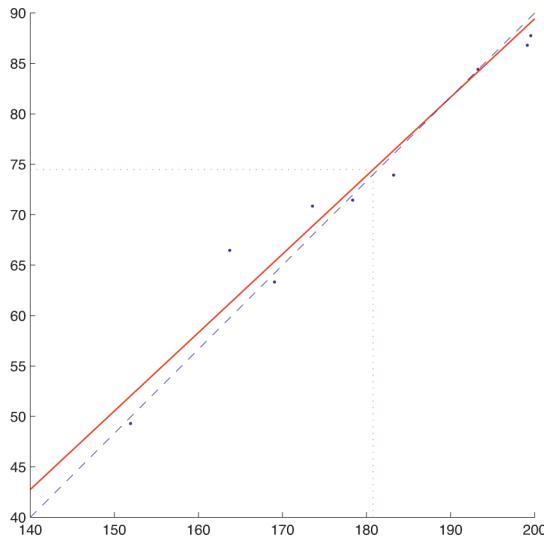


Рис. 7.1. Сплошной **красной** линией показан результат применения линейной регрессии к 10 измерениям зависимости веса тела (отложен по оси y , выражен в килограммах) от роста (отложен по оси x , выражен в сантиметрах). Пунктирные **оранжевые** линии обозначают средний рост $\bar{h} = 181$ и средний вес $\bar{w} = 74.5$; коэффициент регрессии равен $\hat{b} = 0.78$. Измерения были промоделированы путем добавления шума, имеющего нормальное распределение со средним 0 и дисперсией 5, к истинным данным, показанным **синей** штриховой линией ($b = 0.83$)

Следует отметить, что выражение для **коэффициента регрессии**, или углового коэффициента \hat{b} , вычисленное в этом примере, содержит умноженную на n ковариацию между h и w в числителе и умноженную на n дисперсию h знаменателе. Это верно и в общем случае: для признака x и целевой переменной y коэффициент регрессии равен

$$\hat{b} = n \frac{\sigma_{xy}}{n\sigma_{xx}} = \frac{\sigma_{xy}}{\sigma_{xx}}.$$

Здесь σ_{xx} – альтернативное обозначение σ_x^2 дисперсии величины x . Это легко понять, приняв во внимание, что ковариация измеряется в единицах x , умноженных на единицы y (в примере 7.1 в метрах, умноженных на килограммы), а дисперсия – в единицах x в квадрате (в нашем случае в квадратных метрах), поэтому их частное измеряется в единицах y на единицу x (в килограммах на метр).

Можно отметить еще несколько полезных вещей. Свободный член \hat{a} таков, что прямая регрессии проходит через точку (\bar{x}, \bar{y}) . Прибавление константы ко всем значениям x (параллельный перенос) влияет только на свободный член, но не на коэффициент регрессии (поскольку тот определен в терминах отклонения от среднего значения, которое при параллельном переносе не изменяется). Поэтому

мы можем *центрировать значения x относительно нуля*, вычтя из каждого величину \bar{x} , тогда свободный член будет равен \bar{y} . Можно было бы даже вычесть \bar{y} из всех значений y , чтобы получить нулевой свободный член, не изменив задачу существенным образом.

Далее, заменяя x_i на $x'_i = x_i/\sigma_{xx}$ и аналогично \bar{x} на $\bar{x}' = \bar{x}/\sigma_{xx}$, получаем $\hat{b} = \frac{1}{n} \sum_{i=1}^n (x'_i - \bar{x}')(y_i - \bar{y}) = \sigma_{x'y'}$. Иными словами, если *нормировать* x , поделив все значения по этой оси на дисперсию, то в качестве коэффициента регрессии можно будет взять ковариацию между нормированным признаком и целевой переменной. Иначе говоря, можно считать, что одномерная линейная регрессия состоит из двух шагов.

1. Нормировка признака путем деления его значений на дисперсию признака.
2. Вычисление ковариации между целевой переменной и нормированным признаком.

Ниже мы увидим, как эти два шага изменяются в случае, когда признаков больше двух.

Важно отметить еще один момент: сумма невязок решения, полученного методом наименьших квадратов, равна нулю:

$$\sum_{i=1}^n (y_i - (\hat{a} + \hat{b}x_i)) = n(\bar{y} - \hat{a} - \hat{b}\bar{x}) = 0.$$

Это следует из того, что $\hat{a} = \bar{y} - \hat{b}\bar{x}$, как показано в примере 7.1. И хотя интуитивно это свойство кажется красивым, следует иметь в виду, что оно делает линейную регрессию чувствительной к *выбросам*: точкам, далеко отстоящим от прямой регрессии, – нередко из-за ошибок измерения.

Пример 7.2 (влияние выбросов). Предположим, что из-за ошибки при записи данных один вес на рис. 7.1 увеличился на 10 кг. Как видно по рис. 7.2, это оказывает заметное воздействие на прямую регрессии, построенную по методу наименьших квадратов.

Несмотря на чувствительность к выбросам, метод наименьших квадратов при всей своей простоте обычно работает на удивление хорошо. Как это объяснить? Давайте предположим, что истинная функция f действительно линейна, а наблюдаемые значения y загрязнены случайным шумом. То есть наши примеры имеют вид $(x_i, f(x_i) + \epsilon_i)$ вместо $(x_i, f(x_i))$. И еще предположим, что $f(x) = ax + b$ для некоторых a и b . Если бы мы знали, чему равны a и b , то могли бы точно вычислить невязки, а если бы мы знали величину σ^2 , то могли бы посчитать вероятность наблюдения именно такого набора невязок. Но поскольку a и b неизвестны, то мы должны их оценить, причем так, чтобы их оценки обращали в максимум вероятность невязок. В главе 9 мы узнаем, что эта так называемая *оценка максимального правдоподобия* в точности совпадает с решением, полученным по методу наименьших квадратов.

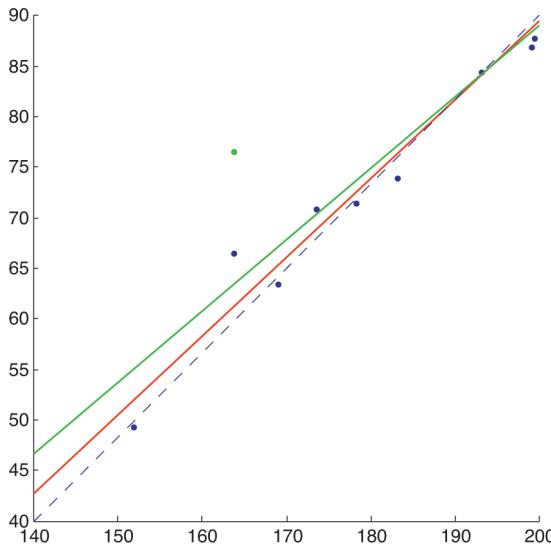


Рис. 7.2. Влияние выброса на одномерную регрессию. Одна из **синих** точек поднялась на 10 единиц вверх и стала **зеленой**, в результате чего **красная** прямая регрессии сменилась **зелёной**

Существуют варианты метода наименьших квадратов. Выше рассматривался **обычный** метод наименьших квадратов, в котором предполагается, что случайнм шумом загрязнены только значения y . В **обобщенном**, или **строгом**, методе наименьших квадратов зашумленными предполагаются также значения x , но при такой постановке решение необязательно единственное.

Буквой \mathbf{X} обычно обозначают матрицу $n \times d$, строки которой представляют n объектов, описываемых d признаками или переменными. \mathbf{X}_r обозначает r -ую строку \mathbf{X} , \mathbf{X}_c – c -ый столбец, а \mathbf{X}_{rc} – элемент на пересечении r -ой строки и c -го столбца. Для пробегания по строкам и столбцам мы будем также использовать индексы i и j соответственно. Среднее по j -му столбцу определяется

по формуле $\mu_j = \frac{1}{n} \sum_{i=1}^n \mathbf{X}_{ij}$; μ^T обозначает вектор-строку, содержащий средние по всем столбцам.

Если $\mathbf{1}$ – n -мерный вектор, содержащий только единицы, то $\mathbf{1}\mu^T$ – матрица размерности $n \times d$, строками которой являются векторы μ^T , поэтому в матрице $\mathbf{X}' = \mathbf{X} - \mathbf{1}\mu^T$ среднее по каждому столбцу равно нулю, из-за чего она называется **центрированной** матрицей.

Матрицей разброса называется матрица размерности $d \times d$ $\mathbf{S} = \mathbf{X}'\mathbf{X}'^T = (\mathbf{X} - \mathbf{1}\mu^T)^T(\mathbf{X} - \mathbf{1}\mu^T) = \mathbf{X}'\mathbf{X} - n\mathbf{M}$, где $\mathbf{M} = \mu\mu^T$ – матрица $d \times d$, элементами которой являются произведения средних по столбцам $\mathbf{M}_{jc} = \mu_j\mu_c$. **Ковариационной матрицей** \mathbf{X} называется матрица $\Sigma = (1/n)\mathbf{S}$, элементами которой являются попарные ковариации $\sigma_{jc} = \frac{1}{n} \sum_{i=1}^n (\mathbf{X}_{ij} - \mu_j)(\mathbf{X}_{ic} - \mu_c) = \frac{1}{n} \left(\sum_{i=1}^n \mathbf{X}_{ij}\mathbf{X}_{ic} - \mu_i\mu_c \right)$.

У двух некоррелированных признаков ковариация близка к нулю. У положительно коррелированных признаков ковариация положительна, это свидетельствует о тенденции к одновременному возрастанию или убыванию. Отрицательная ковариация – знак того, что при возрастании одного признака другой убывает, и наоборот. Величина $\sigma_{jj} = \frac{1}{n} \sum_{i=1}^n (\mathbf{X}_{ij} - \mu_j)^2 = \frac{1}{n} \left(\sum_{i=1}^n \mathbf{X}_{ij}^2 - \mu_j^2 \right)$ на-

зывается *дисперсией* столбца j и обозначается также σ_j^2 . Дисперсия всегда положительна и описывает разброс значений признака относительно среднего.

Для иллюстрации этих определений приведем пример:

$$\mathbf{X} = \begin{pmatrix} 5 & 0 \\ 3 & 5 \\ 1 & 7 \end{pmatrix} \quad \mathbf{1}\mu^T = \begin{pmatrix} 3 & 4 \\ 3 & 4 \\ 3 & 4 \end{pmatrix} \quad \mathbf{X}' = \begin{pmatrix} 2 & -4 \\ 0 & 1 \\ -2 & 3 \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} 25 & 15 & 5 \\ 15 & 34 & 38 \\ 5 & 38 & 50 \end{pmatrix}$$

$$\mathbf{X}'\mathbf{X} = \begin{pmatrix} 35 & 22 \\ 22 & 74 \end{pmatrix} \quad \mathbf{M} = \begin{pmatrix} 9 & 12 \\ 12 & 16 \end{pmatrix} \quad \mathbf{S} = \begin{pmatrix} 8 & -14 \\ -14 & 26 \end{pmatrix} \quad \Sigma = \begin{pmatrix} 8/3 & -14/3 \\ -14/3 & 26-3 \end{pmatrix}$$

Мы видим, что два признака отрицательно коррелируют и что у второго признака дисперсия больше. По-другому матрицу разброса можно вычислить как сумму внешних произведений, по одному для каждого результата измерения: $\mathbf{S} = \sum_{i=1}^n (\mathbf{X}_i - \mu^T)(\mathbf{X}_i - \mu^T)^T$. В нашем примере имеем:

$$(\mathbf{X}_1 - \mu^T)^T(\mathbf{X}_1 - \mu^T) = \begin{pmatrix} 2 \\ -4 \end{pmatrix}(2 \quad -4) = \begin{pmatrix} 4 & -8 \\ -8 & 16 \end{pmatrix};$$

$$(\mathbf{X}_2 - \mu^T)^T(\mathbf{X}_2 - \mu^T) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}(0 \quad 1) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix};$$

$$(\mathbf{X}_3 - \mu^T)^T(\mathbf{X}_3 - \mu^T) = \begin{pmatrix} -2 \\ 3 \end{pmatrix}(-2 \quad 3) = \begin{pmatrix} 4 & -6 \\ -6 & 9 \end{pmatrix}.$$

Замечание 7.2. Еще о матричной нотации

Многомерная линейная регрессия

При рассмотрении случая с произвольным количеством признаков удобно использовать матричную нотацию (см. замечание 7.2). Одномерную линейную регрессию можно представить в матричной форме:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} a + \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} b + \begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix};$$

$$\mathbf{y} = \mathbf{a} + \mathbf{X}\mathbf{b} + \boldsymbol{\epsilon}.$$

Во второй форме этого уравнения \mathbf{y} , \mathbf{a} , \mathbf{X} и $\boldsymbol{\epsilon}$ – n -мерные векторы, а \mathbf{b} – скаляр. В случае d признаков \mathbf{X} просто становится матрицей $n \times d$, а \mathbf{b} – d -мерным вектором коэффициентов регрессии.

Применив уже знакомый прием – переход к однородным координатам, – мы можем следующим образом упростить эти уравнения:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix};$$

$$\mathbf{y} = \mathbf{X}^\circ \mathbf{w} + \boldsymbol{\epsilon},$$

где \mathbf{X}° – матрица размерности $n \times (d + 1)$, первый столбец которой содержит только единицы, а остальные совпадают со столбцами \mathbf{X} ; первым элементом \mathbf{w} является свободный член, а остальные d – коэффициенты регрессии. Для удобства мы часто не будем акцентировать внимание на различии этих формулировок и записывать уравнение регрессии в виде $\mathbf{y} = \mathbf{X}\mathbf{w} + \epsilon$, где матрица \mathbf{X} имеет d столбцов, а \mathbf{w} состоит из d строк, – из контекста должно быть ясно, представляем мы свободный член с помощью однородных координат или центрировали целевую переменную и признаки, чтобы избавиться от свободного члена.

В одномерном случае мы смогли получить решение, \mathbf{w} , в виде замкнутой формулы: можно ли сделать то же самое в многомерном случае? Во-первых, нам, вероятно, понадобятся ковариации между каждым признаком и целевой переменной. Рассмотрим выражение $\mathbf{X}^T\mathbf{y}$, являющееся n -мерным вектором, j -ый элемент которого равен произведению j -ой строки \mathbf{X}^T – то есть j -го столбца \mathbf{X} (x_{1j}, \dots, x_{nj}) – на (y_1, \dots, y_n) :

$$(\mathbf{X}^T\mathbf{y}) = \sum_{i=1}^n x_{ij}y_i = \sum_{i=1}^n (x_{ij} - \mu_j)(y_i - \bar{y}) + n\mu_j\bar{y} = n(\sigma_{jy} + \mu_j\bar{y}).$$

Если предположить, что каждый признак центрирован, то $\mu_j = 0$ и, следовательно, $\mathbf{X}^T\mathbf{y}$ – n -мерный вектор, содержащий все необходимые ковариации (умноженные на n).

В одномерном случае нам пришлось нормировать признаки, чтобы получить единичную дисперсию. В многомерном случае мы можем достичь того же результата с помощью масштабирующей матрицы размерности $d \times d$: диагональной матрицы, в которой на диагонали находятся числа $1/n\sigma_{jj}$. Если \mathbf{S} – диагональная матрица с элементами $n\sigma_{jj}$, то для получения масштабирующей матрицы нужно просто обратить \mathbf{S} . Таким образом, наша первая попытка записать решение задачи *многомерной регрессии* привела к такой формуле:

$$\hat{\mathbf{w}} = \mathbf{S}^{-1}\mathbf{X}^T\mathbf{y}. \quad (7.2)$$

Как выясняется, в общем случае вместо \mathbf{S} надо взять более сложную матрицу:

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \quad (7.3)$$

Попробуем разобраться, в чем смысл члена $(\mathbf{X}^T\mathbf{X})^{-1}$. Предположим, что признаки не только центрированы относительно нуля, но и не коррелируют (то есть ковариация между любой парой различных признаков равна 0). В обозначениях из замечания 7.2 ковариационная матрица Σ будет диагональной с элементами σ_{jj} . Поскольку $\mathbf{X}^T\mathbf{X} = n(\Sigma + \mathbf{M})$ и элементы \mathbf{M} равны 0, так как столбцы \mathbf{X} центрированы, то эта матрица является диагональной с элементами $n\sigma_{jj}$ – то есть не чем иным, как вышеупомянутой матрицей \mathbf{S} . Иными словами, если признаки центрированы и не коррелированы, то $(\mathbf{X}^T\mathbf{X})^{-1}$ сводится к масштабирующей матрице \mathbf{S}^{-1} . В общем случае мы не можем делать никаких предположений относительно признаков, и $(\mathbf{X}^T\mathbf{X})^{-1}$ играет роль преобразования, которое устраняет корреляцию признаков, а также центрирует и нормирует их.

Чтобы добавить конкретики, в следующем примере показано, как все это выглядит в двумерном случае:

Пример 7.3 (двумерная линейная регрессия в матричной нотации). Сначала выпишем общие выражения.

$$\begin{aligned}\mathbf{X}^T \mathbf{X} &= \begin{pmatrix} x_{11} & \cdots & x_{n1} \\ x_{12} & \cdots & x_{n2} \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{n1} & x_{n2} \end{pmatrix} = n \begin{pmatrix} \sigma_{11} + \bar{x}_1^2 & \sigma_{12} + \bar{x}_1 \bar{x}_2 \\ \sigma_{12} + \bar{x}_1 \bar{x}_2 & \sigma_{22} + \bar{x}_2^2 \end{pmatrix}; \\ (\mathbf{X}^T \mathbf{X})^{-1} &= \frac{1}{nD} \begin{pmatrix} \sigma_{22} + \bar{x}_2^2 & \sigma_{12} - \bar{x}_1 \bar{x}_2 \\ -\sigma_{12} - \bar{x}_1 \bar{x}_2 & \sigma_{11} + \bar{x}_1^2 \end{pmatrix}; \\ D &= (\sigma_{11} + \bar{x}_1^2)(\sigma_{22} + \bar{x}_2^2) - (\sigma_{12} + \bar{x}_1 \bar{x}_2)^2; \\ \mathbf{X}^T \mathbf{y} &= \begin{pmatrix} x_{11} & \cdots & x_{n1} \\ x_{12} & \cdots & x_{n2} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = n \begin{pmatrix} \sigma_{1y} + \bar{x}_1 \bar{y} \\ \sigma_{2y} + \bar{x}_2 \bar{y} \end{pmatrix}.\end{aligned}$$

Теперь рассмотрим два частных случая. Во-первых, если \mathbf{X} – матрица в однородных координатах, то мы фактически имеем дело с одномерной задачей. В этом случае $x_{i1} = 1$ для $1 \leq i \leq n$; $\bar{x}_1 = 1$, $\sigma_{11} = \sigma_{12} = \sigma_{1y} = 0$. Тогда получаем (мы пишем x вместо x_2 , σ_{xx} вместо σ_{22} и σ_{xy} вместо σ_{2y}):

$$\begin{aligned}(\mathbf{X}^T \mathbf{X})^{-1} &= \frac{1}{n\sigma_{xx}} \begin{pmatrix} \sigma_{xx} + \bar{x}^2 & -\bar{x} \\ -\bar{x} & 1 \end{pmatrix}; \\ \mathbf{X}^T \mathbf{y} &= n \begin{pmatrix} \bar{y} \\ \sigma_{xy} + \bar{x} \bar{y} \end{pmatrix}; \\ \hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \frac{1}{\sigma_{xx}} \begin{pmatrix} \sigma_{xx} \bar{y} - \sigma_{xy} \bar{x} \\ \sigma_{xy} \end{pmatrix}.\end{aligned}$$

Это тот же самый результат, что был получен в примере 7.1

В качестве второго частного случая предположим, что x_1 , x_2 и y центрированы относительно нуля, то есть свободный член равен нулю и \mathbf{w} содержит два коэффициента регрессии. Тогда получаем

$$\begin{aligned}(\mathbf{X}^T \mathbf{X})^{-1} &= \frac{1}{n(\sigma_{11}\sigma_{22} - \sigma_{12}^2)} \begin{pmatrix} \sigma_{22} & -\sigma_{12} \\ -\sigma_{12} & \sigma_{11} \end{pmatrix}; \\ \mathbf{X}^T \mathbf{y} &= n \begin{pmatrix} \sigma_{1y} \\ \sigma_{2y} \end{pmatrix}; \\ \hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \frac{1}{(\sigma_{11}\sigma_{22} - \sigma_{12}^2)} \begin{pmatrix} \sigma_{22}\sigma_{1y} - \sigma_{12}\sigma_{2y} \\ \sigma_{11}\sigma_{2y} - \sigma_{12}\sigma_{1y} \end{pmatrix}.\end{aligned}$$

Последнее выражение показывает, в частности, что коэффициент регрессии для x_1 может быть отличен от нуля, даже если x_1 не коррелирует с целевой переменной ($\sigma_{1y} = 0$) – за счет корреляции между x_1 и x_2 ($\sigma_{12} \neq 0$).

Отметим, что если мы все-таки предполагаем, что $\sigma_{12} = 0$, то компоненты $\hat{\mathbf{w}}$ сводятся к σ_{jj}/σ_{jj} , что приводит нас опять к уравнению (7.2) *Предположение о некоррелированности признаков по существу позволяет разложить задачу многомерной регрессии на d одномерных задач*. В этой книге мы еще встретимся с другими примерами разложения многомерных проблем обучения на одномерные; на самом деле один такой пример – *наивный байесовский* классификатор – мы уже видели в главе 1. Но тогда, спросите вы, зачем вообще принимать корреляцию признаков в расчет?

Дело в том, что игнорирование корреляции признаков в некоторых ситуациях опасно. Взгляните на рис. 7.3: слева корреляция между признаками невелика, и потому выборка дает много информации об истинной функции. Справа между признаками имеется сильная отрицательная корреляция, поэтому выборочные значения $y = x_1 + x_2 + \epsilon$ кажутся почти постоянными, так как любое увеличение одного признака сопровождается почти равным уменьшением другого. В результате разложение задачи на две одномерные задачи регрессии приводит к обучению почти постоянной функции. Честно говоря, в данном случае даже включение в рассмотрение полной ковариационной матрицы поможет мало. Однако – хотя мы и не станем вдаваться в детали – одно из преимуществ полного учета ковариации состоит в том, что он позволяет осознать, что в этой ситуации нельзя слишком доверять оценкам параметров регрессии. Вычислительная сложность получения решения в замкнутой форме (формула 7.3) обусловлена обращением матрицы $\mathbf{X}^T \mathbf{X}$ размерности $d \times d$, что при большом значении d может оказаться невозможным.

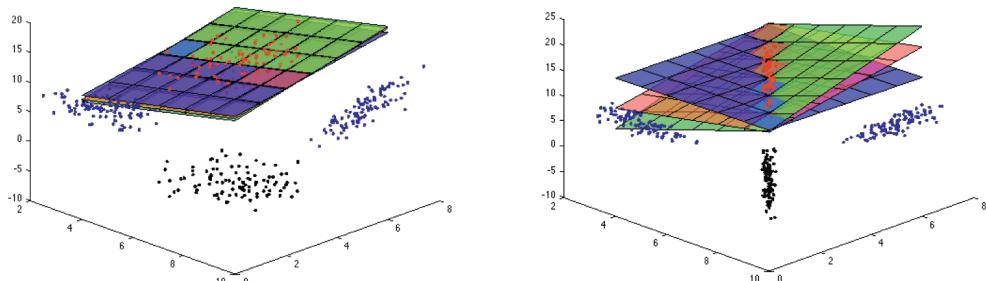


Рис. 7.3. (Слева) Функции регрессии, обученные при решении задачи линейной регрессии. Истинная функция имеет вид $y = x_1 + x_2$ (красная плоскость). Красные точки – это зашумленная выборка данной функции; черные точки – их проекции на плоскость (x_1, x_2) . Зеленая плоскость представляет функцию, обученную в ходе линейной регрессии; синяя плоскость – это результат разложения задачи на две одномерные задачи регрессии (синие точки). Обе являются хорошими аппроксимациями истинной функции. **(Справа)** Истинная функция та же, но теперь между x_1 и x_2 имеется сильная отрицательная корреляция. Выборка дает гораздо меньше информации об истинной функции: на самом деле из разложения на одномерные задачи складывается впечатление, что функция постоянна

Регуляризованная регрессия

Мы только что видели ситуацию, когда регрессия по методу наименьших квадратов может стать *неустойчивой*, то есть сильно зависящей от обучающих данных. Неустойчивость – это проявление тенденции к переобучению. Избежать такого переобучения помогает *регуляризация* – общий метод, заключающийся в наложении дополнительных ограничений на вектор весов. Общепринятый подход заключается в том, чтобы выбирать веса, малые по абсолютной величине в среднем; это называется *стягиванием*. Чтобы показать, как это можно сделать, мы сначала переформулируем задачу регрессии по методу наименьших квадратов как задачу оптимизации:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}).$$

Правая часть – это просто способ записи суммы квадратов невязок в виде скалярного произведения. Тогда регуляризованный вариант этой оптимизации принимает вид:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|^2, \quad (7.4)$$

где $\|\mathbf{w}\|^2 = \sum_i w_i^2$ – квадрат нормы вектора \mathbf{w} , или, эквивалентно, скалярное произведение $\mathbf{w}^T \mathbf{w}$; λ – скаляр, определяющий величину регуляризации. Регуляризованная задача по-прежнему имеет решение в замкнутой форме:

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}, \quad (7.5)$$

где \mathbf{I} – единичная матрица, в которой элементы на главной диагонали равны единице, а все остальные – нулю. Сравнивая это выражение с формулой 7.3, мы видим, что регуляризация сводится к прибавлению λ к элементам на диагонали матрицы $\mathbf{X}^T \mathbf{X}$ – хорошо известный прием, призванный улучшить численную устойчивость обращения матрицы. Эта форма регрессии по методу наименьших квадратов называется *гребневой регрессией*.

Интересную альтернативную форму регуляризованной регрессии дает метод *lasso* (least absolute shrinkage and selection operator). В нем член гребневой регуляризации $\sum_i w_i^2$ заменяется суммой абсолютных весов $\sum_i |w_i|$. (В терминологии, которая будет введена в определении 8.2 на стр. 246: в методе lasso используется L_1 -регуляризация, тогда как в гребневой регрессии – норма L_2 .) В результате некоторые веса стягиваются, а другие обращаются в 0, поэтому регрессия по методу lasso предпочитает *разреженные решения*. Следует отметить, что регрессия по методу lasso очень чувствительна к параметру регуляризации λ , который обычно подбирается на зарезервированных данных или в ходе перекрестной проверки. Кроме того, решения в замкнутой форме не существует, и необходимо применять те или иные методы численной оптимизации.

Применение регрессии по методу наименьших квадратов к задаче классификации

До сих пор мы использовали метод наименьших квадратов для построения аппроксимаций функций. Интересно, что линейную регрессию можно применить и к обучению бинарного классификатора, если представить два класса вещественными числами. Например, мы можем пометить *Pos* положительных примеров числом $y^{\oplus} = +1$, а *Neg* отрицательных примеров – числом $y^{\ominus} = -1$. Тогда $\mathbf{X}^T \mathbf{y} = Pos \mu^{\oplus} - Neg \mu^{\ominus}$, где μ^{\oplus} и μ^{\ominus} – d -мерные векторы, содержащие средние значения каждого признака для положительных и отрицательных примеров соответственно.

Пример 7.4 (классификатор на основе одномерного метода наименьших квадратов). В одномерном случае имеем $\sum_i x_i y_i = Pos \mu^{\oplus} - Neg \mu^{\ominus}$; мы также знаем (см. пример 7.3), что $\sum_i x_i y_i = n(\sigma_{xy} + \bar{x}\bar{y})$, и потому $\sigma_{xy} = pos \mu^{\oplus} - neg \mu^{\ominus} - \bar{x}\bar{y}$. Так как $\bar{x} = pos \mu^{\oplus} + neg \mu^{\ominus}$ и $\bar{y} = pos - neg$, то мы можем переписать ковариацию между x и y в виде $\sigma_{xy} = 2pos \cdot neg (\mu^{\oplus} - \mu^{\ominus})$, а угловой коэффициент прямой регрессии равен

$$\hat{b} = 2pos \cdot neg \frac{\mu^{\oplus} - \mu^{\ominus}}{\sigma_{xx}}. \quad (7.6)$$

Это уравнение показывает, что угловой коэффициент прямой регрессии увеличивается вместе с ростом разделенности классов (которая измеряется как расстояние между средними классов, поделенное на дисперсию признака), но уменьшается, если распределение по классам становится асимметричным.

Тогда уравнение регрессии $y = \bar{y} + \hat{b}(x - \bar{x})$ можно использовать для получения решающей границы. Нам нужно определить точку (x_0, y_0) , так чтобы y_0 лежала посередине между y^{\oplus} и y^{\ominus} (в нашем случае будет $y_0 = 0$). Тогда будем иметь

$$x_0 = \bar{x} + \frac{y_0 - \bar{y}}{\hat{b}} = \bar{x} - \frac{pos - neg}{2pos \cdot neg} \frac{\sigma_{xx}}{\mu^{\oplus} - \mu^{\ominus}}.$$

Следовательно, если положительных и отрицательных примеров поровну, то порог признака просто устанавливается равным среднему значению \bar{x} ; если же распределение по классам асимметрично, то этот порог смещается вправо или влево (рис. 7.4).

В общем случае **классификатор по методу наименьших квадратов** обучается решающей границе $\mathbf{w} \cdot \mathbf{x} = t$, где

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} (Pos \mu^{\oplus} - Neg \mu^{\ominus}). \quad (7.7)$$

Таким образом, мы могли бы назначить объекту x класс $\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x} - t)$, где

$$\text{sign}(x) = \begin{cases} +1 & \text{если } x > 0 \\ 0 & \text{если } x = 0 \\ -1 & \text{если } x < 0 \end{cases}$$

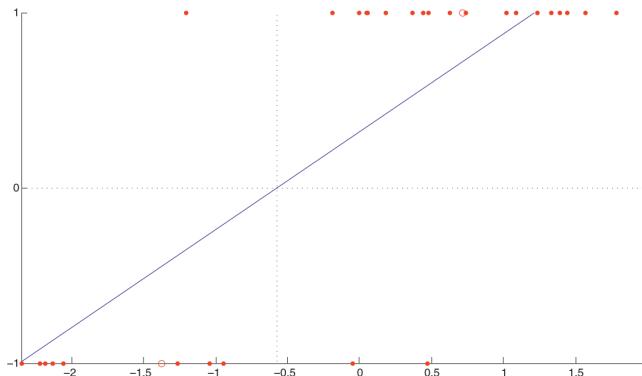


Рис. 7.4. Применение одномерной линейной регрессии для получения границы решений. 10 отрицательных примеров помечены числом $y^{\ominus} = -1$, а 20 положительных – числом $y^{\oplus} = +1$. μ^{\oplus} и μ^{\ominus} обозначены красными кружочками. Синяя линия – это прямая регрессии $y = \bar{y} + b(x - \bar{x})$, а перекрестье обозначает решающую границу $x_0 = \bar{x} - \bar{y}/\hat{b}$. В результате три примера оказываются классифицированы неправильно; отметим, что это лучшее, чего можно достичь при имеющихся данных

Можно делать различные упрощающие предположения, например: признаки центрированы, признаки имеют одинаковую дисперсию, признаки некоррелированы, и встречаемость классов одинакова. В простейшем случае, когда выполнены все эти предположения, уравнение 7.7 сводится к $\mathbf{w} = c(\mu^{\oplus} - \mu^{\ominus})$, где c – некоторый скаляр, который можно включить в порог принятия решения t . Это не что иное, как [☞] **базовый линейный классификатор**, о котором было упомянуто в разделе «Пролог». Таким образом, уравнение 7.7 говорит, как с помощью метода наименьших квадратов модифицировать базовый линейный классификатор, так чтобы принимались во внимание корреляция признаков и асимметричное распределение по классам.

Подводя итог, можно сказать, что *общий способ построения линейного классификатора с решающей границей $\mathbf{w} \cdot \mathbf{x} = t$ заключается в том, чтобы конструировать \mathbf{w} в виде $\mathbf{M}^{-1}(n^{\oplus}\mu^{\oplus} - n^{\ominus}\mu^{\ominus})$* , по-разному выбирая \mathbf{M} , n^{\oplus} и n^{\ominus} . В случае подхода на основе полной ковариации, когда $\mathbf{M} = \mathbf{X}^T \mathbf{X}$, временная сложность равна $O(n^2d)$ для построения матрицы \mathbf{M} и $O(d^3)$ для ее обращения¹, поэтому при большом количестве признаков он становится практически нереализуем.

7.2 Перцептрон

Напомним, что в главе 1 мы называли помеченные данные [☞] **линейно разделимыми**, если существует линейная решающая граница, разделяющая классы. Класси-

¹ С помощью более сложного алгоритма можно добиться сложности $O(d^{2.8})$, но это, по-видимому, лучше, на что можно рассчитывать.

фикатор по методу наименьших квадратов может найти идеально разделяющую решающую границу, если таковая существует, однако гарантии нет. Чтобы убедиться в этом, предположим, что базовый линейный классификатор реализует идеальное разделение на данном обучающем наборе. Теперь будем отодвигать все положительные примеры, кроме одного, подальше от отрицательного класса. Решающая граница также будет отодвигаться и в какой-то момент пересечет единственный оставшийся на месте положительный пример. По построению модифицированные данные по-прежнему линейно разделимы, так как их разделяет первоначальная граница. Однако статистика модифицированных данных такова, что базовый линейный классификатор будет неправильно классифицировать один положительный выброс.

Линейный классификатор, который достигает идеального разделения на линейно разделимых данных, называется *перцептроном*, и первоначально он предлагался как простая нейронная сеть. Перцептрон перебирает обучающий набор, обновляя вектор весов всякий раз, как встречает неправильно классифицированный пример. Пусть, например, \mathbf{x}_i – неправильно классифицированный положительный пример, тогда $y_i = +1$ и $\mathbf{w} \cdot \mathbf{x}_i < t$. Таким образом, мы хотим найти такой вектор \mathbf{w}' , чтобы $\mathbf{w}' \cdot \mathbf{x}_i > \mathbf{w} \cdot \mathbf{x}_i$, при этом решающая граница сдвигается вперед и, хочется надеяться, оставит \mathbf{x}_i позади. Этого можно добиться, если вычислять новый вектор весов в виде $\mathbf{w}' = \mathbf{w} + \eta \mathbf{x}_i$, где величина $0 < \eta \leq 1$ называется *скоростью обучения*. Тогда имеем $\mathbf{w}' \cdot \mathbf{x}_i = \mathbf{w} \cdot \mathbf{x}_i + \eta \mathbf{x}_i \cdot \mathbf{x}_i > \mathbf{w} \cdot \mathbf{x}_i$, что и требовалось. Аналогично, если \mathbf{x}_j – неправильно классифицированный отрицательный пример, то имеем $y_j = -1$ и $\mathbf{w} \cdot \mathbf{x}_j > t$. В этом случае мы вычисляем новый вектор весов в виде $\mathbf{w}' = \mathbf{w} - \eta \mathbf{x}_j$ и, следовательно, $\mathbf{w}' \cdot \mathbf{x}_j = \mathbf{w} \cdot \mathbf{x}_j - \eta \mathbf{x}_j \cdot \mathbf{x}_j < \mathbf{w} \cdot \mathbf{x}_j$. Оба случая можно объединить в одно правило обновления:

$$\mathbf{w}' = \mathbf{w} + \eta y_i \mathbf{x}_i. \quad (7.8)$$

Алгоритм обучения перцептрана приведен в алгоритме 7.1. Он перебирает обучающие примеры, пока все они не будут классифицированы правильно. Этот алгоритм легко превратить в *динамический*, то есть такой, который обрабатывает поток примеров, обновляя вектор весов, только если неправильно классифицирован последний полученный пример. Гарантируется, что перцептрон сходится к решению, если обучающие данные линейно разделимы, а в противном случае он не сходится. На рис. 7.5 графически представлен алгоритм обучения перцептрана. В этом примере для инициализации вектора весов я взял базовый линейный классификатор, при этом скорость обучения действительно влияет на то, как быстро мы будем отдаляться от начальной решающей границы. Однако если вектор весов первоначально нулевой, то легко видеть, что скорость обучения – просто постоянный множитель, не влияющий на сходимость. Далее в этом разделе мы будем считать ее равной 1.

Ключевой момент алгоритма обучения перцептрана заключается в том, что всякий раз, как пример \mathbf{x}_i оказывается классифицирован неправильно, мы прибавляем $y_i \mathbf{x}_i$ к вектору весов. После завершения обучения каждый пример был

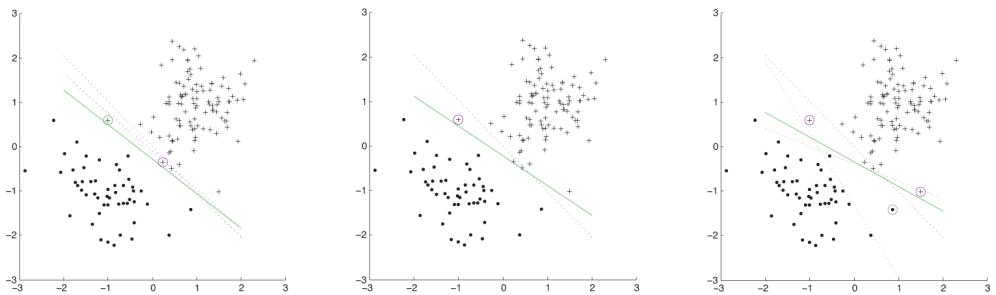


Рис. 7.5. (Слева) Перцентрон, обученный при малой скорости обучения ($\eta = 0.2$). Кружочками обведены примеры, на которых происходило обновление вектора весов. **(В центре)** В этом случае увеличение скорости обучения до $\eta = 0.5$ приводит к ускорению сходимости. **(Справа)** Дальнейшее увеличение скорости обучения до $\eta = 1$ может привести к слишком агрессивному обновлению весов, что отрицательно влияет на сходимость. Во всех трех примерах в качестве отправной точки был взят линейный классификатор

неправильно классифицирован нуль или более раз – пусть для примера \mathbf{x}_i это число равно α_i .

Алгоритм 7.1. *Perceptron(D, η)* – обучить перцентрон для линейной классификации

Вход: помеченные обучающие данные D в однородных координатах, скорость обучения η .

Выход: вектор весов \mathbf{w} , определяющий классификатор $\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x})$.

```

1  $\mathbf{w} \leftarrow \mathbf{0};$  // можно инициализировать вектор весов и по-другому
2 converged  $\leftarrow \text{false};$ 
3 while converged = false do
4   converged  $\leftarrow \text{true};$ 
5   for  $i = 1$  to  $|D|$  do
6     if  $y_i \mathbf{w} \cdot \mathbf{x}_i \leq 0$  // то есть  $\hat{y}_i \neq y_i$ 
7       then
8          $w \leftarrow w + \eta y_i \mathbf{x}_i;$ 
9         converged  $\leftarrow \text{false};$  // Мы изменили  $\mathbf{w}$ , значит, алгоритм еще не сошелся
10      end
11    end
12  end
```

В этих обозначениях вектор весов можно записать в виде

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i. \quad (7.9)$$

Иначе говоря, вектор весов – это линейная комбинация обучающих примеров. Этим свойством обладает не только перцентрон, но и, например, базовый линейный классификатор:

$$\begin{aligned} \mathbf{w}_{blk} &= \mu^{\oplus} - \mu^{\ominus} = \frac{1}{Pos} \sum_{\mathbf{x}^{\oplus} \in Tr^{\oplus}} \mathbf{x}^{\oplus} - \frac{1}{Neg} \sum_{\mathbf{x}^{\ominus} \in Tr^{\ominus}} \mathbf{x}^{\ominus} = \\ &= \sum_{\mathbf{x}^{\oplus} \in Tr^{\oplus}} \alpha^{\oplus} c(\mathbf{x}^{\oplus}) \mathbf{x}^{\oplus} + \sum_{\mathbf{x}^{\ominus} \in Tr^{\ominus}} \alpha^{\ominus} c(\mathbf{x}^{\ominus}) \mathbf{x}^{\ominus}, \end{aligned} \quad (7.10)$$

где $c(\mathbf{x})$ – истинный класс примера \mathbf{x} (то есть $+1$ или -1), $\alpha^{\oplus} = 1/Pos$, $\alpha^{\ominus} = 1/Neg$. В двойственной, основанной на объектах форме линейной классификации мы ищем веса объектов α_i , а не веса признаков w_j . С этой двойственной точки зрения объект \mathbf{x} классифицируется как $\hat{y} = \text{sign}(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x})$. Следовательно, на этапе обучения единственное, что нужно знать об обучающих данных, – это все попарные скалярные произведения: матрица $\mathbf{G} = \mathbf{XX}^T$ размерности $n \times n$, содержащая эти произведения, называется *матрицей Грама*. Двойственная форма алгоритма обучения перцептрона приведена в алгоритме 7.2. Мы еще встретимся с таким взглядом на проблему при обсуждении метода опорных векторов в следующем разделе.

Алгоритм 7.2. DualPerceptron(D) – обучение перцептрона в двойственной форме

```

Вход: помеченные обучающие данные  $D$  в однородных координатах.
Выход: коэффициенты  $\alpha_i$ , определяющие вектор весов  $\mathbf{w} = \sum_{i=1}^{|D|} \alpha_i y_i \mathbf{x}_i$ .
1  $\alpha_i \leftarrow 0$  для  $1 \leq i \leq |D|$ ;
2  $converged \leftarrow \text{false}$ ;
3 while  $converged = \text{false}$  do
4    $converged \leftarrow \text{true}$ ;
5   for  $i = 1$  до  $|D|$  do
6     if формула then
7        $\alpha_i \leftarrow \alpha_i + 1$ ;
8        $converged \leftarrow \text{false}$ ;
9     end
10   end
11 end
```

На рис. 7.6 продемонстрировано различие между базовым линейным классификатором, классификатором по методу наименьших квадратов и перцептроном на случайно взятом наборе данных. На этом наборе ни первый, ни второй классификаторы не достигают идеального разделения, а перцептрон – достигает. Одно из отличий от других линейных методов состоит в том, что мы не можем получить замкнутую формулу для найденного перцептроном вектора весов, так что этот подход в большей степени эвристический.

Перцептрон легко превратить в линейную аппроксимирующую функцию (алгоритм 7.3). Для этой цели правило обновления заменяется на $\mathbf{w} = \mathbf{w} + (y_i - \hat{y}_i)^2 \mathbf{x}_i$ с использованием квадратов невязок. Маловероятно, что этот алгоритм сойдется к истинной функции, поэтому он просто останавливается после фиксированного количества эпох обучения (эпохой называется один полный просмотр обучающих данных). Или можно продолжать выполнение алгоритма, пока сумма квадратов невязок не станет меньше заданного порога.

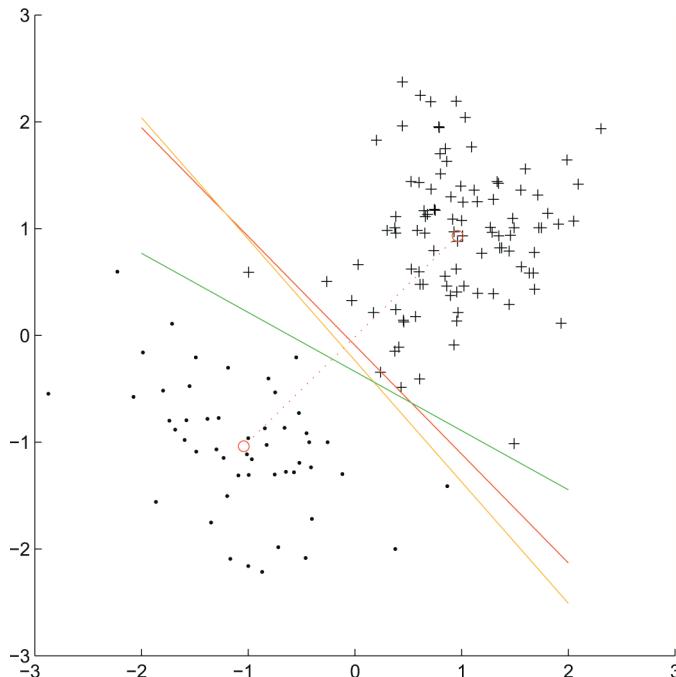


Рис. 7.6. Три линейных классификатора, по-разному обученных на наборе данных из 100 положительных (справа вверху) и 50 отрицательных (слева внизу) примеров: базовый линейный классификатор изображен **красным** цветом, классификатор по методу наименьших квадратов – **оранжевым**, а перцептрон – **зеленым**. Обратите внимание, что перцептрон идеально разделяет обучающие данные, но принятый в нем эвристический подход в некоторых ситуациях приводит к переобучению

Алгоритм 7.3. *PerceptronRegression(D, T)* – обучить перцептрон для регрессии

Вход: помеченные обучающие данные D в однородных координатах, максимальное число эпох обучения T .

Выход: вектор весов \mathbf{w} , определяющий линейную аппроксимирующую функцию
 $\hat{y} = \mathbf{w} \cdot \mathbf{x}$

```

1  $\mathbf{w} \leftarrow \mathbf{0}; t \leftarrow 0;$ 
2 while  $t < T$  do
3   for  $i = 1$  до  $|D|$  do
4      $\mathbf{w} \leftarrow \mathbf{w} + (y_i - \hat{y}_i)^2 \mathbf{x}_i;$ 
5   end
6    $t \leftarrow t + 1;$ 
7 end
```

7.3 Метод опорных векторов

Для линейно разделимых данных может существовать бесконечно много решающих границ, которые разделяют классы, но интуитивно одни из них кажутся лучше других. Например, решающие границы на рис. 7.5 слева и посередине выглядят слишком близко придинутыми к положительным примерам, а граница на рисунке справа, хотя и оставляет больше места с каждой стороны, тоже не кажется особенно хорошей. Чтобы придать точности этим интуитивным соображениям, вспомним, что в разделе 2.2 мы определили *зазор*, назначенный примеру оценивающим классификатором, как величину $c(x)\hat{s}(x)$, где $c(x)$ равно +1 для положительных и -1 для отрицательных примеров, а $\hat{s}(x)$ – оценка примера x . Если положить $\hat{s}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - t$, то у истинно положительного примера \mathbf{x}_i будет зазор $\mathbf{w} \cdot \mathbf{x}_i - t > 0$, а у истинно отрицательного \mathbf{x}_j – зазор $-(\mathbf{w} \cdot \mathbf{x}_j - t) > 0$. Для данного обучающего набора и решающей границы пусть m^+ – наименьший зазор, посчитанный по положительным примерам, а m^- – наименьший зазор по отрицательным примерам. Тогда наша цель – сделать сумму этих величин максимальной. Эта сумма не зависит от порога принятия решения t , коль скоро ближайшие положительные и отрицательные примеры располагаются по нужные стороны от решающей границы, поэтому мы скорректируем t , так чтобы m^+ и m^- стали равны. На рис. 7.7 это изображено графически в двухмерном пространстве объектов. Обучающие примеры, расположенные ближе всего к решающей границе, называются *опорными векторами*; как мы вскоре увидим, решающая граница в методе опорных векторов (SVM) определяется в виде линейной комбинации опорных векторов.

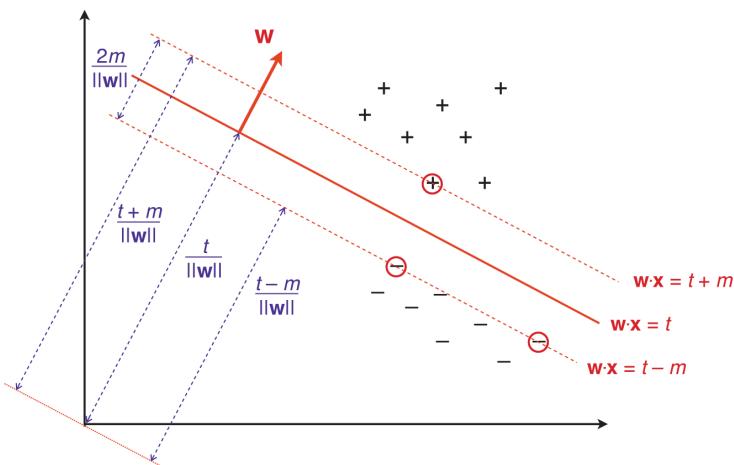


Рис. 7.7. Геометрия классификатора по методу опорных векторов. Кружочками обведены примеры, являющиеся опорными векторами, то есть ближайшие к решающей границе. Метод опорных векторов находит решающую границу, которая обращает в максимум величину $m/\|\mathbf{w}\|$.

Таким образом, зазор определяется как $m/\|\mathbf{w}\|$, где m – расстояние между решающей границей и ближайшими к ней обучающими примерами (по меньшей мере, по одному из каждого класса), измеренное вдоль вектора \mathbf{w} . Поскольку мы вправе изменять масштаб t , $\|\mathbf{w}\|$ и m , то принято выбирать $m = 1$. Тогда максимизация зазора соответствует минимизации $\|\mathbf{w}\|$, или, что более удобно, $\frac{1}{2}\|\mathbf{w}\|^2$, при условии, конечно, что ни один из обучающих примеров не попадает внутрь зазора. Это приводит к задаче квадратичной оптимизации с ограничениями:

$$\mathbf{w}^*, t^* = \underset{\mathbf{w}, t}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{w}\|^2 \text{ при условии } y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1, 1 \leq i \leq n.$$

Наш подход к решению этой задачи будет основан на множителях Лагранжа (см. замечание 7.3). Прибавление ограничений для всех обучающих примеров, помноженных на множители α_i , дает функцию Лагранжа:

$$\begin{aligned} \Lambda(\mathbf{w}, t, \alpha_1, \dots, \alpha_n) &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x}_i - t) - 1) = \\ &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i y_i (\mathbf{w} \cdot \mathbf{x}_i) + \sum_{i=1}^n \alpha_i y_i t + \sum_{i=1}^n \alpha_i = \\ &= \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \mathbf{w} \cdot \left(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \right) + t \left(\sum_{i=1}^n \alpha_i y_i \right) + \sum_{i=1}^n \alpha_i. \end{aligned}$$

Хотя выглядит эта формула пугающе, несложный анализ позволит вывести более простую двойственную форму функции Лагранжа. Взяв частную производную от функции Лагранжа по t и положив ее равной 0, мы найдем, что для оптимального порога t должно быть $\sum_{i=1}^n \alpha_i y_i = 0$. Аналогично, взяв частную производную по \mathbf{w} , мы увидим, что множители Лагранжа определяют вектор весов в виде линейной комбинации обучающих примеров:

$$\frac{\partial}{\partial \mathbf{w}} \Lambda(\mathbf{w}, t, \alpha_1, \dots, \alpha_n) = \frac{\partial}{\partial \mathbf{w}} \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \frac{\partial}{\partial \mathbf{w}} \mathbf{w} \cdot \left(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \right) = \mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i.$$

Оптимизация – широкий термин, обозначающий задачу нахождения наилучшего объекта или значения во множестве альтернатив. Мы уже видели очень простую оптимизацию без ограничений в примере 7.1, где требовалось найти значения a и b , минимизирующие сумму квадратов невязок $f(a, b) = \sum_{i=1}^n (w_i - (a + bh_i))^2$; для этой формулировки применяются такие обозначения:

$$a^*, b^* = \underset{a, b}{\operatorname{argmin}} f(a, b).$$

f называется *целевой функцией*; она может быть линейной, квадратичной (как в этом случае) или более сложной. Мы нашли минимум f , приравняв к нулю частные производные f по a и b и решив получившуюся систему уравнений относительно a и b ; вектор, составленный из частных производных, называется *градиентом* и обозначается ∇f , поэтому лаконичный способ сформулировать задачу оптимизации без ограничений выглядит так: найти a и b – такие, что $\nabla f(a, b) = \mathbf{0}$.

В данном частном случае целевая функция *вытукла*, что по существу означает наличие у нее единственного глобального минимума. Однако это необязательно.

В задаче *оптимизации с ограничениями* на альтернативы налагаются ограничения, например:

$$a^*, b^* = \underset{a,b}{\operatorname{argmin}} f(a,b) \quad \text{при условии } g(a,b) = c.$$

Если связь, выраженная ограничением, линейна, например $a - b = 0$, то, конечно, можно исключить одну из переменных и решить более простую задачу без ограничений. Однако в случае нелинейных ограничений это не всегда возможно. *Множители Лагранжа* – эффективный способ, пригодный в общем случае. Мы определяем функцию Лагранжа:

$$\Lambda(a,b,\lambda) = f(a,b) - \lambda(g(a,b) - c),$$

где λ – множитель Лагранжа, и решаем задачу без ограничений $\nabla \Lambda(a,b,\lambda) = \mathbf{0}$. Поскольку $\nabla_{ab} \Lambda(a,b,\lambda) = \nabla f(a,b) - \lambda \nabla g(a,b)$ и $\nabla_\lambda \Lambda(a,b,\lambda) = g(a,b) - c$, то это лаконичный способ выразить требование, что (i) градиенты f и g однодimensionalны и (ii) ограничение удовлетворяется. Мы можем включить несколько ограничений в виде равенств и неравенств, каждое со своим множителем Лагранжа.

Из функции Лагранжа можно вывести *двойственную* задачу оптимизации, когда мы ищем оптимальные значения множителей Лагранжа. В общем случае решение двойственной задачи – это лишь нижняя граница решения *основной* задачи, но при некоторых условиях, известных как условия *Каруша-Куна-Таккера* (*KKT*), оба решения оказываются одинаковы. Задача квадратичной оптимизации, возникающая в методе опорных векторов, обычно решается в двойственной форме.

Замечание 7.3. Основные понятия и термины математической оптимизации

Поскольку эта частная производная обращается в 0 для оптимального вектора весов, мы можем заключить, что $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$ – то же самое выражение, которое мы вывели для перцептрона в уравнении 7.9. Для перцептрона веса объектов α_i – неотрицательные целые числа, означающие, сколько раз пример был неправильно классифицирован в ходе обучения. В методе опорных векторов α_i – неотрицательные вещественные числа. Общее между ними то, что если $\alpha_i = 0$ для некоторого примера \mathbf{x}_i , то удаление этого примера из обучающего набора не влияет на обученную решающую границу. В случае метода опорных векторов это означает, что $\alpha_i > 0$ только для опорных векторов: обучающих примеров, близайших к решающей границе.

Теперь, подставляя выражения $\sum_{i=1}^n \alpha_i y_i = 0$ и $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$ обратно в лагранжиан, мы сможем исключить \mathbf{w} и t , а значит, получить двойственную задачу оптимизации, сформулированную исключительно в терминах множителей Лагранжа:

$$\begin{aligned} \Lambda(\alpha_1, \dots, \alpha_n) &= -\frac{1}{2} \left(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \right) \cdot \left(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \right) + \sum_{i=1}^n \alpha_i = \\ &= -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^n \alpha_i. \end{aligned}$$

Двойственная задача заключается в максимизации этой функции при ограничениях положительности и одном ограничении в виде равенства:

$$\alpha_1^*, \dots, \alpha_n^* = \arg \max_{\alpha_1, \dots, \alpha_n} -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^n \alpha_i$$

$$\text{при условии } \alpha_i \geq 0, 1 \leq i \leq n \text{ и } \sum_{i=1}^n \alpha_i y_i = 0.$$

Двойственная форма задачи оптимизации для метода опорных векторов иллюстрирует два важных положения. Во-первых, она показывает, что поиск решающей границы с максимальным зазором эквивалентен поиску опорных векторов: это обучающие примеры с ненулевыми множителями Лагранжа, и с помощью выражения $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$ они полностью определяют решающую границу. Во-вторых, она показывает, что задача оптимизации полностью определяется попарными скалярными произведениями обучающих примеров – элементами матрицы Грама. В разделе 7.5 мы увидим, что это наблюдение – прямая дорога к эффективной модификации метода опорных векторов, благодаря которой его можно применять и нелинейным решающим границам.

Следующий пример добавляет конкретики за счет подробных вычислений на простеньком наборе данных.

Пример 7.5 (два классификатора с максимальным зазором и их опорные векторы). Пусть при-
меры и метки заданы следующим образом (рис. 7.8 слева):

$$\mathbf{X} = \begin{pmatrix} 1 & 2 \\ -1 & 2 \\ -1 & -2 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} -1 \\ -1 \\ +1 \end{pmatrix} \quad \mathbf{X}' = \begin{pmatrix} -1 & -2 \\ 1 & -2 \\ -1 & -2 \end{pmatrix}.$$

Матрица \mathbf{X}' справа описывает метки классов, то есть ее строки имеют вид $y_i \mathbf{x}_i$. Матрица Грама в этом случае такова (без меток классов и с ними):

$$\mathbf{X} \mathbf{X}^T = \begin{pmatrix} 5 & 3 & -5 \\ 3 & 5 & -3 \\ -5 & -3 & 5 \end{pmatrix} \quad \mathbf{X}' \mathbf{X}'^T = \begin{pmatrix} 5 & 3 & 5 \\ 3 & 5 & 3 \\ 5 & 3 & 5 \end{pmatrix}.$$

Таким образом, двойственная задача оптимизации записывается в виде:

$$\begin{aligned} \arg \max_{\alpha_1, \alpha_2, \alpha_3} & -\frac{1}{2} (5\alpha_1^2 + 3\alpha_1\alpha_2 + 5\alpha_1\alpha_3 + 3\alpha_2\alpha_1 + 5\alpha_2^2 + 3\alpha_2\alpha_3 + 5\alpha_3\alpha_1 + 3\alpha_3\alpha_2 + 5\alpha_3^2) + \alpha_1 + \alpha_2 + \alpha_3 = \\ & = \arg \max_{\alpha_1, \alpha_2, \alpha_3} -\frac{1}{2} (5\alpha_1^2 + 6\alpha_1\alpha_2 + 10\alpha_1\alpha_3 + 5\alpha_2^2 + 6\alpha_2\alpha_3 + 5\alpha_3^2) + \alpha_1 + \alpha_2 + \alpha_3 \end{aligned}$$

при ограничениях $\alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_3 \geq 0$ и $-\alpha_1 - \alpha_2 + \alpha_3 = 0$. Хотя на практике такие задачи решаются специализированными программами квадратичной оптимизации, здесь мы продемонстрируем решение вручную.

Применив ограничение в виде равенства, мы можем исключить одну из переменных, например α_3 , и упростить целевую функцию, приведя ее к виду:

$$\begin{aligned} \arg \max_{\alpha_1, \alpha_2, \alpha_3} & -\frac{1}{2} (5\alpha_1^2 + 6\alpha_1\alpha_2 + 10\alpha_1(\alpha_1 + \alpha_2) + 5\alpha_2^2 + 6\alpha_2(\alpha_1 + \alpha_2) + 5(\alpha_1 + \alpha_2)^2) + \\ & + 2\alpha_1 + 2\alpha_2 = \arg \max_{\alpha_1, \alpha_2, \alpha_3} -\frac{1}{2} (20\alpha_1^2 + 32\alpha_1\alpha_2 + 16\alpha_2^2) + 2\alpha_1 + 2\alpha_2. \end{aligned}$$

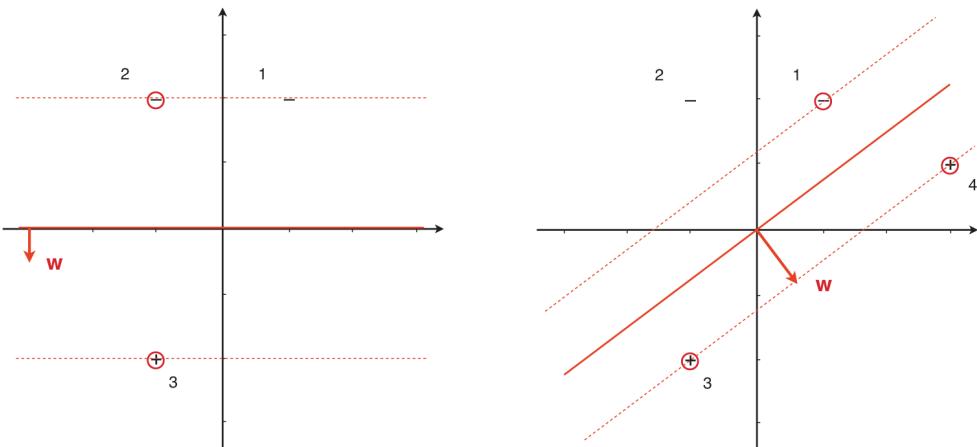


Рис. 7.8. (Слева) Классификатор с максимальным зазором, построенный по трем примерам, в котором $\mathbf{w} = (0, -1/2)$ и зазор равен 2. Кружочками обведены примеры, являющиеся опорными векторами, они определяют решающую границу, и множители Лагранжа для них отличны от нуля. **(Справа)** После добавления второго положительного примера решающая граница повернулась, вектор \mathbf{w} стал равен $(3/5, -4/5)$, а зазор уменьшился до 1

Приравнивая частные производные к 0, получаем $-20\alpha_1 - 16\alpha_2 + 2 = 0$ и $-16\alpha_1 - 16\alpha_2 + 2 = 0$ (отметим, что, поскольку целевая функция квадратичная, эти уравнения гарантированно линейны). Отсюда находим решение $\alpha_1 = 0$ и $\alpha_2 = \alpha_3 = 1/8$. Тогда $\mathbf{w} = 1/8(\mathbf{x}_3 - \mathbf{x}_2) = \begin{pmatrix} 0 \\ -1/2 \end{pmatrix}$ и зазор

равен $1/\|\mathbf{w}\| = 2$. Наконец, t можно найти по любому опорному вектору, например \mathbf{x}_2 , потому что $y_2(\mathbf{w} \cdot \mathbf{x}_2 - t) = 1$; отсюда получаем $-1 \cdot (-1 - t) = 1$, и, значит, $t = 0$. Получающийся классификатор с максимальным зазором изображен на рис. 7.8 слева. Отметим, что первый пример \mathbf{x}_1 не является опорным вектором, хотя он и лежит на границе зазора, и связано это с тем, что его удаление не повлияет на решающую границу.

Теперь добавим еще один положительный пример: $(3, 1)$. Тогда получатся такие матрицы:

$$\mathbf{X}' = \begin{pmatrix} -1 & -2 \\ 1 & -2 \\ -1 & -2 \\ 3 & 1 \end{pmatrix} \quad \mathbf{X}' \mathbf{X}'^T = \begin{pmatrix} 5 & 3 & 5 & -5 \\ 3 & 5 & 3 & 1 \\ 5 & 3 & 5 & -5 \\ -5 & 1 & -5 & 10 \end{pmatrix}.$$

С помощью аналогичных вычислений можно проверить, что зазор уменьшается до 1, а решающая граница поворачивается, так что вектор $\mathbf{w} = \begin{pmatrix} 3/5 \\ -4/5 \end{pmatrix}$ (рис. 7.8 справа). Множители Лагранжа теперь равны $\alpha_1 = 1/2$, $\alpha_2 = 0$, $\alpha_3 = 1/10$, $\alpha_4 = 2/5$. Таким образом, только \mathbf{x}_3 является опорным вектором в исходном и расширенном наборах данных.

Метод опорных векторов с мягким зазором

Если данные не являются линейно разделимыми, то ограничения $\mathbf{w} \cdot \mathbf{x}_i - t \geq 1$, налагаемые примерами, невозможно удовлетворить совместно. Однако существует весьма элегантный способ видоизменить задачу оптимизации так, что она будет допускать решение даже в этом случае. Идея заключается во введении *ослабляющих переменных* ξ_i , по одной для каждого примера, благодаря которым некоторые примеры могут оказаться внутри зазора или даже с неправильной стороны от решающей границы, — мы будем называть это *ошибками зазора*. Таким образом, мы приводим ограничения к виду $\mathbf{w} \cdot \mathbf{x}_i - t \geq 1 - \xi_i$ и добавляем сумму всех ослабляющих переменных к минимизируемой целевой функции, получая в результате следующую задачу оптимизации с *мягким зазором*:

$$\mathbf{w}^*, t^*, \xi_i^* = \arg \min_{\mathbf{w}, t, \xi_i} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad (7.11)$$

при условии $y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1 - \xi_i$ и $\xi_i \geq 0, 1 \leq i \leq n$.

Здесь C – определенный пользователем параметр, определяющий компромисс между максимизацией зазора и минимизацией ослабляющей переменной: большое значение C означает, что штраф за ошибки зазора велик, а при малом значении допускается больше ошибок зазора (в том числе, возможно, неправильных классификаций) с целью увеличить величину зазора. Если мы допускаем больше ошибок зазора, то понадобится меньше опорных векторов, поэтому C в некотором роде управляет «сложностью» SVM, отсюда и часто употребляемое название *параметр сложности*. Эту идею можно рассматривать как одну из форм регуляризации, которая обсуждалась выше в контексте регрессии методом наименьших квадратов.

Функция Лагранжа в этом случае имеет вид:

$$\begin{aligned} \Lambda(\mathbf{w}, t, \xi_i, \alpha_i, \beta_i) &= \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x}_i - t) - (1 - \xi_i)) - \sum_{i=1}^n \beta_i \xi_i = \\ &= \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \mathbf{w} \cdot \left(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \right) + t \left(\sum_{i=1}^n \alpha_i y_i \right) + \sum_{i=1}^n \alpha_i + \sum_{i=1}^n (C - \alpha_i - \beta_i) \xi_i = \\ &= \Lambda(\mathbf{w}, t, \alpha_i) + \sum_{i=1}^n (C - \alpha_i - \beta_i) \xi_i. \end{aligned}$$

Для оптимального решения все частные производные по ξ_i должны быть равны 0, откуда следует, что $C - \alpha_i - \beta_i = 0$ для всех i , а значит, в двойственной задаче добавленный член исчезает. Далее, так как α_i и β_i положительны, то α_i не может быть больше C , и это выливается в дополнительное ограничение сверху на α_i в двойственной задаче:

$$\alpha_1^*, \dots, \alpha_n^* = \arg \max_{\alpha_1, \dots, \alpha_n} -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^n \alpha_i$$

(7.12)

при условии $0 \leq \alpha_i \leq C$ и $\sum_{i=1}^n \alpha_i y_i = 0$.

Это удивительный и красивый результат. Он следует из специального способа добавления ослабляющих переменных в выражении (7.11). Поскольку на ослабляющие переменные наложено ограничение положительности и они добавлены в минимизируемую целевую функцию в качестве слагаемых, то они действуют как штрафные члены, измеряя только отклонения с неправильной стороны зазора. Далее отсутствие множителей β_i в двойственной целевой функции вытекает из того факта, что штрафной член в основной целевой функции линейно зависит от ξ_i . По существу, эти ослабляющие переменные реализуют то, что на рис. 2.6 (стр. 77) было названо *кусочно-линейной функцией потерь*: зазор $z > 1$ не влечет никакого штрафа, а зазор $z = 1 - \xi \leq 1$ влечет штраф $\xi = 1 - z$.

Какое влияние верхняя граница C оказывает на множители α_i ? Поскольку $C - \alpha_i - \beta_i = 0$ для всех i , то из $\alpha_i = C$ следует, что $\beta_i = 0$. Множители β_i получаются из ограничения $\xi_i \geq 0$, и обращение множителя в 0 означает, что нижняя граница не достигается, то есть $\xi_i > 0$ (по аналогии с тем фактом, что условие $\alpha_i = 0$ означает, что \mathbf{x}_j – не опорный вектор и, следовательно, $\mathbf{w} \cdot \mathbf{x}_j - t > 1$). Иными словами, решение задачи оптимизации с мягким зазором в двойственной форме разделяет обучающие примеры на три множества:

$\alpha_i = 0$ те, что лежат вне или на границе зазора;

$0 < \alpha_i < C$ опорные векторы на границе зазора;

$\alpha_i = C$ внутри или на границе зазора.

Отметим, что мы по-прежнему имеем $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$ и что во втором и третьем случаях примеры участвуют в порождении решающей границы.

Пример 7.6 (мягкие зазоры). Продолжим пример 7.5, в котором, как мы видели, добавление положительного примера $\mathbf{x}_4 = (3, 1)$ к первым трем существенно уменьшило зазор – с 2 до 1. Теперь мы покажем, что при достаточно больших значениях параметра сложности C обучаются классификаторы с мягким зазором, в которых величина зазора больше. Напомним, множители Лагранжа для классификатора на рис. 7.8 справа: $\alpha_1 = 1/2$, $\alpha_2 = 0$, $\alpha_3 = 1/10$, $\alpha_4 = 2/5$. Следовательно, α_1 – самый большой множитель, и при условии $C > \alpha_1 = 1/2$ никакие ошибки зазора недопустимы. Для $C = 1/2$ имеем $\alpha_1 = C$, и, значит, при $C < 1/2$ \mathbf{x}_1 становится ошибкой зазора, а оптимальным является классификатор с мягким зазором. При уменьшении C решающая граница и верхний зазор смещаются вверх, а нижний зазор остается на месте.

Верхняя граница достигает \mathbf{x}_2 при $C = 5/16$ (рис. 7.9 слева), и в этой точке мы имеем $\mathbf{w} = \begin{pmatrix} 3/8 \\ -1/2 \end{pmatrix}$,

$t = 3/8$, а зазор увеличился до 1.6. При этом $\xi_1 = 6/8$, $\alpha_1 = C = 5/16$, $\alpha_2 = 0$, $\alpha_3 = 1/16$ и $\alpha_4 = 1/4$. При дальнейшем уменьшении C решающая граница начинает поворачиваться по часовой стрелке, так что \mathbf{x}_4 также становится ошибкой зазора, и лишь \mathbf{x}_2 и \mathbf{x}_3 остаются опорными векторами.

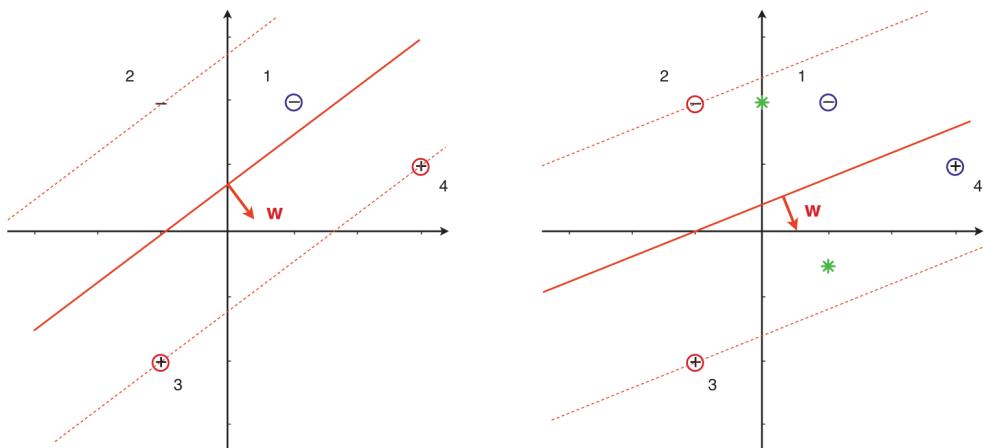


Рис. 7.9. (Слева) Классификатор с мягким зазором, обученный при $C = 5/16$, в котором точка \mathbf{x}_2 готова стать опорным вектором. **(Справа)** Классификатор с мягким зазором, обученный при $C = 1/10$: все примеры дают равный вклад в вектор весов. Звездочки обозначают средние значения классов, а решающая граница параллельна той, что получилась при обучении базового линейного классификатора

Граница поворачивается, пока C не станет равным $1/10$, в этой точке мы имеем $\mathbf{w} = \begin{pmatrix} 1/5 \\ -1/2 \end{pmatrix}$, $t = 1/5$, а зазор увеличился до 1.86. Кроме того, $\xi_1 = 4/10$, $\xi_4 = 7/10$, а все множители стали равны C (рис. 7.9 справа).

Наконец, если продолжить уменьшение C , то решающая граница остается неподвижной, но норма вектора весов постепенно уменьшается, и все точки становятся ошибками зазора.

Пример 7.6 иллюстрирует важное положение: для достаточно малых C все примеры получают один и тот же множитель C , а потом $\mathbf{w} = C \sum_{i=1}^n y_i \mathbf{x}_i = C(\text{Pos} \cdot \mu^\oplus - \text{Neg} \cdot \mu^\ominus)$, где μ^\oplus и μ^\ominus – средние положительных и отрицательных примеров соответственно. Иными словами, *классификатор минимальной сложности с мягким зазором сводит классы к их средним – почти так же, как в случае базового линейного классификатора*. Для промежуточных значений C решающая граница натянута на опорные векторы и средние ошибки зазора для каждого класса.

Подведем итог: машины опорных векторов являются линейными классификаторами, которые строят решающую границу, максимизирующую расстояние до ближайших обучающих примеров (опорных векторов). Параметр сложности C позволяет регулировать количество и серьезность допустимых нарушений зазора. Обучение SVM сводится к решению большой задачи квадратичной оптимизации, нахождение которого лучше поручить специализированной программе.

7.4 Получение вероятностей от линейных классификаторов

Как мы видели, линейный классификатор порождает оценки $\hat{s}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i - t$ – такие, что пороговое значение 0 позволяет классифицировать примеры. Благодаря геометрической природе линейных классификаторов эти оценки можно использовать для получения расстояния (со знаком) от \mathbf{x}_i до решающей границы. Чтобы понять, почему это так, заметим, что длина проекции \mathbf{x}_i на \mathbf{w} равна $\|\mathbf{x}_i\| \cos \theta$, где θ – угол между \mathbf{x}_i и \mathbf{w} . Поскольку $\mathbf{w} \cdot \mathbf{x}_i = \|\mathbf{w}\| \|\mathbf{x}_i\| \cos \theta$, то мы можем записать эту длину в виде $(\mathbf{w} \cdot \mathbf{x}_i) / \|\mathbf{w}\|$. Отсюда получается такое расстояние со знаком:

$$d(\mathbf{x}_i) = \frac{\hat{s}(\mathbf{x}_i)}{\|\mathbf{w}\|} = \frac{\mathbf{w} \cdot \mathbf{x}_i - t}{\|\mathbf{w}\|} = \mathbf{w}' \cdot \mathbf{x}_i - t',$$

где вектор $\mathbf{w}' = \mathbf{w} / \|\mathbf{w}\|$ нормирован на единичную длину, а $t' = t / \|\mathbf{w}\|$ – соответственно масштабированный свободный член. Знак этой величины говорит, с какой стороны от решающей границы мы находимся: положительное расстояние означает, что точка находится с «положительной» стороны границы (в направлении вектора \mathbf{w}), а отрицательное – что с другой стороны (рис. 7.10).

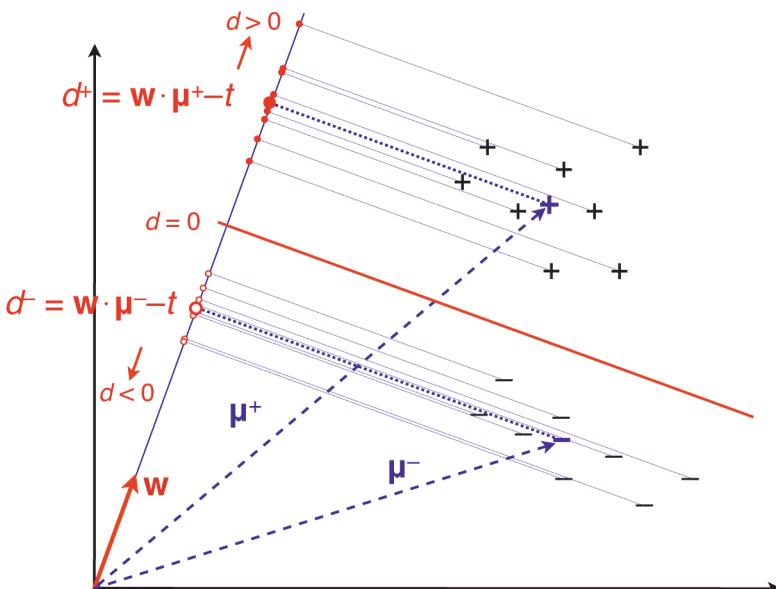


Рис. 7.10. Линейный классификатор можно интерпретировать как проекцию на направление, заданное вектором \mathbf{w} , который здесь предполагается единичным. Величина $\mathbf{w} \cdot \mathbf{x} - t$ – это расстояние со знаком до пересечения решающей границы с линией проекции. Показаны также средние векторы классов μ^+ и μ^- и соответствующие средние расстояния d^+ и d^-

Геометрическая интерпретация оценок, порождаемых линейными классификаторами, предоставляет интересную возможность для превращения их в вероятности, этот процесс в разделе 2.3 был назван *калибровкой*. Пусть \bar{d}^{\oplus} – среднее расстояние от положительных примеров до решающей границы, то есть $\bar{d}^{\oplus} = \mathbf{w} \cdot \mu^{\oplus} - t$, где μ^{\oplus} – средний вектор положительных примеров, а \mathbf{w} – единичный вектор (хотя последнее предположение не является строго необходимым, поскольку вектор весов, как мы увидим, будет масштабирован). Было бы разумно ожидать, что расстояния от положительных примеров до решающей границы имеют нормальное распределение вокруг этого среднего¹: то есть если изобразить гистограмму этих расстояний, то должна получиться знакомая колоколообразная кривая. В этом предположении функция плотности вероятности d имеет вид

$$P(d|\oplus) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(d-\bar{d}^{\oplus})^2}{2\sigma^2}\right) \quad (\text{о нормальном распределении см. замечание 9.1}$$

на стр. 278). Аналогично можно ожидать, что расстояния от отрицательных примеров до решающей границы будут нормально распределены вокруг $\bar{d}^{\ominus} = \mathbf{w} \cdot \mu^{\ominus} - t$, причем $\bar{d}^{\ominus} < 0 < \bar{d}^{\oplus}$. Мы будем предполагать, что оба нормальных распределения имеют одинаковую дисперсию σ^2 .

Теперь предположим, что мы наблюдаем точку \mathbf{x} с расстоянием $d(\mathbf{x})$. Мы классифицируем эту точку как положительную, если $d(\mathbf{x}) > 0$, и как отрицательную, если $d(\mathbf{x}) < 0$, однако хотелось бы связать с этими предсказаниями вероятность $\hat{P}(\mathbf{x}) = P(\oplus|d(\mathbf{x}))$. Согласно формуле Байеса:

$$P(\oplus|d(\mathbf{x})) = \frac{P(d(\mathbf{x})|\oplus)P(\oplus)}{P(d(\mathbf{x})|\oplus)P(\oplus) + P(d(\mathbf{x})|\ominus)P(\ominus)} = \frac{LR}{LR + 1/clr},$$

где LR – отношение правдоподобия, полученное из нормальных распределений оценок, а clr – отношение классов. Для простоты будем далее считать, что $clr = 1$. Кроме того, предположим временно, что $\sigma^2 = 1$ и что $\bar{d}^{\oplus} = -\bar{d}^{\ominus} = 1/2$ (скоро мы ослабим эти предположения). Тогда имеем

$$\begin{aligned} LR &= \frac{P(d(\mathbf{x})|\oplus)}{P(d(\mathbf{x})|\ominus)} = \frac{\exp(-d(\mathbf{x})-1/2)^2/2)}{\exp(-d(\mathbf{x})+1/2)^2/2)} = \\ &= \exp(-d(d(\mathbf{x})-1/2)^2/2 + (d(\mathbf{x})+1/2)^2/2) = \exp(d(\mathbf{x})), \end{aligned}$$

и, следовательно,

$$P(\oplus|d(\mathbf{x})) = \frac{\exp(d(\mathbf{x}))}{\exp(d(\mathbf{x}))+1} = \frac{\exp(\mathbf{w} \cdot \mathbf{x} - t)}{\exp(\mathbf{w} \cdot \mathbf{x} - t)+1}.$$

¹ Например, если число примеров достаточно велико, то это предположение можно обосновать с помощью *центральной предельной теоремы*: сумма большого количества одинаково распределенных независимых случайных величин имеет распределение, близкое к нормальному.

Таким образом, чтобы получить оценки вероятности от линейного классификатора, порождающего оценки расстояния d , мы преобразуем d в вероятность посредством отображения $d \mapsto \frac{\exp(d)}{\exp(d)+1}$ (или, эквивалентно) $d \mapsto \frac{1}{1+\exp(-d)}$. Эта S-образная, или *сигмоидная* функция, называется *логистической функцией*; она находит применения в самых разных областях (рис. 7.11). Предположим теперь, что, как и раньше, $\bar{d}^{\oplus} = -\bar{d}^{\ominus}$, но не будем делать никаких предположений о величине этих средних расстояний и величине σ^2 . В таком случае имеем

$$\begin{aligned} LR &= \exp\left(\frac{-(d(\mathbf{x}) - \bar{d}^{\oplus})^2 + (d(\mathbf{x}) - \bar{d}^{\ominus})^2}{2\sigma^2}\right) = \\ &= \exp\left(\frac{2\bar{d}^{\oplus}d(\mathbf{x}) - (\bar{d}^{\oplus})^2 - 2\bar{d}^{\ominus}d(\mathbf{x}) + (\bar{d}^{\ominus})^2}{2\sigma^2}\right) = \exp(\gamma d(\mathbf{x})), \end{aligned}$$

где $\gamma = (\bar{d}^{\oplus} - \bar{d}^{\ominus})/\sigma^2$ – масштабный коэффициент, который нормирует вектор весов, так что средние расстояния для каждого класса отделены друг от друга на единицу дисперсии. Другими словами, введя в рассмотрение масштабный коэффициент γ , мы можем опустить предположение о том, что \mathbf{w} – единичный вектор.

Если отказаться также от предположения о том, что \bar{d}^{\oplus} и $-\bar{d}^{\ominus}$ симметричны относительно решающей границы, то получится самая общая форма:

$$\begin{aligned} LR &= \frac{P(d(\mathbf{x}) | \oplus)}{P(d(\mathbf{x}) | \ominus)} = \exp(\gamma(d(\mathbf{x}) - d_0)); \\ \gamma &= \frac{\bar{d}^{\oplus} - \bar{d}^{\ominus}}{\sigma^2} = \frac{\mathbf{w} \cdot (\mu^{\oplus} - \mu^{\ominus})}{\sigma^2}; \quad d_0 = \frac{\bar{d}^{\oplus} + \bar{d}^{\ominus}}{2} = \frac{\mathbf{w} \cdot (\mu^{\oplus} + \mu^{\ominus})}{2} - t. \end{aligned} \tag{7.13}$$

Эффект d_0 проявляется в перемещении решающей границы от $\mathbf{w} \cdot \mathbf{x} = t$ до $\mathbf{x} = (\mu^{\oplus} + \mu^{\ominus})/2$, то есть в положение, равноотстоящее от средних по обеим классам. Таким образом, логистическое отображение принимает вид $d \mapsto \frac{1}{1+\exp(-\gamma(d-d_0))}$, а влияние обоих параметров наглядно показано на рис. 7.11.

Пример 7.7 (логистическая калибровка линейного классификатора). Логистическая калибровка принимает особенно простой вид для базового линейного классификатора, для которого $\mathbf{w} = \mu^{\oplus} - \mu^{\ominus}$. Отсюда следует, что

$$\bar{d}^{\oplus} - \bar{d}^{\ominus} = \frac{\mathbf{w} \cdot (\mu^{\oplus} - \mu^{\ominus})}{\|\mathbf{w}\|} = \frac{\|\mu^{\oplus} - \mu^{\ominus}\|^2}{\|\mu^{\oplus} - \mu^{\ominus}\|} = \|\mu^{\oplus} - \mu^{\ominus}\|,$$

и, значит, $\gamma = \|\mu^{\oplus} - \mu^{\ominus}\|/\sigma^2$. Далее $d_0 = 0$, поскольку $(\mu^{\oplus} + \mu^{\ominus})/2$ уже лежит на решающей границе. Поэтому в данном случае логистическая калибровка не сдвигает решающую границу, а лишь изменяет крутизну сигмоида в зависимости от разделенности классов. На рис. 7.12 это проиллюстрировано для двух выборок данных с разными нормальными распределениями, но с одной и той же диагональной матрицей ковариации.

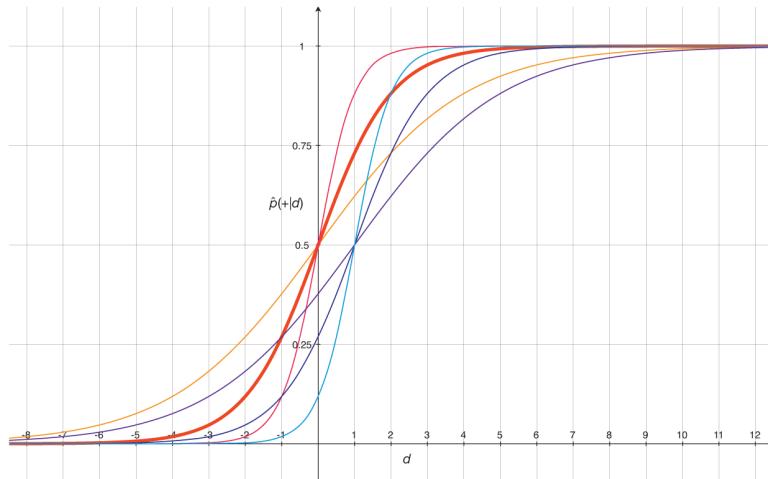


Рис. 7.11. Логистическая функция полезна для отображения расстояний до линейной решающей границы на оценку апостериорной вероятности положительного класса. Жирной красной линией изображена стандартная логистическая функция $\hat{p}(d) = \frac{1}{1 + \exp(-d)}$; эту функцию можно использовать для получения оценок вероятности, если два класса встречаются одинаково часто и их средние отстоят на одно и то же расстояние от решающей границы и на одну единицу дисперсии друг от друга. Более крутая и более пологая красная кривые показывают, как изменяется эта функция, если средние классов отстоят друг от друга на 2 и 1/2 единицы дисперсии соответственно. Три синие кривые показывают, как выглядит функция, если $d_0 = 1$, то есть положительные примеры в среднем отстоят дальше от решающей границы

Подведем итог: чтобы получить калиброванные оценки вероятностей от линейного классификатора, мы сначала вычисляем средние расстояния \bar{d}^{\oplus} и $-\bar{d}^{\ominus}$ и дисперсию σ^2 , а по ним параметр сдвига d_0 и масштабный коэффициент γ . Отношение правдоподобия тогда равно $LR = \exp(\gamma(d(\mathbf{x}) - d_0)) = \exp(\gamma(\mathbf{w} \cdot \mathbf{x} - t - d_0))$. Поскольку логарифм отношения правдоподобия линеен относительно \mathbf{x} , то такие модели называются **логарифмически линейными** (или логлинейными). Отметим, что $\gamma(\mathbf{w} \cdot \mathbf{x} - t - d_0) = \mathbf{w}' \cdot \mathbf{x} - t'$, где $\mathbf{w}' = \gamma \mathbf{w}$ и $t' = \gamma(t + d_0)$. Это означает, что процедура логистической калибровки может изменить положение решающей границы, но не ее направление. Однако может существовать другой вектор весов с иным направлением, который назначает данным более высокое правдоподобие. Задача нахождения линейного классификатора с максимальным правдоподобием с помощью логистической модели называется **логистической регрессией** и рассматривается в разделе 9.3.

В качестве альтернативы логистической калибровке мы можем использовать также метод изотонной калибровки, рассмотренный в разделе 2.3. На рис. 7.12 слева показана кривая РХП базового линейного классификатора, обученного на данных с рис. 7.12, а также ее выпуклая оболочка. Мы можем далее построить

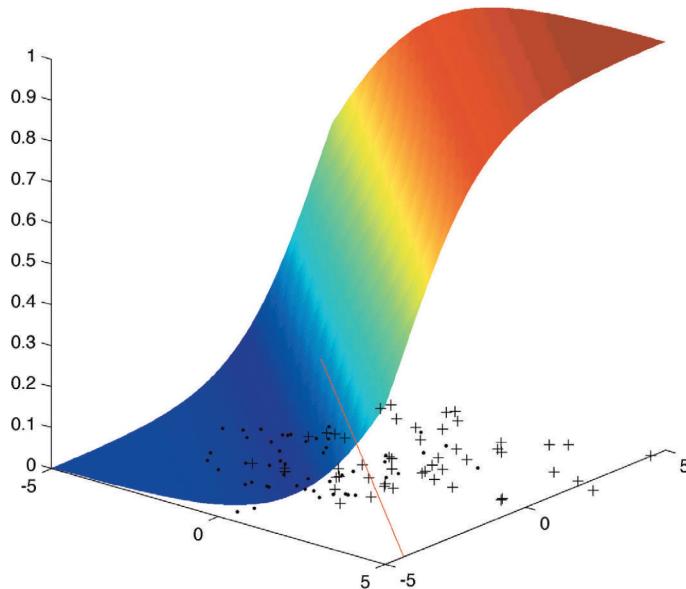


Рис. 7.12. Эта поверхность показывает сигмоидальные оценки вероятностей, получающиеся в результате логистической калибровки базового линейного классификатора на случайных данных, удовлетворяющих допущениям логистической калибровки

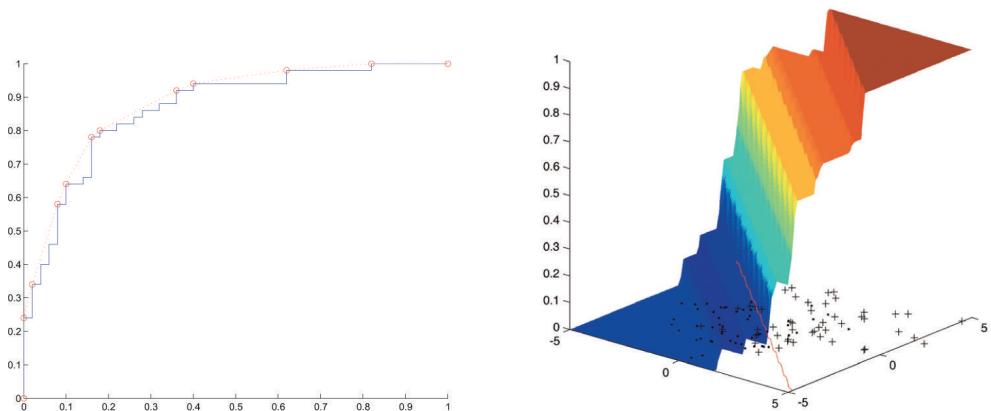


Рис. 7.13. (Слева) Кривая RXH и выпуклая оболочка для той же модели и данных, что на рис. 7.12. **(Справа)** Выпуклую оболочку можно использовать в качестве непараметрического метода калибровки. Каждый сегмент выпуклой оболочки соответствует плато на поверхности вероятностей

кусочно-линейную калибровочную функцию, горизонтальные участки которой соответствуют отрезкам выпуклой оболочки, как показано на рис. 7.13 справа. В отличие от логистического, этот метод калибровки непараметрический, и по-

тому в нем не делается никаких предположений о данных. Чтобы избежать переобучения, непараметрические методы обычно нуждаются в большем объеме данных, чем параметрические. Интересно отметить отсутствие какого-либо ранжирования на горизонтальных участках, которые в чем-то похожи на сегменты в группирующей модели. Иными словами, калибровка выпуклой оболочки потенциально может порождать гибридные ранжирующие-группирующие модели.

7.5 За пределами линейности – ядерные методы

В этой главе мы рассмотрели линейные методы классификации и регрессии. Начав с применения метода наименьших квадратов для регрессии, мы видели, как приспособить его для выполнения бинарной классификации, что привело к варианту базового линейного классификатора, который учитывает корреляцию признаков путем построения матрицы $(\mathbf{X}^T \mathbf{X})^{-1}$ и чувствителен к асимметричному распределению по классам. Затем мы обсудили эвристический алгоритм обучения перцептрона для линейно разделимых данных и метод опорных векторов, который находит единственную решающую границу с максимальным зазором и обобщается на неразделимые данные. В этом разделе мы покажем, как эти методы можно модифицировать для нахождения нелинейных решающих границ. Основная идея проста (и уже исследовалась в примере 1.9 на стр. 55): выполнить нелинейное преобразование данных в пространство признаков, где можно будет применить линейную классификацию.

Пример 7.8 (нахождение квадратичной решающей границы). Данные на рис. 7.14 слева линейно неразделимы, но оба класса имеют отчетливую круговую форму. На рис. 7.14 справа показаны те же данные после возвведения значений признаков в квадрат. В таком преобразованном пространстве признаков данные стали линейно разделимыми, и перцептрон способен разделить классы. Получившаяся решающая граница в исходном пространстве близка к окружности. Также показана решающая граница, найденная базовым линейным классификатором в пространстве квадратичных признаков, в исходном пространстве ей соответствует эллипс.

В общем случае обратное отображение из пространства признаков в пространство объектов нетривиально. Например, в данном примере прообразами средней точки каждого класса в пространстве признаков являются четыре точки в пространстве объектов – в силу квадратичности отображения.

Принято называть преобразованное пространство *пространством признаков*, а исходное – *пространством входов*. Таким образом, идея заключается в том, чтобы как-то отобразить обучающие данные в пространство признаков и там обучить модель. Чтобы классифицировать новые данные, мы также отображаем их в пространство признаков и применяем к ним модель. Замечательно, однако,

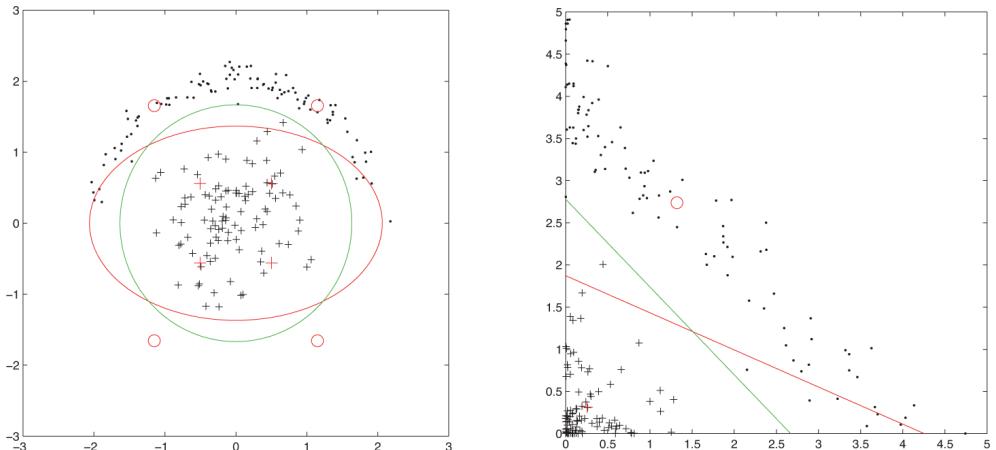


Рис. 7.14. (Слева) Решающие границы, найденные в результате обучения базового линейного классификатора и перцентрона с использованием квадратов признаков. **(Справа)** Данные и решающие границы в преобразованном пространстве признаков

то, что во многих случаях явно строить пространство признаков не нужно, поскольку все необходимые операции можно произвести в пространстве входов.

Возьмем, к примеру, алгоритм обучения перцентрона в двойственной форме (алгоритм 7.2). Это простой алгоритм подсчета – единственная сколько-нибудь сложная операция заключается в проверке правильности классификации примера \mathbf{x}_i путем вычисления величины $y_i \sum_{j=1}^{|D|} \alpha_j y_j \mathbf{x}_i \cdot \mathbf{x}_j$. При этом наиболее трудоемко вычисление скалярного произведения $\mathbf{x}_i \cdot \mathbf{x}_j$. В двумерном случае, когда примеры имеют вид $\mathbf{x}_i = (x_i, y_i)$ и $\mathbf{x}_j = (x_j, y_j)$, скалярное произведение можно записать как $\mathbf{x}_i \cdot \mathbf{x}_j = x_i x_j + y_i y_j$. Соответствующие объекты в пространстве квадратичных признаков: (x_i^2, y_i^2) и (x_j^2, y_j^2) , а их скалярное произведение равно:

$$(x_i^2, y_i^2) \cdot (x_j^2, y_j^2) = x_i^2 x_j^2 + y_i^2 y_j^2.$$

Это почти эквивалентно выражению

$$(\mathbf{x}_i, \mathbf{x}_j)^2 = (x_i x_j + y_i y_j)^2 = (x_i x_j)^2 + (y_i y_j)^2 + 2x_i x_j y_i y_j,$$

если не считать третьего члена – перекрестного произведения. Этот член можно учесть, добавив в вектор признаков третий признак $\sqrt{2}xy$. При этом получается такое пространство признаков:

$$\begin{aligned}\phi(\mathbf{x}_i) &= (x_i^2, y_i^2, \sqrt{2}x_i y_i); & \phi(\mathbf{x}_j) &= (x_j^2, y_j^2, \sqrt{2}x_j y_j); \\ \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) &= x_i^2 x_j^2 + y_i^2 y_j^2 + 2x_i x_j y_i y_j = (\mathbf{x}_i \cdot \mathbf{x}_j)^2.\end{aligned}$$

Определим теперь $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^2$ и заменим $\mathbf{x}_i \cdot \mathbf{x}_j$ на $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ в алгоритме обучения перцентрона в двойственной форме – в результате получим **ядерный**

перцептрон (алгоритм 7.4), который способен находить нелинейные решающие границы вида, показанного в примере 7.8.

Алгоритм 7.4. KernelPerceptron(D, κ) – алгоритм обучения перцептрана с использованием ядра

Вход: помеченные обучающие данные D в однородных координатах, **ядерная функция κ** .

Выход: коэффициенты α_i , определяющие нелинейную решающую границу.

```

1  $\alpha_i \leftarrow 0$  для  $1 \leq i \leq |D|$ ;
2 converged  $\leftarrow \text{false}$ ;
3 while converged = false do
4   converged  $\leftarrow \text{true}$ ;
5   for  $i = 1$  до  $|D|$  do
6     if  $y_i \sum_{j=1}^{|D|} \alpha_j y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \leq 0$  then
7        $\alpha_i \leftarrow \alpha_i + 1$ ;
8     converged  $\leftarrow \text{false}$ ;
9   end
10 end
11 end
```

Введение в рассмотрение ядер открывает целый спектр новых возможностей. Ясно, что можно определить полиномиальное ядро любой степени p в виде $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^p$. Оно преобразует d -мерное пространство входов в пространство признаков большей размерности таким образом, что каждый новый признак является произведением p членов (быть может, повторяющихся). Если включить еще константу, например $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^p$, то получим также все члены низших порядков. Так, если взять двумерное пространство входов и положить $p = 2$, то результирующим пространством признаков будет

$$\phi(\mathbf{x}) = (x^2, y^2, \sqrt{2}xy, \sqrt{2}x, \sqrt{2}y, 1),$$

содержащее как линейные, так и квадратичные признаки.

Но мы не обязаны ограничиваться только полиномиальными ядрами. Часто применяется *гауссово ядро* вида

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right), \quad (7.14)$$

в котором параметр σ называется *полосой пропускания*. Чтобы лучше понять смысл гауссова ядра, заметим, что $\kappa(\mathbf{x}, \mathbf{x}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}) = \|\phi(\mathbf{x})\|^2$ для любого ядра, обладающего рядом стандартных свойств, в совокупности называемых «положительной полуопределенностью». В данном случае мы имеем $\kappa(\mathbf{x}, \mathbf{x}) = 1$, то есть все точки $\phi(\mathbf{x})$ лежат на гиперсфере с центром в начале координат пространства признаков. Однако это пространство бесконечномерно, так что геометрические соображения тут не помогут. Полезнее считать, что гауссово ядро накладывает

гауссову поверхность (то есть многомерную нормальную поверхность, см. замечание 9.1 на стр. 278) на каждый опорный вектор в пространстве объектов, так что решающая граница определяется в терминах этих гауссовых поверхностей.

Ядерные методы чаще всего применяются в сочетании с методом опорных векторов. Отметим, что задача оптимизации с мягким зазором (уравнение 7.12) определена в терминах скалярных произведений между обучающими примерами, поэтому можно применить «трюк с ядром» (kernel trick):

$$\alpha_1^*, \dots, \alpha_n^* = \arg \max_{\alpha_1, \dots, \alpha_n} -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^n \alpha_i$$

при условии $0 \leq \alpha_i \leq C$ и $\sum_{i=1}^n \alpha_i y_i = 0$.

Следует помнить, что решающую границу, обученную с помощью нелинейного ядра, невозможно представить простым вектором весов в пространстве входов. Таким образом, чтобы классифицировать новый пример \mathbf{x} , мы должны вычислить выражение $y_i \sum_{j=1}^n \alpha_j y_j \kappa(\mathbf{x}, \mathbf{x}_j)$; это вычисление требует времени $O(n)$ и включает все обучающие примеры или, по крайней мере, те, для которых множители α_j отличны от нуля. Именно поэтому машины опорных векторов так часто применяются вместе с ядерными методами – ведь они естественным образом приводят к разреженным опорным векторам. И хотя мы здесь ограничились только числовыми признаками, стоит подчеркнуть, что ядра можно определить и над дискретными структурами, в том числе деревьями, графами и логическими формулами, а значит, открывается путь к обобщению геометрических моделей на нечисловые данные.

7.6 Линейные модели: итоги и литература для дальнейшего чтения

После изучения логических моделей в предыдущих трех главах мы в этой главе познакомились с линейными моделями. Логические модели по сути своей нечисловые, поэтому при работе с числовыми признаками они вынуждены использовать пороговые значения для преобразования их в два или более интервалов. Линейные модели почти диаметрально противоположны в том смысле, что могут работать с числовыми признаками непосредственно, а нечисловые должны подвергаться предобработке¹. Геометрически линейные модели строятся с помощью прямых и плоскостей, и по существу это означает, что определенное увеличение или уменьшение одного из признаков дает один и тот же эффект вне зависимости от значения этого признака или всех прочих признаков. Линейные модели

¹ Способы предварительной обработки нечисловых признаков для использования в линейных моделях обсуждаются в главе 10.

просты и устойчивы к изменению обучающих данных, но вследствие этого иногда страдают от недообучения.

- ☞ В разделе 7.1 мы рассмотрели метод наименьших квадратов, который первоначально был предложен для решения задачи регрессии. Этот классический метод, получивший название за минимизацию суммы квадратов невязок между предсказанными и истинными значениями функции, описан в бесчисленных вводных курсах по математике и техническим дисциплинам (и, как мне помнится, был одним из примеров, которые я запускал на программируемом калькуляторе Texas Instruments TI-58, принадлежащем отцу). Сначала мы рассмотрели одномерную задачу, а затем вывели общее решение в виде $\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$, где $(\mathbf{X}^T \mathbf{X})^{-1}$ – преобразование, которое устраивает корреляцию между признаками, центрирует и нормирует их. Затем мы обсудили регуляризованные варианты линейной регрессии: гребневую регрессию, впервые введенную в работе Hoerl, Kennard (1970), и предложенный в работе Tibshirani (1996) метод lasso, который естественно порождает разреженные решения. Мы видели, что метод наименьших квадратов можно применить к задаче бинарной классификации, закодировав классы числами +1 и -1, что приводит к решению $\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} (\text{Pos } \mu^\oplus - \text{Neg } \mu^\ominus)$. Это является обобщением базового линейного классификатора, поскольку мы принимаем в расчет корреляцию признаков и асимметричное распределение по классам, однако цена такого обобщения с точки зрения объема вычислений высока (квадратичная зависимость от количества объектов и кубическая от количества признаков).
- ☞ В разделе 7.2 представлена еще одна классическая линейная модель – перцептрон. В отличие от метода наименьших квадратов, который всегда находит оптимальное решение, обращающее в минимум сумму квадратов невязок, перцептрон – это эвристический алгоритм, который зависит, в частности, от порядка предъявления примеров. Этот алгоритм был впервые описан в работе Rosenblatt (1958), а его сходимость для линейно разделимых данных доказана в работе Новикова (Novikoff, 1962), который также дал оценку сверху числа ошибок, допущенных до того, как перцептрон сойдется. В работе Minsky, Papert (1969) доказаны дополнительные формальные свойства перцептрана, а также продемонстрированы его ограничения как линейного классификатора. Благодаря усилиям многих исследователей эти ограничения со временем были преодолены – путем изобретения многослойного перцептрана и алгоритма его обучения методом обратного распространения ошибки (Rumelhart, Hinton, Williams, 1986). В этом разделе мы также узнали о двойственном взгляде на линейную классификацию, когда в результате обучения находятся веса объектов, а не признаков. В случае перцептрана эти веса – не что иное, как количество неправильных попыток классификации примера за время обучения.
- ☞ Классификация с максимальным зазором методом опорных векторов стала темой раздела 7.3. Этот подход был предложен в работе Boser, Guyon,

- Vapnik (1992). В двойственной формулировке веса объектов отличны от нуля только для опорных векторов, которые представляют собой обучающие примеры на границе зазора. Обобщение на зазоры с мягкой границей предложено в работе Cortes, Vapnik (1995). Допускаются ошибки зазора, но полная ошибка зазора прибавляется в качестве регуляризационного члена к минимизируемой целевой функции с весом, равным параметру сложности C ; всем объектам внутри полосы зазора назначается вес C . Как мы видели, если значение C достаточно мало, то метод опорных векторов сводит классы к их невзвешенному среднему и потому очень похож на базовый линейный классификатор. Общее введение в метод SVM имеется в работе Cristianini, Shawe-Taylor (2000). В часто применяемом алгоритме последовательной минимальной оптимизации (Platt, 1998) итеративно выбираются и аналитически оптимизируются пары множителей.
- ☞ В разделе 7.4 мы рассмотрели два способа превращения линейных классификаторов в оценку вероятностей путем преобразования расстояния со знаком до решающей границы в вероятности классов. Один из них – использовать логистическую функцию, готовую или полученную подгонкой параметров положения и разброса к данным. Хотя зачастую этот метод представляют как некий фокус, мы видели, что его можно обосновать, предположив, что расстояния в каждом классе имеют нормальное распределение с одинаковой дисперсией; последнее предположение необходимо, чтобы преобразование было монотонным. Непараметрическая альтернатива – использовать выпуклую оболочку кривой РХП для получения калиброванных оценок вероятностей. В итогах главы 2 уже упоминалось, что своими корнями это решение уходит в изотонную регрессию (Best, Chakravarti, 1990), и специалисты по машинному обучению познакомились с ним в работе Zadrozny, Elkan (2002). В работах Fawcett, Niculescu-Mizil (2007) и Flach, Matsubara (2007) показана его эквивалентность калибровке с помощью выпуклой оболочки кривой РХП.
- ☞ Наконец, в разделе 7.5 мы вкратце обсудили, как выйти за пределы линейности с помощью ядерных методов. «Трюк с ядром» применим к любому алгоритму обучения, который можно полностью описать в терминах скалярных произведений, а это большинство обсуждавшихся в данной главе подходов. Красота идеи заключается в том, что мы неявно производим классификацию в пространстве признаков высокой размерности, не строя это пространство явно. В качестве примера такого алгоритма я привел ядерный перцептрон; в следующей главе мы познакомимся с другим примером. Работа Shawe-Taylor, Cristianini (2004) – великолепное справочное пособие, в котором представлен богатейший материал по использованию ядер в машинном обучении, а в работе Gartner (2009) обсуждается применение ядерных методов к структурированным нечисловым данным.

Метрические модели

Многие виды обучения основаны на обобщающем переходе от обучающих данных к ранее не виденным с помощью исследования сходства между теми и другими. В группирующих моделях, к которым относятся, в частности, решающие деревья, такое сходство принимает форму отношения эквивалентности, или разбиения пространства объектов: два объекта считаются похожими, если они оказываются в одном и том же сегменте разбиения. В этой главе мы рассмотрим методы обучения, в которых используются в большей степени количественные меры сходства. Есть много способов измерения сходства, и в разделе 8.1 мы поговорим о наиболее важных. Раздел 8.2 посвящен обсуждению двух ключевых понятий машинного обучения на основе расстояния: соседям и эталонам. В разделе 8.3 мы рассмотрим, пожалуй, самый широко известный метрический метод обучения: классификатор по ближайшим соседям. В разделе 8.4 исследуются кластеризация методом K средних и родственные методы, а в разделе 8.5 – иерархическая кластеризация путем построения дендрограмм. Наконец, в разделе 8.6 мы обсудим, как некоторые из этих методов можно обобщить с помощью ядер, с которыми познакомились в предыдущей главе.

8.1 Так много дорог...

Поначалу может показаться странным, что существует много способов измерения расстояния. Я не имею в виду разные меры длины (километры, мили, морские мили и т. д.), поскольку это простые монотонные преобразования, которые не изменяют расстояние сколько-нибудь существенным образом. Приблизиться к существу проблемы поможет учет способа путешествия. Очевидно, что, путешествуя из Бристоля в Амстердам поездом, вы покрываете большее расстояние, чем летая самолетом, потому что самолет не так ограничен в выборе маршрута, как поезд. Мы разовьем эту мысль на примере игры в шахматы.

В шахматах каждая фигура может ходить только определенным образом. Ограничения могут быть связаны с направлением, например король и ферзь могут ходить по горизонтали, по вертикали и по диагонали, слон – только по диагонали, ладья – только по горизонтали и по вертикали, а пешка – только вверх. На короля и на пешку налагается дополнительное ограничение – они могутходить только на одну клетку, тогда как ферзь, слон и ладья могут ходить на любое

число клеток в одном направлении. Наконец, конь ходит очень своеобразно: за один ход смещается на одну клетку по диагонали и на одну по вертикали или по горизонтали – буквой «Г».

Хотя все фигуры перемещаются на одной доске, идея расстояние представляется им совершенно по-разному. Например, следующая клетка вниз находится от короля, ферзя и ладьи в одном ходе, от коня – в трех ходах, а для слона и пешки вообще недостижима. Конечно, это очень похоже на наш опыт в реальном мире. Поезда и автомобили могут перемещаться только по рельсам или по дороге, поэтому дикие места им, как и слону, недоступны. Горный хребет для автомобиля, поезда или пешехода может означать большой крюк, тогда как самолет легко пересекает его сверху. В метро, чтобы добраться до станции, находящейся на соседней улице, возможно, придется сделать несколько пересадок – в частности как коню, которому нужно сделать два или три хода, чтобы попасть в соседнюю клетку. А передвигаясь на своих двоих, мы обретаем такую же максимальную гибкость, но и такую же неспешность, как король.

На рис. 8.1 показаны расстояния с точки зрения короля и ладьи. Обе фигуры могут попасть в любую клетку доски, но ладья перемещается намного быстрее. На самом деле ладья может оказаться в любой клетке за один или два хода (если на пути нет других фигур). Клетки в пределах досягаемости одного хода образуют крест, а для попадания во все остальные нужен один дополнительный ход. Королю, чтобы добраться до определенной клетки, часто нужно более двух ходов (хотя в некоторые клетки король может попасть за один ход, тогда как ладье понадобятся два). Клетки на расстоянии одного хода от короля образуют небольшой квадрат с центром в текущей позиции, клетки на расстоянии двух ходов – больший квадрат, окружающий предыдущий, и т. д. На рис. 8.1 справа показана

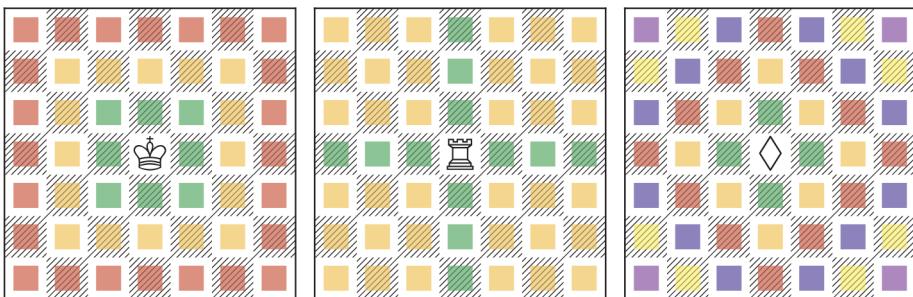


Рис. 8.1. (Слева) Расстояние на шахматной доске с точки зрения короля: зеленые клетки находятся на расстоянии одного хода, оранжевые – двух ходов, красные – трех ходов. **(В центре)** Ладья может перемещаться на любое число клеток, но только по вертикали или по горизонтали. Любая клетка находится от нее не дальше, чем в двух ходах. **(Справа)** Вокруг короля и ладьи симметрично расположены клетки, находящиеся на одинаковом расстоянии от центра. Вокруг короля – это крест, состоящий из 5×5 клеток (включая центральную). Вокруг ладьи – это квадрат, состоящий из 3×3 клеток (включая центральную). Вокруг королевской ладьи – это ромб, состоящий из 7×7 клеток (включая центральную). Равноудаленные клетки теперь образуют ромбы

фигура, которой в шахматах нет, но могла бы быть. Она сочетает ограничения короля и ладьи, поэтому я назвал ее «короладью». Подобно королю, она может ходить только на одну клетку, а подобно ладье – только по горизонтали или по вертикали. Для «короладьи» равноудаленные клетки образуют ромб с центром в текущей позиции.

На рис. 8.2 слева показаны ходы слона. Слон похож на ладью в том отношении, что некоторые клетки (того же цвета, что и та, на которой он находится) отстоят от него не более, чем в двух ходах, однако все прочие – другого цвета – ему недоступны. Объединяя ограничения слона (ходит только по диагонали) и короля (ходит только на одну клетку), мы получаем еще одну воображаемую фигуру – «корослона» (рис. 8.2 справа). Можно сказать, что мир слонов и корослонов повернут на 45° относительно мира ладей и короладей.

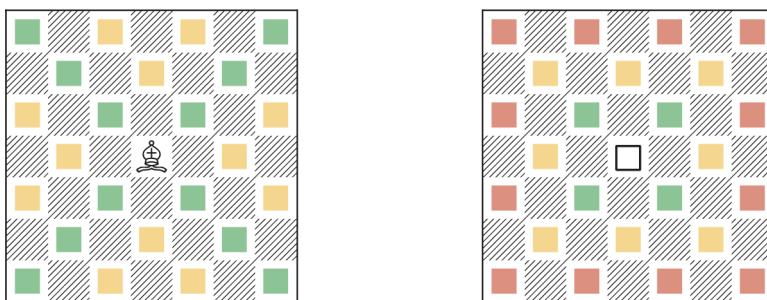


Рис. 8.2. (Слева) Мир слона: любая клетка находится в одном или двух ходах либо недоступна вовсе. **(Справа)** Воображаемый «корослон» сочетает ограничения короля и слона: может ходить только на одну клетку, да и то по диагонали. Равноудаленные клетки образуют квадраты, выкрашенные одним цветом

Ну и какое отношение все это имеет к попытке понять метрические методы машинного обучения, спросите вы. Дело в том, что горизонтали и вертикали на шахматной доске в чем-то похожи на дискретные или категориальные признаки в машинном обучении (на самом деле, поскольку горизонтали и вертикали упорядочены, они являются *порядковыми (ординальными)* признаками, о чем мы будем говорить в главе 10). Мы можем перейти к вещественнонозначным признакам, представив себе «непрерывную» шахматную доску, в которой бесконечно много бесконечно узких горизонталей и вертикалей. Клетки теперь становятся точками, а расстояния выражаются не количеством пройденных клеток, а вещественным числом в некоторой шкале. Если теперь соединить равноудаленные точки, то окажется, что во многих случаях дискретные фигуры превратились в непрерывные. Например, для короля все точки, находящиеся на заданном расстоянии от него, по-прежнему образуют квадрат с центром в текущей позиции, а для короладьи – квадрат, повернутый на 45° , как и раньше. Как выясняется, все это – частные случаи более общего понятия, которое определяется ниже.

Определение 8.1 (расстояние Минковского). Если $X = \mathbb{R}^d$, то расстояние Минковского порядка $p > 0$ определяется формулой

$$\text{Dis}_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{j=1}^d |x_j - y_j|^p \right)^{1/p} = \|\mathbf{x} - \mathbf{y}\|_p,$$

где $\|\mathbf{z}\|_p = (\sum_{j=1}^d |z_j|^p)^{1/p}$ – p -норма (иногда обозначается L_p) вектора \mathbf{z} . Мы часто будем называть Dis_p просто p -нормой.

Так, **2-норма** – не что иное, как всем знакомое *евклидово расстояние*:

$$\text{Dis}_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{j=1}^d |x_j - y_j|^2} = \sqrt{(\mathbf{x} - \mathbf{y})^\top (\mathbf{x} - \mathbf{y})},$$

которое измеряет расстояние «по прямой». Два других значения p можно соотнести с примером из мира шахмат. **1-норма** иначе называется *манхэттенским расстоянием*, или *расстоянием городских кварталов*:

$$\text{Dis}_1(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^d |x_j - y_j|.$$

Это расстояние, которое мы проходим, двигаясь только параллельно осям координат, – как при поездке на такси в Манхэттене или в других городах с продольно-поперечной дорожной сетью. И вместе с тем это расстояние в восприятии нашей воображаемой «королады». Если увеличивать p , то расстояние будет все в большей и большей степени определяться наибольшим расстоянием вдоль одной из осей, откуда можно сделать вывод, что $\text{Dis}_\infty(\mathbf{x}, \mathbf{y}) = \max_j |x_j - y_j|$. Это расстояние в смысле короля, который может двигаться в любом направлении, но только на одну клетку, его также называют *расстоянием Чебышева*. На рис. 8.3 слева показаны точки, равноудаленные от начала координат в смысле расстояний Минковского разного порядка. Легко видеть, что только евклидово расстояние инвариантно относительно поворота, то есть при любом $p \neq 2$ направления осей координат играют особую роль. В определении расстояния Минковского начало координат не фигурирует, поэтому при любом p оно инвариантно относительно параллельных переносов, но ни при каком p не является инвариантным относительно масштабирования.

Иногда можно встретить упоминание о **0-норме** (или норме L_0), которая подсчитывает количество ненулевых элементов в векторе. Соответствующее ей расстояние – это количество позиций, в которых векторы \mathbf{x} и \mathbf{y} различаются. Строго говоря, это не расстояние Минковского, но его можно определить следующим образом:

$$\text{Dis}_0(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^d (x_j - y_j)^0 = \sum_{j=1}^d I[x_j \neq y_j],$$

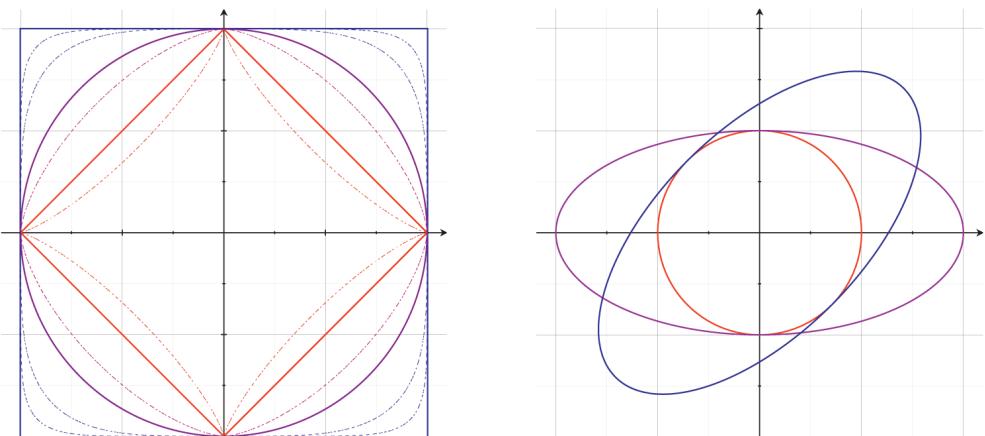


Рис. 8.3. (Слева) Геометрические места точек, удаленных на единичное расстояние Минковского порядка p от начала координат, при (изнутри наружу) $p = 0.8$, $p = 1$ (манхэттенское расстояние, повернутый **красный** квадрат), $p = 1.5$, $p = 2$ (евклидово расстояние, **фиолетовая** окружность), $p = 4$, $p = 8$ и $p = \infty$ (расстояние Чебышева, **синий** прямоугольник). Отметим, что в точках на осях координат все расстояния совпадают, но в остальном с ростом p фигура «разбухает». Однако условию инвариантности относительно поворота удовлетворяет только евклидово расстояние. (**Справа**) Повернутый эллипс $\mathbf{x}^T \mathbf{R}^T \mathbf{S}^2 \mathbf{R} \mathbf{x} = 1/4$, параллельный осям эллипс $\mathbf{x}^T \mathbf{S}^2 \mathbf{x} = 1/4$ и окружность $\mathbf{x}^T \mathbf{x} = 1/4$ (\mathbf{R} и \mathbf{S} , как в примере 8.1)

если принять соглашение, что $x^0 = 0$ при $x = 0$ и 1 в остальных случаях. По сути дела, это расстояние с точки зрения ладьи: если клетка находится на другой горизонтали и вертикали, то до нее два хода, а если отличается только что-то одно, то один ход. Если \mathbf{x} и \mathbf{y} – строки нулей и единиц, то эта величина называется также **расстоянием Хэмминга**. Можно также сказать, что расстояние Хэмминга – это количество битов, которые нужно поменять, чтобы преобразовать \mathbf{x} в \mathbf{y} ; для произвольных строк необязательно равной длины обобщение этой идеи приводит к **расстоянию Левенштейна**, или **редакционному расстоянию**.

Все ли описанные математические конструкции согласуются с нашим представлением о расстоянии? Чтобы ответить на этот вопрос, сформулируем, какими свойствами должно обладать настоящее расстояние, например: неотрицательность и симметричность. Общепризнанным является следующий список, определяющий так называемую метрику.

Определение 8.2 (метрика). Если дано пространство объектов X , то **метрическим расстоянием**, или просто **метрикой**, в нем называется функция $\text{Dis}: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ такая, что для любых $x, y, z \in \mathcal{X}$:

1. Расстояние от точки до нее самой равно нулю: $\text{Dis}(x, x) = 0$;
2. Все остальные расстояния больше нуля: если $x \neq y$, то $\text{Dis}(x, y) > 0$;
3. Расстояние симметрично: $\text{Dis}(y, x) = \text{Dis}(x, y)$;
4. Объезд не может сократить расстояние: $\text{Dis}(x, z) \leq \text{Dis}(x, y) + \text{Dis}(y, z)$.

Если второе условие ослабить до нестрогого неравенства, то есть разрешить $\text{Dis}(x, y)$ быть равным нулю, даже когда $x \neq y$, то функция Dis называется псевдометрикой.

Последнее условие называется *неравенством треугольника* (или субаддитивностью, поскольку оно касается соотношения между расстоянием и сложением). На рис. 8.4 это понятие исследуется для расстояний Минковского различных порядков. Неравенство треугольника говорит, что расстояние от начала координат до точки С не больше суммы расстояний от начала координат до точки А ($\text{Dis}(O, A)$) и от А до С ($\text{Dis}(A, C)$). Точка В находится на том же расстоянии от А, что и С, независимо от используемой метрики, таким образом, $\text{Dis}(O, A) + \text{Dis}(A, C)$ равно расстоянию от начала координат до В. Значит, если нарисовать окружность с центром в начале координат, проходящую через В, то в силу неравенства треугольника точка С не должна оказаться вне этой окружности. На левом рисунке (евклидово расстояние) мы видим, что точка В – единственная, в которой пересекаются окружности с центрами в начале координат и в А, и, стало быть, во всех остальных точках неравенство треугольника строгое.

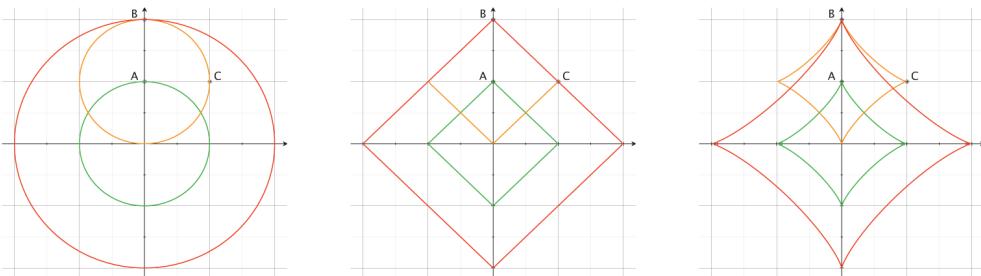


Рис. 8.4. (Слева) Зеленая окружность – геометрическое место точек, равноудаленных от начала координат в смысле евклидова расстояния (расстояния Минковского порядка $p = 2$), таких как А. Оранжевая окружность показывает, что В и С равноудалены от А. Красная окружность показывает, что С ближе к началу координат, чем В, что согласуется с неравенством треугольника. (В центре) В случае манхэттенского расстояния ($p = 1$) В и С расположены на равном расстоянии от начала координат и равноудалены также от А. (Справа) При $p < 1$ (в данном случае $p = 0.8$) С находится дальше от начала координат, чем В; поскольку обе точки по-прежнему равноудалены от А, то получается, что путешествие из начала координат в С через А быстрее, чем напрямую, что противоречит неравенству треугольника

На среднем рисунке та же ситуация показана для манхэттенского расстояния ($p = 1$). Теперь В и С равноудалены от начала координат, так что путешествие из А в С – уже не объезд, а лишь один из многих кратчайших путей. Но если мы будем еще уменьшать p , то в конце концов С окажется вне красной фигуры и, следовательно, дальше, чем В, если смотреть из начала координат; при этом сумма расстояний от начала координат до А и от А до С по-прежнему равна расстоянию от начала координат до В. В этот момент интуиция нам отказывает: расстояния

Минковского с $p < 1$ не слишком полезны в качестве расстояний, так как нарушают неравенство треугольника.

Иногда полезно применять разные шкалы для различных осей координат, если перемещение вдоль них происходит с разной скоростью. Например, люди перемещаются по горизонтали с куда большей легкостью, чем по вертикали, поэтому для определения точек, достижимых за определенное время, реалистичнее использовать не окружность, а эллипс, главная ось которого параллельна направлению наиболее быстрого перемещения. Этот эллипс можно также поворачивать, так что главная ось необязательно параллельна какой-то оси координат; например, это может быть направление шоссе или ветра. Математически гиперсфера (окружность в пространстве $d \geq 2$ измерений) радиуса r определяется уравнением $\mathbf{x}^T \mathbf{x} = r^2$, а гиперэллипс – уравнением $\mathbf{x}^T \mathbf{M} \mathbf{x} = r^2$, где \mathbf{M} – матрица, описывающая поворот и масштабирование.

Пример 8.1 (эллиптическое расстояние). Рассмотрим следующие матрицы:

$$\mathbf{R} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} \quad \mathbf{S} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{M} = \begin{pmatrix} 5/8 & -3/8 \\ -3/8 & 5/8 \end{pmatrix}$$

Матрица \mathbf{R} описывает поворот по часовой стрелке на 45° , а диагональная матрица \mathbf{S} – масштабирование по оси x с коэффициентом $1/2$. Уравнение

$$(\mathbf{SRx})^T (\mathbf{SRx}) = \mathbf{x}^T \mathbf{R}^T \mathbf{S}^T \mathbf{SRx} = \mathbf{x}^T \mathbf{R}^T \mathbf{S}^2 \mathbf{Rx} = \mathbf{x}^T \mathbf{M} \mathbf{x} = 1/4$$

описывает фигуру, которая после поворота по часовой стрелке на 45° и масштабирования по оси x с коэффициентом $1/2$ станет окружностью радиуса $1/2$, то есть наклонный эллипс на рис. 8.3 справа. Уравнение этого эллипса: $(5/8)x^2 + (5/8)y^2 - (3/4)xy = 1/2$.

Часто форму эллипса оценивают по данным как обращение ковариационной матрицы: $\mathbf{M} = \Sigma^{-1}$. Это приводит нас к определению *расстояния Махалонобиса*:

$$\text{Dis}_{\mathcal{M}}(\mathbf{x}, \mathbf{y} | \Sigma) = \sqrt{(\mathbf{x} - \mathbf{y})^T \Sigma^{-1} (\mathbf{x} - \mathbf{y})}. \quad (8.1)$$

Как мы видели в разделе 7.1, такое применение ковариационной матрицы означает устранение корреляции признаков и их нормировку. Очевидно, что евклидово расстояние является частным случаем расстояния Махалонобиса, когда ковариационная матрица единичная: $\text{Dis}_2(\mathbf{x}, \mathbf{y}) = \text{Dis}_{\mathcal{M}}(\mathbf{x}, \mathbf{y} | \mathbf{I})$.

8.2 Соседи и эталоны

Теперь, разобравшись с основами измерения расстояний в пространстве объектов, перейдем к рассмотрению ключевых идей, лежащих в основе метрических моделей. Наиболее важны две из них: формулировка модели в терминах типичных объектов, или *эталонов*, и определение решающего правила в терминах бли-

жайших эталонов, или *соседей*. Лучше понять эти концепции нам поможет наш старый приятель – базовый линейный классификатор. В нем эталонами являются два средних вектора классов μ^{\oplus} и μ^{\ominus} , которые содержат все, что нам нужно знать об обучающих данных, чтобы построить классификатор. Фундаментальное свойство среднего вектора множества векторов заключается в том, что он доставляет минимум сумме квадратов евклидовых расстояний до всех векторов множества.

Теорема 8.1 (среднее арифметическое минимизирует сумму квадратов евклидовых расстояний). Среднее арифметическое μ множества точек D в евклидовом пространстве является единственной точкой, в которой сумма квадратов евклидовых расстояний до этих точек достигает минимума.

Доказательство. Мы покажем, что $\operatorname{argmin}_y \sum_{x \in D} \|x - y\|^2 = \mu$, где $\|\cdot\|$ обозначает 2-норму. Чтобы найти минимум, мы возьмем градиент (вектор частных производных по y_i) суммы и приравняем его к нулю:

$$\nabla_y \sum_{x \in D} \|x - y\|^2 = -2 \sum_{x \in D} (x - y) = -2 \sum_{x \in D} x + 2 |D| y = 0,$$

откуда следует, что $y = \frac{1}{|D|} \sum_{x \in D} x = \mu$.

Отметим, что минимизация суммы квадратов евклидовых расстояний до точек из заданного множества – то же самое, что минимизация *усредненной* суммы квадратов евклидовых расстояний. Может возникнуть вопрос, что случится, если опустить слово «квадратов»: ведь кажется более естественным взять в качестве эталона точку, минимизирующую просто сумму евклидовых расстояний. Такая точка называется *геометрической медианой*, потому что в случае одномерных данных она соответствует медиане, или «срединному значению» множества чисел. Однако в многомерном случае не существует замкнутой формулы для нахождения геометрической медианы, ее приходится вычислять методом последовательных приближений. Это вычислительное преимущество и есть основная причина, почему в метрических методах предпочитают использовать сумму квадратов евклидовых расстояний.

В некоторых ситуациях имеет смысл наложить на эталон ограничение: он должен совпадать с одной из заданных точек. В таком случае мы говорим о *медоиде*, чтобы отличить его от *центроида* – эталона, необязательно принадлежащего множеству данных. Для нахождения медоида мы должны для каждой точки вычислить сумму расстояний до всех остальных точек и выбрать ту точку, в которой эта сумма обращается в минимум. Безотносительно к используемой метрике в случае n точек для этого потребуется $O(n^2)$ операций, поэтому с точки зрения вычисления медоида нет причин предпочесть одну метрику другой. На рис. 8.5 показано множество из 10 точек, для которого различные способы определения эталона дают разные результаты. В частности, средняя точка и медоид, вычисленный с помощью квадрата 2-нормы, могут быть чрезмерно чувствительны к выбросам.

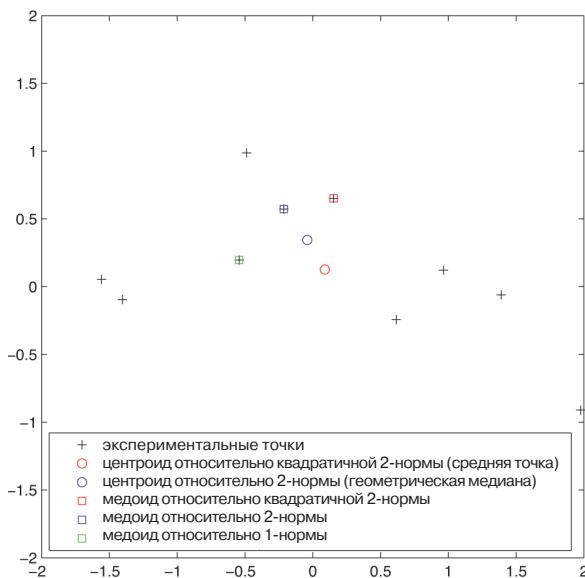


Рис. 8.5. Небольшой набор данных из 10 точек, кружочками обозначены центроиды, квадратиками – медоиды (которые должны совпадать с одной из имеющихся точек) для различных метрик. Обратите внимание, как выброс в правом нижнем углу «оттаскивает» среднюю точку от геометрической медианы, в результате изменяется и соответствующий медоид

Имея определение эталона, базовый линейный классификатор строит решающую границу в виде прямой, проходящей через середину соединяющего их отрезка перпендикулярно к нему. Альтернативный способ метрической классификации объектов без прямого упоминания решающей границы заключается в следующем правиле: если объект \mathbf{x} ближе к μ^\oplus , классифицировать его как положительный, иначе как отрицательный, или – эквивалентно – назначить объекту класс *ближайшего* к нему эталона. Если в качестве меры близости взять евклидово расстояние, то из простейших геометрических соображений следует, что мы получим в точности ту же самую решающую границу (рис. 8.6 слева).

Таким образом, *с метрической точки зрения, базовый линейный классификатор можно интерпретировать как построение эталонов, минимизирующих сумму квадратов расстояний в пределах каждого класса, и последующее применение решающего правила ближайшего эталона*. Такая смена угла зрения открывает массу новых возможностей. Например, можно исследовать, как будет выглядеть решающая граница, если в решающем правиле использовать манхэттенское расстояние (рис. 8.6 справа). Оказывается, что решающая граница может менять направление только под некоторыми углами: в двухмерном случае она может идти по горизонтали, по вертикали и под углом $\pm 45^\circ$. Это можно обосновать следующими рассуждениями. Предположим, что оба эталона имеют разные координаты x и y , то есть являются противоположными углами прямоугольника (я буду

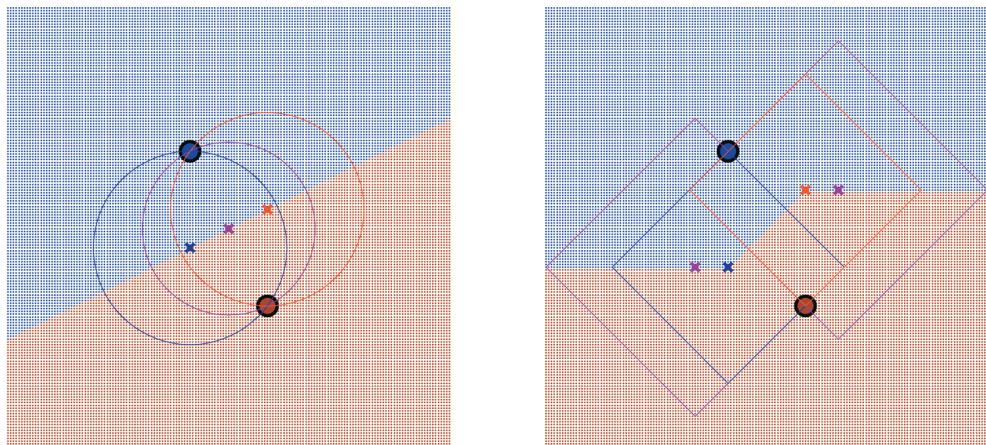


Рис. 8.6. (Слева) При двух эталонах решающее правило ближайшего эталона с евклидовым расстоянием дает линейную решающую границу – прямую, которая проходит через середину отрезка, соединяющего эталоны, перпендикулярно к нему. Крестиками отмечены точки на решающей границе, а окружности с центрами в этих точках наглядно демонстрируют, что они равноудалены от эталонов. При движении вдоль решающей границы из левого нижнего в правый верхний угол радиусы этих окружностей сначала уменьшаются, а после прохождения середины отрезка, соединяющего эталоны, снова начинают расти.
(Справа) В случае использования манхэттенского расстояния окружности заменяются ромбами. При движении слева направо ромбы смещаются вдоль левого горизонтального участка решающей границы, уменьшаясь в размерах, затем движутся вдоль участка решающей границы, наклоненного под углом 45° , сохраняя размер, а потом снова смещаются вдоль горизонтального участка – правого

считать, что прямоугольник вытянут по вертикали, как на рисунке). Представьте, что вы стоите в центре этого прямоугольника и, следовательно, на равных расстояниях от обоих эталонов (на самом деле эта точка принадлежит решающей границе относительно 2-нормы). Если вы теперь отступите на один шаг по горизонтали, то приблизитесь к одному эталону и отдалиитесь от другого; чтобы компенсировать это, необходимо сделать также шаг по вертикали. Таким образом, находясь внутри прямоугольника, вы сохраняете равноудаленность от эталонов, двигаясь под углом 45° . Достигнув периметра прямоугольника, вы можете обеспечить дальнейшее выполнение условия равноудаленности, только двигаясь по горизонтали, то есть решающая граница дальше будет проходить вертикально.

Еще одно полезное следствие взгляда на проблему с метрической точки зрения – тот факт, что решающее правило ближайшего эталона работает и тогда, когда эталонов больше двух. Это дает нам многоклассовый вариант базового линейного классификатора¹. На рис. 8.7 слева это показано для трех эталонов.

¹ В контексте информационного поиска его часто называют *классификатором Рокко*.

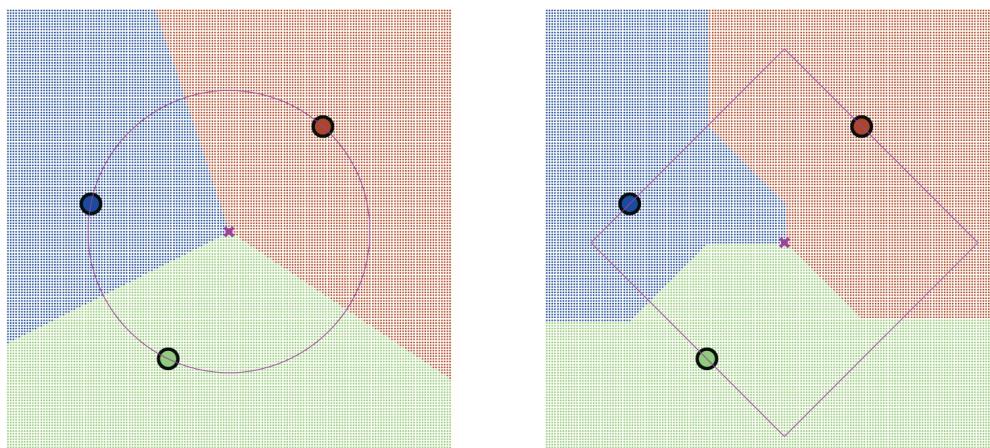


Рис. 8.7. (Слева) Решающие области, определенные решающим правилом ближайшего эталона относительно 2-нормы. **(Справа)** При использовании манхэттенского расстояния решающие области перестают быть выпуклыми

Каждая решающая область теперь ограничена двумя лучами. Как и следовало ожидать, решающие границы при использовании 2-нормы более регулярны, чем при использовании 1-нормы: математики говорят, что решающие области относительно 2-нормы выпуклы, то есть отрезок, соединяющий любые две точки, принадлежащие такой области, сам целиком принадлежит ей. Очевидно, что для решающих областей относительно 1-нормы это уже не так (рис. 8.7 справа). При дальнейшем увеличении числа эталонов некоторые области становятся замкнутыми выпуклыми «ячейками» (далее в этом разделе предполагается евклидово расстояние), что приводит к *диаграмме Вороного*. Поскольку число классов обычно гораздо меньше числа эталонов, решающие правила часто принимают во внимание больше одного эталона. В результате количество решающих областей еще увеличивается.

Пример 8.2 (два соседа знают больше, чем один). На рис. 8.8 слева показана диаграмма Вороного для пяти эталонов. Каждый отрезок лежит на прямой, которая проходит через середину отрезка, соединяющего два эталона, перпендикулярно к нему. Всего существует $\binom{5}{2} = 10$ пар

эталонов, но в двух из них эталоны отстоят слишком далеко друг от друга, поэтому в диаграмме Вороного мы видим только восемь отрезков.

Если мы теперь примем во внимание еще и второй ближайший эталон, то каждая ячейка диаграммы разбивается на более мелкие. Например, поскольку у центральной точки четыре соседа, то содержащая ее ячейка разбивается на четыре подобласти (рис. 8.8 в центре). Эти дополнительные отрезки можно считать частями диаграммы Вороного, замещающими удаленную

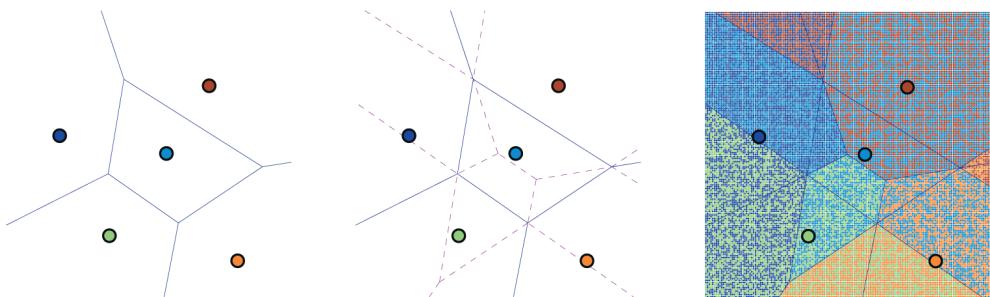


Рис. 8.8. (Слева) Диаграмма Вороного для пяти эталонов. **(В центре)** Если принимать во внимание два ближайших эталона, то каждая ячейка диаграммы Вороного разбивается на более мелкие. **(Справа)** Штриховкой показано, в какие ячейки вносит вклад каждый эталон

центральную ячейку. У других эталонов по три непосредственных соседа, поэтому их ячейки разбиваются на три подобласти. Таким образом, мы получаем 16 решающих областей «с 2 ближайшими эталонами», каждая из которых определяется своей парой ближайшего и второго по близости экземпляра.

На рис. 8.8 справа каждая область заштрихована в соответствии с тем, какие два ближайших эталона ее порождают. Отметим, что мы назначили обоим эталонам одинаковый вес и что пары соседних областей (границящих по отрезкам исходной диаграммы Вороного) заштрихованы одинаково, так что всего понадобилось восемь разных видов штриховки. Это окажется существенным впоследствии, когда мы перейдем к обсуждению уточнения классификаторов по ближайшим соседям.

Итак, основными ингредиентами метрических моделей являются:

- ☞ метрические расстояния, в том числе евклидово, манхэттенское, Минковского и Махalanобиса;
- ☞ эталоны: центроиды, определяющие центр масс в соответствии с выбранной метрикой, или медоиды, определяющие «самую центральную» точку;
- ☞ метрические решающие правила, в которых проводится голосование среди k ближайших эталонов.

В следующих разделах мы увидим, как эти ингредиенты различными способами комбинируются для получения алгоритмов обучения с учителем и без.

8.3 Классификация по ближайшему соседу

В предыдущем разделе мы видели, как обобщить базовый линейный классификатор на число классов, большее двух, обучив эталон в каждом классе и используя решающее правило ближайшего эталона для классификации новых данных. На самом деле большинство популярных метрических классификаторов еще проще: в них эталоном может служить любой обучающий пример. Следовательно, «обучение» такого классификатора сводится просто к запоминанию обучающих

данных. Такой примитивный до крайности классификатор называется *классификатором по ближайшему соседу*. Его решающие области составлены из ячеек диаграммы Вороного, а кусочно-линейные решающие границы выбираются из множества границ диаграммы Вороного (поскольку соседние ячейки могут быть помечены одним и тем же классом).

Каковы свойства классификатора по ближайшему соседу? Прежде всего отметим, что если обучающий набор не содержит одинаковых объектов из разных классов, то мы сможем идеально разделить классы на обучающем наборе – и неудивительно, коль скоро мы запомнили все обучающие примеры! Далее за счет подходящего выбора эталонов мы можем представить более или менее любую решающую границу или, по крайней мере, сколь угодно точное кусочно-линейное приближение к ней. Отсюда следует, что классификатор по ближайшему соседу обладает низким смещением, но высокой дисперсией: сдвиньте любой эталон, порождающий решающую границу, – и сама граница тоже изменится. Отсюда риск переобучения, если обучающие данные ограничены, зашумлены или не репрезентативны.

С алгоритмической точки зрения, обучение классификатора по ближайшему соседу производится очень быстро – за время $O(n)$, где n – число хранимых эталонов. Недостаток в том, что и классификация одного экземпляра тоже занимает время $O(n)$, поскольку этот экземпляр необходимо сравнить с каждым эталоном, чтобы найти ближайший. Время классификации можно сократить за счет увеличения времени обучения путем хранения эталонов в более сложной структуре данных, но такой подход плохо масштабируется на большое число признаков.

На самом деле пространства объектов высокой размерности вызывают проблемы и по другой причине: печально известного *проклятия размерности*. Многомерные пространства обычно оказываются сильно разреженными, то есть каждая точка далеко отстоит практически от любой другой, а потому попарные расстояния не несут полезной информации. Однако настигнет вас проклятие размерности или нет, зависит не только от количества признаков, поскольку имеется ряд причин, из-за которых эффективная размерность пространства объектов может оказаться гораздо меньше числа признаков. Например, некоторые признаки могут быть нерелевантными и заглушать релевантные признаки при вычислении расстояний. В таких случаях полезно до построения метрической модели понизить размерность путем [отбора признаков](#), который мы будем обсуждать в главе 10. Возможно также, что данные располагаются на *многообразии* более низкой размерности, чем размерность пространства объектов (например, на поверхности сферы, которая представляет собой двумерное многообразие, погруженное в трехмерное пространство), что позволяет применить другие методы понижения размерности, например [метод главных компонент](#), рассматриваемый в той же главе. В любом случае, перед тем как применять классификацию по ближайшему соседу, имеет смысл построить гистограмму попарных расстояний между примечаниями в выборке и посмотреть, достаточно ли они разнообразны.

Отметим, что метод ближайшего соседа применим и к задачам регрессии с вещественнонозначной целевой переменной. На самом деле этот метод абсолютно

безразличен к типу целевой переменной и может использоваться для классификации текстовых документов, изображений и видео. Можно также возвращать сам эталон, а не отдельную целевую переменную, и в таком случае обычно говорят о *поиске ближайшего соседа*. Разумеется, мы можем получать только цели (или эталоны), хранящиеся в базе данных эталонов, но если существует способ их агрегирования, то это ограничение можно снять, применив метод *k ближайших соседей*. В простейшей форме классификатор по k ближайшим соседям проводит голосование между $k \geq 1$ эталонами, ближайшими к классифицируемому объекту, и предсказывает класс, получивший большинство голосов (мажоритарный класс). Такой классификатор легко превратить в оценку вероятностей, возвращая нормированные счетчики классов в качестве распределения вероятностей по классам.

Рисунок 8.9 иллюстрирует это на небольшом наборе данных с 20 эталонами из пяти разных классов для $k = 3, 5, 7$. Распределение по классам наглядно представлено путем назначения каждой тестовой точке класса, определяемого соседями, не отдавая предпочтения ни одному из них. Так, в области, для которой два из $k = 3$ соседей красные, а один оранжевый, цвет штриховки на две трети состоит из красного и на одну треть из оранжевого. При $k = 3$ решающие области по большей части хорошо различимы, чего не скажешь о случаях $k = 5$ и $k = 7$. На первый взгляд, это противоречит ранее продемонстрированному в примере 8.2 увеличению числа решающих областей с ростом k . Однако это увеличение уравновешивается тем фактом, что векторы вероятностей становятся все больше похожи друг на друга. Возьмем крайний пример: если k равно числу эталонов n , то у каждого тестового объекта будет одинаковое число соседей, поэтому все они получат один и тот же вектор вероятностей, совпадающий с априорным распределением по эталонам. Если $k = n - 1$, то один из счетчиков классов можно уменьшить на 1, и сделать это можно с способами: число возможностей точно такое же, как при $k = 1$!

Мы делаем вывод, что уточнение метода k ближайших соседей – количество даваемых им различных предсказаний – сначала возрастает с ростом k , а затем

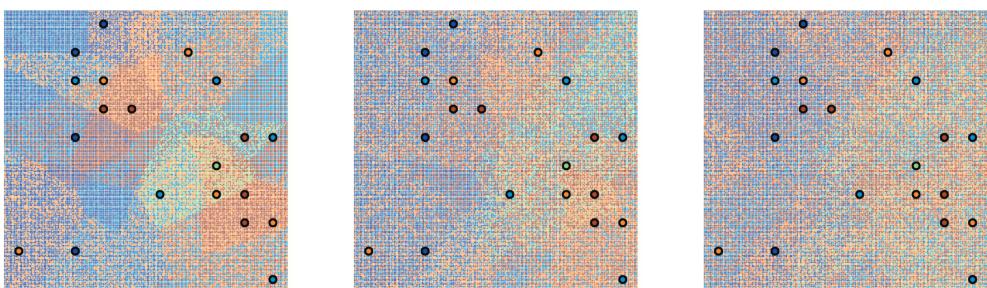


Рис. 8.9. (Слева) Решающие области для классификатора по 3 ближайшим соседям; штриховкой представлено предсказанное распределение вероятностей по пяти классам. **(В центре)** По 5 ближайшим соседям. **(Справа)** По 7 ближайшим соседям

снова убывает. Кроме того, мы можем сказать, что при увеличении k смещение растет, а дисперсия снижается. Не существует простого рецепта, который позволил бы решить, какое значение k лучше всего отвечает имеющемуся набору данных. Однако можно в какой-то степени обойти этот вопрос, применив к голосам *взвешивание расстояний*: то есть чем ближе эталон к классифицируемому объекту, тем весомее его голос. Это демонстрируется на рис. 8.10, где в качестве веса голоса используется величина, обратная расстоянию. При этом решающие границы размываются, поскольку теперь в модели используется сочетание группировки посредством границы Вороного и ранжирования посредством взвешивания расстояний. Далее, поскольку веса быстро убывают с ростом расстояния, эффект увеличения k гораздо менее заметен, чем при голосовании без взвешивания. Фактически, если применяется взвешивание, мы можем просто положить $k = n$ и тем не менее получить модель, которая дает разные предсказания в разных частях пространства объектов. Можно сказать, что взвешивание расстояний делает классификатор по k ближайшим соседям в большей степени глобальной моделью, тогда как без него (и для малых k) это скорее агрегирование локальных моделей.

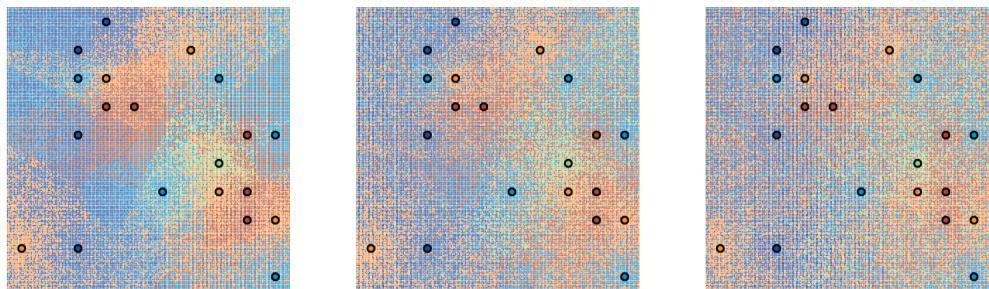


Рис. 8.10. (Слева) По 3 ближайшим соседям с взвешиванием расстояний на тех же данных, что на рис. 8.9. **(В центре)** По 5 ближайшим соседям. **(Справа)** По 7 ближайшим соседям

При использовании метода k ближайших соседей для задач регрессии очевидный способ агрегировать предсказания от k соседей заключается в том, чтобы взять среднее значение, которое опять-таки может быть взвешено с учетом расстояний. Это сообщило бы модели дополнительную предсказательную способность за счет предсказания значений, которые не наблюдались в хранимых эталонах. Более общо, мы можем применить метод k средних к любой проблеме обучения, в которой имеется подходящий «агрегатор» для нескольких целевых значений.

8.4 Метрическая кластеризация

В метрическом контексте обучение без учителя обычно связывают с кластеризацией, и сейчас мы дадим обзор нескольких метрических методов кластеризации.

Все методы, рассматриваемые в этом разделе, основаны на эталонах и потому являются прогностическими: они естественно обобщаются на ранее не предъявлявшиеся объекты (о различии между прогностической и дескриптивной кластеризациями см. раздел 3.3). В следующем разделе будет рассмотрен метод кластеризации, не основанный на эталонах и, следовательно, дескриптивный.

Прогностические метрические методы кластеризации состоят из тех же ингредиентов, что и метрические классификаторы: метрическое расстояние, способ построения эталонов и основанное на расстоянии решающее правило. В отсутствие явной целевой переменной предполагается, что цель обучения неявно закодирована в метрике, то есть мы стремимся найти кластеры, *компактные* относительно метрики. Для этого необходимо ввести понятие компактности кластера, которое может служить критерием оптимизации. С этой целью обратимся снова к матрице разброса, введенной в замечании 7.2 на стр. 211.

Определение 8.3 (разброс). Если дана матрица данных \mathbf{X} , то **матрицей разброса** называется матрица

$$\mathbf{S} = (\mathbf{X} - \mathbf{1}\mu)^T(\mathbf{X} - \mathbf{1}\mu) = \sum_{i=1}^n (\mathbf{X}_i - \mu)^T(\mathbf{X}_i - \mu),$$

где μ – вектор-строка, содержащий средние по всем столбцам \mathbf{X} . **Разброс** \mathbf{X} определяется как $\text{Scat}(\mathbf{X}) = \sum_{i=1}^n \|\mathbf{X}_i - \mu\|^2$, то есть след матрицы разброса (сумма элементов на ее главной диагонали).

Представим теперь, что мы разбили D на K подмножеств $D_1 \uplus \dots \uplus D_K = D$, и пусть μ_j – среднее по D_j . Пусть \mathbf{S} – матрица разброса D , а \mathbf{S}_j – матрицы разброса D_j . Эти матрицы связаны следующим соотношением:

$$\mathbf{S} = \sum_{j=1}^K \mathbf{S}_j + \mathbf{B}. \quad (8.2)$$

Здесь \mathbf{B} – матрица разброса, которая получается в результате замены каждой точки D соответствующим μ_j . Матрицы \mathbf{S}_j называются *матрицами внутрикластерного разброса* и описывают компактность j -го кластера. \mathbf{B} называется *матрицей межкластерного разброса* и характеризует разброс центроидов кластеров. Из (8.2) следует аналогичное разложение следов матриц:

$$\text{Scat}(D) = \sum_{j=1}^K \text{Scat}(D_j) + \sum_{j=1}^K |D_j| \|\mu_j - \mu\|^2. \quad (8.3)$$

Это говорит о том, что минимизация полного разброса по всем кластерам эквивалентна максимизации (взвешенного) разброса центроидов. *Проблема K средних* заключается в нахождении такого разбиения, которое минимизирует полный внутрикластерный разброс.

Пример 8.3 (уменьшение разброса путем разбиения данных). Рассмотрим такие пять точек: $(0,3)$, $(3,3)$, $(3,0)$, $(-2,-4)$, $(-4,-2)$. Эти точки центрированы относительно $(0,0)$ – очень удобно. Матрица разброса равна

$$\mathbf{S} = \begin{pmatrix} 0 & 3 & 3 & -2 & -4 \\ 3 & 3 & 0 & -4 & -2 \end{pmatrix} \begin{pmatrix} 0 & 3 \\ 3 & 3 \\ 3 & 0 \\ -2 & -4 \\ -4 & -2 \end{pmatrix} = \begin{pmatrix} 38 & 25 \\ 25 & 38 \end{pmatrix},$$

а ее след $\text{Scat}(D) = 76$. Если поместить первые две точки в один кластер, а оставшиеся три – в другой, то средние векторы кластеров будут равны $\mu_1 = (1.5, 3)$ и $\mu_2 = (-1, -2)$, а матрицы внутрикластерного разброса:

$$\mathbf{S}_1 = \begin{pmatrix} 0-1.5 & 3-1.5 \\ 3-3 & 3-3 \end{pmatrix} \begin{pmatrix} 0-1.5 & 3-3 \\ 3-1.5 & 3-3 \end{pmatrix} = \begin{pmatrix} 4.5 & 0 \\ 0 & 0 \end{pmatrix};$$

$$\mathbf{S}_2 = \begin{pmatrix} 3-(-1) & -2-(-1) & -4-(-1) \\ 0-(-2) & -4-(-2) & -2-(-2) \end{pmatrix} \begin{pmatrix} 3-(-1) & 0-(-2) \\ -2-(-1) & -4-(-2) \\ -4-(-1) & -2-(-2) \end{pmatrix} = \begin{pmatrix} 26 & 10 \\ 10 & 8 \end{pmatrix}$$

со следами $\text{Scat}(D_1) = 4.5$ и $\text{Scat}(D_2) = 34$. Две копии μ_1 и три копии μ_2 по определению имеют тот же центр тяжести, что полный набор данных: в данном случае $(0,0)$. Таким образом, мы вычисляем матрицу межкластерного разброса как

$$\mathbf{B} = \begin{pmatrix} 1.5 & 1.5 & -1 & -1 & -1 \\ 3 & 3 & -2 & -2 & -2 \end{pmatrix} \begin{pmatrix} 1.5 & 3 \\ 1.5 & 3 \\ -1 & -2 \\ -1 & -2 \\ -1 & -2 \end{pmatrix} = \begin{pmatrix} 7.5 & 15 \\ 15 & 30 \end{pmatrix}$$

со следом 37.5.

С другой стороны, если рассматривать как кластер первые три точки, а остальные две поместить во второй кластер, то средние векторы кластеров будут равны $\mu'_1 = (2,2)$ и $\mu'_2 = (-3,-3)$, а матрицы внутрикластерного разброса:

$$\mathbf{S}'_1 = \begin{pmatrix} 0-2 & 3-2 & 3-2 \\ 3-2 & 3-2 & 0-2 \end{pmatrix} \begin{pmatrix} 0-2 & 3-2 \\ 3-2 & 3-2 \\ 3-2 & 0-2 \end{pmatrix} = \begin{pmatrix} 6 & -3 \\ -3 & 6 \end{pmatrix};$$

$$\mathbf{S}'_2 = \begin{pmatrix} -2-(-3) & -4-(-3) \\ -4-(-3) & -2-(-3) \end{pmatrix} \begin{pmatrix} -2-(-3) & -4-(-3) \\ -4-(-3) & -2-(-3) \end{pmatrix} = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix}$$

со следами $\text{Scat}(D'_1) = 12$ и $\text{Scat}(D'_2) = 4$. Матрица межкластерного разброса равна:

$$\mathbf{B}' = \begin{pmatrix} 2 & 2 & 2 & -3 & -3 \\ 2 & 2 & 2 & -3 & -3 \end{pmatrix} \begin{pmatrix} 2 & 2 \\ 2 & 2 \\ 2 & 2 \\ -3 & -3 \\ -3 & -3 \end{pmatrix} = \begin{pmatrix} 30 & 30 \\ 30 & 30 \end{pmatrix}$$

со следом 60. Очевидно, что второй способ кластеризации порождает более компактные кластеры, центроиды которых дальше отстоят друг от друга.

Алгоритм K средних

Проблема K средних является NP-полной, то есть эффективного способа найти глобальный минимум не существует, и приходится прибегать к эвристическим алгоритмам. Самый известный из них обычно называют точно так же: K средних, хотя встречается также название «алгоритм Ллойда». Его набросок приведен в алгоритме 8.1. Алгоритм поочередно разбивает данные, применяя решающее правило ближайшего центроида, и пересчитывает центроиды по разбиению. На рис. 8.11 этот алгоритм продемонстрирован на небольшом наборе данных с тремя кластерами, а в примере 8.4 приведен результат на наборе данных, описывающем свойства различных методов машинного обучения.

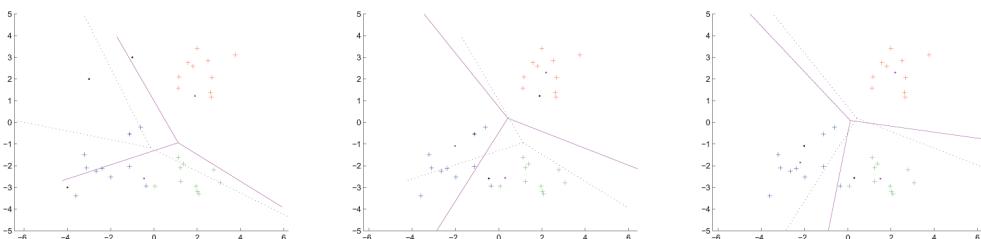


Рис. 8.11. (Слева) Первая итерация с 3 средними на данных с гауссовым распределением. Пунктирными линиями показаны границы диаграммы Вороного для центроидов, инициализированных случайным образом; **фиолетовые** сплошные линии – результат пересчета средних. **(В центре)** Вторая итерация, для которых начальными данными служит предыдущее разбиение. **(Справа)** Третья итерация с устойчивой кластеризацией

Пример 8.4 (кластеризация данных о методах машинного обучения). Обратимся к набору данных ММО о методах машинного обучения, показанному в табл. 1.4 на стр. 52 (полезно также взглянуть на его двухмерную аппроксимацию на рис. 1.7 на стр. 50). Прогнав алгоритм K средних на этих данных при $K = 3$, получим кластеры {Ассоциации, Деревья, Правила}, {GMM, наивная байесовская} и более крупный кластер, содержащий остальные точки. При $K = 4$ больший кластер разбивается на два: {kNN, линейный классификатор, линейная регрессия} и {K средних, логистическая регрессия, метод опорных векторов}; кроме того, GMM перемещается в последний кластер, а наивная байесовская классификация остается в одиночестве.

Можно показать, что одна итерация алгоритма K никогда не увеличивает внутрекластерного разброса, откуда следует, что алгоритм достигает **стационарного состояния**, после которого дальнейшее улучшение невозможно. Стоит отметить, что даже для простейшего набора данных стационарных состояний может быть много.

Алгоритм 8.1. KMeans(D, K) – кластеризация методом K средних с применением евклидова расстояния Dis_2

Вход: данные $D \subseteq \mathbb{R}^d$; число кластеров $K \in \mathbb{N}$.

Выход: средние K кластеров $\mu_1, \dots, \mu_K \in \mathbb{R}^d$.

- 1 случайным образом инициализировать K векторов $\mu_1, \dots, \mu_K \in \mathbb{R}^d$;
 - 2 **repeat**
 - 3 отнести каждую точку $\mathbf{x} \in D$ к кластеру $\arg\min_j \text{Dis}_2(\mathbf{x}, \mu_j)$;
 - 4 **for** $j = 1$ до K **do**
 - 5 $D_j \leftarrow \{\mathbf{x} \in D \mid \mathbf{x}$ отнесена к кластеру $j\}$;
 - 6 $\mu_j = \frac{1}{|D_j|} \sum_{\mathbf{x} \in D_j} \mathbf{x}$;
 - 7 **end**
 - 8 **until** μ_1, \dots, μ_K перестают изменяться;
 - 9 **return** μ_1, \dots, μ_K ;
-

Пример 8.5 (стационарные состояния кластеризации). Рассмотрим задачу разбиения множества чисел $\{8, 44, 50, 58, 84\}$ на два кластера. Всего возможны четыре разбиения, которые мог бы найти алгоритм 2 средних: $\{8\}$, $\{44, 50, 58, 84\}$; $\{8, 44\}$, $\{50, 58, 84\}$; $\{8, 44, 50\}$, $\{58, 84\}$ и $\{8, 44, 50, 58\}$, $\{84\}$. Легко проверить, что каждое из них является стационарным состоянием алгоритма 2 средних и, следовательно, будет найдено при подходящей инициализации. Оптимальна только первая кластеризация, именно она минимизирует полный внутрикластерный разброс.

В общем случае, хотя алгоритм K средних сходится к стационарному состоянию за конечное время, не гарантируется, что найденное состояние является глобальным минимумом, и даже неизвестно, как далеко от него оно отстоит. На рис. 8.12 показано, что после неудачной инициализации центроидов может быть найдено неоптимальное решение. На практике рекомендуется прогнать алгоритм несколько раз и выбрать решение с наименьшим внутрикластерным разбросом.

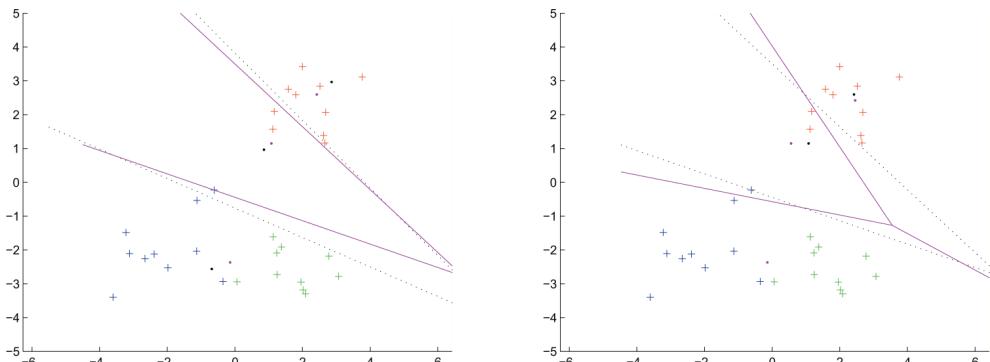


Рис. 8.12. (Слева) Первая итерация алгоритма 3 средних на тех же данных, что на рис. 8.11, но с центроидами, инициализированными иначе. **(Справа)** Алгоритм 3 средних сорвался к неоптимальной кластеризации

Кластеризация вокруг медоидов

Нетрудно модифицировать алгоритм K средних для другой метрики; отметим, что при этом изменится также минимизируемая целевая функция. В алгоритме 8.2 приведен алгоритм ***K* медоидов**, в котором дополнительно требуется, чтобы каждый эталон совпадал с какой-то точкой набора данных. Отметим, что для вычисления медоида кластера требуется перебрать все пары точек (тогда как для вычисления средней точки нужен лишь один проход по данным), для больших наборов это может оказаться вычислительно нереализуемым. Алгоритм 8.3 предлагает альтернативу – ***разбиение по медоидам (PAM)***, – в которой делается попытка локально улучшить кластеризацию, меняя медоиды местами с другими точками. Качество кластеризации Q вычисляется как сумма расстояний от каждой точки до ближайшего к ней медоида. Отметим, что существует $k(n - k)$ пар, состоящих из медоида и немедоида, а для вычисления Q необходимо перебрать $n - k$ точек, поэтому вычислительная сложность одной итерации квадратично зависит от количества точек в наборе данных. Для больших наборов можно прогнать алгоритм **PAM** на небольшой выборке, а Q посчитать по всему набору и повторить эту процедуру несколько раз для различных выборок.

Важным ограничением методов кластеризации, обсуждаемых в этом разделе, является тот факт, что они представляют кластеры только эталонами. При этом не учитывается форма кластеров, и иногда это приводит к результатам, противоречащим интуиции. Два набора данных на рис. 8.13 отличаются только масштабом по оси y . Тем не менее алгоритм K средних находит для них совершенно разные кластеризации. Это нельзя считать недостатком алгоритма K средних как такового, поскольку на рис. 8.13 справа оба центроида отстоят дальше друг от друга, чем в предполагаемом решении, и потому представляют лучшее решение в смысле критерия (8.3). Настоящая же проблема заключается в том, что в данном случае мы хотели бы оценить не только центроиды, но и «форму» кластеров, а значит, принимать во внимание не только следы матриц разброса. Мы обсудим эту тему в следующей главе.

Алгоритм 8.2. *KMedoids(D, K, Dis)* – кластеризация методом K медоидов с произвольной метрикой Dis

Вход: данные $D \subseteq \mathcal{X}$; число кластеров $K \in \mathbb{N}$; метрика $\text{Dis} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$

Выход: K медоидов $\mu_1, \dots, \mu_K \in D$, представляющие прогнозистическую кластеризацию \mathcal{X} .

1 случайным образом выбрать K точек $\mu_1, \dots, \mu_K \in D$;

2 **repeat**

3 отнести каждую точку $x \in D$ к кластеру $\arg\min_j \text{Dis}(x, \mu_j)$;

4 **for** $j = 1$ до K **do**

5 $D_j \leftarrow \{x \in D \mid x \text{ отнесена к кластеру } j\}$;

6 $\mu_j = \arg\min_{x \in D_j} \sum_{x' \in D_j} \text{Dis}(x, x')$;

7 **end**

8 **until** μ_1, \dots, μ_K перестают изменяться;

9 **return** μ_1, \dots, μ_K ;

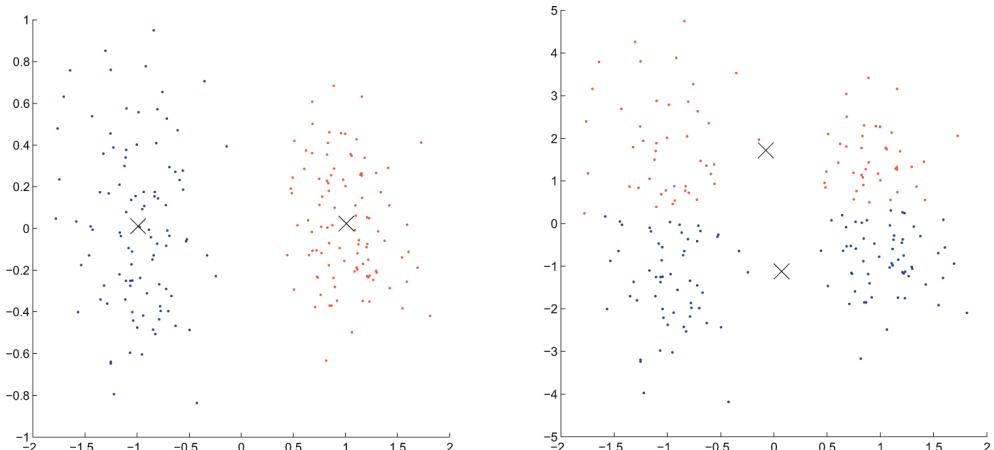


Рис. 8.13. (Слева) На этих данных алгоритм 2 средних обнаруживает правые кластеры.
(Справа) После изменения масштаба по оси y у этой конфигурации межкластерный разброс оказался выше, чем у предполагаемой

Алгоритм 8.3. $\text{PAM}(D, K, \text{Dis})$ – кластеризация методом разбиения по медоидам с произвольной метрикой Dis

Вход: данные $D \subseteq \mathcal{X}$; число кластеров $K \in \mathbb{N}$; метрика $\text{Dis} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$.
Выход: K медоидов $\mu_1, \dots, \mu_K \in D$, представляющие прогностическую кластеризацию \mathcal{X} .

- 1 случайным образом выбрать K точек $\mu_1, \dots, \mu_K \in D$;
- 2 **repeat**
- 3 отнести каждую точку $\mathbf{x} \in D$ к кластеру $\operatorname{argmin}_j \text{Dis}(\mathbf{x}, \mu_j)$;
- 4 **for** $j = 1$ до K **do**
- 5 $D_j \leftarrow \{\mathbf{x} \in D \mid \mathbf{x}$ отнесена к кластеру $j\}$;
- 6 **end**
- 7 $Q = \sum_j \sum_{\mathbf{x} \in D_j} \text{Dis}(\mathbf{x}, \mu_j)$;
- 8 **for** каждого медоида \mathbf{m} и каждого немедоида \mathbf{o} **do**
- 9 вычислить улучшение Q в результате замены \mathbf{m} на \mathbf{o} ;
- 10 **end**
- 11 выбрать пару, дающую максимальное улучшение, и произвести перестановку;
- 12 **until** дальнейшие улучшения невозможны;
- 13 **return** μ_1, \dots, μ_K ;

Силуэты

Как можно было бы обнаружить плохое качество кластеризации на рис. 8.13 справа? Интересный ответ дают силуэты. Для любой точки \mathbf{x}_i обозначим $d(\mathbf{x}_i, D_j)$ среднее расстояние от \mathbf{x}_i до точек в кластере D_j , а $j(i)$ – индекс кластера, которо-

му принадлежит \mathbf{x}_i . Пусть далее $a(\mathbf{x}_i) = d(\mathbf{x}_i, D_{j(i)})$ – среднее расстояние от точки \mathbf{x}_i до точек в ее собственном кластере $D_{j(i)}$, и пусть $b(\mathbf{x}_i) = \min_{k \neq j(i)} d(\mathbf{x}_i, D_k)$ – среднее расстояние до точек в соседнем с ней кластере. Мы хотели бы, чтобы $a(\mathbf{x}_i)$ было значительно меньше, чем $b(\mathbf{x}_i)$, но гарантировать это нельзя. Поэтому можно взять разность $b(\mathbf{x}_i) - a(\mathbf{x}_i)$ как меру того, насколько «хорошо кластеризована» точка \mathbf{x}_i , и разделить эту величину на $b(\mathbf{x}_i)$, чтобы получилось число, меньшее или равное 1.

Однако может случиться, что $a(\mathbf{x}_i) > b(\mathbf{x}_i)$, и в этом случае разность $b(\mathbf{x}_i) - a(\mathbf{x}_i)$ будет отрицательна. Это ситуация, когда в среднем члены соседнего кластера ближе к точке \mathbf{x}_i , чем члены ее собственного кластера. В этом случае для получения нормированного значения мы делим разность на $a(\mathbf{x}_i)$. Таким образом, мы приходим к следующему определению:

$$s(\mathbf{x}_i) = \frac{b(\mathbf{x}_i) - a(\mathbf{x}_i)}{\max(a(\mathbf{x}_i), b(\mathbf{x}_i))}. \quad (8.4)$$

Затем для получения *силуэта* необходимо рассортировать и нанести на график $s(\mathbf{x})$ для каждого объекта, сгруппировав данные по кластерам. На рис. 8.14 показаны примеры для двух кластеризаций, изображенных на рис. 8.13. В данном конкретном случае мы использовали для построения силуэта квадрат евклидова расстояния, но сам метод применим и к другим метрикам. Отчетливо видно, что первая кластеризация гораздо лучше второй. В дополнение к графическому представлению мы можем вычислить средние значения силуэта в каждом кластере и по всему набору данных.

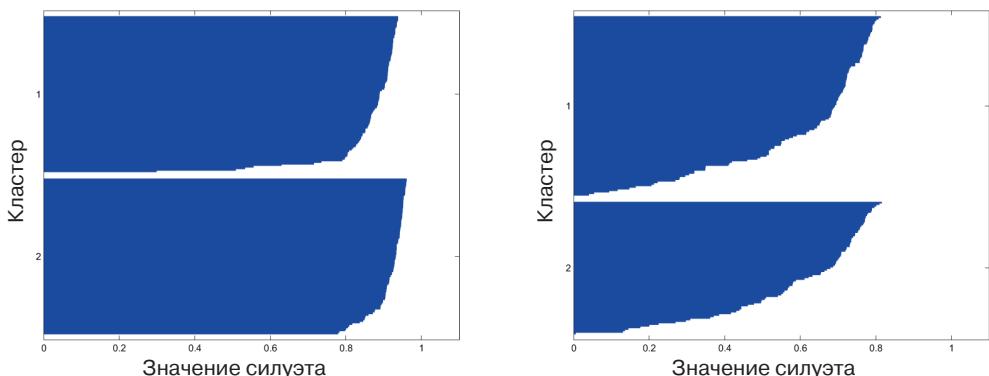


Рис. 8.14. (Слева) Силуэт для кластеризации, показанной на рис. 8.13 слева, с использованием евклидовой метрики. Почти для всех точек значение $s(\mathbf{x})$ велико, то есть в среднем они гораздо ближе к другим членам своего кластера, чем к членам соседнего кластера. **(Справа)** Силуэт для кластеризации, показанной на рис. 8.13 справа, гораздо менее убедителен

8.5 Иерархическая кластеризация

В методах, обсуждавшихся в предыдущем разделе, для представления прогностической кластеризации – разбиения всего пространства объектов – использовались эталоны. В этом разделе мы рассмотрим методы, позволяющие представить кластеры с использованием деревьев. Мы уже встречались с *кластеризующими деревьями* в разделе 5.3: в них для навигации по пространству объектов используются признаки, как в решающих деревьях, и от понятия расстояния они не зависят. А сейчас мы займемся деревьями, которые называются дендрограммами и определяются исключительно в терминах расстояния. Поскольку в дендрограммах признаки используются лишь косвенно, как основа для вычисления расстояний, то они разбивают предъявленные данные, а не все пространство объектов, и, следовательно, описывают не прогностическую, а дескриптивную кластеризацию.

Пример 8.6 (иерархическая кластеризация данных о методах машинного обучения). Продолжим пример 8.4. Иерархическая кластеризация набора ММО приведена на рис. 8.15. Дерево показывает, что все три логических метода сверху образуют сильно связанный кластер. Если бы мы запросили три кластера, то получили бы кластер логических методов, второй кластер поменьше *{GMM, наивная байесовская}* и все остальное. Если бы мы запросили четыре кластера, то *GMM* и *наивная байесовская* классификация разделились бы, поскольку, если верить дереву, этот кластер связан слабее прочих (отметим, что этот вывод несколько отличается от решения, найденного методом 4 средних). Если бы мы запросили пять кластеров, то был бы построен отдельный кластер *{Линейная регрессия, Линейный классификатор}*. Это иллюстрирует основное достоинство иерархической кластеризации: она не требует фиксировать количество кластеров заранее.

Точное определение дендрограммы формулируется так.

Определение 8.4 (дендрограмма). Пусть имеется набор данных D , *дендрограммой* называется двоичное дерево, в котором листьями являются элементы D . Внутренний узел дерева представляет подмножество элементов в листьях поддерева с корнем в этом узле. Уровень узла – это расстояние между двумя кластерами, представленными потомками этого узла. Для листьев уровень равен 0.

Чтобы это определение заработало, нам нужен способ измерить близость двух кластеров. На первый взгляд, все кажется простым: нужно лишь вычислить расстояние между средними точками кластеров. Однако иногда это приводит к проблемам, обсуждаемым ниже в этом разделе. Кроме того, взятие средних точек в качестве эталонов предполагает использование евклидовой метрики, а мы хотим, чтобы определение были пригодно и для других метрик. Все это привело к понятию функции связи – общего способа превратить попарные расстояния между точками в попарные расстояния между кластерами.

Определение 8.5 (функция связи). Функция связи $L: 2^{\mathcal{X}} \times 2^{\mathcal{X}} \rightarrow \mathbb{R}$ вычисляет расстояние между произвольными подмножествами пространства объектов, если задана метрика $\text{Dis}: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$.

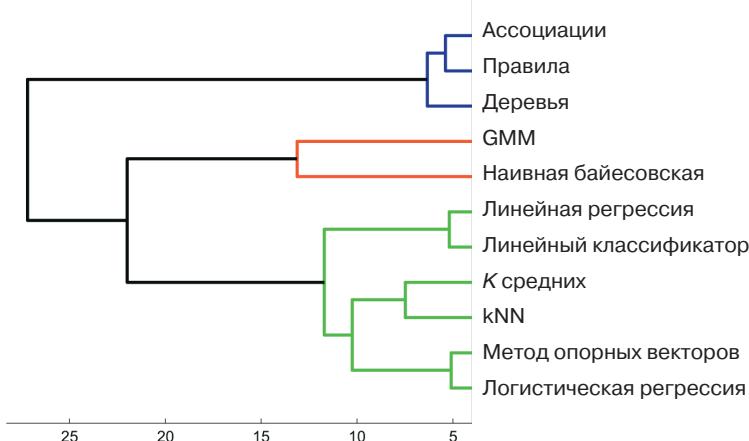


Рис. 8.15. Дендрограмма (напечатана слева направо для простоты восприятия), построенная по иерархической кластеризации данных в табл. 1.4 на стр. 52

Наиболее часто встречаются следующие функции связи.

- | | |
|--------------------------|---|
| Одиночная связь | определяет расстояние между двумя кластерами как <i>наименьшее</i> попарное расстояние между элементами, взятыми из разных кластеров. |
| Полная связь | определяет расстояние между двумя кластерами как <i>наибольшее</i> расстояние между точками из разных кластеров. |
| Средняя связь | определяет расстояние между двумя кластерами как <i>среднее</i> расстояние между точками из разных кластеров. |
| Центроидная связь | определяет расстояние между двумя кластерами как расстояние между средними точками кластеров. |

Математически эти функции можно определить следующим образом:

$$\begin{aligned} L_{\text{single}}(A, B) &= \min_{x \in A, y \in B} \text{Dis}(x, y); \\ L_{\text{complete}}(A, B) &= \max_{x \in A, y \in B} \text{Dis}(x, y); \\ L_{\text{average}}(A, B) &= \frac{\sum_{x \in A, y \in B} \text{Dis}(x, y)}{|A| \cdot |B|}; \\ L_{\text{centroid}}(A, B) &= \text{Dis}\left(\frac{\sum_{x \in A} x}{|A|}, \frac{\sum_{y \in B} y}{|B|}\right). \end{aligned}$$

Понятно, что все эти функции связи дают один и тот же результат для кластеров, состоящих из одного элемента: $L(\{x\}, \{y\}) = \text{Dis}(x, y)$. Однако для более крупных кластеров они расходятся. Например, предположим, что $\text{Dis}(x, y) < \text{Dis}(x, z)$, тогда связь между $\{x\}$ и $\{y, z\}$ во всех четырех случаях различна:

$$\begin{aligned} L_{\text{single}}(\{x\}, \{y, z\}) &= \text{Dis}(x, y); \\ L_{\text{complete}}(\{x\}, \{y, z\}) &= \text{Dis}(x, z); \\ L_{\text{average}}(\{x\}, \{y, z\}) &= (\text{Dis}(x, y) + \text{Dis}(x, z))/2; \\ L_{\text{centroid}}(\{x\}, \{y, z\}) &= \text{Dis}(x, (y + z)/2). \end{aligned}$$

Общий алгоритм построения дендрограммы описан в алгоритме 8.4. Дерево строится, начиная от точек набора данных, вверх, то есть это алгоритм восходящий, или, как еще говорят, *агломеративный*. На каждой итерации алгоритм строит новое разбиение данных, объединяя два ближайших кластера. В общем случае алгоритм НАС дает различные результаты при использовании разных функций связи. Проще всего понять его работу в случае одиночной связи, потому что при этом граф строится путем добавления все более длинных ребер между точками, по одному за раз, так что в конечном итоге будет существовать путь между любыми двумя точками (отсюда и термин «связь»). В любой момент этой процедуры связанными компонентами графа являются найденные на текущей итерации кластеры, а связь последнего найденного кластера равна длине последнего добавленного ребра. Для выполнения иерархической кластеризации с применением одиночной связи нужно по существу вычислить и отсортировать все попарные расстояния между точками наборами данных, что требует времени порядка $O(n^2)$ для n точек. Для других функций связи требуется время не менее $O(n^2 \log n)$. Отметим, что сложность неоптимизированного алгоритма 8.4 составляет $O(n^3)$.

Алгоритм 8.4. $\text{HAC}(D, L)$ – иерархическая агломеративная кластеризация

Вход: данные $D \subseteq \mathcal{X}$; функция связи $L: 2^{\mathcal{X}} \times 2^{\mathcal{X}} \rightarrow \mathbb{R}$, определенная в терминах метрики.

Выход: дендрограмма, представляющая дескриптивную кластеризацию D .

- 1 инициализировать кластеры, включив в каждый по одной точке;
- 2 создать узел уровня 0 для каждого такого кластера;
- 3 **repeat**
- 4 | найти пару кластеров X, Y с наименьшей связью l и объединить их;
- 5 | создать родителя X, Y на уровне l ;
- 6 **until** все точки не окажутся в одном кластере;
- 7 **return** построенное двоичное дерево с уровнями связи;

Пример 8.7 (связь имеет значение). Рассмотрим регулярную сетку из 8 точек, расположенных в двух строках по четыре точки (рис. 8.16). Мы предполагаем, что неопределенности устраниены за счет небольших нерегулярностей. Каждая функция связи объединяет одни и те же кластеры в одном и том же порядке, но сами связи в каждом случае различны. Полная связь создает впечатление, что D далеко отстоит от всех остальных, но если чуть-чуть подвинуть D вправо, то он был бы добавлен к E раньше C. В случае центроидной связи мы видим, что у E такая же связь, как у A и B, а это значит, что A и B не различимы как отдельные кластеры, хотя они и обнаруживаются первыми. В данном случае предпочтительной кажется одиночная связь, потому что она наиболее отчетливо демонстрирует, что никакой значимой кластерной структуры в этом наборе точек нет.

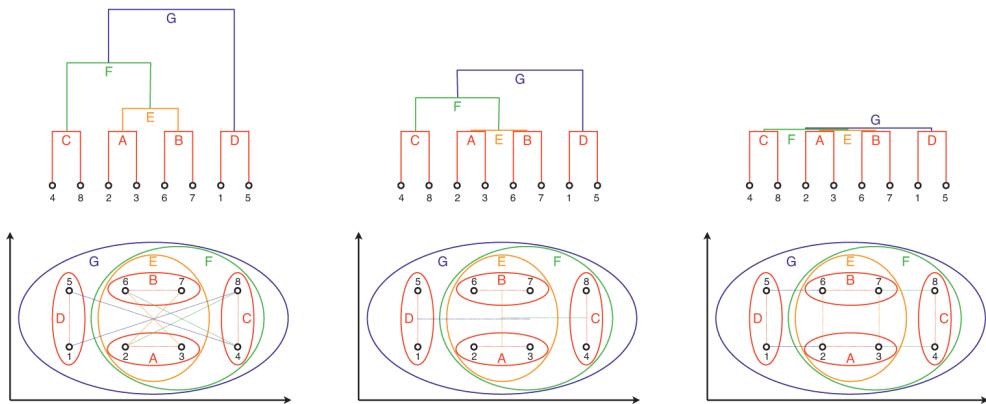


Рис. 8.16. (Слева) Полная связь определяет расстояние между кластерами как наибольшее попарное расстояние между их элементами; пары элементов показаны цветными линиями. Найденную кластеризацию можно представить в виде вложенных разбиений (снизу) или в виде дендрограммы (сверху); уровень горизонтального соединения между кластерами в дендрограмме соответствует длине связующего отрезка. В примере предполагается, что неопределенности устранены за счет небольших нерегулярностей сетки. **(В центре)** Центроидная связь определяет расстояние между кластерами как расстояние между их средними точками. Отметим, что у Е оказывается такая же связь, как у А и В, поэтому два последних кластера по существу исчезают. **(Справа)** Одиночная связь определяет расстояние между кластерами как наименьшее попарное расстояние между их элементами. Дендрограмма практически схлопывается, то есть при данной конфигурации сетки обнаружить значимых кластеров не удалось

Одиночная и полная связи определяют расстояние между кластерами в терминах расстояния между определенной парой точек. Следовательно, они не могут учесть форму кластера, и именно поэтому средняя и центроидная связи могут оказаться предпочтительнее. Однако центроидная связь ведет к интуитивно неочевидным дендрограммам, как показано на рис. 8.17. Проблема в том, что $L(\{1\}, \{2\}) < L(\{1\}, \{3\})$ и $L(\{1\}, \{2\}) < L(\{2\}, \{3\})$, но $L(\{1\}, \{2\}) > L(\{1,2\}, \{3\})$. Первые два неравенства означают, что точки 1 и 2 должны быть объединены в кластер первыми, но третье неравенство говорит, что уровень кластера $\{1,2,3\}$ в дендрограмме оказывается ниже уровня $\{1,2\}$. Центроидная связь нарушает требование **монотонности**, согласно которому из $L(A,B) < L(A,C)$ и $L(A,B) < L(B,C)$ должно следовать, что $L(A,B) < L(A \cup B, C)$ для любых кластеров A , B и C . Остальные три функции связи монотонны (пример также иллюстрирует, почему средняя и центроидная связь – не одно и то же).

При построении дендрограмм следует также иметь в виду, что иерархический метод кластеризации детерминирован и всегда завершается успешным построением кластеров. Взгляните на рис. 8.18, где показан набор из 20 случайных точек с равномерным распределением. Непредвзятоому наблюдателю трудно обнаружить в этих данных хоть какую-нибудь кластерную структуру, однако дендрограмма,

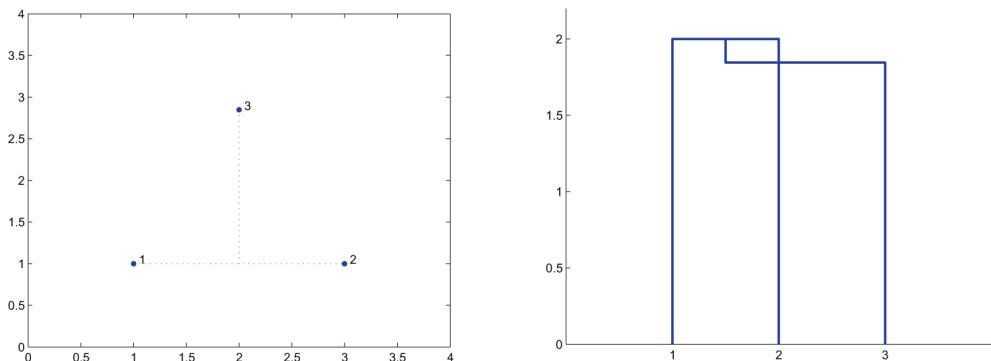


Рис. 8.17. (Слева) Точки 1 и 2 ближе друг к другу, чем к точке 3. Однако расстояние от точки 3 до центроида двух остальных точек меньше любого из попарных расстояний. **(Справа)** Это приводит к уменьшению связи после добавления точки 3 в кластер {1, 2}, а значит, к не-монотонной дендрограмме

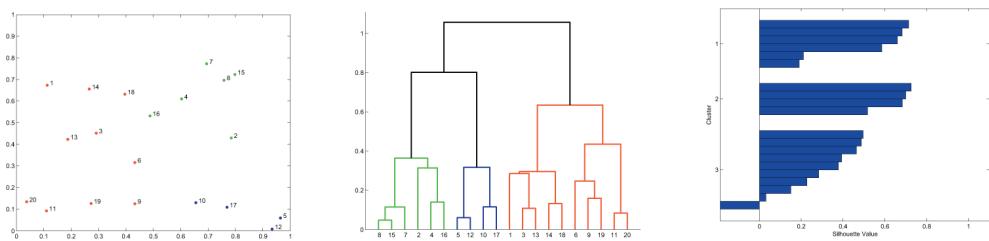


Рис. 8.18. (Слева) 20 случайных точек с равномерным распределением. **(В центре)** Дендрограмма, построенная с использованием полной связи. Все три предложенных ей кластера иллюзорные, поскольку в данных они не наблюдаются. **(Справа)** Быстрое убывание значений силуэта подтверждает отсутствие выраженной кластерной структуры. У точки 18 силуэт отрицательный, потому что в среднем она ближе к зеленым точкам, чем к остальным красным

построенная с помощью функции полной связи и евклидовой метрики, показывает существование трех или четырех отчетливых кластеров. Но, взглянувшись пристальнее, мы замечаем, что уровни связи очень близки друг к другу в нижней части дерева, а тот факт, что с ростом дерева они становятся выше, объясняется главным образом использованием полной связи, которая рассчитывается на основе максимального попарного расстояния. Силуэт на рис. 8.18 (справа) подтверждает, что кластерная структура не слишком развита. По существу, мы здесь наблюдаем своеобразную, свойственную кластеризации форму переобучения, уже знакомую нам по другим древовидным моделям, обсуждавшимся в главе 5. Кроме того, дендрограммы – как и другие древовидные модели – обладают высокой дисперсией, то есть небольшие изменения данных могут привести к значительному изменению дендрограммы.

В заключение отмечу, что иерархические методы кластеризации обладают тем несомненным достоинством, что количество кластеров заранее задавать не нужно. Однако за это приходится расплачиваться высокой стоимостью вычислений. И кроме того, теперь мы должны выбирать не только метрику, но и функцию связи.

8.6 От ядер к расстояниям

В разделе 7.5 мы обсудили, как с помощью ядер можно значительно расширить возможности линейных моделей. Напомним, что ядром называется функция $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$, которая вычисляет скалярное произведение в некотором пространстве признаков, не строя векторов признаков $\phi(\mathbf{x})$ явно. Любой метод обучения, который можно определить исключительно в терминах скалярных произведений обучающих примеров, пригоден для такого «перехода к ядру». В силу наличия тесной связи между евклидовой метрикой и скалярными произведениями этот «трюк с ядром» можно применить ко многим метрическим методам обучения.

В основе лежит тот факт, что евклидово расстояние можно записать в виде скалярных произведений:

$$\text{Dis}_2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{(\mathbf{x} - \mathbf{y}) \cdot (\mathbf{x} - \mathbf{y})} = \sqrt{\mathbf{x} \cdot \mathbf{x} - 2\mathbf{x} \cdot \mathbf{y} + \mathbf{y} \cdot \mathbf{y}}.$$

Из этой формулы с очевидностью следует, что расстояние между \mathbf{x} и \mathbf{y} убывает с ростом скалярного произведения $\mathbf{x} \cdot \mathbf{y}$, а это наводит на мысль, что и само скалярное произведение может служить мерой сходства. Однако оно не инвариантно относительно параллельных переносов, поскольку зависит от положения начала координат. В обеспечение такой инвариантности вносят вклад оба члена $\mathbf{x} \cdot \mathbf{x}$ и $\mathbf{y} \cdot \mathbf{y}$. Заменяя скалярное произведение ядром κ , мы можем построить такое ядерное расстояние:

$$\text{Dis}_\kappa(\mathbf{x}, \mathbf{y}) = \sqrt{\kappa(\mathbf{x}, \mathbf{x}) - 2\kappa(\mathbf{x}, \mathbf{y}) + \kappa(\mathbf{y}, \mathbf{y})}. \quad (8.5)$$

Как выясняется, Dis_κ определяет псевдометрику (см. определение 8.2), если κ – положительно полуопределенное ядро¹.

Для иллюстрации рассмотрим алгоритм 8.5, который адаптирует алгоритм K средних (алгоритм 8.1) к использованию ядерного расстояния. Таким образом, этот алгоритм производит кластеризацию согласно нелинейному расстоянию в пространстве объектов, которое соответствует евклидову расстоянию в неявном пространстве признаков. Однако возникает одна сложность: дело в том, что теорема 8.1 неприменима к нелинейным расстояниям, поэтому мы не можем постро-

¹ Она является метрикой, только если отображение признаков инъективно. Предположим, что это не так, тогда различные точки \mathbf{x} и \mathbf{y} отображаются в один и тот же вектор признаков $\phi(\mathbf{x}) = \phi(\mathbf{y})$, откуда следует, что $\kappa(\mathbf{x}, \mathbf{x}) - 2\kappa(\mathbf{x}, \mathbf{y}) + \kappa(\mathbf{y}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}) - 2\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) + \phi(\mathbf{y}) \cdot \phi(\mathbf{y}) = 0$.

ить средние точки кластеров в пространстве объектов. Поэтому в алгоритме 8.5 кластеризация трактуется как разбиение, а не как набор эталонов. Следовательно, отнесение каждой точки \mathbf{x} к ближайшему кластеру (шаг 3) теперь является операцией квадратичной сложности, так как для каждого кластера необходимо вычислить сумму расстояний от \mathbf{x} до всех его членов. Для алгоритма K средних сложность этого шага линейно зависит от $|D|$. Существует другой способ превратить скалярные произведения в расстояния. Поскольку скалярное произведение можно записать в виде $\|\mathbf{x}\| \cdot \|\mathbf{y}\| \cos \theta$, где θ – угол между векторами \mathbf{x} и \mathbf{y} , то можно определить *косинусоидальную меру сходства* как

$$\cos \theta = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} = \frac{\mathbf{x} \cdot \mathbf{y}}{\sqrt{(\mathbf{x} \cdot \mathbf{x})(\mathbf{y} \cdot \mathbf{y})}}. \quad (8.6)$$

Алгоритм 8.5. Kernel-KMeans(D, K) – кластеризация методом K средних с использованием ядерного расстояния Dis_κ

Вход: данные $D \subseteq \mathcal{X}$; число кластеров $K \in \mathbb{N}$.
Выход: K -разбиение $D_1 \cup \dots \cup D_K = D$.

- 1 случайным образом инициализировать K кластеров D_1, \dots, D_K ;
- 2 **repeat**
- 3 отнести каждую точку $\mathbf{x} \in D$ к кластеру $\operatorname{argmin}_j \frac{1}{|D_j|} \sum_{\mathbf{y} \in D_j} \text{Dis}_\kappa(\mathbf{x}, \mathbf{y})$;
- 4 **for** $j = 1$ до K **do**
- 5 $D_j \leftarrow \{\mathbf{x} \in D | \mathbf{x}$ отнесена к кластеру $j\}$;
- 6 **end**
- 7 **until** D_1, \dots, D_K перестают изменяться;
- 8 **return** D_1, \dots, D_K ;

Косинусоидальное сходство отличается от евклидова расстояния тем, что не зависит от длины векторов \mathbf{x} и \mathbf{y} . С другой стороны, оно не инвариантно относительно параллельных переносов, но придает специальное значение началу координат. Одна из возможных интерпретаций заключается в том, что векторы проецируются на единичную сферу с центром в начале координат, и расстояние между ними измеряется по поверхности этой сферы. Косинусоидальное сходство обычно преобразуют в метрику, рассматривая величину $1 - \cos \theta$. Будучи определено исключительно в терминах скалярных произведений, оно легко включается в ядерные алгоритмы, играя роль евклидовой метрики.

8.7 Метрические модели: итоги и литература для дальнейшего чтения

Наряду с линейными метрические модели образуют вторую группу моделей, апеллирующих к геометрической интуиции. Литература по метрическим моде-

лям богата и разнообразна; в этой главе я стремился познакомить вас только с основными интуитивными соображениями.

- ☞ В разделе 8.1 мы ввели в рассмотрение наиболее употребительные метрики: метрику Минковского, или p -норму, и ее частные случаи – евклидово расстояние ($p = 2$) и манхэттенское расстояние ($p = 1$), расстояние Хэмминга, которое подсчитывает, в скольких разрядах отличаются битовые векторы, и расстояние Махalanобиса, которое устраниет корреляцию и нормирует признаки (Mahalanobis, 1936). Можно рассмотреть и другие метрики при условии, что они удовлетворяют условиям, перечисленным в определении 8.2.
- ☞ В разделе 8.2 изучались ключевые понятия: соседи и эталоны. В роли эталонов могут выступать как центроиды – центры масс в соответствии с выбранной метрикой, так и медоиды – самые «центральные» точки выборки. Чаще всего в качестве центроида берут среднее арифметическое, которое минимизирует сумму квадратов евклидовых расстояний до всех остальных точек. Возможны и другие определения центроида, но они вычислительно сложнее: например, геометрическая медиана – это точка, минимизирующая сумму евклидовых расстояний, однако для ее нахождения не существует замкнутой формулы. Сложность нахождения медоида всегда квадратична безотносительно к используемой метрике. Далее мы рассмотрели решающее правило ближайшего соседа и проанализировали различие между решающими границами, найденными с помощью ближайшего эталона при использовании 2-нормы и 1-нормы, а затем показали, как их можно уточнить, перейдя к решающему правилу 2 ближайших эталонов.
- ☞ В разделе 8.3 мы обсудили модель ближайшего соседа, в которой эталонами являются просто сами обучающие данные. Эта модель очень широко применяется для классификации, а ее истоки можно проследить до работы Fix, Hodges (1951). Несмотря на простоту, можно доказать, что при достаточноном объеме обучающих данных частота ошибок в этой модели не более чем вдвое превышает оптимальную (Cover, Hart, 1967). Классификатор по 1 ближайшему соседу имеет низкое смещение, но высокую дисперсию; за счет увеличения числа соседей, голоса которых агрегируются, мы можем уменьшить дисперсию, но одновременно увеличить смещение. Решающее правило ближайшего соседа можно также применять к вещественнонезначимым целевым переменным и – более общо – к любым задачам, в которых имеется подходящий агрегатор нескольких целевых значений.
- ☞ В разделе 8.4 рассмотрен ряд алгоритмов метрической кластеризации с применением средних арифметических или медоидов. Алгоритм K средних – это простой эвристический подход к решению проблемы K средних, первоначально он был предложен в 1957 году и иногда называется алгоритмом Ллойда (Lloyd, 1982). Он зависит от начальной конфигурации и легко может сойтись к неправильному стационарному состоянию. Мы познакомились также с алгоритмами K медоидов и разбиения по медоидам.

дам, последний был предложен в работе Kaufman, Rousseeuw (1990). Их вычислительная сложность велика вследствие использования медоидов. Силуэты (Rousseeuw, 1987) – полезная техника, позволяющая проверить, верно ли, что точки в среднем ближе к другим членам своего кластера, чем к членам соседних кластеров. Подробнее об этих и других методах кластеризации можно прочитать в работе Jain, Murty, Flynn (1999).

- ☞ Все рассмотренные ранее методы кластеризации приводят к разбиению пространства объектов и потому являются прогностическими. Иерархическая же кластеризация, ставшая предметом обсуждения в разделе 8.5, применима только к предъявленным данным и, значит, является дескриптивной. Ее достоинством является тот факт, что кластеризация строится в виде дендрограммы, то есть нет необходимости задавать количество кластеров заранее, его можно выбрать путем анализа дендрограммы. Однако это вычислительно накладный метод, непригодный для больших наборов данных. Кроме того, не всегда очевидно, какую взять функцию связи.
- ☞ Наконец, в разделе 8.6 мы кратко обсудили ядерные метрики и привели один пример: ядерный алгоритм K средних. Использование неевклидовой метрики приводит к квадратичной сложности из-за необходимости пересчитывать кластеры на каждой итерации.



Вероятностные модели

Третье и последнее семейство моделей машинного обучения, рассматриваемое в этой книге, – вероятностные модели. Мы уже видели, как полезны могут быть вероятности для выражения ожиданий модели относительно класса предъявленного объекта. Например, в *дереве оценивания вероятностей* (раздел 5.2) к каждому листу присоединено распределение вероятностей классов, а объект, попавший в определенный лист древовидной модели, помечается хранящимся в нем распределением. Аналогично калиброванная линейная модель преобразует расстояние до решающей границы в вероятность класса (раздел 7.4). Это все примеры так называемых *дискриминантных* вероятностных моделей. Они моделируют апостериорное распределение вероятностей $P(Y|X)$, где Y – целевая переменная, а X – признаки. Иначе говоря, они возвращают распределение вероятностей Y при заданном X .

Другой важный класс вероятностных моделей – *порождающие* модели. Они моделируют совместное распределение $P(Y, X)$ целевой переменной Y и вектора признаков X . Зная совместное распределение, мы можем вывести любое условное или маргинальное распределение с участием тех же случайных величин. В частности, поскольку $P(X) = \sum_y P(Y = y, X)$, то апостериорное распределение можно получить как

$$P(Y|X) = \frac{P(Y, X)}{\sum_y P(Y = y, X)}.$$

С другой стороны, порождающие модели можно описать функцией правдоподобия $P(X|Y)$, так как $P(Y, X) = P(X|Y)P(Y)$, а распределение целевой переменной, или априорное распределение, можно легко оценить или постулировать. Такие модели называются «порождающими», потому что мы можем сделать выборку из совместного распределения для получения новых данных вместе с их метками. Альтернативно можно использовать $P(Y)$ для выборки класса и $P(X|Y)$ для выборки объекта этого класса – это было показано на примере почтового спама (стр. 41). Напротив, дискриминантная модель, например дерево оценивания вероятностей или линейный классификатор, моделирует $P(Y|X)$, но не $P(X)$, и потому может использоваться для пометки данных, но не для порождения новых.

Поскольку порождающие модели умеют делать все то же, что дискриминантные, они могут показаться более предпочтительными. Однако же они не лише-

ны недостатков. Прежде всего отметим, что объем памяти, необходимой для хранения совместного распределения, экспоненциально растет вместе с числом признаков. Поэтому приходится делать упрощающие предположения, например о независимости признаков. А если такие предположения в конкретной предметной области неверны, то результаты могут оказаться неточными. Самое распространенное критическое замечание, выдвигаемое против порождающих моделей, – это то, что точность моделирования $P(X)$ на самом деле, возможно, достигается ценой менее точного моделирования $P(Y|X)$. Однако в этом вопросе еще нет окончательной ясности, и, безусловно, существуют ситуации, когда знание $P(X)$ дает желанное дополнительное понимание предметной области. Например, нас, скорее всего, не так уж и заботит неправильная классификация тех объектов, которые маловероятны согласно распределению $P(X)$.

Одна из самых привлекательных черт вероятностного взгляда на вещи – тот факт, что он позволяет рассматривать обучение как процесс уменьшения неопределенности. Например, если априорное распределение по классам равномерно, то до получения каких-либо знаний о классифицируемом объекте неопределенность в части того, какому классу он принадлежит, максимальна. Если же апостериорное распределение после наблюдения объекта не столь равномерно, значит, мы уменьшили эту неопределенность. Этот процесс можно повторять всякий раз после прихода новой информации, используя апостериорное распределение, полученное на предыдущем шаге, как априорное для следующего. В принципе, этот процесс применим к любой неизвестной величине.

Пример 9.1 (спам или неспам?). Допустим, мы хотим оценить вероятность θ того, что произвольно взятое почтовое сообщение – спам, чтобы можно было использовать подходящее априорное распределение. Напрашивющееся решение – проверить n сообщений, определить, сколько из них спамных – d , и положить $\hat{\theta} = d/n$; для этого не нужны никакие сложные статистики. Однако, хотя это наиболее правдоподобная оценка θ – оценка апостериорного максимума (MAP), пользуясь терминологией, введенной на стр. 40, это не означает, что все прочие значения θ следуют полностью исключить. Мы моделируем это распределением вероятности θ , которое обновляется всякий раз при поступлении новой информации. На рис. 9.1 показано распределение, которое все сильнее и сильнее смещается в сторону спама.

Явное моделирование апостериорного распределения параметра θ обладает рядом достоинств, которые обычно ассоциируются с «байесовским» взглядом.

- ☞ Мы можем точно охарактеризовать оставшуюся неопределенность оценки, количественно оценив размах апостериорного распределения.
- ☞ Мы можем получить порождающую модель для параметра, произведя выборку из апостериорного распределения, которая содержит намного больше информации, чем может предложить сводная статистика типа MAP, – так, вместо использования единственного сообщения с $\theta = \theta_{\text{MAP}}$ порождающая модель может содержать ряд сообщений с θ , полученными в результате выборки из апостериорного распределения.

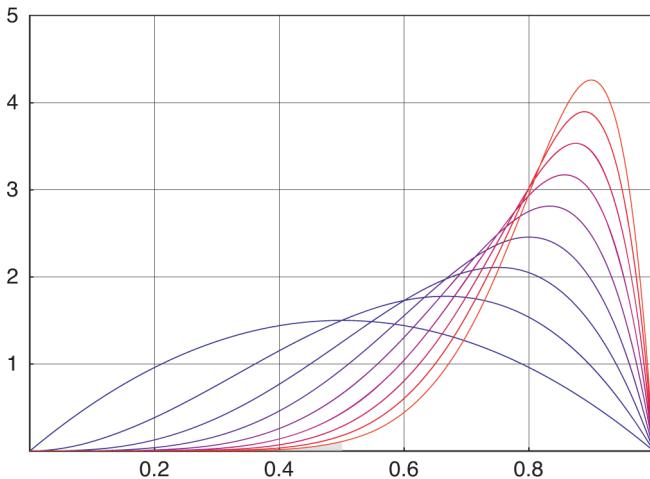


Рис. 9.1. При проверке каждого почтового сообщения мы уменьшаем неопределенность, связанную с априорной вероятностью спама θ . После просмотра двух сообщений, из которых одно оказалось спамом, возможные значения θ характеризуются симметричным распределением с центром 1/2. Если мы проверим три, четыре, ..., десять сообщений, и все они окажутся спамом, то распределение сужается и сдвигается все правее. Было бы естественно ожидать, что распределение для n сообщений достигает максимума при $\hat{\theta}_{\text{MAP}} = (n - 1)/n$ (например, $\hat{\theta}_{\text{MAP}} = 0.8$ при $n = 5$); однако асимметричные распределения, подобные этому, содержат информацию, которую невозможно выразить одним числом, скажем, средним или максимумом

- ☞ Мы можем количественно оценить вероятность утверждений типа «сообщения смешены в сторону хороших» (крохотная заштрихованная область на рис. 9.1 показывает, что после наблюдения одного хорошего и девяти спамных сообщений эта вероятность очень мала – примерно 0.6%).
- ☞ Мы можем использовать одно из этих распределений для описания априорных гипотез: например, если мы полагаем, что доли спама и неспама 50–50, то можем взять в качестве априорного распределение для $n = 2$ (на рис. 9.1 оно самое нижнее, симметричное)¹.

Ключевой момент – что *вероятности необязательно интерпретировать как оценки относительных частот, они могут нести и более общий смысл: степень доверия (возможно, субъективная)*. Следовательно, мы можем связать распределение вероятностей практически с чем угодно: не только с признаками и целями,

¹ Статистики называют априорное распределение, имеющее такую же форму, как апостериорное, *сопряженным априорным* – в данном случае мы использовали бета-распределение, сопряженное биномиальному. Сопряженные априорные распределения не только упрощают математический аппарат, но также допускают интуитивно более понятные интерпретации; в данном случае мы делаем вид, что уже проверили два сообщения, одно из которых оказалось спамом, – очень полезная идея, которую мы уже фактически применяли в разделе 2.3 в форме поправки Лапласа.

но и с параметрами моделей и даже самими моделями. Например, в только что приведенном примере мы рассматривали распределение $P(\theta|D)$, где D представляет данные (то есть классы проверяемых почтовых сообщений).

С вероятностными моделями связано важное понятие *оптимальности по Байесу*. Классификатор называется оптимальным по Байесу, если он всегда назначает объекту x класс $\operatorname{argmax}_y P^*(Y = y|X = x)$, где P^* обозначает истинное апостериорное распределение. И хотя на практике истинное распределение почти никогда не известно, существует несколько способов конкретизации этого понятия. Например, мы можем поставить эксперименты с искусственно сгенерированными данными, для которых сами выберем истинное распределение: это позволяет экспериментально оценить, насколько качество модели близко к оптимальному по Байесу. С другой стороны, при выводе вероятностного метода обучения обычно делаются предположения об истинном распределении, которые позволяют теоретически доказать, что модель будет оптимальной по Байесу, если эти предположения выполняются. Например, ниже в этой главе мы сформулируем условия, при которых базовый линейный классификатор является оптимальным по Байесу. Таким образом, это свойство лучше всего рассматривать как мерилом качества вероятностных моделей.

Поскольку многие модели, рассмотренные в предыдущих главах, способны оценивать вероятности классов и потому являются дискриминантными вероятностными моделями, стоит отметить, что выбор конкретной модели, который часто так и называют – *выбор модели*, необязательно означает байесовскую оптимальность – даже если выбранная модель оказывается наилучшей для истинного распределения. Для иллюстрации предположим, что m^* – наилучшее дерево оценивания вероятностей, которое мы обучили на достаточном объеме данных. С помощью m^* мы могли бы предсказать класс $\operatorname{argmax}_y P(Y = y|M = m^*, X = x)$ для объекта x , где M – случайная величина над классом моделей, из которого выбрана модель m^* . Однако эти предсказания необязательно оптимальны по Байесу, потому что

$$\begin{aligned} P(Y|X = x) &= \sum_{m \in M} P(Y, M = m|X = x) && \text{путем маргинализации по } M; \\ &= \sum_{m \in M} P(Y|M = m, X = x)P(M = m|X = x) && \text{по цепному правилу;} \\ &= \sum_{m \in M} P(Y|M = m, X = x)P(M = m) && \text{в силу независимости } M \text{ и } X. \end{aligned}$$

Здесь $P(M)$ можно интерпретировать как апостериорное распределение моделей после наблюдения обучающих данных (следовательно, модель МАР имеет вид $m^* = \operatorname{argmax}_m P(M = m)$). Последнее выражение в приведенном выше выводе означает, что мы должны усреднить предсказания по всем моделям, приспав им веса в соответствии с апостериорными вероятностями. Очевидно, что это распределение совпадает с $P(Y|M = m^*, X = x)$, только если $P(M)$ равно нулю для всех моделей, кроме m^* , то есть если мы видели достаточно обучающих данных, чтобы исключить все модели, кроме одной. Понятно, что это нереалистичное предположение¹.

¹ Отметим, что на самом деле не требуется, чтобы оба распределения совпадали, достаточно, чтобы у них был одинаковый максимум.

Эта глава устроена следующим образом. В разделе 9.1 мы увидим некоторые полезные связи между геометрическим и вероятностным взглядом на вещи, которые проявляются, когда признаки имеют нормальное распределение. Как уже отмечалось, это позволит нам сформулировать условия, при которых базовый линейный классификатор является оптимальным по Байесу. В разделе 8.2 мы рассмотрим случай категориальных признаков, который ведет к хорошо известному наивному байесовскому классификатору. В разделе 9.3 мы вернемся к линейному классификатору, но взглянем на него с вероятностной точки зрения, это позволит разработать новый алгоритм обучения с явно поставленной целью оптимизировать апостериорную вероятность примеров. В разделе 9.4 обсуждаются способы учета скрытых переменных. Наконец, в разделе 9.5 мы вкратце познакомимся с методами обучения на основе сжатия, которым можно придать вероятностную интерпретацию с помощью понятий из теории информации.

9.1 Нормальное распределение и его геометрические интерпретации

Мы можем установить связь между вероятностными и геометрическими моделями, рассмотрев распределения вероятности, определенные в евклидовых пространствах. Самыми известными из них являются *нормальные распределения*, называемые также *гауссианами* (в замечании 9.1 приведены важнейшие факты об одномерных и многомерных нормальных распределениях). Начнем с рассмотрения одномерного случая с двумя классами. Предположим, что значения $x \in \mathbb{R}$ следуют *смесовой модели* (mixture model), то есть у каждого класса имеется собственное распределение вероятности (*компонент* смесовой модели). Будем предполагать гауссову смесовую модель, когда обе компоненты смеси – гауссианы. Тогда имеем:

$$P(x|\oplus) = \frac{1}{\sqrt{2\pi}\sigma^{\oplus}} \exp\left(-\frac{1}{2}\left[\frac{x-\mu^{\oplus}}{\sigma^{\oplus}}\right]^2\right);$$

$$P(x|\ominus) = \frac{1}{\sqrt{2\pi}\sigma^{\ominus}} \exp\left(-\frac{1}{2}\left[\frac{x-\mu^{\ominus}}{\sigma^{\ominus}}\right]^2\right),$$

где μ^{\oplus} и Σ^{\oplus} – среднее и стандартное отклонения положительного класса, а μ^{\ominus} и Σ^{\ominus} – среднее и стандартное отклонения отрицательного класса. Это дает следующее отношение правдоподобия:

$$\text{LR}(x) = \frac{P(x|\oplus)}{P(x|\ominus)} = \frac{\sigma^{\ominus}}{\sigma^{\oplus}} \exp\left(-\frac{1}{2}\left[\left(\frac{x-\mu^{\oplus}}{\sigma^{\oplus}}\right)^2 - \left(\frac{x-\mu^{\ominus}}{\sigma^{\ominus}}\right)^2\right]\right). \quad (9.1)$$

Одномерное нормальное, или гауссово, распределение имеет следующую функцию плотности вероятности:

$$P(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \frac{1}{E} \exp\left(-\frac{1}{2}\left[\frac{x-\mu}{\sigma}\right]^2\right) = \frac{1}{E} \exp(-z^2/2), \quad E = \sqrt{2\pi}\sigma.$$

Это распределение имеет два параметра: μ – среднее, или математическое, ожидание, оно же медиана (то есть точка, которая делит пополам площадь под кривой функции плотности) и мода (то есть точка, в которой функция плотности достигает максимума), и σ – стандартное отклонение, определяющее ширину колоколообразной кривой.

$z = (x - \mu)/\sigma$ называется ***z-оценкой***, ассоциированной с x ; она измеряет количество стандартных отклонений, укладывающихся между x и средним (сама она имеет среднее 0 и стандартное отклонение 1). Отсюда следует, что $P(x|\mu, \sigma) = (1/\sigma)P(z|0, 1)$, где $P(z|0, 1)$ обозначает ***стандартное нормальное распределение***. Иными словами, любое нормальное распределение можно получить из стандартного нормального распределения, выполнив масштабирование по оси x с коэффициентом σ , масштабирование по оси y с коэффициентом $1/\sigma$ (чтобы площадь под кривой оставалась равной 1) и параллельный перенос начала координат на μ .

Многомерное нормальное распределение d -мерных векторов $\mathbf{x} = (x_1, \dots, x_d)^T \in \mathbb{R}^d$ имеет вид:

$$P(\mathbf{x}|\mu, \Sigma) = \frac{1}{E_d} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\right), \quad E_d = (2\pi)^{d/2} \sqrt{|\Sigma|}. \quad (9.2)$$

Его параметрами являются средний вектор $\mu = (\mu_1, \dots, \mu_d)^T$ и ковариационная матрица Σ размерности $d \times d$ (см. замечание 7.2 на стр. 211). Σ^{-1} – обратная ковариационная матрица, $|\Sigma|$ – ее определитель. Можно считать, что компонентами вектора \mathbf{x} являются d признаков, возможно, коррелированных.

Если $d = 1$, то $\Sigma = \sigma^2 = |\Sigma|$ и $\Sigma^{-1} = 1/\sigma^2$, что дает одномерную гауссиану в качестве частного случая.

Для $d = 2$ имеем $\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}$, $|\Sigma| = \sigma_1^2\sigma_2^2 - (\sigma_{12})^2$ и $\Sigma^{-1} = \frac{1}{|\Sigma|} \begin{pmatrix} \sigma_2^2 & -\sigma_{12} \\ -\sigma_{12} & \sigma_1^2 \end{pmatrix}$. Применяя z -оценки, мы можем вывести следующее выражение для двумерного нормального распределения:

$$P(x_1, x_2 | \mu_1, \mu_2, \sigma_1, \sigma_2, \rho) = \frac{1}{E_2} \exp\left(-\frac{1}{2(1-\rho^2)}(z_1^2 + z_2^2 - 2\rho z_1 z_2)\right), \quad E_2 = 2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}, \quad (9.3)$$

где $z_i = (x_i - \mu_i)/\sigma_i$ для $i = 1, 2$, а $\rho = \sigma_{12}/\sigma_1\sigma_2$ – ***коэффициент корреляции*** между двумя признаками.

Для ***многомерного стандартного нормального распределения*** $\mu = \mathbf{0}$ (d -мерный вектор, состоящий из одних нулей) и $\Sigma = \mathbf{I}$ (единичная матрица размерности $d \times d$), и, следовательно, $P(\mathbf{x}|\mathbf{0}, \mathbf{I}) =$

$$= \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\mathbf{x} \cdot \mathbf{x}\right).$$

Замечание 9.1. Нормальное распределение

Рассмотрим сначала случай, когда обе компоненты имеют одинаковое стандартное отклонение, то есть $\sigma^\oplus = \sigma^\ominus = \sigma$. Тогда аргумент экспоненциальной функции в уравнении (9.1) можно упростить следующим образом:

$$\begin{aligned} -\frac{1}{2\sigma^2}[(x - \mu^\oplus)^2 - (x - \mu^\ominus)^2] &= -\frac{1}{2\sigma^2}[x^2 - 2\mu^\oplus x + \mu^{\oplus^2} - (x^2 - 2\mu^\ominus x + \mu^{\ominus^2})] = \\ &= -\frac{1}{2\sigma^2}[-2(\mu^\oplus - \mu^\ominus)x + (\mu^{\oplus^2} - \mu^{\ominus^2})] = \frac{\mu^\oplus - \mu^\ominus}{\sigma^2} \left[x - \frac{\mu^\oplus + \mu^\ominus}{2} \right]. \end{aligned}$$

Следовательно, отношение максимального правдоподобия можно записать в виде $\text{LR}(x) = \exp(\gamma(x - \mu))$ с параметрами $\gamma = (\mu^{\oplus} - \mu^{\ominus})/\sigma^2$ – разность между средними, поделенная на дисперсию, и $\mu = (\mu^{\oplus} + \mu^{\ominus})/2$ – среднее арифметическое двух средних классов. Отсюда следует, что порог принятия решения с максимальным правдоподобием (значение x такое, что $\text{LR}(x) = 1$) равен $x_{\text{ML}} = \mu$.

Если $\sigma^{\oplus} \neq \sigma^{\ominus}$, то члены, содержащие x^2 , в уравнении (9.1) не сокращаются. В результате мы получаем две решающие границы и разрывную решающую область для одного из классов.

Пример 9.2 (одномерная смесовая модель с неравными дисперсиями). Предположим, что $\mu^{\oplus} = 1$, $\mu^{\ominus} = 2$ и $\sigma^{\ominus} = 2\sigma^{\oplus} = 2$. Тогда $\text{LR}(x) = 2\exp(-[(x - 1)^2 - (x - 2)^2/4]/2) = 2\exp(3x^2/8)$. Отсюда следует, что решающие границы с максимальным правдоподобием есть $x = \pm(8/3)\ln 2 = \pm 1.85$. На рис. 9.2 видно, что это точки, в которых пересекаются две гауссианы. Если же $\sigma^{\ominus} = \sigma^{\oplus}$, то получается единственная решающая граница с максимальным правдоподобием при $x = 1.5$.

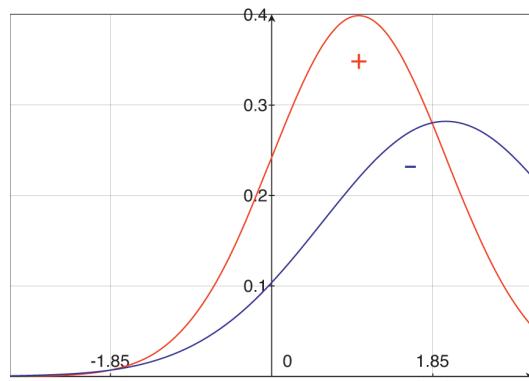


Рис. 9.2. Если положительные примеры лежат на гауссиане со средним и стандартным отклонением 1, а отрицательные – на гауссиане со средним и стандартным отклонением 2, то оба распределения пересекаются в точках $x = \pm 1.85$. Это означает, что областью максимального правдоподобия для положительных примеров является замкнутый отрезок $[-1.85, 1.85]$, а значит, соответствующая область для отрицательных примеров разрывна

Разрывные решающие области встречаются также в многомерных пространствах. В следующем примере это продемонстрировано для $m = 2$.

Пример 9.3 (двумерная гауссова смесь). Мы воспользуемся уравнением (9.3) для получения явных выражений для решающей границы с максимальным правдоподобием в двумерном случае. В этом примере мы будем считать, что $\mu_1^{\oplus} = \mu_2^{\oplus} = 1$ и $\mu_1^{\ominus} = \mu_2^{\ominus} = -1$.

(i) Если все дисперсии равны 1 и обе корреляции равны 0, то решающая граница с максимальным правдоподобием описывается уравнением $(x_1 - 1)^2 + (x_2 - 1)^2 - (x_1 + 1)^2 - (x_2 + 1)^2 = -2x_1 - 2x_2 - 2x_1 - 2x_2 = 0$, то есть $x_1 + x_2 = 0$ (рис. 9.3 слева).

(ii) Если $\sigma_1^{\oplus} = \sigma_1^{\ominus} = 1$, $\sigma_2^{\oplus} = \sigma_2^{\ominus} = \sqrt{2}$ и $\rho^{\oplus} = \rho^{\ominus} = \sqrt{2}/2$, то решающая граница с максимальным правдоподобием описывается уравнением $(x_1 - 1)^2 + (x_2 - 1)^2/2 - \sqrt{2}(x_1 - 1)(x_2 - 1)/\sqrt{2} - (x_1 + 1)^2 - (x_2 + 1)^2/2 + \sqrt{2}(x_1 + 1)(x_2 + 1)/\sqrt{2} = -2x_1 = 0$ (рис. 9.3 в центре).

(iii) Если все дисперсии равны 1 и $\rho^{\oplus} = \rho^{\ominus} = \rho$, то решающая граница с максимальным правдоподобием описывается уравнением $(x_1 - 1)^2 + (x_2 - 1)^2 - 2\rho(x_1 - 1)(x_2 - 1) - (x_1 + 1)^2 - (x_2 + 1)^2 - 2\rho(x_1 + 1)(x_2 + 1) = -4x_1 - 4x_2 - 4\rho x_1 x_2 - 4\rho = 0$, то есть $x_1 + x_2 + \rho x_1 x_2 + \rho = 0$, и является гиперболой. На рис. 9.3 справа это показано для $\rho = 0.7$. Отметим, что левая нижняя часть пространства объектов – положительная решающая область, хотя она не содержит ни одного обучающего примера и ближе к среднему отрицательного, чем к среднему положительного класса.

Обратите внимание на окружности и эллипсы на рис. 9.3, которые дают наглядное представление о ковариационной матрице. Проецируя фигуру, описывающую положительный класс, на ось x , мы получаем отрезок $[\mu_1^{\oplus} - \sigma_1^{\oplus}, \mu_1^{\oplus} + \sigma_1^{\oplus}]$ шириной в одно стандартное отклонение вокруг среднего – и аналогично для отрицательного класса и оси y . Можно выделить три случая: (i) стандартные отклонения x и y равны, и коэффициент корреляции равен нулю, тогда получается окружность; (ii) стандартные отклонения различны, и коэффициент корреляции равен нулю, тогда получается эллипс, параллельный оси с наибольшим стандартным отклонением; (iii) коэффициент корреляции не равен нулю: ориентация эллипса дает знак коэффициента корреляции¹. Математически эти фигуры определяются приравниванием $f(\mathbf{x})$ в выражении $(1/E_d)\exp(-\frac{1}{2}f(\mathbf{x}))$ к единице и решением относительно \mathbf{x} , чтобы найти точки, находящиеся на расстоянии одного стандартного отклонения от среднего. В двумерном случае это приводит к уравнению $(z_1^2 + z_2^2 - 2\rho z_1 z_2) = 1 - \rho^2$, которое можно преобразовать в уравнение эллипса относительно x_1 и x_2 , если раскрыть z -оценки. Отметим, что при $\rho = 0$ получается окружность с центром в начале координат, а когда $\rho \rightarrow 1$, эта окружность стремится к прямой $z_2 = z_1$ (мы не можем положить $\rho = 1$, потому что это приведет к сингулярной ковариационной матрице).

В общем многомерном случае условие $(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = 1$ определяет гиперэллипс, поскольку матрица $\boldsymbol{\Sigma}^{-1}$ обладает определенными свойствами². Для стандартного нормального распределения линии с единичным стандартным отклонением лежат на гиперсфере (d -мерном аналоге окружности), определенной уравнением $\mathbf{x} \cdot \mathbf{x} = 1$. Очень полезное геометрическое соображение заключается в том, что точно так же, как гиперсферу можно преобразовать в произвольный гиперэллипс масштабированием и поворотом, любую многомерную гауссиану можно получить из стандартной гауссианы масштабированием, поворотом (для

¹ Типичная ошибка – думать, что угол поворота эллипса зависит от коэффициента корреляции; на самом деле он определяется исключительно отношением абсолютных величин маргинальных стандартных отклонений.

² Точнее, $\mathbf{x}^T \mathbf{A} \mathbf{x}$ определяет гиперэллипс, если матрица \mathbf{A} симметричная и положительно определенная. Оба эти свойства удовлетворены, если \mathbf{A} – матрица, обратная к несингулярной ковариационной матрице.

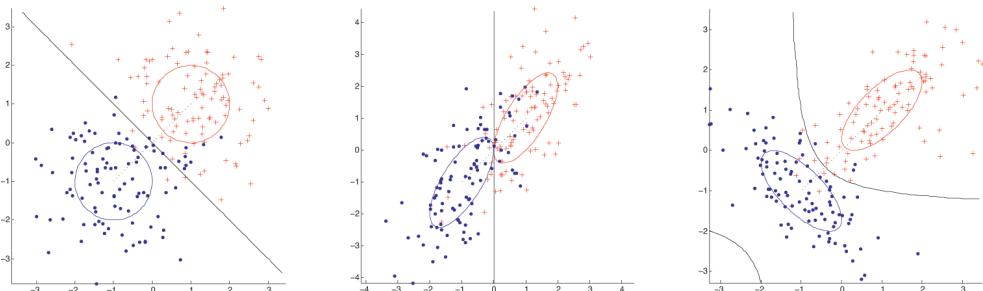


Рис. 9.3. (Слева) Если признаки некоррелированы и имеют одинаковую дисперсию, то классификация по максимальному правдоподобию приводит к базовому линейному классификатору, решающая граница которого ортогональна прямой, соединяющей средние. **(В центре)** При условии, что ковариационные матрицы идентичны, оптимальная по Байесу решающая граница линейна – если бы мы устранили корреляцию с помощью поворота и масштабирования, то снова получили бы базовый линейный классификатор. **(Справа)** При различных ковариационных матрицах получаются гиперболические решающие границы, то есть одна из решающих областей разрывна

получения требуемой ковариационной матрицы) и параллельным переносом (для получения желаемого среднего). Обратно произвольную многомерную гауссиану можно преобразовать в стандартное нормальное распределение с помощью параллельного переноса, поворота и масштабирования, как уже было сказано в замечании 1.2 на стр. 36. Это приводит к устранению корреляции признаков и их нормировке.

Общую формулу для отношения правдоподобия можно вывести из уравнения (9.2) в виде:

$$LR(\mathbf{x}) = \sqrt{\frac{|\Sigma^{\ominus}|}{|\Sigma^{\oplus}|}} \exp\left(-\frac{1}{2}[(\mathbf{x} - \mu^{\oplus})^T (\Sigma^{\oplus})^{-1} (\mathbf{x} - \mu^{\oplus}) - (\mathbf{x} - \mu^{\ominus})^T (\Sigma^{\ominus})^{-1} (\mathbf{x} - \mu^{\ominus})]\right),$$

где μ^{\oplus} и μ^{\ominus} – средние классов, а Σ^{\oplus} и Σ^{\ominus} – ковариационные матрицы для каждого класса. Чтобы лучше разобраться в этом, предположим, что $\Sigma^{\oplus} = \Sigma^{\ominus} = \mathbf{I}$ (то есть в каждом классе признаки некоррелированы и имеют единичную дисперсию), тогда имеем:

$$\begin{aligned} LR(\mathbf{x}) &= \exp\left(-\frac{1}{2}[(\mathbf{x} - \mu^{\oplus})^T (\mathbf{x} - \mu^{\oplus}) - (\mathbf{x} - \mu^{\ominus})^T (\mathbf{x} - \mu^{\ominus})]\right) = \\ &= \exp\left(-\frac{1}{2}[\|\mathbf{x} - \mu^{\oplus}\|^2 - \|\mathbf{x} - \mu^{\ominus}\|^2]\right). \end{aligned}$$

Отсюда следует, что $LR(\mathbf{x}) = 1$ для любого \mathbf{x} , равноудаленного от μ^{\oplus} и μ^{\ominus} . Но это означает, что решающая граница с максимальным правдоподобием – прямая линия, проходящая на равном расстоянии от средних точек классов, – а это не

что иное, как наш старый приятель, базовый линейный классификатор! Другими словами, *для некоррелированных гауссовых признаков с единичной дисперсией базовый линейный классификатор является оптимальным по Байесу*. Это хороший пример того, как с помощью вероятностного подхода можно обосновать конкретные модели.

В общем случае при условии, что ковариационные матрицы для каждого класса одинаковы, решающая граница с максимальным правдоподобием будет гиперплоскостью, пересекающей отрезок $\mu^{\oplus} - \mu^{\ominus}$ в середине, но не под прямым углом, если признаки коррелированы. Это означает, что в этом случае базовый линейный классификатор будет оптимален по Байесу, только если предварительно устраниТЬ корреляцию и нормировать признаки. При неравных ковариационных матрицах классов решающая граница будет гиперболической. Таким образом, все три случая, показанные на рис. 9.3, обобщаются на многомерную ситуацию.

Мы видели несколько примеров того, как нормальное распределение связывает геометрический и вероятностный взгляды на вещи. Многомерное нормальное распределение по существу преобразует расстояния в вероятности. Это становится очевидным, если подставить определение *расстояния Махalanобиса* (формула (8.1) на стр. 248) в уравнение (9.2):

$$P(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{E_d} \exp\left(-\frac{1}{2} (\text{Dis}_M(\mathbf{x}, \boldsymbol{\mu} | \boldsymbol{\Sigma}))^2\right). \quad (9.4)$$

Аналогично стандартное нормальное распределение преобразует евклидовы расстояния в вероятности:

$$P(\mathbf{x} | \mathbf{0}, \mathbf{I}) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2} (\text{Dis}_2(\mathbf{x}, \mathbf{0}))^2\right).$$

Обратно, мы видим, что *взятый со знаком минус логарифм гауссова правдоподобия можно интерпретировать как квадрат расстояния*:

$$-\ln P(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \ln E_d + \frac{1}{2} (\text{Dis}_M(\mathbf{x}, \boldsymbol{\mu} | \boldsymbol{\Sigma}))^2.$$

Интуиция подсказывает, что логарифм преобразует мультипликативную шкалу вероятностей в аддитивную шкалу (которая в случае гауссова распределения соответствует квадрату расстояния). Поскольку с аддитивной шкалой работать проще, логарифмическое правдоподобие применяется в статистике очень часто.

Еще один пример связи между геометрическим и вероятностным взглядами возникает, когда мы рассматриваем вопрос об оценивании параметров нормального распределения. Например, предположим, что нужно оценить среднее $\boldsymbol{\mu}$ многомерного гауссова распределения с заданной ковариационной матрицей $\boldsymbol{\Sigma}$, имея набор данных X . Принцип *оценки максимального правдоподобия* утверждает, что следует искать значение $\boldsymbol{\mu}$, которое обращает в максимум совместное правдоподобие X . В предположении, что элементы X выбрались независимо, совместное

правдоподобие разлагается в произведение по отдельным точкам X , и оценку максимального правдоподобия можно найти следующим образом:

$$\begin{aligned}\hat{\mu} &= \arg \max_{\mu} \prod_{x \in X} P(x | \mu, \Sigma) = \\ &= \arg \max_{\mu} \prod_{x \in X} \frac{1}{E_d} \exp \left(-\frac{1}{2} (\text{Dis}_M(x, \mu | \Sigma))^2 \right) = && \text{из уравнения (9.4)} \\ &= \arg \min_{\mu} \sum_{x \in X} \left[\ln E_d + \frac{1}{2} (\text{Dis}_M(x, \mu | \Sigma))^2 \right] = && \text{путем взятия логарифмов} \\ &= \arg \min_{\mu} \sum_{x \in X} (\text{Dis}_M(x, \mu | \Sigma))^2. && \text{опускаем постоянный член} \\ &&& \text{и коэффициент}\end{aligned}$$

Таким образом, находим, что оценка максимального правдоподобия среднегомерного распределения – это точка, в которой обращается в минимум сумма квадратов расстояния Махalanобиса до всех точек X . В случае единичной ковариационной матрицы $\Sigma = \mathbf{I}$ мы можем заменить расстояние Махalanобиса евклидовым, и тогда по теореме 8.1 точкой, минимизирующей сумму квадратов евклидовых расстояний до всех точек X , является среднее арифметическое

$$\frac{1}{|X|} \sum_{x \in X} x.$$

И в качестве последнего примера того, как геометрический и вероятностный взгляды на одну и ту же проблему могут быть сильно связаны, я продемонстрирую вывод *решения задачи линейной регрессии по методу наименьших квадратов* (раздел 7.1) как оценки максимального правдоподобия. Для простоты обозначений будем рассматривать одномерный случай, обсуждавшийся в примере 7.1. Отправной точкой является гипотеза, что обучающие примеры (h_i, y_i) – это зашумленные измерения истинной функции $(x_i, f(x_i))$, то есть $y_i = f(x_i) + \epsilon_i$, где ϵ_i – независимые невязки с одинаковым распределением. (Обратите внимание на небольшое изменение обозначений – y_i больше не обозначает истинного значения функции.) Мы хотим вывести оценки максимального правдоподобия \hat{y}_i значений $f(x_i)$. Их можно вывести, если предположить конкретное распределение шума, например гауссово с дисперсией σ^2 . Тогда каждое y_i имеет нормальное распределение со средним $a + bx_i$ и дисперсией σ^2 , а значит:

$$P(y_i | a, b, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left(-\frac{(y_i - (a + bx_i))^2}{2\sigma^2} \right).$$

Поскольку описывающие шум члены ϵ_i независимы для различных i , то также и y_i , и, следовательно, совместное распределение вероятностей по всем i – это просто произведение n гауссийан:

$$\begin{aligned} P(y_1, \dots, y_n | a, b, \sigma^2) &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - (a + bx_i))^2}{2\sigma^2}\right) = \\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(-\frac{\sum_{i=1}^n (y_i - (a + bx_i))^2}{2\sigma^2}\right). \end{aligned}$$

Для простоты алгебраических манипуляций возьмем натуральный логарифм со знаком минус:

$$-\ln P(y_1, \dots, y_n | a, b, \sigma^2) = \frac{n}{2} \ln 2\pi + \frac{n}{2} \ln \sigma^2 + \frac{\sum_{i=1}^n (y_i - (a + bx_i))^2}{2\sigma^2}.$$

Приравняв к нулю частные производные по a , b и σ^2 для нахождения максимума отрицательного логарифмического правдоподобия, мы получим три уравнения:

$$\begin{aligned} \sum_{i=1}^n y_i - (a + bx_i) &= 0; \\ \sum_{i=1}^n (y_i - (a + bx_i))x_i &= 0; \\ \frac{n}{2} \frac{1}{\sigma^2} - \frac{\sum_{i=1}^n (y_i - (a + bx_i))^2}{2(\sigma^2)^2} &. \end{aligned}$$

Первые два – по существу те же самые, что были выведены в примере 7.1, они дают $\hat{a} = \bar{y} - \hat{b}\bar{x}$ и $\hat{b} = \sigma_{xy}/\sigma_{xx}$ соответственно. Третье уравнение говорит, что сумма квадратов невязок равна $n\sigma^2$ и дает оценку максимального правдоподобия дисперсии шума в виде $(\sum_{i=1}^n (y_i - (a + bx_i))^2)/n$.

Обнадеживает, что вероятностный взгляд позволяет вывести решение задачи регрессии по (обычному) методу наименьших квадратов из чисто теоретических соображений. С другой стороны, полное рассмотрение потребовало бы учитывать также зашумленность значений x (обобщенный метод наименьших квадратов), но это усложняет математику, и решение может быть не единственным. Это показывает, что *хороший вероятностный подход к проблеме машинного обучения позволяет достичь баланса между солидным теоретическим основанием и pragmatizmом, необходимым для получения рабочего решения*.

9.2 Вероятностные модели для категориальных данных

Чтобы чем-то занять себя во время долгих поездок на удаленное место отдыха, мы с сестрами часто развлекаемся играми, в которых участвуют проезжающие мимо машины. Например, просим друг друга отмечать машины определенного

цвета, из определенной страны или с определенной буквой в номере. Вопрос с двумя возможными ответами типа «является ли машина синей?» в статистике называется *испытанием Бернулли*. Такие испытания моделируются бинарной случайной величиной, для которой вероятность успеха фиксирована и одинакова в каждом независимом испытании. Мы использовали распределение Бернулли для моделирования события «почтовое сообщение хорошее» в примере 9.1. Над такой случайной величиной можно построить и другие распределения вероятности. Например, можно угадывать, сколько следующих машин будут синими: соответствующее распределение называется *биномиальным*. Или оценить, сколько проедет машин до появления первой машины из Голландии: это определение геометрического распределения. В замечании 9.2 приведены соответствующие определения.

Категориальные величины или признаки (их еще называют дискретными, или номинальными) в машинном обучении встречаются повсеместно. Пожалуй, самая распространенная форма распределения Бернулли моделирует вхождение слова в документ. То есть для i -го слова в словаре мы имеем случайную величину X_i с распределением Бернулли. Совместное распределение *битового вектора* $\mathbf{X} = (X_1, \dots, X_k)$ называется *многомерным распределением Бернулли*. Часто встречаются также величины с более чем двумя исходами: например, позиции слова в почтовом сообщении соответствует категориальная величина с k исходами, где k – размер словаря. Мультиномиальное распределение выступает в виде *вектора счетчиков*: гистограммы количества вхождений всех словарных слов в документ. Это дает альтернативный способ моделирования текстовых документов, позволяющий учитывать количество вхождений слова при классификации документа.

Распределение Бернулли, названное в честь швейцарского математика XVII века Яакоба Бернулли, относится к булевым, или бинарным, событиям с двумя возможными исходами: успех (1) и неудача (0). У распределения Бернулли имеется единственный параметр θ , определяющий вероятность успеха: $P(X = 1) = \theta$ и $P(X = 0) = 1 - \theta$. Математическое ожидание распределения Бернулли $\mathbb{E}[X] = \theta$, а дисперсия $\mathbb{E}[(X - \mathbb{E}[X])^2] = \theta(1 - \theta)$.

Биномиальное распределение возникает при подсчете числа успехов S в n независимых испытаниях Бернулли с одним и тем же параметром θ . Оно описывается формулой

$$P(S = s) = \binom{n}{s} \theta^s (1 - \theta)^{n-s} \text{ для } s \in \{0, \dots, n\}.$$

Его математическое ожидание равно $\mathbb{E}[S] = n\theta$, а дисперсия $\mathbb{E}[(S - \mathbb{E}[S])^2] = n\theta(1 - \theta)$.

Категориальное распределение обобщает распределение Бернулли на $k \geq 2$ исходов. Его параметром является k -мерный вектор $\theta = (\theta_1, \dots, \theta_k)$ – такой, что $\sum_{i=1}^k \theta_i = 1$.

Наконец, *мультиномиальное распределение* имеет исходы n независимых категориальных испытаний с одинаковым распределением. То есть $\mathbf{X} = (X_1, \dots, X_k)$ – k -мерный вектор целочисленных счетчиков и

$$P(\mathbf{X} = (x_1, \dots, x_k)) = n! \frac{\theta_1^{x_1} \dots \theta_k^{x_k}}{x_1! \dots x_k!}$$

при условии $\sum_{i=1}^k x_i = n$. Отметим, что задание $n = 1$ дает альтернативный способ определения категориального распределения в виде $P(\mathbf{X} = (x_1, \dots, x_k)) = \theta_1^{x_1} \dots \theta_k^{x_k}$ при условии, что ровно один x_i

равен 1, а все остальные – 0. А задание $k = 2$ дает альтернативное выражение для распределения Бернулли в виде $P(X = x) = \theta^x(1 - \theta)^{1-x}$ при $x \in \{0, 1\}$. Полезно также заметить, что если \mathbf{X} имеет мультиномиальное распределение, то каждая компонента X_i имеет биномиальное распределение с параметром θ_i .

Мы можем оценить параметры этих распределений прямым подсчетом. Предположим, что $a \ b \ a \ c \ c \ b \ a \ b \ c$ – последовательность слов. Нас может интересовать, совпадает слово с a или нет, при том что данные интерпретируются как результаты независимых одинаково распределенных испытаний Бернулли; это даст оценку $\hat{\theta}_a = 4/10 = 0.4$. Тот же самый параметр порождает биномиальное распределение количества вхождений слова a в аналогичные последовательности. Точно так же можно оценить параметры категориального (вхождение слов) и мультиномиального (счетчики слов) в виде $\hat{\theta} = (0.4, 0.3, 0.3)$.

Почти всегда полезно сглаживать эти распределения, включая *псевдосчетчики*. Представьте, что словарь содержит слово d , но мы его еще не наблюдали, тогда оценка максимального правдоподобия имела бы вид $\hat{\theta}_d = 0$. Этую оценку можно сгладить, добавив виртуальное вхождение каждого слова в наблюдения, что дает $\hat{\theta}' = (5/14, 4/14, 4/14, 1/14)$. В случае биномиального распределения это не что иное, как поправка Лапласа.

Замечание 9.2. Распределения вероятности для категориальных данных

Обе эти модели документов широко употребляются. Несмотря на различия, в обеих предполагается независимость вхождений слов, что обычно называют *наивным байесовским предположением*. В мультиномиальной модели документа это следует из самого определения мультиномиального распределения, в котором предполагается, что слова в разных позициях независимо выбираются из одного и того же категориального распределения. В многомерной модели Бернулли мы предполагаем, что отдельные биты битового вектора статистически независимы, что позволяет вычислить совместную вероятность конкретного битового вектора (x_1, \dots, x_k) как произведение вероятностей его компонент $P(X_i = x_i)$. На практике предположение о независимости часто не выполняется: если мы знаем, что почтовое сообщение содержит слово «виагра», то можно с высокой степенью уверенности утверждать, что в нем встречается и слово «таблетка». Как бы то ни было, опыт показывает, что хотя наивное байесовское предположение почти всегда дает плохие оценки вероятностей, это зачастую не вредит качеству ранжирования. Это означает, что при условии разумного выбора порога классификации мы обычно получаем также и хорошую классификацию.

Использование наивной байесовской модели для классификации

Предположим, что мы выбрали одно из возможных распределений для моделирования данных \mathbf{X} . В контексте классификации мы можем далее предположить, что это распределение зависит от класса, так что распределения $P(\mathbf{X}|Y = \text{спам})$ и $P(\mathbf{X}|Y = \text{неспам})$ различны. Чем сильнее они различаются, тем полезнее признаки \mathbf{X} для классификации. Таким образом, для конкретного почтового сообщения мы вычисляем обе величины $P(\mathbf{X} = x|Y = \text{спам})$ и $P(\mathbf{X} = x|Y = \text{неспам})$ и применяем одно из нескольких возможных решающих правил:

- ☞ максимального правдоподобия (ML) – предсказать $\operatorname{argmax}_y P(X = x|Y = y)$;
- ☞ апостериорного максимума (MAP) – предсказать $\operatorname{argmax}_y P(X = x|Y = y \cdot P(Y = y))$;
- ☞ откалиброванного правдоподобия – предсказать $\operatorname{argmax}_y w_y P(X = x|Y = y)$.

Первые два решающих правила соотносятся следующим образом: классификация по методу ML эквивалентна классификации по методу MAP, если распределение по классам равномерно. Третье правило обобщает первые два в том смысле, что заменяет распределение по классам набором весов, обученным на данных: как мы увидим ниже, это позволяет исправлять ошибки оценивания в правдоподобиях.

Пример 9.4 (предсказание с помощью наивной байесовской модели). Предположим, что словарь содержит три слова a, b и c и что для почтовых сообщений используется многомерная модель Бернулли с параметрами

$$\theta^{\oplus} = (0.5, 0.67, 0.33) \quad \theta^{\ominus} = (0.67, 0.33, 0.33)$$

Это, в частности, означает, что присутствие b в два раза более вероятно в спаме (+), чем в хороших сообщениях.

Подлежащее классификации сообщение содержит слова a и b , но не c , и, следовательно, описывается битовым вектором $\mathbf{x} = (1, 1, 0)$. Получаются такие правдоподобия:

$$P(\mathbf{x}|\oplus) = 0.5 \cdot 0.67 \cdot (1 - 0.33) = 0.222 \quad P(\mathbf{x}|\ominus) = 0.67 \cdot 0.33 \cdot (1 - 0.33) = 0.148$$

Следовательно, по правилу ML \mathbf{x} классифицируется как спам. В случае двух классов часто удобнее работать с отношениями правдоподобия и шансами. Отношение правдоподобия можно вы-

числить как $\frac{P(\mathbf{x}|\oplus)}{P(\mathbf{x}|\ominus)} = \frac{0.5}{0.67} \frac{0.67}{0.33} \frac{1 - 0.33}{1 - 0.33} = 3/2 > 1$. Это означает, что по правилу MAP \mathbf{x} также классифицируется как спам, если априорный шанс больше $2/3$, и как неспам, если меньше. Например, при 33% спама и 67% неспама априорный шанс равен $\frac{P(\oplus)}{P(\ominus)} = \frac{0.33}{0.67} = 1/2$, а апостериор-

ный, следовательно, $\frac{P(\oplus|\mathbf{x})}{P(\ominus|\mathbf{x})} = \frac{P(\mathbf{x}|\oplus)}{P(\mathbf{x}|\ominus)} \frac{P(\oplus)}{P(\ominus)} = 3/2 \cdot 1/2 = 3/4 < 1$. В этом случае отношение правдоподобия для \mathbf{x} недостаточно велико, чтобы изменить решение, по сравнению с априорным.

Альтернативно можно использовать мультиномиальную модель. Ее параметры задают распределение слов из словаря, например:

$$\theta^{\oplus} = (0.3, 0.5, 0.2) \quad \theta^{\ominus} = (0.6, 0.2, 0.2)$$

Подлежащее классификации сообщение содержит три вхождения слова a , одно вхождение слова b и не содержит слова c , то есть описывается вектором счетчиков $\mathbf{x} = (3, 1, 0)$. Общее число вхождений словарных слов равно $n = 4$. Получаются такие правдоподобия:

$$P(\mathbf{x}|\oplus) = 4! \frac{0.3^3 \cdot 0.5^1 \cdot 0.2^0}{3! \cdot 1! \cdot 0!} = 0.054; \quad P(\mathbf{x}|\ominus) = 4! \frac{0.6^3 \cdot 0.2^1 \cdot 0.2^0}{3! \cdot 1! \cdot 0!} = 0.1728.$$

Отношение правдоподобия равно $\left(\frac{0.3}{0.6}\right)^3 \left(\frac{0.5}{0.2}\right)^1 \left(\frac{0.2}{0.2}\right)^0 = 5/16$. По правилу ML \mathbf{x} классифицируется как неспам, то есть прямо противоположно многомерной модели Бернулли. Это случилось главным образом из-за трех вхождений слова a , что является сильным свидетельством в пользу неспама.

Отметим, что отношение правдоподобия для многомерной модели Бернулли является произведением сомножителей $\theta_i^{\oplus}/\theta_i^{\ominus}$, если $x_i = 1$ в классифицируемом битовом векторе и $(1 - \theta_i^{\oplus})/(1 - \theta_i^{\ominus})$, если $x_i = 0$. Для мультиномиальной модели сомножители равны $(\theta_i^{\oplus}/\theta_i^{\ominus})^{x_i}$. Из этого следует, в частности, что мультиномиальная модель принимает во внимание только присутствие слов, тогда как многомерная модель Бернулли – еще и их отсутствие. В примере выше отсутствию слова b соответствует сомножитель $(1 - 0.67)/(1 - 0.33) = 1/2$ в отношении правдоподобия. Еще одно существенное различие между этими двумя моделями заключается в том, что в мультиномиальной многократные вхождения слов трактуются как повторяющиеся признаки – благодаря «весу» x_i в показателе степени. Это становится понятнее, если взять логарифм отношения правдоподобия, равный $\sum_i x_i (\ln \theta_i^{\oplus} - \ln \theta_i^{\ominus})$: это выражение линейно зависит от $\ln \theta_i^{\oplus}$ и $\ln \theta_i^{\ominus}$ с весовыми коэффициентами x_i . Отметим, что это еще не означает линейность байесовских классификаторов в смысле, обсуждавшемся в главе 7, если только мы не сможем продемонстрировать линейное соотношение между $\ln \theta$ и значением соответствующего признака. Но мы можем сказать, что наивные байесовские модели линейны в специальном пространстве (пространстве «логарифмических шансов»), получающемся путем применения корректно определенного преобразования к признакам. Мы вернемся к этому моменту, когда будем обсуждать [«калибровку признаков»](#) в разделе 10.2.

Тот факт, что отношение совместного правдоподобия наивной байесовской модели разлагается в произведение отношений правдоподобия отдельных слов, – прямое следствие наивного байесовского предположения. Иными словами, задача обучения разлагается на одномерные задачи, по одной для каждого слова в словаре. Мы уже встречались с подобным разложением при обсуждении [«многомерной линейной регрессии»](#) в разделе 7.1. Там мы видели пример вреда, который может нанести игнорирование корреляции признаков. Можно ли привести аналогичные примеры для наивных байесовских классификаторов? Рассмотрим ситуацию, когда некоторое слово встречается в словаре дважды. В таком случае один и тот же сомножитель будет дважды входить в произведение, представляющее отношение правдоподобия, и по существу удваивает вес данного слова, по сравнению со всеми остальными. Хотя это крайний пример, такой двойной подсчет действительно дает заметный эффект на практике. Я уже выше указывал, что сообщение, содержащее слово «виагра», скорее всего, содержит и слово «таблетка», поэтому наблюдение обоих слов вместе не является существенно более сильным свидетельством в пользу спама, чем наблюдение одного лишь первого слова, а отношение правдоподобия для обоих слов не должно быть намного больше, чем для первого слова. Однако перемножение двух отношений правдоподобия, больших 1, дает величину, большую каждого из сомножителей. В результате оценки вероятностей, даваемые наивным байесовским классификатором, зачастую смещаются слишком далеко в сторону 0 или 1.

Если нас интересует только классификация, а не оценки вероятностей как таковые, то может показаться, что это не страшно. Однако *часто забывают* о

том, что для таких неоткалиброванных оценок вероятностей, какие порождаются наивным байесовским классификатором, решающие правила *ML* и *MAP* становятся неадекватными. Если у нас нет свидетельств, доказывающих, что предположения модели верны, то в таком случае единственная разумная вещь – воспользоваться *решающим правилом откалиброванного правдоподобия*, которое требует обучить вектор весов классов, чтобы исправить ошибки оценивания в правдоподобиях. Точнее, мы хотим найти веса w_i – такие, что предсказанная величина $\text{argmax}_y w_y P(X = x|Y = y)$ приводит к наименьшей возможной потере – числу неправильно классифицированных примеров – на тестовом наборе. Для двух классов эту задачу можно решить с помощью той же процедуры *преобразования ранжировщиков в классификаторы*, которую мы рассматривали в разделе 2.2. Чтобы убедиться в этом, заметим, что для двух классов решающее правило откалиброванного правдоподобия можно переписать в виде:

- ☞ предсказывать положительный класс, если $w^\oplus P(X = x|Y = \oplus) > w^\ominus P(X = x|Y = \ominus)$, и отрицательный в противном случае, что эквивалентно;
- ☞ предсказывать положительный класс, если $P(X = x|Y = \oplus)/P(X = x|Y = \ominus) > w^\ominus/w^\oplus$, и отрицательный в противном случае.

Это показывает, что в случае двух классов у нас на самом деле есть всего одна степень свободы, поскольку умножение весов на константу не влияет на принятие решений. Иными словами, нам интересно найти наилучший порог $t = w^\ominus/w^\oplus$ отношения правдоподобия, а это фактически та же задача, что и нахождение наилучшей рабочей точки на кривой РХП. Решением является точка на изолинии самой высокой верности. На рис. 9.4 изображены два реальных набора данных; слева мы видим, что порог принятия решения *MAP* более-менее оптимален, а справа – что оптимальная точка находится в правом верхнем углу.

Если классов больше двух, то объем вычислений, необходимых для нахождения глобально оптимального вектора весов, превышает наши возможности, а значит, необходимо прибегнуть к эвристическому методу. В разделе 3.1 такой метод был продемонстрирован для трех классов. Идея заключается в том, чтобы поочередно фиксировать веса, используя некоторое упорядочение классов. Иначе говоря, мы применяем двухклассовую процедуру, чтобы оптимально отделить i -й класс от предыдущих $i - 1$ классов.

Обучение наивной байесовской модели

Обучение вероятностной модели обычно сводится к оцениванию параметров, используемых в этой модели распределений. Параметр распределения Бернулли можно оценить, подсчитав число успехов d в n испытаниях и положив $\hat{\theta} = d/n$. Иными словами, мы для каждого класса подсчитываем, сколько сообщений содержит рассматриваемое слово. Такие оценки на основе относительной частоты обычно сглаживаются путем введения *псевдосчетчиков*, представляющих исходы виртуальных испытаний с некоторыми фиксированными распределениями. В случае распределения Бернулли в качестве операции сглаживания чаще все-

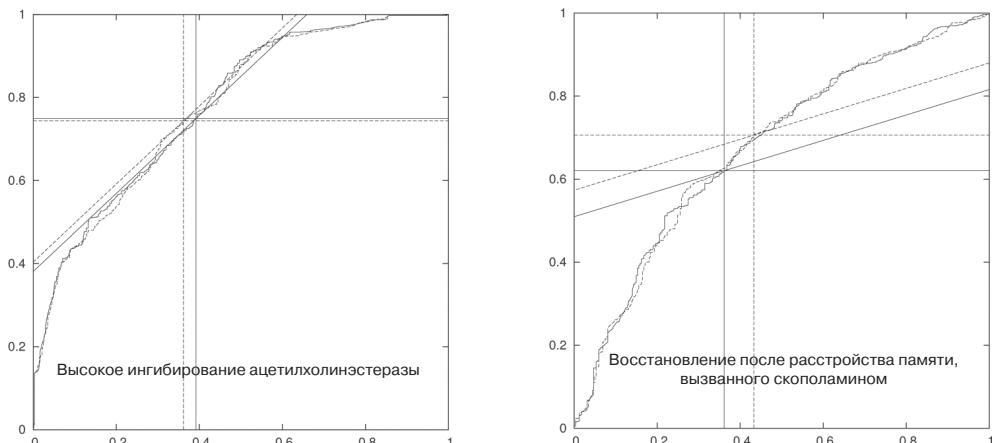


Рис. 9.4. (Слева) Кривые РХП, порожденные двумя наивными байесовскими классификаторами (сплошная линия: вариант многомерной модели Бернулли; штриховая линия: вариант мультиномиальной модели). У обеих моделей схожее качество ранжирования и почти одинаковый – более-менее оптимальный – порог принятия решения МАР. **(Справа)** На другом наборе данных из той же предметной области порог МАР для мультиномиальной модели несколько лучше, что наводит на мысль о лучших калиброванных оценках вероятностей. Но поскольку угловой коэффициент изолиний верность показывает, что на каждый отрицательный пример приходится примерно четыре положительных, то оптимальное решающее правило фактически всегда будет предсказывать положительный класс

го берут поправку Лапласа, которая подразумевает два виртуальных испытания: одно успешное, другое неудачное. Следовательно, оценка на основе относительной частоты заменяется на $(d + 1)/(n + 2)$. С точки зрения байесовской модели, это сводится к принятию равномерного априорного распределения, выражающего нашу веру в том, что успех и неудача равновероятны. Если того требует ситуация, мы можем усилить влияние априорного знания, включив большее количество виртуальных испытаний, что означает, что для отодвигания оценки от априорной потребуется больше данных. В случае категориального распределения сглаживание добавляет один псевдосчетчик для каждой из k категорий, что приводит к сглаженной оценке $(d + 1)/(n + k)$. ***m*-оценка** представляет собой дальнейшее обобщение, считая параметрами как общее число псевдосчетчиков m , так и их распределение по категориям. Оценка для i -й категории определяется как $(d + p_i m)/(n + m)$, где p_i – распределение по категориям (то есть $\sum_{i=1}^k p_i = 1$). Отметим, что оценки на основе сглаженной относительной частоты – а значит, и произведение таких оценок – никогда не могут достигать крайних значений $\hat{\theta} = 0$ и $\hat{\theta} = 1$.

Пример 9.5 (обучение наивной байесовской модели). Сейчас мы покажем, как можно было бы получить векторы параметров в предыдущем примере. Рассмотрим следующие сообщения, состоящие из пяти слов a, b, c, d, e :

$e_1: b d e b b d e$	$e_5: a b a b a b a e d$
$e_2: b c e b b d d e c c$	$e_6: a c a c a c a e d$
$e_3: a d a d e a e e$	$e_7: e a e d a a e a$
$e_4: b a d b e d a b$	$e_8: d e d e d$

Нам известно, что сообщения слева – спам, а справа – неспам, и мы хотим использовать их в качестве небольшого обучающего набора для обучения байесовского классификатора. Сначала мы принимаем решение, что d и e – так называемые *стоп-слова*, которые встречаются настолько часто, что не несут никакой информации о классе. Остальные слова, a , b и c , составляют наш словарь.

В случае мультиномиальной модели мы представляем каждое сообщение вектором счетчиков, как в табл. 9.1 слева. Чтобы оценить параметры распределения, мы вычисляем сумму векторов счетчиков для каждого класса и получаем $(5, 9, 3)$ для спама и $(11, 3, 3)$ для неспама. Чтобы сгладить эти оценки вероятностей, мы добавляем по одному псевдосчетчику для каждого словарного слова, что доводит общее число вхождений словарных слов до 20 для каждого класса. Таким образом, оценочные векторы параметров равны $\hat{\theta}^{\oplus} = (6/20, 10/20, 4/20) = (0.3, 0.5, 0.2)$ для спама и $\hat{\theta}^{\ominus} = (12/20, 4/20, 4/20) = (0.6, 0.2, 0.2)$ для неспама.

В случае многомерной модели Бернулли сообщения представлены битовыми векторами, как в табл. 9.1 справа. Сложение битовых векторов для каждого класса дает $(2, 3, 1)$ спама и $(3, 1, 1)$ для неспама. Каждый счетчик следует разделить на количество документов в классе, чтобы получить оценку вероятности того, что документ содержит конкретное словарное слово. Сглаживание вероятности теперь означает добавление двух псевдодокументов, один из которых содержит все слова, а другой – ни одного. Это дает такие оценочные векторы параметров: $\hat{\theta}^{\oplus} = (3/6, 4/6, 2/6) = (0.5, 0.67, 0.33)$ для спама и $\hat{\theta}^{\ominus} = (4/6, 2/6, 2/6) = (0.67, 0.33, 0.33)$ для неспама.

Сообщение	# a	# b	# c	Класс	Сообщение	$a?$	$b?$	$c?$	Класс
e_1	0	3	0	+	e_1	0	1	0	+
e_2	0	3	3	+	e_2	0	1	1	+
e_3	3	0	0	+	e_3	1	0	0	+
e_4	2	3	0	+	e_4	1	1	0	+
e_5	4	3	0	-	e_5	1	1	0	-
e_6	4	0	3	-	e_6	1	0	1	-
e_7	3	0	0	-	e_7	1	0	0	-
e_8	0	0	0	-	e_8	0	0	0	-

Таблица 9.1. (Слева) Небольшой набор данных о почтовых сообщениях, описываемый векторами счетчиков. **(Справа)** Тот же самый набор данных, описываемый битовыми векторами

Существует много других вариантов наивного байесовского классификатора. На самом деле в том, что обычно понимается под «настоящим» байесовским классификатором, используется не мультиномиальная модель и не многомерная модель Бернулли, а многомерная категориальная модель. Это означает, что все признаки категориальные и что вероятность того, что i -й признак принимает свое l -ое значение для примеров из класса c , равна $\theta_{il}^{(c)}$ при условии, что $\sum_{l=1}^{k_i} \theta_{il}^{(c)} = 1$, где k_i – количество различных значений i -го признака. Эти параметры можно оценить с помощью сглаженных относительных частот в обучающем наборе, как

и в случае многомерной модели Бернулли. Совместная вероятность вектора признаков снова равна произведению вероятностей отдельных признаков, а значит, $P(F_i, F_j|C) = P(F_i|C)P(F_j|C)$ для всех пар признаков и для всех классов.

Отметим попутно, что условная независимость не имеет ничего общего с безусловной: ни одна не следует из другой. Чтобы убедиться в том, что из условной независимости не вытекает безусловная, представим два слова, появление которых в спаме очень вероятно, но при этом они независимы (то есть вероятность их одновременного появления в спамном сообщении равна произведению маргинальных вероятностей). Предположим также, что их появление в хорошем сообщении крайне маловероятно, хотя эти события также независимы. Допустим, я говорю вам, что неклассифицированное сообщение содержит одно из этих слов; надо думать, вы решите, что сообщение спамное, а отсюда сделаете вывод, что оно содержит и другое слово, – демонстрация того, что слова не являются безусловно независимыми. Чтобы понять, почему безусловная независимость не влечет за собой условной, рассмотрим два разных независимых слова, и пусть сообщение является спамом, если оно содержит хотя бы одно из этих слов, и неспамом в противном случае. Тогда среди спамных сообщений оба слова зависимы (поскольку если я знаю, что спамное сообщение не содержит одного из них, то должно содержать другое).

Другое обобщение наивной байесовской модели необходимо, когда некоторые признаки принимают вещественные значения. Одна из возможностей – дискретизировать их на стадии предобработки – обсуждается в главе 10. Другая – предположить, что значения признаков имеют нормальное распределение в каждом классе, как обсуждалось в предыдущем разделе. В этом контексте стоит отметить, что наивное байесовское предположение сводится к предположению о диагональности ковариационной матрицы в каждом классе, так что все признаки можно рассматривать независимо. Третья возможность, которая также применяется на практике, – смоделировать условное относительно класса правдоподобие каждого признака с помощью непараметрической оценки плотности. Все три варианта показаны на рис. 9.5.

Короче говоря, наивная байесовская модель популярна при работе с текстовыми, категориальными и смешанными – категориальными и вещественными – данными. Ее основной недостаток в качестве вероятностной модели – плохо откалиброванные оценки вероятностей – перевешивается в общем неплохим качеством ранжирования. Еще один очевидный парадокс наивной байесовской модели – то, что она вовсе и не байесовская! Во-первых, мы видели, что плохое качество оценок вероятности вынуждает использовать взвешенные по-другому правдоподобия, то есть вообще отказываться от применения правила Байеса. Во-вторых, при обучении наивной байесовской модели мы оцениваем параметры методом максимального правдоподобия, тогда как в настоящем байесовском подходе мы не интересуемся значениями отдельных параметров, а используем полное апостериорное распределение. Лично мне кажется, что смысл наивной байесовской модели заключается в разложении совместных правдоподобий

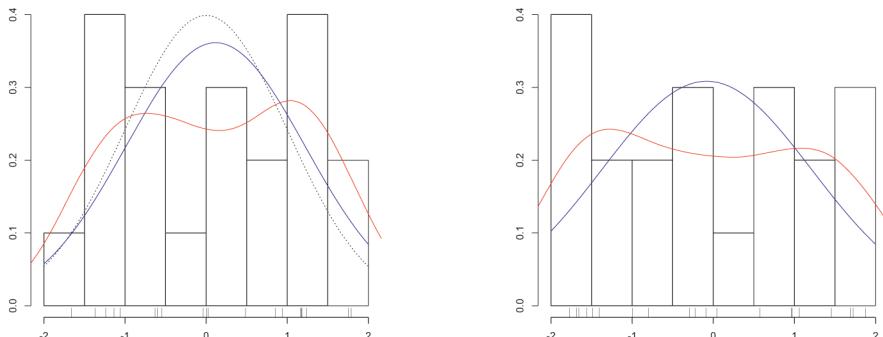


Рис. 9.5. (Слева) Примеры трех оценок плотности на выборке из 20 точек, имеющей нормальное распределение с нулевым средним и единичной дисперсией (штриховая линия). Гистограмма представляет собой простой непараметрический метод, в котором используется фиксированное число интервалов равной длины. Ядерная оценка плотности (**красная** линия) получается применением интерполяции для сглаживания функции плотности. Сплошная колоколообразная кривая (**синяя**) получена путем оценивания выборочного среднего и дисперсии в предположении, что истинное распределение нормально. (**Справа**) Здесь 20 точек распределены равномерно на отрезке $[-2, 2]$, и непараметрические методы в общем случае работают лучше

в произведение маргинальных. Это разложение очень выразительно представлено в виде шотландского пледа в клетку на рис. 1.3, поэтому я называю наивный байесовский классификатор «шотландским».

9.3 Дискриминантное обучение путем оптимизации условного правдоподобия

Во введении к этой главе мы провели различие между порождающими и дискриминантными вероятностными моделями. Наивные байесовские модели порождающие: после обучения их можно использовать для порождения новых данных. В этом разделе мы рассмотрим наиболее распространенную дискриминантную модель: *логистическую регрессию*¹. Проще всего понять логистическую регрессию, рассматривая ее как линейный классификатор, оценки вероятностей которого логистически калиброваны методом, описанным в разделе 7.4, но с одним существенным отличием: калибровка является неотъемлемой частью алгоритма обучения, а не шагом постобработки. Если в порождающих моделях решающая граница – побочный продукт моделирования распределений каждого класса, то логистическая регрессия моделирует решающую границу непосредственно. На-

¹ Отметим, что слово «регрессия» здесь не вполне уместно, поскольку, несмотря на то что оценка вероятности аппроксимирует неизвестную функцию, обучающие метки являются классами, а не значениями функции.

пример, если классы перекрываются, это значит, что логистическая регрессия демонстрирует тенденцию располагать решающую границу в области, где перекрытие классов максимально, вне зависимости от «форм» выборок из каждого класса. Получающие в результате решающие границы заметно отличаются от тех, что строят обученные порождающие классификаторы (рис. 9.6).

Уравнение (7.13) на стр. 233 выражает отношение правдоподобия в виде $\exp(\gamma(d(\mathbf{x}) - d_0))$, где $d(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - t$. Так как в случае дискриминантного обучения мы обучаем все параметры сразу, то γ и d_0 можно включить в \mathbf{w} и t . Поэтому модель логистической регрессии описывается просто уравнением:

$$\hat{p}(\mathbf{x}) = \frac{\exp(\mathbf{w} \cdot \mathbf{x} - t)}{\exp(\mathbf{w} \cdot \mathbf{x} - t) + 1} = \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} - t))}.$$

В предположении, что метки классов $y = 1$ для положительных примеров и $y = 0$ для отрицательных, эта формула определяет распределение Бернулли для каждого обучающего примера:

$$P(y_i | \mathbf{x}_i) = \hat{p}(\mathbf{x}_i)^{y_i} (1 - \hat{p}(\mathbf{x}_i))^{(1-y_i)}.$$

Важно отметить, что параметры этих распределений Бернулли связаны между собой посредством \mathbf{w} и t , и, следовательно, существует один параметр для каждого признака, а не по одному для каждого обучающего примера.

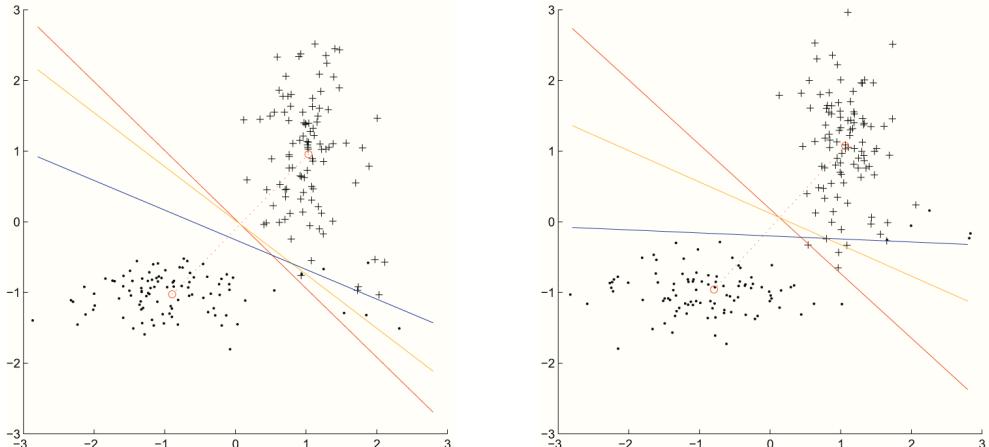


Рис. 9.6. (Слева) На этом наборе данных логистическая регрессия (**синяя линия**) лучше базового линейного классификатора (**красная линия**) и классификатора по методу наименьших квадратов (**оранжевая линия**), потому что последние два более чувствительны к форме классов, тогда как логистическая регрессия сосредоточена на области, где классы перекрываются. **(Справа)** На этом, лишь немного отличающемся наборе данных оба метода оказываются лучше логистической регрессии, потому что она чрезмерно концентрируется на отслеживании перехода от преимущественно положительной к преимущественно отрицательной области

Функция правдоподобия имеет вид:

$$\text{CL}(\mathbf{w}, t) = \prod_i P(y_i | \mathbf{x}_i) = \prod_i \hat{p}(\mathbf{x}_i)^{y_i} (1 - \hat{p}(\mathbf{x}_i))^{(1-y_i)}.$$

Она называется *условным правдоподобием*, чтобы подчеркнуть, что дает *условную* вероятность $P(y_i | \mathbf{x}_i)$, а не $P(\mathbf{x}_i)$, как в порождающей модели. Отметим, что для использования произведения необходимо предположение о независимости значений y при заданном \mathbf{x} ; но это абсолютно разумное предположение, далеко не такое сильное, как наивное байесовское предположение о независимости \mathbf{x} в каждом классе. Как обычно, проще работать с логарифмом функции правдоподобия:

$$\begin{aligned} \text{LCL}(\mathbf{w}, t) &= \sum_i y_i \ln \hat{p}(\mathbf{x}_i) + (1-y_i) \ln(1 - \hat{p}(\mathbf{x}_i)) = \\ &= \sum_{\mathbf{x}^{\oplus} \in Tr^{\oplus}} \ln \hat{p}(\mathbf{x}^{\oplus}) + \sum_{\mathbf{x}^{\ominus} \in Tr^{\ominus}} \ln(1 - \hat{p}(\mathbf{x}^{\ominus})). \end{aligned}$$

Мы хотим максимизировать логарифмическое условное правдоподобие относительно этих параметров, а это означает, что все частные производные должны быть равны нулю:

$$\begin{aligned} \nabla_{\mathbf{w}} \text{LCL}(\mathbf{w}, t) &= \mathbf{0}; \\ \frac{\partial}{\partial t} \text{LCL}(\mathbf{w}, t) &= 0. \end{aligned}$$

Хотя эти уравнения не дают аналитического решения, их можно использовать для лучшего понимания природы логистической регрессии. Сосредоточимся на t и проделаем сначала подготовительные алгебраические преобразования:

$$\begin{aligned} \ln \hat{p}(\mathbf{x}) &= \ln \frac{\exp(\mathbf{w} \cdot \mathbf{x} - t)}{\exp(\mathbf{w} \cdot \mathbf{x} - t) + 1} = \mathbf{w} \cdot \mathbf{x} - t - \ln(\exp(\mathbf{w} \cdot \mathbf{x} - t) + 1); \\ \frac{\partial}{\partial t} \ln \hat{p}(\mathbf{x}) &= -1 - \frac{\partial}{\partial t} \ln(\exp(\mathbf{w} \cdot \mathbf{x} - t) + 1) = \\ &= -1 - \frac{1}{\exp(\mathbf{w} \cdot \mathbf{x} - t) + 1} \exp(\mathbf{w} \cdot \mathbf{x} - t) \cdot (-1) = \hat{p}(\mathbf{x}) - 1. \end{aligned}$$

Аналогично для отрицательных примеров:

$$\begin{aligned} \ln(1 - \hat{p}(\mathbf{x})) &= \ln \frac{1}{\exp(\mathbf{w} \cdot \mathbf{x} - t) + 1} = -\ln(\exp(\mathbf{w} \cdot \mathbf{x} - t) + 1); \\ \frac{\partial}{\partial t} \ln(1 - \hat{p}(\mathbf{x})) &= \frac{\partial}{\partial t} - \ln(\exp(\mathbf{w} \cdot \mathbf{x} - t) + 1) = \\ &= \frac{-1}{\exp(\mathbf{w} \cdot \mathbf{x} - t) + 1} \exp(\mathbf{w} \cdot \mathbf{x} - t) \cdot (-1) = \hat{p}(\mathbf{x}). \end{aligned}$$

Отсюда следует, что частная производная LCL по t записывается просто:

$$\frac{\partial}{\partial t} LCL(\mathbf{w}, t) = \sum_{\mathbf{x}^{\oplus} \in Tr^{\oplus}} (\hat{p}(\mathbf{x}) - 1) + \sum_{\mathbf{x}^{\ominus} \in Tr^{\ominus}} \hat{p}(\mathbf{x}^{\ominus}) = \sum_{\mathbf{x}_i \in Tr} (\hat{p}(\mathbf{x}_i) - y_i).$$

Для оптимального решения эта частная производная равна нулю. Это означает, что в среднем предсказанная вероятность должна быть равна доле положительных примеров *pos*. Это удовлетворительный результат, поскольку это, очевидно, желательное глобальное свойство калиброванного классификатора.

Отметим, что группирующие модели, например деревья оценивания вероятностей, обладают этим свойством по построению, поскольку предсказанная вероятность в них считается равной эмпирической вероятности в сегменте.

Очень похожее рассуждение ведет к частной производной логарифмической условной вероятности по j -му весу w_j . Здесь следует отметить, что в то время как $\frac{\partial}{\partial t} (\mathbf{w} \cdot \mathbf{x} - t) = -1$, мы имеем $\frac{\partial}{\partial w_j} (\mathbf{w} \cdot \mathbf{x} - t) = \frac{\partial}{\partial w_j} (\sum_i w_j x_j - t) = x_j - j$ -му признаку объекта. Отсюда

$$\frac{\partial}{\partial t} LCL(\mathbf{w}, t) = \sum_{\mathbf{x}_i \in Tr} (y_i - \hat{p}(\mathbf{x}_i)) x_{ij}. \quad (9.5)$$

Приравнивание этой частной производной к нулю выражает еще одно калибровочное свойство на уровне отдельных признаков. Например, если j -ый признак является булевым и разреженным, преимущественно равным нулю, то это калибровочное свойство включает только объекты \mathbf{x}_i , для которых $x_{ij} = 1$: в среднем предсказанная вероятность для этих объектов должна быть равна доле положительных среди них.

Пример 9.6 (одномерная логистическая регрессия). Рассмотрим данные на рис. 9.7, по 20 точек в каждом классе. Хотя оба класса были выбраны из нормальных распределений, перекрытие классов в данном примере меньше, чем можно было бы ожидать на основе их средних. Логистическая регрессия может воспользоваться этим фактом и дает гораздо более крутой сигмоид, чем базовый линейный классификатор с логистической калибровкой (см. объяснение в примере 7.7 на стр. 233), который формулируется вслед за терминах средних и дисперсий классов. Показаны также оценки вероятностей, полученные из выпуклой оболочки кривой РХП (см. рис. 7.13 на стр. 235); эта процедура калибровки непараметрическая и потому способна лучше обнаружить ограниченное перекрытие классов.

В терминах статистики логистическая регрессия имеет лучшую среднеквадратичную ошибку (0.040), чем логистически калиброванный классификатор (0.057). Изотонная калибровка дает наименьшую ошибку (0.021), но отметим, что не применялось слаживание вероятностей для снижения риска переобучения. Сумма предсказанных вероятностей равна 18.7 для логистически калиброванного классификатора и 20 для двух других, то есть равна числу примеров, что является необходимым условием полной калибровки. Наконец, $\sum_{\mathbf{x}_i \in Tr} (y_i - \hat{p}(\mathbf{x}_i)) x_i$ равна 2.6 для логистически калиброванного классификатора, 4.7 – для РХП-калиброванного классификатора и 0 – для логистической регрессии, чего и следовало ожидать из уравнения (9.5).

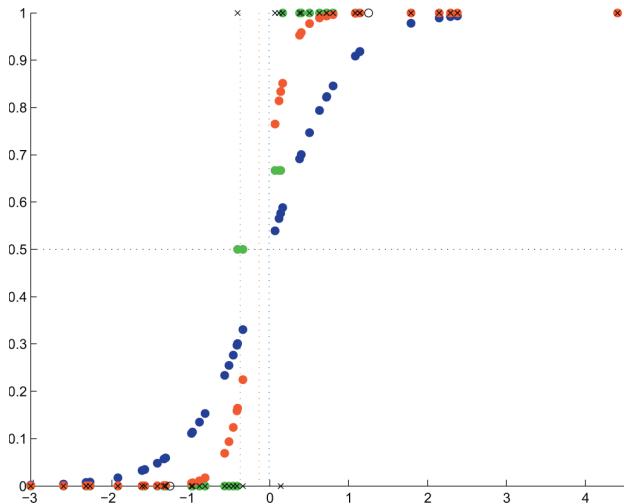


Рис. 9.7. Логистическая регрессия (красная линия) в сравнении с оценками вероятностей, полученными логистической калибровкой (синяя линия) и изотонной калибровкой (зеленая линия); последние две применены к базовому линейному классификатору (оценочные средние точки классов обозначены кружочками). Три соответствующие решающие границы показаны вертикальными пунктирными линиями

Для обучения логистической регрессионной модели нам нужно найти

$$\mathbf{w}^*, t^* = \underset{\mathbf{w}, t}{\operatorname{argmax}} \text{CL}(\mathbf{w}, t) = \underset{\mathbf{w}, t}{\operatorname{argmax}} \text{LCL}(\mathbf{w}, t).$$

Можно доказать, что это выпуклая задача оптимизации, а значит, у нее есть всего один максимум. Существует целый ряд методов ее решения. Простой подход, основанный на алгоритме обучения перцентрана, предполагает перебор примеров с использованием следующего правила обновления:

$$\mathbf{w} = \mathbf{w} + \eta(y_i - \hat{p}_i)\mathbf{x}_i,$$

где η — скорость обучения. Обратите внимание на связь с частной производной в уравнении (9.5). По существу, мы используем одиночные примеры для аппроксимации направления наискорейшего подъема.

9.4 Вероятностные модели со скрытыми переменными

Предположим, что перед нами стоит четырехклассовая задача классификации с классами A, B, C и D . При наличии достаточно объемной и репрезентативной обучающей выборки размера n мы можем использовать относительные частоты

в выборке n_A, \dots, n_D для оценивания априорного распределения по классам $\hat{p}_A = n_A/n, \dots, \hat{p}_D = n_D/n$, как уже много раз делали прежде¹. Обратно, если известно априорное распределение и нужно узнать наиболее вероятное распределение по классам в случайной выборке из n объектов, то мы могли бы воспользоваться априорным распределением для вычисления математических ожиданий $\mathbb{E}[n_A] = p_A n, \dots, \mathbb{E}[n_D] = p_D n$. Таким образом, полное знание одного позволяет оценить или вывести другое. Однако иногда мы обладаем лишь частичными знаниями о том и другом. Например, мы можем знать, что $p_A = 1/2$ и что C в два раза вероятнее B , не зная всего априорного распределения. И еще мы можем знать, что выборка, которую мы видели на прошлой неделе, была поровну разнесена между $A \cup B$ и $C \cup D$ и что C и D имели одинаковый размер, но не можем вспомнить размеров A и B порознь. И что теперь делать?

Формализуя все, что нам известно об априорном распределении, мы имеем: $p_A = 1/2; p_B = \beta$ (пока неизвестна); $p_C = 2\beta$, поскольку она в два раза больше p_B , и $p_D = 1/2 - 3\beta$, поскольку сумма всех четырех вероятностей должна быть равна 1. Далее: $n_A + n_B = a + b = s, n_C = c$ и $n_D = d$, где s, c и d известны. Требуется вычислить a, b и β , однако мы, похоже, столкнулись с проблемой курицы и яйца. Если бы мы знали β , то располагали бы полным знанием об априорном распределении и могли бы вывести математические ожидания a и b :

$$\frac{\mathbb{E}[a]}{\mathbb{E}[b]} = \frac{1/2}{\beta}; \quad \mathbb{E}[a] + \mathbb{E}[b] = s,$$

откуда

$$\mathbb{E}[a] = \frac{1}{1+2\beta} s; \quad \mathbb{E}[b] = \frac{2\beta}{1+2\beta} s. \quad (9.6)$$

Так, например, если $s = 20$ и $\beta = 1/10$, то $\mathbb{E}[a] = 16\frac{2}{3}$ и $\mathbb{E}[b] = 3\frac{1}{3}$.

Обратно, зная a и b , мы могли бы оценить β , применив оценку максимального правдоподобия и используя мультиномиальное распределение a, b, c и d :

$$P(a,b,c,d|\beta) = K(1/2)^a \beta^b (2\beta)^c (1/2 - 3\beta)^d, \\ \ln P(a,b,c,d|\beta) = \ln K + a \ln(1/2) + b \ln \beta + c \ln(2\beta) + d \ln(1/2 - 3\beta).$$

Здесь K – комбинаторная постоянная, которая не влияет на значение β , доставляющее максимум правдоподобию. Частная производная по β равна

$$\frac{\partial}{\partial \beta} \ln P(a,b,c,d|\beta) = \frac{b}{\beta} + \frac{2c}{2\beta} - \frac{3d}{1/2 - 3\beta}.$$

¹ Разумеется, если вы не уверены, достаточно ли велика выборка, то лучше сгладить эти оценки на основе относительных частот, например с помощью [поправки Лапласа](#) (раздел 2.3).

Приравнивая ее к нулю и решая уравнение относительно β , мы, наконец, получаем:

$$\hat{\beta} = \frac{b+c}{6(b+c+d)}. \quad (9.7)$$

Например, если $b = 5$ и $c = d = 10$, то $\hat{\beta} = 1/10$.

Чтобы разорвать этот замкнутый круг, мы можем повторять следующие два шага: (i) вычислить математическое ожидание отсутствующих частот a и b по предполагаемому или ранее оцененному значению параметра β ; (ii) вычислить оценку максимального правдоподобия параметра β по предполагаемым или ранее оцененным математическим ожиданиям отсутствующих частот a и b . Эти два шага повторяются до достижения стационарного состояния. Так, если начать с $a = 15$, $b = 5$ и $c = d = 10$, то, как мы только что видели, $\hat{\beta} = 1/10$. Подставляя это значение β в уравнение (9.6), получаем $\mathbb{E}[a] = 16\frac{2}{3}$ и $\mathbb{E}[b] = 3\frac{1}{3}$. Подставляя эти значения обратно в уравнение (9.7), получаем $\hat{\beta} = 2/21$, что, в свою очередь, дает $\mathbb{E}[a] = 16.8$ и $\mathbb{E}[b] = 3.2$ и т. д. Для достижения стационарного состояния, в котором $\beta = 0.0948$, $a = 16.813$ и $b = 3.187$, требуется менее 10 итераций. В данном простом случае это действительно глобальный оптимум, который достигается независимо от начальной точки просто потому, что связь между b и β монотонна ($\mathbb{E}[b]$ возрастает вместе с β , согласно уравнению (9.6)), а $\hat{\beta}$ возрастает вместе с b , согласно уравнению (9.7)). Однако в общем случае это не так, мы еще вернемся к этому моменту ниже.

EM-алгоритм

Проблема, которую мы только что обсуждали, – пример задач с отсутствующими данными, когда все множество данных Y разбивается на наблюдаемые переменные X и *скрытые переменные* Z (иногда их называют еще *латентными переменными*). В нашем примере наблюдаемыми были переменные c , d и s , а скрытыми – переменные a и b . Имеются также параметры модели θ – в нашем случае один параметр β^1 . Обозначим оценку θ на t -ой итерации θ^t . Существуют две относящиеся к делу величины:

- ☞ математическое ожидание $\mathbb{E}[Z|X,\theta^t]$ скрытой переменной при условии наблюдаемых переменных и текущей оценки параметров (так, в уравнении (9.6) математические ожидания a и b зависят от s и β);
- ☞ правдоподобие $P(Y|\theta)$, которое используется для нахождения максимизирующего значения θ .

¹ Параметры модели также в некотором роде «скрыты», но отличаются от скрытых переменных тем, что мы и не рассчитываем наблюдать их значения (например, средней точки класса), тогда как скрытую переменную, в принципе, можно было бы наблюдать, но вот в данном конкретном случае с этим не сложилось.

В функции правдоподобия нам нужны значения $Y = X \cup Z$. Очевидно, что мы используем наблюдаемые значения X , но для Z нам необходимы ранее вычисленные математические ожидания. Это означает, что на самом деле мы хотим максимизировать величину $P(X \cup \mathbb{E}[Z|X, \theta'] | \theta)$ или, что эквивалентно, ее логарифм. Сделаем теперь предположение, что логарифм функции правдоподобия линейно зависит от Y : отметим, что в рассмотренном выше примере это предположение выполнялось. Для любой линейной функции f справедливо равенство $f(\mathbb{E}[Z]) = \mathbb{E}[f(Z)]$, и, следовательно, мы можем вынести математическое ожидание из целевой функции:

$$\ln P(X \cup \mathbb{E}[Z|X, \theta'] | \theta) = \mathbb{E}[\ln P(X \cup Z | \theta) | X, \theta'] = \mathbb{E}[\ln P(Y | \theta) | X, \theta']. \quad (9.8)$$

Это последнее выражение обычно обозначают $Q(\theta | \theta')$, поскольку оно говорит, как вычислить следующее значение θ , зная текущее:

$$\theta^{t+1} = \arg \max_{\theta} Q(\theta | \theta^t) = \arg \max_{\theta} \mathbb{E}[\ln P(Y | \theta) | X, \theta^t]. \quad (9.9)$$

Это и есть общая форма знаменитого алгоритма *Expectation-Maximisation (EM-алгоритма)* – эффективного подхода к построению вероятностных моделей со скрытыми переменными или отсутствующими данными. Как в рассмотренном выше примере, мы производим итерации, каждая из которых состоит из двух шагов: присваивание ожидаемых значений скрытым переменным при условии текущих оценок параметров и переоценка параметров с учетом обновленных ожидаемых значений – и так до тех пор, пока не будет достигнуто стационарное состояние. Чтобы начать итерации, мы можем каким-то образом инициализировать либо параметры, либо скрытые переменные. Этот алгоритм очень напоминает алгоритм [«К средних»](#) (алгоритм 8.1 на стр. 260), в котором также производятся итерации из двух шагов: назначение точек в качестве текущих средних кластеров и переоценка средних кластеров с учетом новых назначений. Вскоре мы увидим, что это сходство не случайно. Как и в случае алгоритма K средних, можно доказать, что EM-алгоритм сходится к стационарному состоянию для широкого класса вероятностных моделей. Однако EM-алгоритм может остановиться в локальном минимуме при неудачном выборе начальной конфигурации.

Гауссовые смесевые модели

Хорошо известное применение EM-алгоритма – оценивание параметров *гауссовой смесевой модели* по данным. В такой модели данные берутся из K нормальных распределений, каждое со своим средним μ_j и ковариационной матрицей Σ_j , причем доля точек, взятых из каждой гауссианы, определяется априорным распределением $\tau = (\tau_1, \dots, \tau_K)$. Если бы каждая точка в выборке была помечена индексом гауссианы, из которой она взята, то мы имели бы простую задачу классификации, которую было бы легко решить, независимо оценив μ_j и Σ_j каждой гауссианы по данным, принадлежащим классу j . Однако мы сейчас рассматриваем гораздо

более трудную задачу прогностической кластеризации, в которой метки классов скрыты и должны быть восстановлены по наблюдаемым значениям признаков.

Для построения соответствующей модели удобно для каждой точки \mathbf{x}_i иметь битовый вектор $\mathbf{z}_i = (z_{i1}, \dots, z_{iK})$, в котором ровно один бит z_{ij} равен 1, а все остальные – 0. Такой вектор говорит о том, что i -я точка взята из j -й гауссианы. В этих обозначениях мы можем модифицировать формулу [многомерного нормального распределения](#) (уравнение (9.2)) для получения общего выражения для гауссовой смесевой модели:

$$P(\mathbf{x}_i, \mathbf{z}_i | \theta) = \sum_{j=1}^K z_{ij} \tau_j \frac{1}{(2\pi)^{d/2} \sqrt{|\Sigma_j|}} \exp\left(-\frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j)\right). \quad (9.10)$$

Здесь θ – конгломерат всех параметров $\tau, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$ и $\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K$. Интерпретация в качестве порождающей модели следующая: сначала случайным образом выбираем гауссиану, применяя априорное распределение τ , а затем используем выбранную гауссиану с помощью индикаторных переменных z_{ij} .

Для применения ЕМ-алгоритма выписываем Q-функцию:

$$\begin{aligned} Q(\theta | \theta^\ell) &= \mathbb{E}[\ln P(\mathbf{X} \cup \mathbf{Z} | \theta) | \mathbf{X}, \theta^\ell] = \\ &= \mathbb{E}\left[\ln \prod_{i=1}^n P(\mathbf{x}_i \cup \mathbf{z}_i | \theta) \middle| \mathbf{X}, \theta^\ell\right] = \\ &= \mathbb{E}\left[\sum_{i=1}^n \ln P(\mathbf{x}_i \cup \mathbf{z}_i | \theta) \middle| \mathbf{X}, \theta^\ell\right] = \\ &= \mathbb{E}\left[\sum_{i=1}^n \ln \sum_{j=1}^K z_{ij} \tau_j \frac{1}{(2\pi)^{d/2} \sqrt{|\Sigma_j|}} \exp\left(-\frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j)\right) \middle| \mathbf{X}, \theta^\ell\right] = \\ &= \mathbb{E}\left[\sum_{i=1}^n \sum_{j=1}^K z_{ij} \ln \left(\tau_j \frac{1}{(2\pi)^{d/2} \sqrt{|\Sigma_j|}} \exp\left(-\frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j)\right) \right) \middle| \mathbf{X}, \theta^\ell\right] = (*) \\ &= \mathbb{E}\left[\sum_{i=1}^n \sum_{j=1}^K z_{ij} \left(\ln \tau_j - \frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln |\Sigma_j| - \frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j) \right) \middle| \mathbf{X}, \theta^\ell\right] = \\ &= \sum_{i=1}^n \sum_{j=1}^K \mathbb{E}[z_{ij} | \mathbf{X}, \theta^\ell] \left(\ln \tau_j - \frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln |\Sigma_j| - \frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j) \right). \end{aligned} \quad (9.11)$$

Шаг, помеченный звездочкой (*), возможен, потому что для любого i среди чисел z_{ij} только одно отлично от 0, следовательно, мы можем вынести индикаторные переменные из-под знака логарифма. В последней строке Q -функция представлена в желаемой форме, включающей, с одной стороны, математические ожидания скрытых переменных при условии наблюдаемых данных \mathbf{X} и ранее

оцененных параметров θ^t , а с другой – зависящими от θ выражениями, которые позволяют найти θ^{t+1} путем максимизации.

Шаг Expectation EM-алгоритма сводится, таким образом, к вычислению ожидаемых значений индикаторных переменных $\mathbb{E}[z_{ij} | \mathbf{X}, \theta^t]$. Отметим, что математические ожидания булевых переменных принимают значения во всем отрезке $[0, 1]$ с тем ограничением, что $\sum_{j=1}^K z_{ij} = 1$ для всех i . По существу, жесткое назначение кластеров в алгоритме K средних заменяется мягким назначением – это один из видов обобщения алгоритма K средних с помощью гауссовых смесевых моделей. Теперь предположим, что $K = 2$ и что мы ожидаем, что кластеры имеют одинаковые размеры и одинаковые ковариации. Если заданная точка \mathbf{x}_i равнодалена от средних точек обоих кластеров (а точнее, наших оценок этих точек), то очевидно, что $\mathbb{E}[z_{i1} | \mathbf{X}, \theta^t] = \mathbb{E}[z_{i2} | \mathbf{X}, \theta^t] = 1/2$. В общем случае эти математические ожидания распределяются пропорционально массе вероятности, назначенной этой точке каждой гауссианой:

$$\mathbb{E}[z_{ij} | \mathbf{X}, \theta^t] = \frac{\tau_j^t f(\mathbf{x}_i | \mu_j^t, \Sigma_j^t)}{\sum_{k=1}^K \tau_k^t f(\mathbf{x}_i | \mu_k^t, \Sigma_k^t)}, \quad (9.12)$$

где $f(\mathbf{x} | \mu, \Sigma)$ – многомерная гауссова функция плотности.

На шаге Maximisation мы оптимизируем параметры в уравнении (9.11). Отметим, что не существует никакой связи между членами, содержащими τ_j , и членами, содержащими другие параметры, поэтому априорное распределение τ можно оптимизировать отдельно:

$$\begin{aligned} \tau^{t+1} &= \arg \max_{\tau} \sum_{i=1}^n \sum_{j=1}^K \mathbb{E}[z_{ij} | \mathbf{X}, \theta^t] \ln \tau_j = \\ &= \arg \max_{\tau} \sum_{j=1}^K E_j \ln \tau_j \quad \text{с ограничением } \sum_{j=1}^K \tau_j = 1, \end{aligned}$$

где E_j обозначает сумму $\sum_{i=1}^n \mathbb{E}[z_{ij} | \mathbf{X}, \theta^t]$ – полное членство j -го кластера; отметим, что $\sum_{j=1}^K E_j = n$. Для простоты предположим, что $K = 2$, так что $\tau_2 = 1 - \tau_1$, тогда

$$\tau_1^{t+1} = \arg \max_{\tau_1} E_1 \ln \tau_1 + E_2 \ln(1 - \tau_1).$$

Приравнивая к нулю частную производную по τ_1 и решая получившееся уравнение относительно τ_1 , легко проверить, что $\tau_1^{t+1} = E_1 / (E_1 + E_2) = E_1 / n$ и, следовательно, $\tau_2^{t+1} = E_2 / n$. В общем случае K кластеров имеем аналогично:

$$\tau_j^{t+1} = \frac{E_j}{\sum_{k=1}^K E_k} = \frac{1}{n} \sum_{i=1}^n \mathbb{E}[z_{ij} | \mathbf{X}, \theta^t]. \quad (9.13)$$

Средние и ковариационные матрицы можно оптимизировать для каждого кластера по отдельности:

$$\begin{aligned}\mu_j^{t+1}, \Sigma_j^{t+1} &= \arg \max_{\mu_j, \Sigma_j} \sum_{i=1}^n \mathbb{E}[z_{ij} | \mathbf{X}, \theta^t] \left(-\frac{1}{2} \ln |\Sigma_j| - \frac{1}{2} (\mathbf{x}_i - \mu_j)^T \Sigma_j^{-1} (\mathbf{x}_i - \mu_j) \right) = \\ &= \arg \max_{\mu_j, \Sigma_j} \sum_{i=1}^n \mathbb{E}[z_{ij} | \mathbf{X}, \theta^t] \left(\frac{1}{2} \ln |\Sigma_j| + \frac{1}{2} (\mathbf{x}_i - \mu_j)^T \Sigma_j^{-1} (\mathbf{x}_i - \mu_j) \right).\end{aligned}$$

Отметим, что выражение в скобках – квадрат расстояния, а математические ожидания играют роль весов, соответствующих каждому объекту. Эта формула описывает обобщенный вариант задачи нахождения точки, которая *обращает в минимум сумму квадратов евклидовых расстояний* до множества точек (теорема 8.1 на стр. 249). И если решением той задачи являлось среднее арифметическое, то здесь мы просто берем *взвешенное* среднее по всем точкам:

$$\mu_j^{t+1} = \frac{1}{E_j} \sum_{i=1}^n \mathbb{E}[z_{ij} | \mathbf{X}, \theta^t] \mathbf{x}_i = \frac{\sum_{i=1}^n \mathbb{E}[z_{ij} | \mathbf{X}, \theta^t] \mathbf{x}_i}{\sum_{i=1}^n \mathbb{E}[z_{ij} | \mathbf{X}, \theta^t]}. \quad (9.14)$$

Аналогично ковариационная матрица вычисляется как взвешенное среднее ковариационных матриц по всем точкам, с учетом новой оценки среднего:

$$\begin{aligned}\Sigma_j^{t+1} &= \frac{1}{E_j} \sum_{i=1}^n \mathbb{E}[z_{ij} | \mathbf{X}, \theta^t] (\mathbf{x}_i - \mu_j^{t+1}) (\mathbf{x}_i - \mu_j^{t+1})^T = \\ &= \frac{\sum_{i=1}^n \mathbb{E}[z_{ij} | \mathbf{X}, \theta^t] (\mathbf{x}_i - \mu_j^{t+1}) (\mathbf{x}_i - \mu_j^{t+1})^T}{\sum_{i=1}^n \mathbb{E}[z_{ij} | \mathbf{X}, \theta^t]}.\end{aligned} \quad (9.15)$$

Уравнения (9.12)–(9.15) составляют полученное с помощью ЕМ-алгоритма решение для обучения гауссовой смесевой модели по непомеченной выборке. Я представил его в самой общей форме, когда явно моделируются неравные размеры кластеров и различные ковариационные матрицы. Последнее важно, так как позволяет иметь кластеры различной формы – в отличие от алгоритма K средних, в котором предполагается, что все кластеры имеют одинаковую сферическую форму. Следовательно, границы между кластерами не будут линейными, как в случае кластеризации методом K средних. На рис. 9.8 демонстрируется сходимость ЕМ-алгоритма на простом одномерном наборе данных, а также существование нескольких стационарных состояний.

В заключение повторим, что алгоритм Expectation-Maximisation – гибкий и эффективный способ решения задач с отсутствующими данными, имеющий твердые теоретические основания. Как мы видели на примере гауссовой смесевой модели, основным его ингредиентом является выражение для параметрической функции правдоподобия $P(X \cup Z | \theta)$, из которого с помощью Q -функции можно вывести уравнения обновления. Нелишним будет предостережение о том, что, за исключением простейших случаев, стационарных состояний может быть несколько. Поэтому, как и в случае алгоритма K средних, оптимизацию следует выполнять несколько раз с различными начальными значениями.

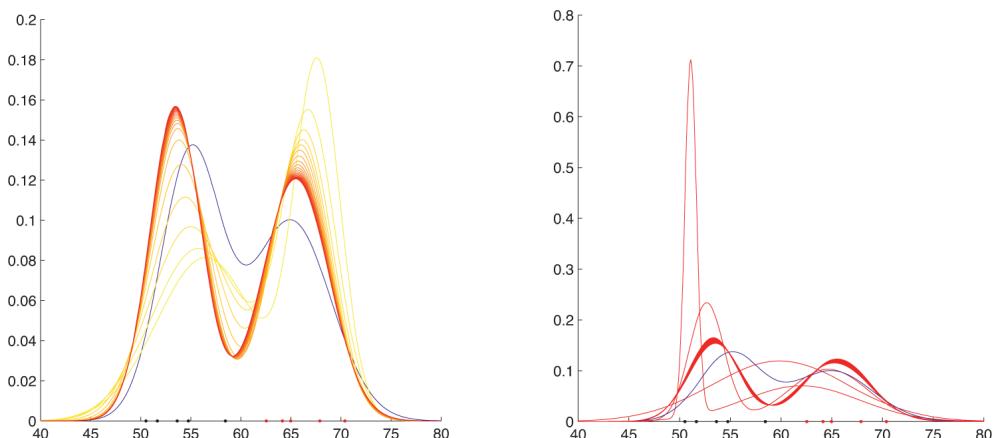


Рис. 9.8. (Слева) Синей линией показана истинная гауссова смесовая модель, из которой были выбраны 10 точек на оси x ; цвет точек обозначает, из какой гауссианы они взяты: левой или правой. Остальные линии демонстрируют сходимость ЕМ-алгоритма к стационарному состоянию после случайной инициализации. **(Справа)** На этом графике показаны четыре стационарных состояния для одного и того же набора данных. Было выполнено 20 итераций ЕМ-алгоритма; большая толщина одной из линий говорит о том, что в этой конфигурации время сходимости оказалось дольше

9.5 Модели на основе сжатия

Мы закончим эту главу кратким обсуждением подхода к машинному обучению, который одновременно тесно связан и разительно отличается от вероятностного.

$$y_{\text{MAP}} = \arg \max_y P(X = x | Y = y)P(Y = y).$$

Взяв логарифмы со знаком минус, мы можем преобразовать эту задачу к эквивалентной задаче минимизации:

$$y_{\text{MAP}} = \arg \min_y -\log P(X = x | Y = y) - \log P(Y = y). \quad (9.16)$$

Это следует из того, что для любых двух вероятностей $0 < p < p' < 1$ имеем $\infty > -\log p > -\log p' > 0$. Если вероятность события равна p , то логарифм p со знаком минус дает количественное выражение *объема информации*, содержащейся в сообщении о том, что событие произошло. Это интуитивно понятно, потому что чем менее ожидаемо событие, тем больше информации содержит объявление о нем. Единица информации зависит от основания логарифма: принято брать логарифмы по основанию 2, и в таком случае объем информации измеряется в битах. Например, если вы один раз подбросите правильную монету и скажете мне, что выпала решка, то в этом сообщении будет содержаться $-\log_2 1/2 =$

= 1 бит информации; если вы один раз бросите правильную кость и скажете, что выпала шестерка, то объем информации в сообщении будет равен $-\log_2 1/6 = -2.6$ бита. Уравнение (9.16) говорит, что решающее правило МАР выбирает наименее неожиданный, или самый ожидаемый, класс объекта x при условии заданных априорных распределений и правдоподобий. Мы пишем $IC(X|Y) = -\log_2 P(X|Y)$ и $IC(Y) = -\log_2 P(Y)$ ¹.

Пример 9.7 (классификация на основе информации). В табл. 9.2 воспроизведена левая часть табл. 1.3 на стр. 41 вместе с соответствующими количественными показателями объема информации. Если Y распределена равномерно, то $IC(Y = \text{спам}) = 1$ бит и $IC(Y = \text{неспам}) = 1$ бит. Отсюда следует, что

$$\arg \min_y (IC(\text{виагра} = 1|Y = y) + IC(Y = y)) = \text{спам};$$

$$\arg \min_y (IC(\text{виагра} = 0|Y = y) + IC(Y = y)) = \text{неспам}.$$

Если неспам в четыре раза вероятнее спама, то $IC(Y = \text{спам}) = 2.32$ бита, $IC(Y = \text{неспам}) = 0.32$ бита и $\arg \min_y (IC(\text{виагра} = 1|Y = y) + IC(Y = y)) = \text{неспам}$.

Y	$P(\text{виагра} = 1 Y)$	$IC(\text{виагра} = 1 Y)$	$P(\text{виагра} = 0 Y)$	$IC(\text{виагра} = 0 Y)$
Спам	0.40	1.32 бита	0.60	0.74 бита
Неспам	0.12	3.06 бита	0.88	0.18 бита

Таблица 9.2. Примеры маргинальных правдоподобий

Очевидно, что при равномерном распределении k исходов каждый исход содержит один и тот же объем информации $-\log_2 1/k = \log_2 k$. Если распределение неравномерно, то объем информации будет различаться, поэтому имеет смысл вычислить средний объем информации, или [энтропию](#) $\sum_{i=1}^k p_i \log_2 p_i$. Мы уже встречались с энтропией в разделе 5.1, только тогда называли ее [мерой нечистоты](#).

До сих пор в том, что я сказал, нового было разве что наличие взаимно однозначного соответствия между вероятностью и объемом информации. А настоящим двигателем обучения на основе сжатия является фундаментальный результат из теории информации, доказанный в 1948 году Клодом Шенноном. Результат Шеннона гласит – без излишнего формализма, – что невозможно передавать информацию со скоростью, большей энтропии, но с помощью хитроумных двоичных кодов можно достичь скорости, сколь угодно близкой к оптимальной. К числу хорошо известных относятся коды Шеннона-Фано и Хаффмана, с которыми стоит познакомиться, потому что в них используется простая древовидная структура для построения кода на основе эмпирических вероятностей. Существуют и более эффективные коды, например арифметическое кодирование, в которых несколько сообщений объединяются в одно кодовое слово.

¹ Здесь IC – сокращение от Information Content (объем информации). – Прим. перев.

Предполагая наличие почти оптимального кода, мы можем сменить точку зрения и использовать объем информации – или, как его чаще называют, «длину описания» – вместо вероятности. Упрощенный вариант принципа минимальной длины описания (МДО) формулируется следующим образом.

Определение 9.1 (принцип минимальной длины описания). Обозначим $L(m)$ длину в битах описания модели m , а $L(D|m)$ – длину в битах описания данных D при условии модели m . Согласно принципу минимальной длины описания, предпочтительной является модель, минимизирующая сумму длин описания модели и данных при условии модели:

$$m_{\text{MDL}} = \arg \min_{m \in M} (L(m) + L(D|m)). \quad (9.17)$$

В контексте прогностического обучения «описание данных при условии модели» относится к той информации – помимо самой модели и значений признаков данных, – которая необходима для вывода целевых меток. Если модель на 100% точна, то никакой дополнительной информации не нужно, так что этот член, по существу, служит для количественного выражения степени некорректности модели. Например, в случае двух равномерно распределенных классов нам нужен один бит для каждого объекта, неправильно классифицированного моделью. Член $L(m)$ является количественным выражением сложности модели. Например, если мы аппроксимируем данные полиномом, то должны закодировать степень полинома и его корни с некоторой разрешающей способностью. Таким образом, принцип МДО устанавливает компромисс между верностью и сложностью модели: член, описывающий сложность, позволяет избежать переобучения – как *регуляризующий член* в гребневой регрессии (раздел 7.1) и *ослабляющая переменная* в методе опорных векторов с мягким зазором (раздел 7.3).

Какое кодирование использовать для определения сложности модели $L(m)$ – вопрос, часто неочевидный и в какой-то мере субъективный. Тут можно провести аналогию с байесовским подходом, когда мы должны определить априорное распределение моделей. Подход на основе МДО предлагает конкретный способ определения априорного распределения моделей с помощью кодов.

9.6 Вероятностные модели: итоги и литература для дальнейшего чтения

В этой главе мы рассмотрели ряд моделей машинного обучения, в основе которых лежит идея моделирования признаков и целевых переменных случайными величинами, что дает возможность явно представлять и манипулировать уровнем доверия к этим величинам. Такие модели обычно являются прогностическими в том смысле, что их результатом является условное распределение вероятности $P(Y|X)$, с помощью которого Y можно предсказать, зная X . Порождающие мо-

дели оценивают совместное распределение $P(Y, X)$ – часто с помощью функции правдоподобия $P(X|Y)$ и априорного распределения $P(Y)$, – из которого можно получить апостериорное распределение $P(Y|X)$. Напротив, в условных моделях обученное апостериорное распределение $P(Y|X)$ получается непосредственно, без затрат ресурсов на обучение $P(X)$. Для «байесовского» подхода к машинному обучению характерно стремление получить полное апостериорное распределение, когда это осуществимо, а не просто найти максимизирующее значение.

- ☞ В разделе 9.1 мы видели, что нормальное, или гауссово, распределение поддерживает многие полезные геометрические представления, прежде всего потому, что логарифм со знаком минус гауссова правдоподобия можно интерпретировать как квадрат расстояния. Однаковые ковариационные матрицы классов приводят к прямолинейным решающим границам, а это означает, что модели, дающие такие линейные границы, в том числе линейные классификаторы, линейную регрессию и кластеризацию методом K средних, можно интерпретировать с вероятностной точки зрения, что делает встроенные в них предположения явными. Мы рассмотрели два иллюстративных примера: (i) базовый линейный классификатор является оптимальным по Байесу в случае некоррелированных гауссовых признаков с единичной дисперсией и (ii) регрессия по методу наименьших квадратов оптимальна для линейных функций, загрязненных гауссовым шумом.
- ☞ Раздел 9.2 был посвящен различным вариантам наивного байесовского классификатора, в котором делается упрощающее предположение о независимости признаков внутри каждого класса. Обзор и история вопроса приведены в работе Lewis (1998). Эта модель широко используется для информационного поиска и классификации текстов, поскольку она часто дает хорошее ранжирование, пусть даже порождаемые ей оценки вероятностей далеки от совершенства. Хотя обычно в модели, именуемой наивной, признаки рассматриваются как категориальные, или случайные, величины с распределением Бернулли, варианты, в которых применяется мультиномиальное распределение, как правило, лучше моделируют количество вхождений слов в документ (McCallum, Nigam, 1998). Учесть вещественные признаки можно двумя способами: моделируя их как нормально распределенные в пределах каждого класса или с помощью непараметрического оценивания плотности, – в работе John, Langley (1995) утверждается, что последний способ дает лучшие эмпирические результаты. В работе Webb, Boughton, Wang (2005) обсуждается, как можно ослабить требование независимости, лежащее в основе наивной байесовской классификации. Сглаживание вероятностей с помощью t -оценки было впервые предложено в работе Cestnik (1990).
- ☞ Хотя это и звучит парадоксально, я не считаю, что в наивном байесовском классификаторе есть что-то «байесовское». Да, это порождающая вероятностная модель для оценивания апостериорного распределения $P(Y|X)$ по совместному распределению $P(Y, X)$, но на практике апостериорное

распределение очень плохо откалибровано из-за нереалистичных предположений о независимости. В результате анализа, проведенного в работе Domingos, Pazzani (1997), установлено, что наивный байесовский классификатор так часто оказывается успешным вследствие высокого качества $\text{argmax}_Y P(Y|X)$, а не апостериорного распределения как такового. Более того, даже без использования правила Байеса, при определении доставляющего максимум значения Y , можно обойтись, поскольку оно служит лишь для преобразования неоткалиброванных правдоподобий в неоткалиброванные апостериорные вероятности. Поэтому я рекомендую использовать наивные байесовские правдоподобия как оценки с неизвестной шкалой, для которых порог принятия решения следует откалибровать посредством анализа кривой РХП, как обсуждалось в нескольких местах выше.

- ☞ В разделе 9.3 мы рассмотрели широко распространенную модель логистической регрессии. Основная идея – объединить линейную решающую границу с логистической калибровкой, но обучить ее в дискриминантной манере путем оптимизации условного правдоподобия. Таким образом, вместо моделирования классов в виде облаков точек и вывода из них решающей границы логистическая регрессия сосредоточивается на областях перекрытия классов. Это пример более широкого класса обобщенных линейных моделей (Nelder, Wedderburn, 1972). В работе Jebraa (2004) обсуждаются преимущества дискриминантного обучения, по сравнению с порождающими моделями. Дискриминантное обучение применимо также к последовательным данным в форме условных случайных полей (Lafferty et al., 2001).
- ☞ В разделе 9.4 был в общем виде представлен алгоритм Expectation-Maximisation как способ обучения моделей со скрытыми переменными. Эта общая форма ЕМ-алгоритма предложена в работе Dempster, Laird, Rubin (1977), основанной на более ранних работах. Мы видели, как ее можно применить к гауссовым смесевым моделям для получения более общего варианта прогностической кластеризации методом K средних, который умеет также оценивать формы и размеры кластеров. Однако при этом увеличивается число параметров модели, а значит, и риск застрять в неоптимальном стационарном состоянии. Работы Little, Rubin, 1987 – стандартный источник по работе в условиях отсутствия данных.
- ☞ Наконец, в разделе 9.5 мы вкратце обсудили некоторые идеи, относящиеся ко взгляду на обучение как на сжатие. Связь с вероятностным моделированием заключается в том, что в обоих случаях цель состоит в моделировании и использовании неслучайных аспектов данных. В упрощенной форме принцип минимальной длины описания можно вывести из правила Байеса, взяв логарифм со знаком минус; он утверждает, что следует предпочесть модель, которая минимизирует сумму длин описания модели и данных при условии модели. Первый член количественно выражает сложность модели, а второй – ее верность (поскольку в явном кодировании нуждаются лишь ошибки модели). Достоинство принципа МДО заключ-

чается в том, что схемы кодирования зачастую более осозаемы, их проще определить, чем априорные распределения вероятностей. Однако подойдет не любое кодирование: как и в случае вероятностей, для этих схем необходимо обоснование в моделируемой предметной области. Основополагающими в этой области были работы Solomonoff (1964 a, b); Wallace, Boulton (1968); Rissanen (1978) и другие. Отличным введением и обзором может послужить работа Grunwald (2007).



Признаки

В [ыше я назвал](#) признаки «рабочими лошадками машинного обучения». Настало время рассмотреть их детально. Признаки, или атрибуты, определяются как отображения $f_i: \mathcal{X} \rightarrow \mathcal{F}_i$ пространства объектов \mathcal{X} в область значений признака \mathcal{F}_i . Признаки можно различать по области значений: чаще всего в роли области значений выступает множество целых или вещественных чисел, но бывают и дискретные множества, например цвета, булевы величины и т. д. Можно также различать признаки по доступным операциям. Например, можно вычислить средний возраст совокупности людей, но не среднюю группу крови, поэтому операция нахождения среднего значения для одних признаков допустима, а для других нет. Различные виды признаков мы будем рассматривать в разделе 10.1.

Хотя многие наборы данных уже содержат предопределенные признаки, ими можно манипулировать различными способами. Например, можно изменить область значений признака путем масштабирования или дискретизации; можно выбрать наилучшие признаки из большего набора и работать только с ними; можно объединить два или более признаков в один новый. В действительности сама модель – это способ построения нового признака, решающего поставленную задачу. Преобразования признаков рассматриваются в разделе 10.2, а построение и отбор – тема раздела 10.3.

10.1 Виды признаков

Рассмотрим два признака – возраст человека и номер дома, в котором он живет. Оба признака – целые числа, но используем мы их совершенно по-разному. Вычисление среднего возраста группы людей – осмысленная операция, тогда как средний номер дома вряд ли можно назвать полезной информацией! Иными словами, важна не только область значений признака, но и множество допустимых операций. Оно, в свою очередь, зависит от того, существует ли для значений признака осмысленная *шкала*. Номера домов – на самом деле не целые числа, хотя кажутся таковыми, а *порядковые числа*, или *ординалы*: они позволяют сказать, что номера соседних с десятым домов – 8 и 12, но предположение о том, что расстояние между 8 и 10 такое же, как между 10 и 12, ни на чем не основано. Из-за отсутствия линейной шкалы сложение и вычитание номеров домов, а значит, и операция усреднения лишены смысла.

Вычисления с признаками

Рассмотрим более пристально специальные операции с признаками, которые часто называют агрегатами, или статистиками. Три основные категории: *статистики центральной тенденции*, *статистики разброса* и *статистики формы*. Каждую из них можно интерпретировать либо как теоретическое свойство неизвестной генеральной совокупности, либо как конкретное свойство заданной выборки – нас здесь будут интересовать выборочные статистики.

Начнем со статистик центральной тенденции. Из них наиболее важны следующие:

- ☞ *среднее* значение;
- ☞ *медиана* – значение, которое окажется посередине, если упорядочить признаки по возрастанию значений;
- ☞ *мода* – значение (или значения), которое встречается наиболее часто.

Из этих статистик моду можно вычислить при любой области значений признака; например, мы можем сказать, что в данной совокупности людей чаще всего встречается группа крови О+. Для вычисления медианы значения признака должны быть упорядочены; так, можно вычислить моду и медиану номеров домов в совокупности адресов¹. Для вычисления среднего необходимо, чтобы значение признака было выражено относительно некоторой шкалы; чаще всего шкала линейна и вычисляется как обычное среднее арифметическое, но в замечании 10.1 обсуждаются средние для других шкал. Нередко можно встретить утверждение, что медиана обычно находится между модой и средним, но существует сколько угодно исключений из этого «правила». Выдающийся статистик Карл Пирсон предложил уточненное эвристическое правило (естественно, исключений из него еще больше): медиана обычно находится на трети пути от среднего к моде.

Представьте себе пловца, который проплывает одну и ту же дистанцию d в два разных дня: за a секунд в первый день и за b секунд во второй. Среднее время прохождения дистанции составляет $c = (a + b)/2$ секунд, а средняя скорость равна $d/c = 2d/(a + b)$. Обратите внимание, что средняя скорость *не есть* *среднее арифметическое* двух скоростей, которое равно $(d/a + d/b)/2$; для вычисления средней скорости на фиксированном отрезке пути требуется так называемое *среднее гармоническое*. Если даны два числа x и y (в нашем примере скорости в первый и во второй день, d/a и d/b), то их среднее гармоническое h определяется как

$$h(x, y) = \frac{2}{\frac{1}{x} + \frac{1}{y}} = \frac{2xy}{x + y}.$$

Поскольку $1/h(x, y) = (1/x + 1/y)/2$, то можно сделать вывод, что вычисление среднего гармонического по шкале с единицей *и* соответствует вычислению среднего арифметического по

¹ Если выборка содержит четное число объектов, то срединных значений два. Если для признака определена шкала, то принято в качестве медианы брать среднее этих двух значений; если шкалы нет или если важно, чтобы значение медианы было элементом выборки, то можно либо взять оба значения, назвав их нижней и верхней медианами, либо выбрать одно произвольно.

обратной шкале с единицей $1/u$. В нашем примере скорость на фиксированной дистанции выражается в шкале, обратной временной, и потому для усреднения времени мы пользуемся средним арифметическим, а для усреднения скорости – средним гармоническим. (Если бы мы усредняли скорость за фиксированный интервал *времени*, то она выражалась бы в той же шкале, что и расстояние, и тогда мы воспользовались бы средним арифметическим.)

Хороший пример использования среднего гармонического в машинном обучении возникает при усреднении точности и полноты классификатора. Напомним, что точность – это доля правильных положительных предсказаний ($prec = TP/(TP + FP)$), а полнота – доля положительных результатов, которые правильно предсказаны ($rec = TP/(TP + FN)$). Предположим, что мы сначала вычисляем количество ошибок, усредненное по классам: это среднее арифметическое $Fm = (FP + FN)/2$. Отсюда получаем:

$$\frac{TP}{TP + Fm} = \frac{TP}{TP + (FP + FN)/2} = \frac{2TP}{(TP + FP) + (TP + FN)} = \frac{2}{1/prec + 1/rec}.$$

В последнем члене мы узнаем среднее гармоническое точности и полноты. Поскольку числитель в формулах точности и полноты фиксирован, то вычисление среднего арифметического знаменателей соответствует вычислению среднего гармонического дробей. В контексте информационного поиска среднее гармоническое точности и полноты называется *F-мерой*.

Для других шкал существуют и другие средние. В музыке переход от одной ноты к ноте на октаву выше соответствует удвоению частоты. Поэтому частоты f и $4f$ разделены двумя октавами, и в качестве их среднего имеет смысл взять октаву с частотой $2f$. Для этого используется *среднее геометрическое* $g(x, y) = \sqrt{xy}$. Поскольку $\log\sqrt{xy} = (\log xy)/2 = (\log x + \log y)/2$, то среднему геометрическому соответствует среднее арифметическое по логарифмической шкале. У всех этих средних есть общее свойство: среднее двух значений не меньше наименьшего и не больше наибольшего, и все они обобщаются на случай, когда значений больше двух.

Замечание 10.1. О шкалах и средних

Второй вид вычислений с признаками – статистики разброса. Самые известные из них – *дисперсия*, или среднеквадратичное отклонение от среднего арифметического, и квадратный корень из нее – *стандартное отклонение*. Дисперсия и стандартное отклонение, по существу, измеряют одно и то же, но у последнего то преимущество, что оно выражено в той же шкале, что сами значения признака. Например, дисперсия веса тела, измеренного в килограммах для группы людей, измеряется в k^2 , а стандартное отклонение – в килограммах. Абсолютная разность между средним и медианой никогда не превышает стандартного отклонения – это следствие *неравенства Чебышева*, которое утверждает, что самое большее $1/k^2$ значений отстоят от среднего на расстояние, превышающее k стандартных отклонений.

Более простая статистика разброса – разность между максимальным и минимальным значениями, которая называется *размахом*. Вместе с размахом естественно используется статистика центральной тенденции – *среднее значение размаха*, равное полусумме крайних значений. В этих определениях предполагается линейная шкала, но они обобщаются и на другие шкалы с помощью подходящих преобразований. Например, если признак выражен в логарифмической шкале, как частота, то в качестве размаха мы взяли бы отношение самой высокой и са-

мой низкой частот, а в качестве среднего значения размаха – среднее геометрическое крайних значений.

К статистикам разброса относятся также *процентили*. p -й процентиль – это такое значение, что p процентов объектов оказываются меньше него. Если имеется 100 объектов, то 80-й процентиль – это значение 81-го элемента в списке значений, расположенных в порядке возрастания¹. Если p кратно 25, то процентили называют также квартилями, а если p кратно 10, то децилями. Отметим, что 50-й процентиль, 5-й дециль и второй квартиль – то же самое, что медиана. Процентили, децили и квартили – частные случаи *квантилей*. Зная квантили, мы можем измерить разброс как расстояние между различными квантителями. Например, *межквартильный размах* – это разность между третьим и первым квартилями (то есть 75-м и 25-м процентилями).

Пример 10.1 (график процентилей). Предположим, что требуется обучить модель на пространстве объектов, состоящем из стран, и один из рассматриваемых признаков – валовый внутренний продукт (ВВП) на душу населения. На рис. 10.1 показан *график процентилей* для этого признака. Чтобы получить p -й процентиль, мы находим точку пересечения прямой $y = p$ с пунктирной кривой и считываем значение с оси x . На рисунке показаны 25-й, 50-й и 75-й процентили. Показано также среднее (которое следует вычислять по исходным данным). Как видите, среднее расположено значительно выше медианы; объясняется это главным образом тем, что в нескольких странах очень высокий ВВП на душу населения. Иначе говоря, среднее более чувствительно к *выбросам*, чем медиана, поэтому медиану часто предпочитают среднему в такого рода асимметричных распределениях.

Вам может показаться, что я нарисовал график процентилей вверх ногами: не лучше ли было бы откладывать p по оси x , а процентили по оси y ? Но у такого способа изображения есть одно достоинство: если интерпретировать ось y как вероятности, то график можно воспринимать как *кумулятивное распределение вероятностей*: график функции $P(X \leq x)$ переменной x для случайной величины X . Например, этот график показывает, что $P(X \leq \mu)$ приблизительно равно 0.70, где $\mu = \$11284$ – средний ВВП на душу населения. Иначе говоря, если случайным образом выбрать страну, то вероятность того, что ВВП на душу населения в ней меньше среднего, составляет примерно 0.70.

Поскольку ВВП на душу населения – вещественномозначный признак, то говорить о его mode, возможно, и не имеет смысла, потому что если он измерен достаточно точно, то значения для всех стран будут различны. Эту трудность можно обойти с помощью *гистограммы*, которая подсчитывает количество значений признака в заданном интервале, или столбике.

¹ Как и в случае медианы, имеются трудности с нецелочисленными рангами, и преодолевать их можно разными способами; однако существенных расхождений не возникает, если только размер выборки не слишком мал.



Рис. 10.1. График процентилей ВВП на душу населения для 231 страны (данные получены с сайта www.wolframalpha.com по запросу «GDP per capita»). Вертикальные пунктирные линии обозначают слева направо: первый квартиль (\$900), медиану (\$3600), среднее (\$11 284) и третий квартиль (\$14 750). Межквартильный размах равен \$13 850, а стандартное отклонение \$16 189.

Пример 10.2 (гистограмма). На рис. 10.2 показана гистограмма данных, изображенных на рис. 10.1. Самый левый столбик – мода, более чем в трети стран ВВП на душу населения меньше 2000 долларов. Отсюда следует, что это распределение сильно *скошено вправо* (имеет длинный правый хвост), поэтому среднее оказывается значительно больше медианы.

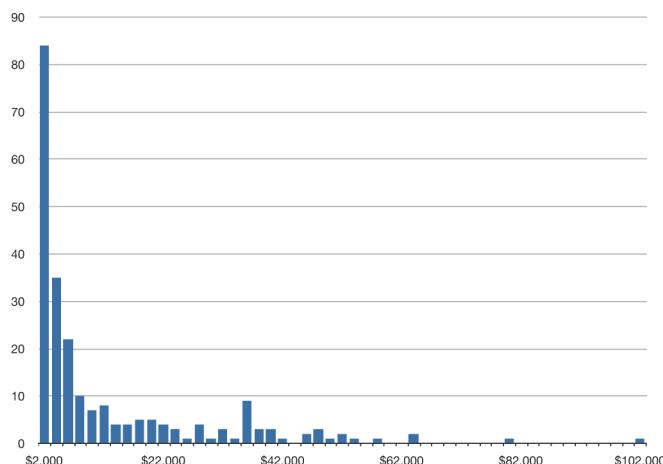


Рис. 10.2. Гистограмма данных, изображенных на рис. 10.1, ширина столбиков равна 2000 долларов

Асимметрию и «островершинность» распределения можно измерить с помощью таких статистик формы, как асимметрия и куртозис. Основная идея состоит в том, чтобы измерить третий и четвертый *центральные моменты* выборки. В общем случае k -й центральный момент выборки $\{x_1, \dots, x_n\}$ определяется формулой $m_k = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^k$, где μ – выборочное среднее. Ясно, что первый центральный момент – среднее отклонение от среднего – всегда равен нулю, потому что положительные и отрицательные отклонения взаимно уничтожаются, а второй центральный момент – среднеквадратичное отклонение от среднего – уже известная нам дисперсия. Третий центральный момент m_3 может быть положительным или отрицательным. *Асимметрия* определяется как m_3/σ_3 , где σ – выборочное стандартное отклонение. Если асимметрия положительна, то распределение скошено вправо, то есть его правый хвост длиннее левого. Если асимметрия отрицательна, то мы имеем противоположный случай скошенного влево распределения. *Куртозис* по определению равен m_4/σ_4 . Поскольку можно показать, что нормальное распределение имеет куртозис 3, то часто вместо куртозиса используют *экссес*: $m_4/\sigma_4 - 3$. Положительный экссес означает, что распределение более островершинное, чем нормальное.

Пример 10.3 (асимметрия и куртозис). В примере ВВП на душу населения асимметрия равна 2.12, а экссес – 2.53. Это подтверждает, что распределение сильно скошено вправо и что оно более островершинное, чем нормальное.

Категориальные, порядковые и количественные признаки

Имея различные статистики, мы можем выделить три основных вида признаков: с осмысленной числовой шкалой, без шкалы, но с отношением порядка и не имеющие ни того, ни другого. Признаки первого типа мы будем называть *количественными*; чаще всего они отображаются на вещественные числа (другой распространенный термин – «непрерывные»). Даже если признак отображается на подмножество вещественных чисел, например возраст, выраженный в годах, различные статистики, в том числе среднее и стандартное отклонения, все же нуждаются в полной вещественной шкале. Признаки, для которых определено отношение порядка, но нет шкалы, называются *порядковыми*. Область значений порядкового признака – некоторое вполне упорядоченное множество, например множество символов или строк. Даже если областью значений признака является множество целых чисел, его порядковость означает, что шкалу следует игнорировать, как в случае с номерами домов. Еще один типичный пример – признаки, выражющие ранговый порядок: первый, второй, третий и т. д. Для порядковых признаков можно определить моду и медиану в качестве статистики центральной тенденции и квантили в качестве статистик разброса.

Признаки, для которых не определены ни порядок, ни шкала, называются *категориальными* (иногда «номинальными»). У них не может быть никакой сводной статистики, кроме моды. Подвидом категориальных признаков являются *булевы признаки*, которым сопоставляются значения истинности *true* и *false*. Эта классификация показана в табл. 10.1.

Вид	Порядок	Шкала	Тенденция	Разброс	Форма
Категориальный	✗	✗	мода	не применимо	не применимо
Порядковый	✓	✗	медиана	квантили	не применимо
Количественный	✓	✓	среднее	размах, межквартильный размах, дисперсия, стандартное отклонение	асимметрия, куртозис

Таблица 10.1. Виды признаков, их свойства и допустимые статистики. Каждый вид наследует статистики от видов, расположенных над ним. Например, мода – это статистика центральной тенденции, которую можно вычислить для признака любого вида

В моделях признаки различных видов трактуются по-разному. Для начала рассмотрим древовидные модели, например решающие деревья. Разделение по категориальному признаку порождает столько дочерних узлов, сколько значений может принимать признак. С другой стороны, порядковые и количественные признаки определяют бинарное разделение: выбирается некое значение v_0 – такое, что все объекты, для которых значение признака меньше или равно v_0 , попадают в левый дочерний узел, а все остальные – в правый. Отсюда следует, что древовидные модели нечувствительны к шкале количественного признака. Например, для обученного дерева не важно, измерена температура по шкале Цельсия или Фаренгейта. Несуществен и переход от линейной шкалы к логарифмической: порог разделения просто будет равен $\log v_0$, а не v_0 . Вообще, древовидные модели нечувствительны к *монотонным* преобразованиям шкалы признака, то есть преобразованиям, не изменяющим относительного порядка его значений. Таким образом, *древовидные модели игнорируют шкалу количественных признаков, трактуя их как порядковые*. То же самое справедливо для моделей на основе правил.

Теперь рассмотрим наивный байесовский классификатор. Мы видели, что в основе работы этой модели лежит оценивание функции правдоподобия $P(X|Y)$ для каждого признака X при условии класса Y . Для категориальных и порядковых признаков с k значениями это сводится к оцениванию $P(X = v_1|Y), \dots, P(X = v_k|Y)$. По существу, порядковые признаки трактуются как категориальные, порядок игнорируется. С количественными признаками работать вообще невозможно, если не дискретизировать их, преобразовав в конечное множество интервалов, то есть в категориальные признаки. Альтернативно можно было бы предположить некоторую параметрическую форму $P(X|Y)$, например нормальное распределение. Мы вернемся к этому вопросу ниже, когда будем обсуждать калибровку признаков.

Если наивный байесовский классификатор умеет работать только с категориальными признаками, то многим геометрическим моделям свойственно прямо

противоположное: они способны иметь дело лишь с количественными признаками. В этом отношении показательны линейные модели: само понятие линейности подразумевает евклидово пространство объектов, в котором признаки играют роль декартовых координат и потому обязаны быть количественными. Для метрических моделей, таких как k ближайших соседей и K средних, количественные признаки необходимы, если используется евклидова метрика, однако метрику можно подобрать и так, что можно будет работать с категориальными признаками, положив расстояние равным 0 для равных значений и 1 – для неравных (\bowtie *расстояние Хэмминга*, определенное в разделе 8.1). Аналогично для порядковых признаков мы можем подсчитать количество значений между двумя значениями признака (если значения порядкового признака кодируются целыми числами, то это будет просто их разность). Это означает, что метрические модели могут применяться к признакам любого вида, если подходящим образом определить метрику. Похожие приемы можно использовать для обобщения метода опорных векторов и других ядерных методов на категориальные и порядковые признаки.

Структурированные признаки

Обычно молчаливо предполагается, что объект – это вектор значений признаков. Иначе говоря, пространство объектов – это декартово произведение d областей значений признаков: $\mathcal{X} = \mathcal{F}_1 \times \dots \times \mathcal{F}_d$. Это означает, что нет никакой другой информации об объекте, помимо содержащейся в значениях его признаков. Идентификация объекта его вектором значений признаков – пример того, что в информатике называют *абстракцией*, – результат отбрасывания несущественной информации. Примером абстракции может служить представление почтового сообщения вектором частот слов.

Однако иногда таких абстракций следует избегать и сохранять об объекте больше информации, чем можно запомнить в конечном векторе значений признаков. Например, мы могли бы представить почтовое сообщение в виде длинной строки. Или в виде последовательности слов и знаков препинания. Или в виде дерева, сохраняющего HTML-разметку. И так далее. Признаки, которые определены в таких структурированных пространствах объектов, называются *структурированными признаками*.

Пример 10.4 (структурированные признаки). Предположим, что почтовое сообщение представлено в виде последовательности слов. Это позволяет нам определить, помимо обычной частоты слов, еще целый ряд признаков, в том числе:

- \bowtie встречается ли в сообщении словосочетание «машинное обучение» – или вообще любой другой набор последовательных слов;
- \bowtie содержит ли сообщение не менее восьми последовательных слов на любом языке, кроме английского;
- \bowtie является ли сообщение палиндромом, например: «А роза упала на лапу Азора».

Можно не ограничиваться свойствами одиночных сообщений, а выражать отношения между ними, например верно ли, что одно сообщение цитируется в другом или что в двух сообщениях есть один или более одинаковых фрагментов текста.

Структурированные признаки в некотором роде похожи на *запросы* на языке запросов к базе данных типа **SQL** или на декларативном языке программирования типа **Пролог**. На самом деле мы уже видели примеры структурированных признаков в разделе 6.4, когда занимались обучением предложений на **Прологе**, например:

```
fish(X) :-bodyPart(X,Y).  
fish(X) :-bodyPart(X,pairOf(Z)).
```

В первом предложении есть один структурированный признак в теле – там, где проверяется существование некоторой неуказанный части тела, а во втором есть еще один структурированный признак – проверяющий существование пары неуказанных частей тела. Определяющей характеристикой структурированных признаков является наличие *локальных переменных*, которые ссылаются на объекты, отличные от текущего. В логическом языке типа **Пролог** локальные переменные естественно интерпретируются как неявно предваряемые квантором существования, как мы только что и поступили. Однако ничто не мешает использовать и другие формы *агрегирования* локальных переменных, например мы можем подсчитать число частей тела (или пар частей тела) у объекта.

Структурированные признаки можно конструировать как до обучения модели, так и одновременно с этим. Первый случай называется *пропозиционализацией*, поскольку признаки можно рассматривать как трансляцию логики первого порядка в пропозициональную логику без локальных переменных. Основная проблема здесь – как справиться с комбинаторным ростом количества потенциальных признаков. Отметим, что признаки могут быть логически связаны, например второе из приведенных выше предложений покрывает подмножество объектов, покрытых первым. Этим фактом можно воспользоваться, если интегрировать конструирование структурированных признаков в построение модели, как в индуктивном логическом программировании.

10.2 Преобразования признаков

Цель *преобразования признаков* – повысить их полезность путем удаления, изменения или добавления информации. Можно упорядочить виды признаков по степени их детальности: количественные признаки детальнее порядковых, далее следуют категориальные признаки, и на последнем месте булевы. Наиболее известны те преобразования признаков, которые превращают признак одного типа в признак следующего за ним типа в этом списке. Но существуют также преоб-

разования, которые изменяют шкалу количественных признаков или добавляют шкалу (либо вводят порядок) для порядковых, категориальных или булевых признаков.

\downarrow в, из \rightarrow	Количественный	Порядковый	Категориальный	Булев
Количественный	нормализация	калибровка	калибровка	калибровка
Порядковый	дискретизация	упорядочение	упорядочение	упорядочение
Категориальный	дискретизация	разупорядочение	группировка	
Булев	задание порога	задание порога	бинаризация	

Таблица 10.2. Сводка возможных преобразований признаков. **Нормировка и калибровка** изменяют шкалу количественных признаков либо добавляют шкалу признакам, у которых ее нет. **Упорядочение** вводит или изменяет порядок на множестве значений признаков безотносительно к шкале. Прочие операции абстрагируют несущественные детали – либо дедуктивно (**разупорядочение**, **бинаризация**), либо путем введения новой информации (**задание порога**, **дискретизация**)

Простейшие преобразования признаков всецело дедуктивны в том смысле, что достигают четко определенного результата, не требуя производить какого-либо выбора. **Бинаризация** преобразует категориальный признак в набор булевых, по одному для каждого значения категориального признака. При этом теряется информация, потому что значения одного категориального признака являются взаимно исключающими, но иногда это необходимо, если модель не приспособлена к работе более чем с двумя значениями признака. **Разупорядочение** просто превращает порядковый признак в категориальный, отбрасывая существующий порядок на множестве значений. Часто без этого не обойтись, потому что большинство моделей обучения не могут работать с порядковыми признаками напрямую. Интересная альтернатива, которую мы рассмотрим ниже, – наделить признак шкалой с помощью калибровки.

Далее в этом разделе мы рассмотрим преобразования признаков, которые добавляют информацию. Самые важные из них – дискретизация и калибровка.

Задание порога и дискретизация

Задание порога преобразует количественный или порядковый признак в булев путем отыскания такого значения признака, по которому лучше всего произвести разделение. Я уже касался этого вопроса в главе 5, когда говорил о разделении по количественным признакам в решающих деревьях. Конкретно, пусть $f : \mathcal{X} \rightarrow \mathbb{R}$ – количественный признак и пусть $t \in \mathbb{R}$ – пороговое значение, тогда $f_t : \mathcal{X} \rightarrow \{\text{true}, \text{false}\}$ – булев признак, определенный следующим образом: $f_t(x) = \text{true}$, если $f(x) \geq t$, и $f_t(x) = \text{false}$, если $f(x) < t$. Пороговые значения можно выбирать путем обучения с учителем или без учителя.

Пример 10.5 (задание порога с учителем и без учителя). Снова рассмотрим признак «ВВП на душу населения», изображенный на рис. 10.1. Если мы не знаем, как этот признак будет использоваться в модели, то самыми разумными пороговыми значениями будут статистики центральной тенденции – среднее или медиана. Такой подход называется *заданием порога без учителя*.

При *обучении с учителем* можно добиться большего. Предположим, например, что мы собираемся использовать ВВП на душу населения как признак в решающем дереве для предсказания того, входит ли страна в число тех 23, что используют в качестве официальной валюты (или одной из валют) евро. Используя этот признак как ранжировщик, мы можем построить кривую покрытия (рис. 10.3 слева). Мы видим, что для этого признака среднее значение – не самый очевидный порог, потому что разделение по нему оказывается прямо посередине серии отрицательных объектов. Лучше было бы произвести разделение в начале этой серии отрицательных объектов или в конце следующей за ней серии положительных объектов, эти позиции обозначены красными точками по обе стороны от разделения по среднему. Более общо, любая точка на границе выпуклой оболочки кривой покрытия является кандидатом на пороговое значение; какую выбрать, зависит от того, что для нас важнее: объекты положительного или отрицательного класса. В данном примере получилось так, что медиана лежит на границе выпуклой оболочки, но, вообще говоря, полагаться на это нельзя, потому что, по определению, методы задания порога без учителя выбирают пороговое значение независимо от цели.

На рис. 10.3 справа показан тот же признак с другой целью: требуется узнать, находится ли страна в одной из Америк. Мы видим, что часть кривой расположена под восходящей диагональю, а это означает, что по сравнению с полным набором данных доля американских стран на начальном отрезке ранжирования невелика. Следовательно, потенциально полезные пороги можно найти также и на *нижней выпуклой оболочке*.

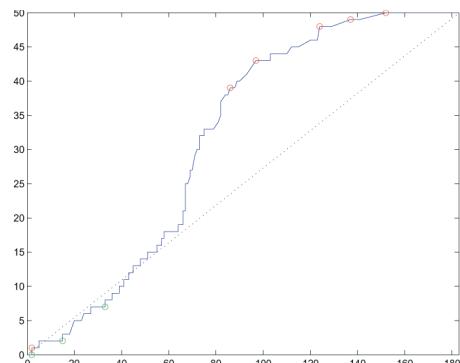
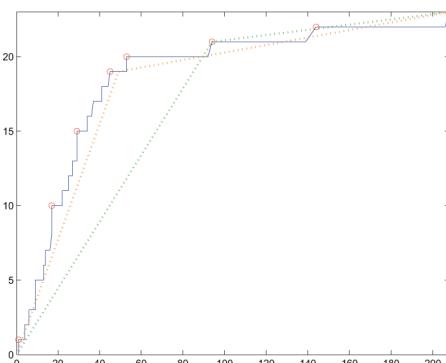


Рис. 10.3. (Слева) Кривая покрытия, полученная путем ранжирования стран по убыванию ВВП на душу населения; в роли положительного класса выступают 23 страны в зоне евро. **Оранжевое** разделение задает порог, равный среднему, отбирающий 19 стран в зоне евро и 49 стран вне этой зоны. **Зеленое** разделение задает порог, равный медиане; он отбирает 21 страну в зоне евро и 94 страны вне этой зоны. **Красные** точки лежат на границе выпуклой оболочки кривой покрытия и обозначают потенциально оптимальные разделения, когда учитывается метка класса. **(Справа)** Кривая покрытия того же признака, но в качестве положительного класса взяты 50 стран в обеих Америках. **Красные** разделения обозначают потенциально оптимальные пороги, при задании которых выше порога оказывается относительно много положительных объектов, **зеленые** – потенциально оптимальные пороги, при задании которых относительно много положительных оказывается объектов ниже порога

Резюмируем: задание порога без учителя обычно подразумевает вычисление некоторой статистики данных, тогда как задание порога с учителем требует сортировки данных по значению признака и обхода их в этом порядке для оптимизации некоторой целевой функции, например прироста информации. Неоптимальные точки разделения можно отфильтровать путем построения верхней и нижней выпуклой оболочки, но на практике это вряд ли окажется вычислитель-но более эффективным, чем прямой перебор отсортированных объектов.

Если обобщить задачу, допустив несколько пороговых значений, то мы при-дем к одному из наиболее распространенных недедуктивных преобразований признаков. *Дискретизация* преобразует количественный признак в порядковый. Каждое значение порядкового признака соответствует некоторому интервалу значений исходного количественного. И снова можно провести различие между обучением с учителем и без. В методах дискретизации без учителя обычно требуется выбрать количество интервалов заранее. Простой способ, который зачастую дает хорошие результаты, – выбирать интервалы так, чтобы в каждом было при-мерно одинаковое число объектов; это называется *дискретизацией с равной час-тотой*. Если интервалов всего два, то это то же самое, что выбор медианы в ка-честве порога. В общем случае границы интервалов – квантили, например при 10 интервалах границами интервалов равной частоты будут децили. Еще один метод дискретизации без учителя – *дискретизация с равной шириной*, когда гра-ница интервалов выбираются так, чтобы все интервалы имели равную ширину. Для задания ширины интервала можно поделить область определения признака на количество интервалов, если эта область ограничена сверху и снизу. Можно вместо этого установить границы в точках, отстоящих от среднего на целое число стандартных отклонений влево и вправо. Интересная альтернатива – рассмотря-вать дискретизацию признака как одномерную задачу кластеризации. Например, для генерации K интервалов мы можем произвести равномерную выборку K началь-ных центров и выполнить алгоритм K средних, пока он не сойдется. Или исполь-зовать любой другой алгоритм кластеризации из числа обсуждавшихся в главе 8: K медоидов, разбиение по медоидам или иерархическую агломератив-ную кластеризацию.

Переходя к методам *дискретизации с учителем*, мы можем выделить *нисходя-щие*, или *дивизивные*, методы, с одной стороны, и *восходящие*, или *агломератив-ные*, методы – с другой. Принцип работы дивизивных методов – постепенное разбиение интервалов, а в случае агломеративных методов мы сначала помещаем каждый объект в отдельный интервал, а затем постепенно объединяем их. В лю-бом случае важную роль играет *критерий остановки*, который определяет, имеет ли смысл продолжать разбиение или объединение. Мы приведем примеры обеих стратегий. Естественное обобщение задания порога приводит к нисходящему ал-горитму *рекурсивного разбиения* (алгоритм 10.1). Этот алгоритм дискретизации находит наилучший порог в соответствии с некоторой оценочной функцией Q и рекурсивно продолжает разбивать левый и правый интервалы. В качестве оце-ночной функции часто используется прирост информации.

Пример 10.6 (рекурсивное разбиение с использованием прироста информации). Рассмотрим следующие значения признака, для удобства расположенные в порядке возрастания.

Объект	Значение	Класс
e_1	-5.0	⊖
e_2	-3.1	⊕
e_3	-2.7	⊖
e_4	0.0	⊖
e_5	7.0	⊖
e_6	7.1	⊕
e_7	8.5	⊕
e_8	9.0	⊖
e_9	9.0	⊕
e_{10}	13.7	⊖
e_{11}	15.1	⊖
e_{12}	20.1	⊖

Этот признак порождает следующее ранжирование: $\ominus\oplus\ominus\ominus\oplus\oplus[\ominus\oplus]\ominus\ominus$, где квадратные скобки обозначают неопределенность, существовавшую между объектами e_8 и e_9 . Соответствующая кривая покрытия показана на рис. 10.4. Прослеживая изолинии прироста информации при каждом возможном разбиении, мы видим, что наилучшим является разбиение $\ominus\oplus\ominus\ominus\oplus\oplus[\ominus\oplus]\|\ominus\ominus$. Еще одна итерация этого процесса дает дискретизацию $\ominus\oplus\ominus\ominus\|\oplus\oplus[\ominus\oplus]\|\ominus\ominus$.

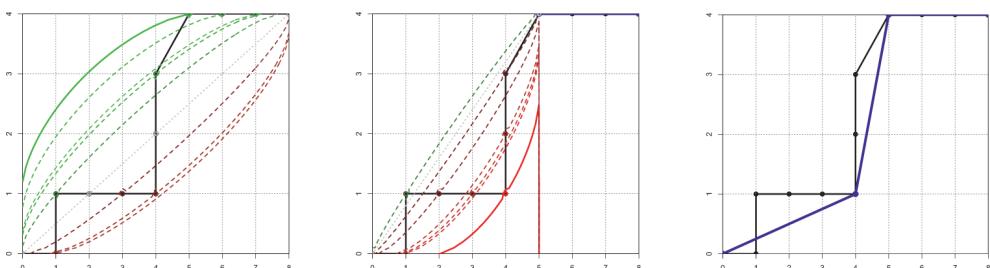


Рис. 10.4. (Слева) Кривая покрытия наглядно показывает ранжирование четырех положительных и восьми отрицательных примеров по дискретизируемому признаку. Кривые линии – это изолинии прироста информации при различных возможных точках разбиения; сплошная изолиния соответствует наилучшему разбиению $[4+, 5-][0+, 3-]$ по критерию прироста информации. **(В центре)** Рекурсивное разбиение продолжается, и отрезок $[4+, 5-]$ разбивается на $[1+, 4-][3+, 1-]$. **(Справа)** Если остановиться в этом месте, то **синяя** линия показывает дискретизированный (но все еще порядковый) признак

Очевидно, что мы можем остановить алгоритм рекурсивного разбиения, когда эмпирические вероятности окажутся одинаковы для всего ранжирования; это дает чистые интервалы и интервалы с постоянным значением признака в качестве

Алгоритм 10.1. $\text{RecPart}(S, f, Q)$ – дискретизация с учителем методом рекурсивного разбиения

Вход: множество помеченных объектов S , ранжированное по значениям признака $f(x)$; оценочная функция Q .

Выход: последовательность пороговых значений t_1, \dots, t_{k-1} .

- 1 **if** применим критерий остановки **then return** \emptyset ;
 - 2 Разбить S на S_l и S_r по пороговому значению t , оптимизирующему Q ;
 - 3 $T_l = \text{RecPart}(S_l, f, Q)$;
 - 4 $T_r = \text{RecPart}(S_r, f, Q)$;
 - 5 **return** $T_l \cup \{t\} \cup T_r$;
-

все частных случаев. При таком критерии остановки алгоритм успешно выявит все прямые отрезки в ранжировании. На самом деле нетрудно видеть, что это справедливо и при изменении оценочной функции – точки разбиения, возможно, будут найдены в другом порядке, но конечный результат не изменится. На практике применяются более агрессивные критерии остановки, и тогда конечный результат все же зависит от оценочной функции. Так, на рис. 10.4 мы видим, что у разбиения $\ominus|\oplus\ominus|\oplus\oplus\ominus\ominus\oplus\oplus$ второй по величине прирост информации, но в итоге оно вообще не выбирается, тогда как при другой оценочной функции оно могло бы быть выбрано в первом раунде. Один из самых популярных критериев остановки применяет рассуждение на основе минимального описания длины для решения о том, нужно ли дальше разбивать данный интервал.

Следует отметить, что набор данных в примере 10.6, по-видимому, настолько мал, что критерий остановки сработает сразу и алгоритм рекурсивного разбиения не продвинется дальше одного интервала. В общем случае такой способ дискретизации ведет себя довольно консервативно. Например, для набора данных о зоне евро на рис. 10.3 слева рекурсивное разбиение порождает два интервала, отбирая 20 стран в зоне евро и 53 страны вне этой зоны (**красная точка** между разделением по **среднему** и по **медиане**). Для данных об американских странах на рис. 10.3 справа мы снова получаем два интервала, соответствующие третьей справа **красной точке**.

Алгоритм 10.2 выполняет восходящее **агломеративное объединение**. Он также может делать различный выбор в зависимости от оценочной функции и критерия остановки; часто на обе роли берут статистику χ^2 .

Пример 10.7 (агломеративное объединение по критерию χ^2). Продолжим пример 10.6. Алгоритм 10.2 инициализирует интервалы следующим образом: $\ominus|\oplus|\ominus\ominus|\oplus\oplus|[\ominus\oplus]|\ominus\ominus\ominus$. Вычисление статистики χ^2 мы проиллюстрируем на примере последних двух интервалов. Строим следующую таблицу сопряженности:

	Левый интервал	Правый интервал	
\oplus	1	0	1
\ominus	1	3	4
	2	3	5

Алгоритм 10.2. *AggloMerge(S, f, Q)* – дискретизация с учителем методом агломеративного объединения

Вход: множество помеченных объектов S , ранжированное по значениям признака $f(x)$; оценочная функция Q .

Выход: последовательность пороговых значений.

- 1 инициализировать интервалы, поместив в каждый точки с одинаковыми оценками;
 - 2 объединить соседние чистые интервалы; // факультативная оптимизация
 - 3 **repeat**
 - 4 | вычислить Q для пар соседних интервалов;
 - 5 | объединить пары с наилучшими значениями Q (если на них не срабатывает критерий остановки);
 - 6 **until** дальнейшее объединение невозможно;
 - 7 **return** пороговые значения между интервалами;
-

В основе статистики χ^2 лежит сравнение наблюдаемых частот с ожидаемыми частотами, полученными на основе строчных и столбцовых маргиналов. Например, маргиналы говорят, что верхняя строка содержит 20% общей массы, а левый столбец – 40%; поэтому если бы строки и столбцы были статистически независимы, то следовало бы ожидать, что в левой верхней ячейке сосредоточится 8% массы, то есть 0.4 из пяти объектов. В направлении по часовой стрелке ожидаемые частоты для остальных ячеек составляют 0.6, 2.4 и 1.6. Если наблюдаемые частоты близки к ожидаемым, то эти два интервала можно было бы считать кандидатами на объединение, потому что разбиение в этой точке, похоже, не оказывает влияния на распределение по классам. Статистика χ^2 равна сумме квадратов разностей между наблюдаемыми и ожидаемыми частотами, в которой каждое слагаемое нормировано на ожидаемую частоту:

$$\chi^2 = \frac{(1-0.4)^2}{0.4} + \frac{(0-0.6)^2}{0.6} + \frac{(3-2.4)^2}{2.4} + \frac{(1-1.6)^2}{1.6} = 1.88.$$

Проходя слева направо по другим парам соседних интервалов, мы получаем такие значения χ^2 : 2, 4, 5, 1.33 (существует простой способ вычислить χ^2 для пары чистых интервалов, оставляю его нахождение в качестве упражнения). Это говорит о том, что четвертый и пятый интервалы следует объединить в первую очередь, что дает разбиение $\Theta \oplus \Theta \Theta \Theta \oplus \Theta \oplus \Theta \Theta \Theta$. Затем мы пересчитываем χ^2 (на самом деле пересчитать нужно только значения, включающие образовавшийся в результате объединения интервал), что дает 2, 4, 3.94, 3.94. Теперь мы объединяем первые два интервала, что дает разбиение $\Theta \oplus \Theta \Theta \Theta \oplus \Theta \Theta \Theta \oplus \Theta \Theta \Theta$. Тогда первое значение χ^2 становится равно 1.88, и мы снова объединяем первые два интервала, получая разбиение $\Theta \oplus \Theta \Theta \Theta \oplus \Theta \Theta \Theta \oplus \Theta \Theta \Theta$ (те же три интервала, что в примере 10.6).

В алгоритме агломеративной дискретизации критерий остановки обычно имеет вид простого порога для оценочной функции. В случае статистики χ^2 этот порог можно вывести из значения p , ассоциированного с распределением χ^2 , то есть с вероятностью наблюдения значения χ^2 , большего, чем порог, если обе величины действительно независимы. Для двух классов (то есть при одной степени свободы) при значении p , равном 0.10, порог χ^2 равен 2.71; в нашем примере это означает, что остановиться следует после построения трех показанных выше интервалов. При меньшем значении $p = 0.05$ порог χ^2 составляет 3.84, и тогда мы объединили бы все интервалы в один.

Отметим, что как восходящая, так и нисходящая дискретизация с учителем напоминают уже встречавшиеся ранее алгоритмы: рекурсивное разбиение по своей природе – разделяй и властвуй – близко к алгоритму обучения *решающего дерева* (алгоритм 5.1 на стр. 145), а агломеративная дискретизация посредством объединения соседних интервалов – к *иерархической агломеративной классификации* (алгоритм 8.4 на стр. 266). Стоит также подчеркнуть, что хотя наши примеры относились в основном к бинарной классификации, большинство методов безо всяких затруднений переносится на случай, когда классов больше двух.

Нормировка и калибровка

Задание порога и дискретизация – это преобразования, отбирающие шкалу у количественных признаков. Теперь мы обратимся к изменению шкалы количественного признака, а также к наделению шкалой порядкового или категориального признака. Когда эта операция совершается без учителя, она называется нормировкой, а когда с учителем, принимая во внимание метки классов (обычно двоичные), – калибровкой. *Нормировка признаков* часто необходима для нейтрализации эффекта применения различных шкал для измерения различных количественных признаков. Если признаки имеют распределение, близкое к нормальному, то мы можем преобразовать их в *z-оценки* (см. замечание 9.1 на стр. 278) путем центрирования относительно среднего и деления на стандартное отклонение. Как мы видели в разделе 7.1, в некоторых случаях математически удобнее делить вместо этого на дисперсию. Если предполагать нормальность нежелательно, то можно центрировать относительно медианы и делить на межквартильный размах.

Иногда нормировку признаков понимают в более строгом смысле, как приведение признака к шкале $[0,1]$. Этого можно добиться разными способами. Если нам известны наибольшее и наименьшее значения признака h и l , то можно просто применить линейное масштабирование $f \rightarrow (f - l)/(h - l)$. Иногда приходится строить гипотезы о значении h или l и обрезать значения, оказавшиеся вне отрезка $[l, h]$. Например, если признак – это возраст в годах, то можно взять $l = 0$ и $h = 100$, а любое значение $f > h$ заменять на 100. Если есть возможность сделать предположение о распределении признака, то можно подобрать такое преобразование, при котором почти все значения признака попадут в определенный диапазон. Например, мы знаем, что свыше 99% массы вероятности нормального распределения попадает в диапазон $\pm 3\sigma$ вокруг среднего, где σ – стандартное отклонение, поэтому линейное масштабирование $f \rightarrow (f - \mu)/6\sigma + 1/2$ практически устраняет необходимость в обрезании.

Под *калибровкой признака* понимают преобразование с учителем, наделяющее произвольный признак осмысленной шкалой, которая несет информацию о классе. У этого подхода есть ряд важных преимуществ. Например, он позволяет применять модели, требующие наличия шкалы, например линейные классификаторы, к категориальным и порядковым признакам. Он также позволяет алгоритму обучения выбирать, трактовать ли признак как категориальный, порядковый или

количественный. Мы будем предполагать контекст бинарной классификации, поэтому естественной шкалой для калиброванного признака будет апостериорная вероятность положительного класса при условии значения признака. Как мы увидим, у этого выбора есть то дополнительное преимущество, что модели, основанные на таких вероятностях, например наивный байесовский классификатор, не потребуют дополнительного обучения после калибровки признаков. Итак, задачу калибровки признака можно поставить следующим образом: пусть дан признак $F : \mathcal{X} \rightarrow \mathcal{F}$, требуется построить калиброванный признак $F^c : \mathcal{X} \rightarrow [0,1]$ – такой, что $F^c(x)$ является оценкой вероятности $F^c(x) = P(\oplus|v)$, где $v = F(x)$ – значение исходного признака для объекта x .

Для категориальных признаков это сводится просто к подсчету относительных частот в обучающем наборе.

Пример 10.8 (калибровка категориальных признаков). Пусть требуется предсказать, есть ли у пациента диабет, имея ряд категориальных признаков: наличие избыточного веса, курение и т. д. Собранная статистика говорит, что среди тучных людей диабетом страдает 1 из 18, а среди людей с нормальным весом – 1 из 55 (данные получены на сайте www.wolframalpha.com по запросу «diabetes»). Если $F(x) = 1$ для человека x , страдающего избыточным весом, и $F(y) = 0$ для остальных, то значения калиброванного признака будут равны $F^c(x) = 1/18 = 0.055$ и $F^c(y) = 1/55 = 0.018$.

На самом деле было бы лучше компенсировать неравномерность распределения по классам, чтобы избежать придания слишком большого значения априорному распределению, которое лучше учитывать в решающем правиле. Сделать это можно следующим образом. Если m из n тучных людей страдают диабетом, то соответствующий апостериорный шанс равен $m/(n-m)$, а функция правдоподобия $m/c(n-m)$, где c – априорный шанс наличия диабета (поскольку апостериорный шанс равен коэффициенту правдоподобия, умноженному на априорный шанс). Использование коэффициента правдоподобия эквивалентно предположению о равномерном распределении по классам. Преобразование коэффициента правдоподобия в вероятность дает:

$$F^c(x) = \frac{\frac{m}{c(n-m)}}{\frac{m}{c(n-m)} + 1} = \frac{m}{m + c(n-m)}.$$

В нашем примере если априорный шанс наличия диабета $c = 1/48$, то $F^c(x) = 1/(1 + 17/48) = 48/(48 + 17) = 0.74$. Превышение этой величины над $1/2$ количественно выражает то, насколько у тучного человека вероятность заболеть диабетом больше, по сравнению со средней. Для людей с нормальным весом вероятность равна $1/(1 + 54/48) = 48/(48 + 54) = 0.47$, так что у них вероятность заболеть диабетом чуть ниже средней. Не забывайте еще и о том, что эти оценки

вероятностей имеет смысл сглаживать с помощью поправки Лапласа, прибавляя 1 к m и 2 к n . Таким образом, мы получаем окончательное выражение для калибровки категориального признака:

$$F^c(x) = \frac{m+1}{m+1+c(n-m+1)}.$$

Порядковые и количественные признаки можно дискретизировать, а затем откалибровать как категориальные. Далее в этом разделе мы рассмотрим методы калибровки, сохраняющие порядок на множестве значений признака. Предположим, к примеру, что мы хотим использовать вес тела как признак диабета. Признак «калибранный вес» связывает с весами вероятности, не убывающие вместе с весом. Это имеет прямое отношение к обсуждению [«калиброванных оценок классификатора»](#) в разделе 7.4, поскольку калиброванные вероятности точно так же должны учитывать ранжирование предсказаний классификатора. Фактически оба подхода к калибровке классификатора – с помощью логистической функции и путем построения выпуклой оболочки РХП – непосредственно применимы и к калибровке признаков, поскольку количественный признак можно рассматривать как одномерный оценивающий классификатор.

Кратко повторим основные положения [логистической калибровки](#), но будем использовать немного другие обозначения. Пусть $F : \mathcal{X} \rightarrow \mathbb{R}$ – количественный признак со средними значениями классов μ^\oplus и μ^\ominus и дисперсией σ^2 . В предположении, что в каждом классе признак имеет нормальное распределение с одинаковой дисперсией, мы можем выразить коэффициент правдоподобия значения признака v в виде:

$$\begin{aligned} LR(v) &= \frac{P(v|\oplus)}{P(v|\ominus)} = \exp\left(\frac{-(v-\mu^\oplus)^2 + (v-\mu^\ominus)^2}{2\sigma^2}\right) = \\ &= \exp\left(\frac{\mu^\oplus - \mu^\ominus}{\sigma} \frac{v - (\mu^\oplus + \mu^\ominus)/2}{\sigma}\right) = \exp(d'z), \end{aligned}$$

где $d' = (\mu^\oplus - \mu^\ominus)/\sigma$ – разность средних, поделенная на стандартное отклонение (в теории обнаружения сигналов эту величину называют [d-штирих](#)), а $z = (v - \mu)/\sigma$ – z -оценка, ассоциированная с v (отметим, что мы берем среднее $\mu = (\mu^\oplus + \mu^\ominus)/2$, чтобы имитировать одинаковое распределение в классах). И снова мы напрямую работаем с коэффициентом правдоподобия, чтобы нейтрализовать эффект неоднородного распределения по классам, и получаем калиброванное значение признака в виде:

$$F^c(x) = \frac{LR(F(x))}{1+LR(F(x))} = \frac{\exp(d'z(x))}{1+\exp(d'z(x))}.$$

Наверное, вы узнаете логистическую функцию, которую мы обсуждали в главе 7 (см. рис. 7.11 на стр. 234).

По существу, логистическая калибровка признака состоит из следующих шагов:

1. Оценить средние значения классов μ^{\oplus} и μ^{\ominus} и стандартное отклонение σ .
2. Преобразовать $F(x)$ в z -оценки $z(x)$, не забывая, что среднее значение признака должно быть равно $\mu = (\mu^{\oplus} + \mu^{\ominus})/2$.
3. Изменить шкалу z -оценок на $F^d(x) = d'z(x)$, где $d' = (\mu^{\oplus} - \mu^{\ominus})/\sigma$.
4. Применить сигмоидное преобразование к $F^d(x)$ для получения калиброванных вероятностей:

$$F^c(x) = \frac{\exp(F^d(x))}{1 + \exp(F^d(x))}.$$

Иногда предпочтительнее работать непосредственно с функцией $F^d(x)$, поскольку она выражена в шкале, линейно связанной со шкалой исходного признака, а из предположения о гауссовом распределении следует, что мы ожидаем от этой шкалы аддитивности. Например, в метрических моделях при вычислении евклидова расстояния мы ожидаем, что признаки аддитивны. Напротив, шкала F^c мультипликативна. Отметим, что каждую шкалу можно определить через другую: $F^d(x) = \ln \frac{F^c(x)}{1 - F^c(x)} = \ln F^c(x) - \ln(1 - F^c(x))$. Я буду называть пространство признаков со шкалой F^d **пространством логарифмических шансов**, потому что $\exp(F^d(x)) = LR(x)$, а отношение правдоподобия равно шансу, если предположить равномерность априорного распределения по классам. Калиброванные признаки F^c обитают в **вероятностном пространстве**.

Пример 10.9 (логистическая калибровка двух признаков). Логистическая калибровка признаков показана на рис. 10.5. Я сгенерировал два набора данных по 50 точек в каждом, выбрав их из двумерных гауссиан с тождественной ковариационной матрицей с центрами в точках (2,2) и (4,4). Затем я построил базовый линейный классификатор и две прямые, параллельные решающей границе и проходящие через средние точки классов. Прослеживание трех этих прямых в калиброванном пространстве помогает лучше понять суть калибровки признаков.

На среднем рисунке мы видим преобразованные данные в пространстве логарифмических шансов; очевидно, что это просто результат изменения масштаба на осях. Базовый линейный классификатор теперь является прямой $F_1^d(x) + F_2^d(x) = 0$, проходящей через начало координат. Иначе говоря, для этого простого классификатора калибровка признаков устранила необходимость в дальнейшем обучении: вместо подгонки решающей границы к данным мы подогнали данные к фиксированной решающей границе! (Должен добавить, что я тут немного смешал, поскольку зафиксировал $\sigma = 1$ в процессе калибровки – если бы я оценивал стандартное отклонение каждого признака по данным, то решающая граница, скорее всего, имела бы чуть отличающийся угловой коэффициент.)

Справа мы видим преобразованные данные в вероятностном пространстве, и тут налицо нелинейное соотношение с двумя другими пространствами признаков. В этом пространстве базовый линейный классификатор остается линейным, но это уже неверно, если признаков больше двух. Чтобы убедиться в этом, заметим, что уравнение $F_1^c(x) + F_2^c(x) = 1$ можно переписать в виде

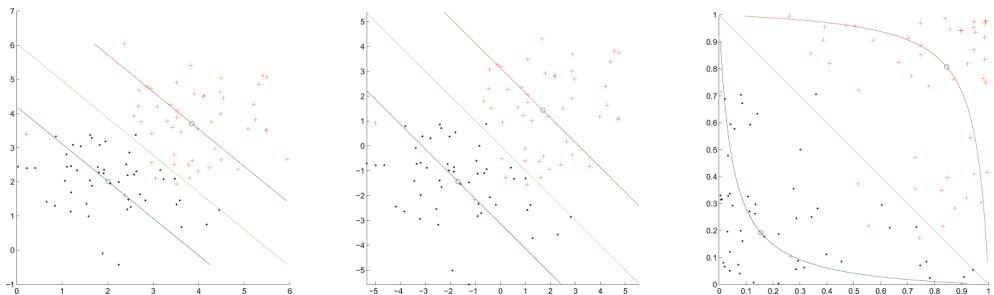


Рис. 10.5. (Слева) Данные с двумя классами, имеющими гауссово распределение. Средняя прямая – решающая граница, найденная в ходе обучения базового линейного классификатора; две другие параллельные ей прямые проходят через средние точки классов. **(В центре)** Логистическая калибровка с переходом в пространство логарифмических шансов – линейное преобразование; в предположении, что стандартные отклонения равны 1, базовым линейным классификатором теперь является фиксированная прямая $F_1^d(x) + F_2^d(x) = 0$. **(Справа)** Логистическая калибровка с переходом в вероятностное пространство – нелинейное преобразование, которое отодвигает данные от решающей границы

$$\frac{\exp(F_1^d(x))}{1+\exp(F_1^d(x))} + \frac{\exp(F_2^d(x))}{1+\exp(F_2^d(x))} = 1,$$

допускающем упрощение до $\exp(F_1^d(x))\exp(F_2^d(x)) = 1$, а отсюда следует, что $F_1^d(x) + F_2^d(x) = 0$. Однако если добавить третий признак, то не все члены сокращаются, и мы получаем нелинейную границу.

Представление в пространстве логарифмических шансов интересно и в другом отношении. Произвольная линейная решающая граница в этом пространстве описывается уравнением $\sum_i w_i F_i^d(x) = t$. После потенцирования получаем:

$$\begin{aligned} \exp\left(\sum_i w_i F_i^d(x)\right) &= \prod_i \exp(w_i F_i^d(x)) = \prod_i (\exp(F_i^d(x)))^{w_i} = \\ &= \prod_i LR_i(x)^{w_i} = \exp(t) = t'. \end{aligned}$$

Это обнажает связь с наивными байесовскими моделями, рассмотренными в разделе 9.2, в которых решающие границы также определяются как произведения отношений правдоподобия для отдельных признаков. В базовой наивной байесовской модели $w_i = 1$ для всех i и $t = 1$, а это означает, что *подгонку данных к фиксированной линейной решающей границе в пространстве логарифмических шансов с помощью калибровки признаков можно интерпретировать как обучение наивной байесовской модели*. Изменение углового коэффициента решающей границы соответствует введению отличных от единицы весов признаков, что аналогично причинам появления весов в мультиномиальной наивной байесовской мо-

дели. Поучительно исследовать распределение калиброванного признака более пристально (технические детали я опускаю). Если предположить, что до калибровки мы имели два гауссовых распределения, то как будут выглядеть калиброванные распределения? Мы уже видели, что калиброванные точки отодвигаются от решающей границы, поэтому следовало бы ожидать, что пики калиброванных распределений будут находиться ближе к краям. Насколько ближе, зависит исключительно от d' ; на рис. 10.6 показаны калиброванные распределения для различных значений d' .

Перейдем к *изотонной калибровке* – методу, для которого необходимо отношение порядка, но не нужна шкала и который применим как к порядковым, так и к количественным признакам. По существу, мы используем признак как одномерный ранжировщик: строим его кривую РХП и выпуклую оболочку для получения кусочно-постоянного калибрующего отображения. Предположим, что имеется кривая РХП и что ее i -й отрезок включает n_i обучающих примеров, из которых m_i положительны. Соответствующий отрезок РХП имеет угловой коэффициент $l_i = m_i / (c(n_i - m_i))$, где c – априорный шанс. Предположим, что кривая РХП выпукла, то есть из $i < j$ следует $l_i \geq l_j$. В таком случае мы можем использовать для получения калиброванного значения признака ту же формулу, что для категориальных признаков:

$$v_i^c = \frac{m_i + 1}{m_i + 1 + c(n_i - m_i + 1)}. \quad (10.1)$$

Как и раньше, при этом достигается как сглаживание вероятностей за счет применения поправки Лапласа, так и компенсация неравномерного распределения по классам. Если кривая РХП не выпукла, что существуют такие $i < j$, что $l_i < l_j$. В предположении, что желательно сохранить исходный порядок значений признака, мы сначала строим выпуклую оболочку кривой РХП. В результате соседние отрезки кривой РХП, являющиеся частью вогнутости, объединяются до тех пор, пока вогнутостей не останется. Затем мы пересчитываем отрезки и присваиваем признаку калиброванные значения, как в уравнении (10.1).

Пример 10.10 (изотонная калибровка признака). В таблице ниже приведены выборочные значения признака «вес» в контексте задачи о классификации диабета. На рис. 10.7 показаны кривая РХП и выпуклая оболочка признака, а также калибрующее отображение, полученное с помощью изотонной калибровки.

Вес	Диабет?	Калиброванный вес	Вес	Диабет?	Калиброванный вес
130	⊕	0.83	81	⊖	0.43
127	⊕	0.83	80	⊕	0.43
111	⊕	0.83	79	⊖	0.43
106	⊕	0.83	77	⊕	0.43
103	⊖	0.60	73	⊖	0.40

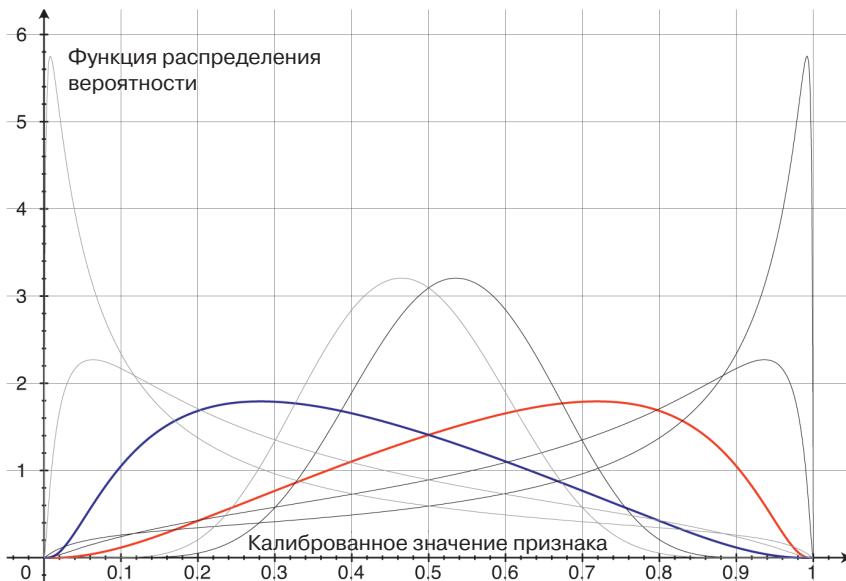


Рис. 10.6. Распределения в каждом классе логистически калиброванного признака для различных значений d' – расстояния между некалиброванными средними классов, поделенного на стандартное отклонение. **Красная** и **синяя** кривые изображают распределения для **положительного** и **отрицательного** классов признака, средние значения которого отстоят друг от друга на одно стандартное отклонение ($d' = 1$). Остальные кривые соответствуют значениям $d' \in \{0.5, 1.4, 1.8\}$

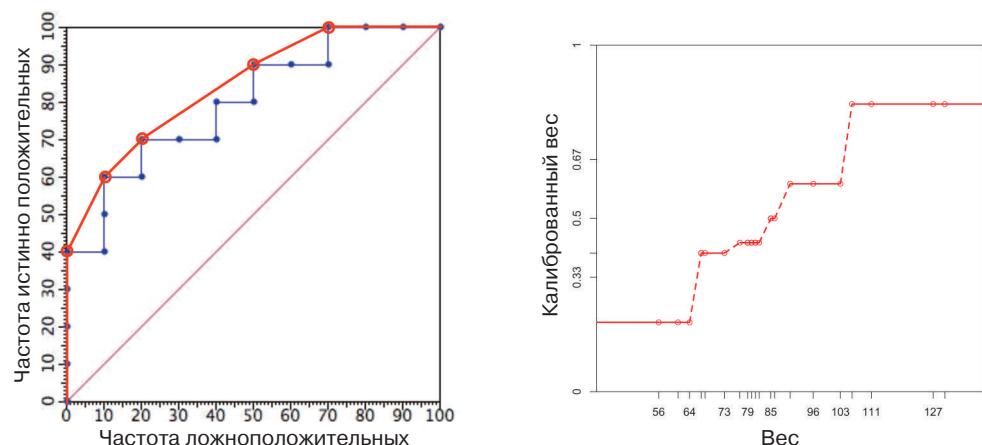


Рис. 10.7. (Слева) Кривая РХП и выпуклая оболочка некалиброванного признака. Значения калиброванного признака получаются на основе доли положительных объектов в каждом отрезке выпуклой оболочки РХП. **(Справа)** Соответствующее кусочно-постоянное калибрующее отображение, которое сопоставляет некалиброванным значениям признака на оси x калиброванные значения на оси y

Вес	Диабет?	Калиброванный вес	Вес	Диабет?	Калиброванный вес
96	⊕	0.60	68	⊖	0.40
90	⊕	0.60	67	⊕	0.40
86	⊖	0.50	64	⊖	0.20
85	⊕	0.50	61	⊖	0.20
82	⊕	0.43	56	⊖	0.20

Например, вес 80 кг после калибровки принимает значение 0.43, то есть три из семи людей в этом интервале весов страдают диабетом (после применения поправки Лапласа).

В примере 10.11 приведена иллюстрация для двумерного случая. Ясно видно, что для количественных признаков процедура сводится к дискретизации значений признаков с учителем, а это значит, что много точек отображается в одну и ту же точку откалиброванного пространства. Этим она отличается от логистической калибровки, которая обратима.

Пример 10.11 (изотонная калибровка двух признаков). На рис. 10.8 показан результат изотонной калибровки тех же данных, что в примере 10.9, – как в пространстве логарифмических шансов, так и в вероятностном пространстве. Из-за дискретной природы изотонной калибровки даже преобразование в пространство логарифмических шансов больше не является линейным: базовый линейный классификатор превращается в ряд отрезков, параллельных оси. Это верно и в противоположном направлении: если вообразить линейную решающую границу в пространстве логарифмических шансов или в вероятностном пространстве, то в исходном пространстве признаков ей будет соответствовать решающая граница, проходящая по пунктирным линиям. По сути дела, изотонная калибровка признаков превратила линейную ранжирующую модель в группирующую.

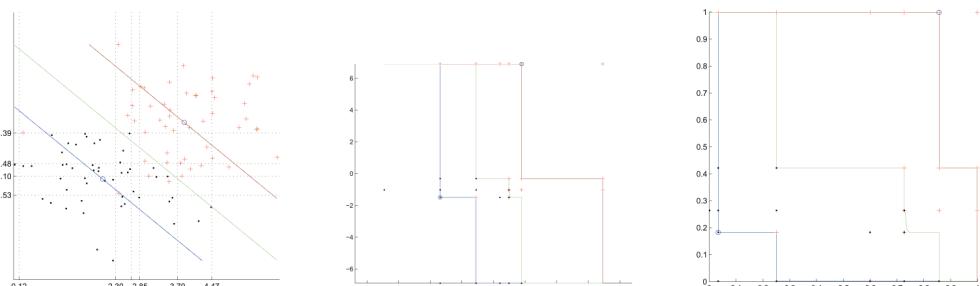


Рис. 10.8. (Слева) Те же данные, что на рис. 10.5, линии сетки обозначают дискретизацию, полученную путем изотонной калибровки признака. **(В центре)** Данные в пространстве логарифмических шансов после изотонной калибровки. **(Справа)** Данные в вероятностном пространстве после изотонной калибровки

Короче говоря, изотонная калибровка признака состоит из следующих шагов.

1. Отсортировать обучающие примеры по значению признака и построить кривую РХП. Порядок сортировки выбирается так, чтобы площадь под кривой РХП была $\geq 1/2$.
 2. Построить выпуклую оболочку этой кривой и подсчитать количество положительных примеров m_i и общее количество примеров n_i в каждом отрезке выпуклой оболочки.
 3. Дискретизировать признак в соответствии с отрезками выпуклой оболочки и сопоставить каждому отрезку калиброванное значение признака
- $$v_i^c = \frac{m_i + 1}{m_i + 1 + c(n_i - m_i + 1)}.$$
4. Если необходима аддитивная шкала, использовать значения $v_i^d = \ln \frac{v_i^c}{1 - v_i^c} = \ln v_i^c - \ln(1 - v_i^c)$.

Неполные признаки

В завершение этого раздела о преобразовании признаков вкратце поговорим о том, что делать, когда для некоторых объектов значение признака неизвестно. Мы сталкивались с такой ситуацией в примере 1.2 на стр. 38, когда обсуждали, как классифицировать почтовое сообщение, если мы не знаем, содержит оно какое-нибудь из словарных слов или нет. В вероятностных моделях этот вопрос решается довольно изящно – путем взвешенного усреднения по всем возможным значениям признака:

$$P(Y | X) = \sum_z P(Y, Z = z | X) = \sum_z P(Y | X, Z = z)P(Z = z).$$

Здесь Y – как обычно, целевая переменная, X обозначает признаки, наблюдавшиеся для классифицируемого объекта, а Z – признаки, не наблюдавшиеся во время классификации. Распределение $P(Z)$ можно получить из обученной модели, по крайней мере если модель порождающая; если модель дискриминантная, то оценивать $P(Z)$ придется отдельно.

Обработка отсутствия значений у признаков на этапе обучения сложнее. Прежде всего сам факт отсутствия значения может быть коррелирован с целевой переменной. Например, набор анализов, назначаемых пациенту, скорее всего, зависит от истории болезни. Для таких признаков, возможно, наилучшим выходом будет завести специальное значение «отсутствует», чтобы по нему можно было, к примеру, проводить разделение в древовидной модели. Однако для линейной модели это не годится. В таких случаях мы можем пополнить признак, «внедрив» отсутствующие значения, этот процесс называется *подстановкой* (imputation). Например, в задаче классификации мы можем вычислить средние, медианы или моды в каждом классе по наблюдавшимся значениям признака и использовать их для подстановки отсутствующих значений. Несколько более сложный метод

позволяет учесть корреляцию признаков, построив прогностическую модель для каждого неполного признака и использовав ее для «предсказания» отсутствующего значения. Можно также воспользоваться [«EM-алгоритмом](#) (раздел 9.4), то есть поступить примерно так: предположив некоторую многомерную модель для всех признаков, использовать наблюдавшиеся значения для получения оценки максимального правдоподобия всех параметров модели, затем вывести математические ожидания ненаблюдавшихся значений и повторить.

10.3 Конструирование и отбор признаков

Из предыдущего раздела, посвященного преобразованию признаков, стало понятно, что в машинном обучении есть большой простор для манипуляций с исходными признаками, входящими в состав данных. Можно сделать следующий шаг – конструировать новые признаки из исходных. Простой вариант этого подхода можно использовать для улучшения [«наивного байесовского](#) классификатора, который обсуждался в разделе 9.2. Напомним, что в приложениях для классификации текстов имеется по одному признаку для каждого словарного слова, но эти признаки игнорируют не только порядок слов, но и их соседство. Это означает, что предложения «they write about machine learning» (они пишут о машинном обучении) и «they are learning to write about a machine» (они учатся писать о машине) фактически неразличимы, хотя первое относится к машинному обучению, а второе – нет. Поэтому иногда необходимо в словарь включать фразы, состоящие из нескольких слов, и рассматривать их как неделимые признаки. В литературе по информационному поиску такие словосочетания называются *n-граммами* (униграмма, биграмма, триграмма и т. д.).

Развивая эту идею, мы можем сконструировать новый признак из двух булевых или категориальных признаков, образовав декартово произведение. Например, если имеется один признак **Форма** со значениями **Окружность**, **Треугольник** и **Квадрат** и другой признак **Цвет** со значениями **Красный**, **Зеленый** и **Синий**, то их декартовым произведением будет признак (**Форма**, **Цвет**) со значениями (**Окружность, Красный**), (**Окружность, Зеленый**), (**Окружность, Синий**), (**Треугольник, Красный**) и т. д. Результат зависит от обучаемой модели. Конструирование признаков в виде декартова произведения для наивного байесовского классификатора означает, что два исходных признака более не рассматриваются как независимые, и это уменьшает сильное смещение, характерное для наивных байесовских моделей. Это не так для древовидных моделей, которые уже умеют различать все возможные пары значений признаков. С другой стороны, вновь введенный признак в виде декартова произведения может означать большой прирост информации и, возможно, окажет влияние на обученную модель.

Существует и много других способов комбинирования признаков. Например, мы можем взять арифметические или полиномиальные комбинации количественных признаков (примеры встречались при использовании [«ядра](#) в приме-

ре 1.9 на стр. 55 и в разделе 7.5). Интересная идея – сначала применить концептуальное обучение или выявление подгрупп, а потом использовать найденные концепты или подгруппы как новые булевы признаки. Например, в примере с дельфинами мы могли бы выявить такие подгруппы, как **Длина = [3,5] \wedge Жабры = нет**, и использовать их в качестве булевых признаков в последующей древовидной модели. Отметим, что тем самым мы увеличиваем выразительность древовидных моделей, допуская использование отрицания, например (**Длина = [3,5] \wedge Жабры = нет**) = **false** эквивалентно дизъюнкции (**Длина \neq [3,5] \vee Жабры = да**), которая не допускает непосредственного выражения в дереве признаков.

После того как новые признаки сконструированы, часто бывает полезно выбрать подходящее их подмножество до начала обучения. Это не только ускорит обучение, поскольку придется рассматривать меньше потенциальных признаков, но и поможет предотвратить переобучение. Существуют два основных подхода к отбору признаков. В случае **фильтрации** мы оцениваем признаки по какому-то показателю и отбираем только те, у которых оценка наивысшая. Для оценивания можно воспользоваться многими из уже встречавшихся нам показателей, в том числе приростом информации, статистикой χ^2 , коэффициентом корреляции и другими. Интересный вариант этой идеи – алгоритм отбора признаков **Relief**, в котором многократно производится выборка случайног объекта x и ищутся ближайшее к нему попадание h (пример из того же класса) и ближайший промах m (пример из противоположного класса). Затем оценка i -го признака уменьшается на $\text{Dis}(x_i, h_i)^2$ и увеличивается на $\text{Dis}(x_i, m_i)^2$, где Dis – некоторая метрика (например, евклидово расстояние для количественных признаков или расстояние Хэмминга для категориальных). Интуиция подсказывает, что наша цель – приблизиться к ближайшему попаданию и отдалиться от ближайшего промаха.

Одним из недостатков простой фильтрации является то, что в ней не учитывается избыточность, присутствующая в признаках. Допустим для определенности, что многообещающий признак в наборе данных дублируется: обе копии получат одинаково высокую оценку и будут отобраны, хотя вторая не дает ничего нового, по сравнению с первой. Другой недостаток заключается в том, что отбор признаков не обнаруживает зависимости между признаками, потому что они основаны исключительно на маргинальных распределениях. Например, рассмотрим два булевых признака – таких, что у половины положительных примеров оба они равны **true**, а у другой половины оба равны **false**, тогда как у отрицательных примеров значения противоположные (и, естественно, отрицательные примеры тоже разделены по этим признакам на две равные группы). Отсюда следует, что каждый признак в отдельности дает нулевой прирост информации и потому вряд ли будет отобран фильтром признаков, несмотря на то что в сочетании они образуют идеальный классификатор. Можно сказать, что фильтры признаков хороши для выбора потенциальных корней решающего дерева, но хуже для отбора полезных признаков на более низких уровнях дерева.

Для выявления признаков, которые могли бы быть полезны в контексте других признаков, мы должны оценивать множества признаков; обычно в таких

случаях используют обертки. Идея в том, что отбор признаков «обертывается» процедурой поиска, которая включает обучение и оценку модели с помощью потенциального множества признаков. В методах *прямого отбора* мы начинаем с пустого множества и добавляем в него по одному признаку при условии, что он улучшает качество модели. В случае *обратного исключения* мы начинаем с полного множества признаков и ставим целью улучшить качество, исключая признаки по одному. Поскольку число подмножеств множества признаков выражается экспоненциальной зависимостью, перебрать их все обычно не представляется возможным, и в большинстве подходов применяются «жадные» алгоритмы поиска, которые никогда не пересматривают ранее сделанного выбора.

Преобразование и разложение матриц

Конструирование и отбор количественных признаков можно также рассматривать с геометрической точки зрения. Для этого представим набор данных в виде матрицы \mathbf{X} , в которой n точкам соответствуют строки, а d признакам – столбцы. Наша цель – построить из нее новую матрицу \mathbf{W} с n строками и r столбцами, пользуясь различными операциями над матрицами. Чтобы немного упростить задачу, будем предполагать, что \mathbf{X} центрирована относительно нуля и что $\mathbf{W} = \mathbf{XT}$ для некоторой матрицы преобразования \mathbf{T} размерности $d \times r$. Например, изменению шкалы признаков соответствует диагональная матрица \mathbf{T} размерности $d \times d$; эту операцию можно скомбинировать с отбором признаков путем удаления некоторых столбцов \mathbf{T} . Поворот описывается ортогональной матрицей \mathbf{T} , для которой $\mathbf{TT}^T = \mathbf{I}$. Понятно, что несколько таких преобразований можно объединить (см. также замечание 1.2 на стр. 36).

Один из самых известных алгебраических методов конструирования признаков называется *методом главных компонент* (principal component analysis – *PCA*). Главные компоненты – это новые признаки, сконструированные в виде линейных комбинаций заданных признаков. Первая главная компонента определяется направлением максимального разброса данных; вторая – направлением максимального разброса данных, ортогональным первой компоненте, и т. д. Метод РСА можно обосновать по-разному, здесь мы выведем его с помощью *сингулярного разложения матрицы* (singular value decomposition – *SVD*). Любую матрицу $n \times d$ можно единственным образом представить в виде произведения трех матриц со специальными свойствами:

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T. \quad (10.2)$$

Здесь \mathbf{U} – матрица размерности $n \times r$, Σ – матрица размерности $r \times r$, а \mathbf{V} – матрица размерности $d \times r$ (временно предположим, что $r = d < n$). Кроме того, матрицы \mathbf{U} и \mathbf{V} ортогональны (то есть описывают повороты), а Σ диагональна (то есть соответствует масштабированию). Столбцы матриц \mathbf{U} и \mathbf{V} называются соответственно левыми и правыми сингулярными векторами, а значения на диагонали Σ – соответствующими им сингулярными значениями. Принято упорядочи-

вать столбцы \mathbf{V} и \mathbf{U} , так чтобы сингулярные значения убывали при перемещении по диагонали из левого верхнего в правый нижний угол.

Теперь рассмотрим матрицу $\mathbf{W} = \mathbf{U}\Sigma$ размерности $n \times r$ и заметим, что $\mathbf{X}\mathbf{V} = \mathbf{U}\Sigma\mathbf{V}^T\mathbf{V} = \mathbf{U}\Sigma = \mathbf{W}$ в силу ортогональности \mathbf{V} . Иными словами, мы можем по матрице \mathbf{X} построить матрицу \mathbf{W} с помощью преобразования \mathbf{V} : это и есть представление \mathbf{X} в терминах ее главных компонент. Новые сконструированные признаки находятся в матрице $\mathbf{U}\Sigma$: первая строка – первая главная компонента, вторая строка – вторая главная компонента и т. д. У этих главных компонент имеется геометрическая интерпретация: направление максимального разброса данных в \mathbf{X} , второго по величине разброса и т. д. В предположении, что данные центрированы относительно нуля, эти направления можно получить с помощью комбинации поворота и масштабирования, в этом и заключается смысл метода PCA.

Разложение SVD можно использовать и для того, чтобы переписать матрицу разброса в стандартной форме:

$$\mathbf{S} = \mathbf{X}^T\mathbf{X} = (\mathbf{U}\Sigma\mathbf{V}^T)^T(\mathbf{U}\Sigma\mathbf{V}^T) = (\mathbf{V}\Sigma\mathbf{U}^T)(\mathbf{U}\Sigma\mathbf{V}^T) = \mathbf{V}\Sigma^2\mathbf{V}^T.$$

Это так называемое *спектральное разложение* матрицы \mathbf{S} : столбцами \mathbf{V} являются собственные векторы \mathbf{S} , а элементы на диагонали матрицы Σ^2 , которая и сама является диагональной, – собственные значения. Правые сингулярные векторы матрицы данных \mathbf{X} – это собственные векторы матрицы разброса $\mathbf{S} = \mathbf{X}^T\mathbf{X}$, а сингулярные значения \mathbf{X} – квадратные корни из собственных значений \mathbf{S} . Аналогичное выражение можно вывести для матрицы Грама $\mathbf{G} = \mathbf{X}\mathbf{X}^T = \mathbf{U}\Sigma^2\mathbf{U}^T$, откуда видно, что собственные векторы матрицы Грама – это левые сингулярные векторы \mathbf{X} . Тем самым показано, что для выполнения анализа главных компонент достаточно найти спектральное разложение матрицы разброса или матрицы Грама, а не полное сингулярное разложение. Нечто похожее на SVD мы видели в разделе 1.1, когда рассматривали следующее произведение матриц:

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Матрица в левой части выражает предпочтения людей в части фильмов (по столбцам). Правая часть – это разложение, или факторизация данной матрицы по жанрам фильмов: первая матрица количественно выражает оценку жанров, последняя ассоциирует фильмы с жанрами, а средняя дает веса каждого жанра при определении предпочтений. Это разложение не совпадает с SVD, поскольку левый и правый сомножители не являются ортогональными матрицами. Однако

можно сказать, что такое разложение лучше отражает особенности данных, потому что матрицы соответствия людей и жанров и фильмов и жанров булевы и разреженные, а в SVD-разложении они такими не были бы. Недостаток же в том, что из-за наложения ограничения целочисленности или булевости задача разложения перестает быть выпуклой (существуют локальные оптимумы) и становится вычислительно более сложной. Разложение матриц с дополнительными ограничениями – область, в которой ведутся очень активные исследования.

Методы разложения матриц часто применяются для понижения размерности. *Рангом* матрицы размерности $n \times d$ называется число d (в предположении, что $d < n$ и ни один столбец не является линейной комбинацией других столбцов). Описанные выше разложения имеют полный ранг, потому что $r = d$, и, значит, матрица данных реконструируется точно. Аппроксимацией меньшего ранга называется такое разложение матрицы, в котором r – наименьшее возможное значение, обеспечивающее достаточную аппроксимацию исходной матрицы. Ошибка реконструкции обычно измеряется как сумма квадратов разностей между элементами \mathbf{X} и соответственными элементами $\mathbf{U}\Sigma\mathbf{V}^T$. Можно показать, что усеченное сингулярное разложение с $r < d$ дает наименьшую в этом смысле ошибку реконструкции из всех разложений ранга, не превосходящего r . Усеченное SVD и PCA – популярные методы конструирования и отбора количественных признаков.

Интересной особенностью таких разложений матриц, как SVD, является тот факт, что они вскрывают ранее скрытые в данных переменные. Это можно показать следующим образом. Рассмотрим разложение или аппроксимацию $\mathbf{U}\Sigma\mathbf{V}^T$, где матрица Σ диагональная, но матрицы \mathbf{U} и \mathbf{V} необязательно ортогональны, и обозначим i -й столбец \mathbf{U} и \mathbf{V} как $\mathbf{U}_{:,i}$ (n -мерный вектор) и $\mathbf{V}_{:,i}$ (d -мерный вектор) соответственно. Тогда $\mathbf{U}_{:,i}\sigma_i(\mathbf{V}_{:,i})^T$ – внешнее произведение, порождающее матрицу размерности $n \times d$ ранга 1 (σ_i обозначает i -й диагональный элемент Σ). Матрица ранга 1 характеризуется тем, что каждый ее столбец является произведением некоторого базисного вектора на скаляр (и то же самое верно для строк). В предположении, что \mathbf{U} и \mathbf{V} имеют ранг r , эти базисные векторы независимы, поэтому сумма полученных матриц ранга 1 по всем i дает исходную матрицу:

$$\mathbf{U}\Sigma\mathbf{V}^T = \sum_{i=1}^r \mathbf{U}_{:,i}\sigma_i(\mathbf{V}_{:,i})^T.$$

Например, матрицу рейтингования фильмов можно записать в таком виде:

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Матрицы в правой части можно интерпретировать как модели рейтингования, обусловленные жанром.

Выявление скрытых переменных – одно из основных приложений методов разложения матриц. Например, в информационном поиске метод РСА известен под названием *латентное семантическое индексирование (LSA)* (слово «латентное» – синоним слова «скрытое»). Но вместо жанров фильмов метод LSA выявляет темы документов путем разложения матриц, содержащих счетчики слов в документе, в предположении, что количество слов, относящихся к каждой теме, – независимая величина и, значит, их можно просто складывать¹. Другое важное применение разложения матриц – *восполнение* отсутствующих элементов матрицы; идея заключается в том, что наилучшая аппроксимация наблюдавшихся значений матрицы с помощью разложения меньшего ранга позволит вывести отсутствующие значения.

10.4 Признак: итоги и литература для дальнейшего чтения

В этой главе мы уделили внимание признакам, что давно уже пора было сделать. Признаки – это телескопы, в которые мы наблюдаем вселенную данных, и потому они играют важную унифицирующую роль в машинном обучении. Признаки связаны с измерениями в научных дисциплинах, но не существует общепринятого мнения о том, как формализовать и классифицировать различные измерения, – я основывался на шкалах измерения Стивенса (Stevens, 1946), но в остальном стремился не отдаляться от современной практики машинного обучения.

- ☞ В разделе 10.1 обсуждались основные виды признаков: категориальные, порядковые и количественные. Последние выражаются в некоторой количественной шкале и допускают вычисление самых разных статистик центральной тенденции (среднее значение, медиана, мода; см. обсуждение относящихся к ним эвристических правил в работе von Hippel, 2005), разброса (дисперсия и стандартное отклонение, размах, межквартильный размах) и формы (асимметрия и куртозис). В машинном обучении количественные признаки нередко называют непрерывными, но мне этот термин кажется неподходящим, поскольку подразумевает, что значение признака может быть определено с неограниченной точностью, а это неверно. Важно понимать, что шкала количественных признаков необязательно аддитивная, например количественный признак можно выразить в виде

¹ Возможны и другие модели, например в случае булева разложения матрицы матричное произведение заменяется булевым, при котором место целочисленного сложения занимает логическая дизъюнкция (то есть $1 + 1 = 1$). В результате дополнительные темы не повышают объяснительную способность вхождения слова в документ.

вероятности по мультипликативной шкале, при этом использовать для неаддитивных признаков, скажем, евклидову расстояние недопустимо. Для порядковых признаков определен порядок на множестве значений, но не шкала, а у категориальных (называемых также номинальными, или дискретными) признаков нет ни шкалы, ни порядка.

- ☞ Структурированные признаки – это предложения в логике первого порядка, в которых имеются ссылки на части объектов с помощью локальных переменных и применяется тот или иной вид агрегирования, например квантор существования или подсчет, для извлечения свойства главного объекта. Конструирование признаков первого порядка до обучения часто называют пропозиционализацией; в работах Kramer et al. (2000) и Lachiche (2010) имеются обзоры, а в работе Krogel et al. (2003) – результаты экспериментального сравнения различных подходов.
- ☞ В разделе 10.2 мы рассмотрели ряд преобразований признаков. Из них наиболее известны дискретизация и задание порога, позволяющие превратить количественный признак в категориальный или булев. Один из самых эффективных методов дискретизации – предложенный в работе Fayyad, Irani (1993) алгоритм рекурсивного разбиения, в котором для нахождения порогов используется прирост информации, а критерий остановки выводится из принципа минимального описания длины. Обзор других методов приведен в работах Boulle (2004, 2006). Подход на основе агломеративного объединения с использованием статистики χ^2 был предложен в работе Kerber (1992).
- ☞ Мы видели, что в случае двух классов дискретизацию можно наглядно представить с помощью кривых покрытия. Это естественно подводит к идею использования кривых покрытия и их выпуклых оболочек для калибровки, а не просто дискретизации признаков. В конце концов, порядковые и количественные признаки – это не что иное, как одномерные ранжировщики и оценивающие классификаторы, и потому к ним могут быть применены те же самые методы калибровки классификаторов, в частности логистическая и изотонная калибровка, обсуждавшиеся в разделе 7.4. Калиброванные признаки обитают в вероятностном пространстве, но иногда предпочтительнее пространство логарифмических шансов, поскольку оно аддитивное, а не мультипликативное. Подгонка данных к фиксированной решающей границе в калиброванном пространстве логарифмических шансов тесно связана с обучением наивной байесовской модели. Изотонная калибровка приводит к кусочно-постоянным решающим границам, состоящим из отрезков, параллельных оси; в силу дискретизирующей природы изотонной калибровки это можно интерпретировать как построение группирующей модели, пусть даже оригинальная модель в некалиброванном пространстве была ранжирующей.
- ☞ Раздел 10.3 был посвящен конструированию и отбору признаков. Ранние подходы к конструированию признаков и конструктивной индукции были

предложены в работах Ragavan, Rendell (1993); Donoho, Rendell (1995). Основанный на объектах метод отбора признаков Relief был предложен в работе Kira, Rendell (1992) и обобщен в работе Robnik-Sikonja, Kononenko (2003). Различие между подходами к отбору признаков на базе фильтров, оценивающих признаки по их индивидуальным достоинствам, и оберток, позволяющих оценивать целые множества признаков, впервые исследовалось в работе Kohavi, John (1997). В работе Hall (1999) предложен фильтрационный подход, названный корреляционным отбором признаков, цель которого – взять лучшее из обоих миров. Работа Guyon, Elisseeff (2003) – великолепное введение в проблематику отбора признаков.

- ☞ Наконец, мы познакомились с конструированием и отбором признаков с применением аппарата линейной алгебры. Разложение матриц – область активных исследований; эта техника легла в основу программы, завоевавшей приз в 1 млн долл. на недавно завершившемся конкурсе по рекомендованию фильмов (Koren et al., 2009). К числу методов разложения с дополнительными ограничениями относится неотрицательное матричное разложение (Lee et al., 1999). Булево матричное разложение изучалось в работе Miettinen (2009). В работе Mahoney, Drineas (2009) описывается метод матричного разложения, в котором истинные столбцы и строки матрицы данных используются для сохранения разреженности (в отличие от метода SVD, который порождает плотные матрицы даже в случае, когда исходная была разрежена). Латентное семантическое индексирование и вероятностное обобщения описаны в работе Hofmann (1999). В работе Ding, He (2004) обсуждается связь между кластеризацией методом K средних и анализом главных компонент.



Ансамбли моделей

Одна голова хорошо, а две лучше – согласно этой известной пословице, два человека, работающие вместе, зачастую добиваются лучших результатов. Если заменить «голова» на «признак», то эта пословица будет в полной мере применима и к машинному обучению, в чем мы имели возможность убедиться в предыдущих главах. Но можно улучшить результаты еще сильнее, если комбинировать не просто признаки, а целые модели – это мы и продемонстрируем в данной главе. Комбинации моделей часто называют *ансамблями моделей*. Это один из самых мощных методов машинного обучения, нередко превосходящий другие по качеству и эффективности. Правда, за это приходится расплачиваться увеличением сложности алгоритмов и моделей.

Тема комбинирования моделей имеет богатую и разнообразную историю, которой мы можем посвятить лишь немного времени в этой короткой главе. Своими корнями она уходит в вычислительную теорию обучения, с одной стороны, и в статистику – с другой. В статистике хорошо известно интуитивное соображение, согласно которому усреднение результатов измерений может дать более устойчивую и надежную оценку, поскольку сокращается влияние случайных флуктуаций в отдельном измерении. Поэтому если бы удалось построить ансамбль немного различающихся моделей по одним и тем же обучающим данным, то мы смогли бы уменьшить влияние случайных флуктуаций в отдельных моделях. Ключевой вопрос – как обеспечить необходимое разнообразие моделей? Как мы увидим, зачастую этого можно добиться путем обучения моделей на случайно выбранных подмножествах данных и даже путем их конструирования из случайно выбранных подмножеств имеющихся признаков.

Из теории вычислительного обучения заимствован следующий ход рассуждений. Как мы видели в разделе 4.4, обучаемость языков гипотез исследовалась в контексте модели обучения, которая и определяет, что понимать под обучаемостью. PAC-обучаемость предполагает, что гипотеза почти правильна в большинстве случаев. Альтернативная модель, известная под названием *слабая обучаемость*, требует только, чтобы обученная гипотеза была лишь немного лучше случайной. И хотя кажется очевидным, что PAC-обучаемость сильнее, чем слабая обучаемость, оказывается, что на самом деле обе модели обучения эквивалентны: язык гипотез PAC-обучаем тогда и только тогда, когда он слабо обучаем. Это было доказано конструктивно – с помощью итеративного алгоритма, который повторяет конструирование гипотезы, имеющей целью исправление ошибок предыдущей гипотезы, и тем самым «усиливает» ее. Окончательная

модель комбинирует гипотезы, обученные на каждой итерации, и потому определяет ансамбль.

По существу, у всех методов построения ансамблей в машинном обучении есть две общие черты:

- ☞ они конструируют несколько различающихся прогностических моделей из адаптированных вариантов обучающих данных (чаще всего с помощью изменения весов или повторной выборки);
- ☞ они каким-то образом комбинируют предсказания этих моделей, часто с помощью простого усреднения или голосования (возможно, взвешенного).

Однако необходимо подчеркнуть, что, несмотря на наличие этих общих черт, разнообразие применяемых методов очень велико, и не следует удивляться тому, что некоторые методы на практике сильно отличаются, пусть даже и обладают поверхностным сходством. Например, очень многое зависит от того, учитываются ли при адаптации обучающих данных для следующей итерации предсказания предыдущих моделей. Из всего разнообразия мы рассмотрим два наиболее известных метода построения ансамблей: баггинг (раздел 11.1) и усиление (раздел 11.2). Далее в разделе 11.3 мы кратко обсудим эти и родственные им методы построения ансамблей и завершим главу обычным подведением итогов и рекомендацией литературы для дальнейшего чтения.

11.1 Баггинг и случайные леса

Баггинг (сокращение от «bootstrap aggregating») – простой, но весьма эффективный метод создания различающихся моделей на основе различных случайных выборок из исходного набора данных. Выборки производятся равномерно с заменой и называются *усиливающими выборками* (bootstrap samples). Поскольку выборка производится с заменой, то в общем случае усиливающая выборка содержит дубликаты, и потому некоторые исходные примеры будут отсутствовать, даже если размер усиливающей выборки такой же, как размер исходного набора. Чтобы понять, насколько различными могут быть усиливающие выборки, заметим, что вероятность того, что конкретный пример не входит в усиливающую выборку размера n , равна $(1 - 1/n)^n$; при $n = 5$ она равна примерно $1/3$ и стремится к $1/e = 0.368$, когда $n \rightarrow \infty$. Это означает, что в каждой усиливающей выборке отсутствует примерно треть исходных данных.

В алгоритме 11.1 описан базовый алгоритм баггинга, который возвращает ансамбль в виде множества моделей. Для комбинирования предсказаний моделей мы можем воспользоваться голосованием – выигрывает класс, предсказанный большинством моделей, – или усреднением, которое подходит лучше, если базовые классификаторы возвращают оценки или вероятности. На рис. 11.1 приведена иллюстрация. Я обучил пять базовых линейных классификаторов на усиливающих выборках из набора, содержащего по 20 положительных и отри-

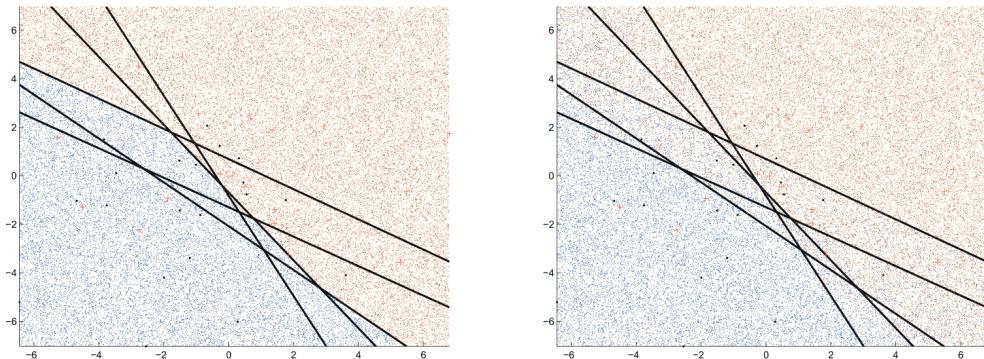


Рис. 11.1. (Слева) Ансамбль из пяти базовых линейных классификаторов, построенный из усиливающих выборок с помощью баггинга. Решающим правилом является большинство голосов, оно порождает кусочно-линейную решающую границу. **(Справа)** Если преобразовать голоса в вероятности, то ансамбль превращается в группирующую модель: каждый сегмент пространства объектов получает немного отличающуюся вероятность

цательных примеров. Отчетливо видно, насколько различны классификаторы, этому способствует также то, что набор данных очень мал. Рисунок демонстрирует различие между комбинированием предсказаний путем голосования (слева) и создания вероятностного классификатора (справа). В случае голосования баггинг порождает кусочно-линейную решающую границу – вещь, невозможную для одного линейного классификатора. Если преобразовать голоса моделей в оценки вероятностей, то мы увидим, что различные решающие границы разбивают пространство объектов на сегменты, каждый из которых потенциально может получить различные оценки.

Алгоритм 11.1. Bagging(D, T, \mathcal{A}) – обучить ансамбль моделей на усиливающих выборках

Вход: набор данных D ; размер ансамбля T ; алгоритм обучения \mathcal{A} .

Выход: ансамбль моделей, предсказания которых необходимо скомбинировать путем голосования или усреднения.

```

1 for  $t = 1$  до  $T$  do
2   | построить усиливающую выборку  $D_t$  из  $D$ , выбрав  $|D|$  примеров с заменой;
3   | прогнать  $\mathcal{A}$  на  $D_t$  и получить в результате модель  $M_t$ ;
4 end
5 return  $\{M_t \mid 1 \leq t \leq T\}$ 
```

Баггинг особенно полезен в сочетании с древовидными моделями, которые очень чувствительны к небольшому изменению обучающих данных. В применении к древовидным моделям баггинг нередко сочетается еще с одной идеей: строить каждое дерево по разным случайно выбранным подмножествам признаков; этот процесс называется *выборкой подпространства* (subspace sampling). В результате разнообразие ансамбля еще повышается, и в качестве дополнитель-

ногого бонуса мы получаем уменьшение времени обучения каждого дерева. Получающийся ансамбль называется *случайным лесом*, а соответствующий алгоритм описан в алгоритме 11.2.

Алгоритм 11.2. *RandomForest(D, T, d)* – обучить ансамбль древовидных моделей на усиливающих выборках и случайных подпространствах

Вход: набор данных D ; размер ансамбля T ; размерность подпространства d .

Выход: ансамбль древовидных моделей, предсказания которых необходимо скомбинировать путем голосования или усреднения.

```

1 for  $t = 1$  до  $T$  do
2   | построить усиливающую выборку  $D_t$  из  $D$ , выбрав  $|D|$  примеров с заменой;
3   | случайным образом выбрать  $d$  признаков и соответственно уменьшить размерность  $D_t$ ;
4   | обучить древовидную модель  $M_t$  по  $D_t$  без редукции;
5 end
6 return  $\{M_t \mid 1 \leq t \leq T\}$ 
```

Поскольку решающее дерево – это группирующая модель, а его листья образуют разбиение пространства объектов, то таковым является и случайный лес: соответствующее ему разбиение пространства объектов представляет собой пересечение разбиений, образуемых входящими в ансамбль деревьями. Хотя разбиение, образуемое случайнм лесом, мельче разбиений, образуемых большинством деревьев, его, в принципе, можно отобразить обратно на одиночную древовидную модель (потому что пересечение соответствует комбинированию ветвей двух разных деревьев). В этом состоит отличие от баггинга линейных классификаторов, при котором решающая граница ансамбля не может быть получена в результате обучения одного базового классификатора. Таким образом, можно сказать, что алгоритм построения случайного леса в случае древовидных моделей – это реализация альтернативного алгоритма обучения.

11.2 Усиление

Усиление, или бустинг (boosting), – техника построения ансамблей, на первый взгляд, похожая на баггинг, однако для внесения разнообразия в обучающие наборы в ней используются более изощренные методы. Основная идея проста и соблазнительна. Допустим, что мы обучили линейный классификатор на некотором наборе данных и обнаружили, что частота ошибок обучения равна ϵ . Мы хотим добавить в ансамбль еще один классификатор, который ошибается реже, чем первый. Добиться этого можно, например, продублировав неправильно классифицированные объекты: если наша модель представляет собой базовый линейный классификатор, то это приведет к смещению средних значений классов в сторону дубликатов. Еще лучше было бы сопоставить неправильно классифицированным объектам больший вес и изменить классификатор, так чтобы он

принимал веса во внимание (например, базовый линейный классификатор умеет вычислять средние значения классов как средневзвешенные).

Но насколько следует изменить вес? Идея в том, чтобы половину общего веса отдать неправильно классифицированным примерам, а другую половину – всем остальным. Поскольку мы начинали с одинаковых весов, которые в сумме дают 1, то текущий вес, приходящийся на долю неправильно классифицированных примеров, в точности равен частоте ошибок ϵ , поэтому мы умножаем их веса на $1/2\epsilon$. В предположении, что $\epsilon < 0.5$, это и будет желаемым увеличением веса. Веса правильно классифицированных примеров умножаются на $1/2(1 - \epsilon)$, так чтобы сумма подправленных весов осталась равной 1. В следующем раунде мы делаем то же самое, только при вычислении частоты ошибок учитываем неодинаковые веса.

Пример 11.1 (изменение весов в методе усиления). Предположим, что качество линейного классификатора такое, как в таблице сопряженности слева. Частота ошибок равна $\epsilon = (9 + 16)/100 = 0.25$. Веса неправильно классифицированных примеров умножаются на $1/2\epsilon = 2$, а правильно классифицированных – на $1/2(1 - \epsilon) = 2/3$.

		Предсказано \oplus	Предсказано \ominus		\oplus	\ominus	
Фактически \oplus	24	16	40	16	32	48	
	9	51	60		18	34	52
	33	67	100		24	66	100

С учетом измененных весов получаем таблицу сопряженности справа, в которой взвешенная частота ошибки равна 0.5.

В алгоритме усиления необходим еще один компонент – коэффициент доверия α для каждой модели в ансамбле; им мы воспользуемся для формирования ансамблевого предсказания, равного взвешенному среднему отдельных моделей. Понятно, что нам хотелось бы, что α увеличивался при уменьшении ϵ : обычно его вычисляют по формуле

$$\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t} = \ln \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}, \quad (11.1)$$

обоснование которой мы приведем чуть ниже. Базовый алгоритм усиления приведен в алгоритме 11.3. На рис. 11.2 слева показано, как усиленный ансамбль из пяти базовых линейных классификаторов может достичь нулевой ошибки обучения. Очевидно, что получившаяся решающая граница гораздо сложнее той, что может построить один базовый линейный классификатор. Напротив, ансамбль из пяти базовых линейных классификаторов, построенный методом багтинга, дал пять очень похожих решающих границ, и объясняется это тем, что усиленные выборки были очень похожи.

Алгоритм 11.3. Boosting(D, T, \mathcal{A}) – обучить ансамбль бинарных классификаторов по обучающим наборам с пересчитываемыми весами

Вход: набор данных D ; размер ансамбля T ; алгоритм обучения \mathcal{A} .

Выход: взвешенный ансамбль моделей.

```

1  $w_{1i} \leftarrow 1/|D|$  для всех  $x_i \in D$ ; // начинаем с одинаковых весов
2 for  $t = 1$  до  $T$  do
3   прогнать  $\mathcal{A}$  на  $D$  с весами  $w_t$ , получив в результате модель  $M_t$ ;
4   вычислить взвешенную ошибку  $\epsilon_t$ ;
5   if  $\epsilon_t \geq 1/2$  then
6     положить  $T \leftarrow t-1$  и выйти из цикла
7   end
8    $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$ ; // доверие для этой модели
9    $w_{(t+1)i} \leftarrow w_{ti}/2\epsilon_t$  для неправильно классифицированных  $x_i \in D$ ; // увеличить вес
10   $w_{(t+1)j} \leftarrow w_{tj}/2(1-\epsilon_t)$  для правильно классифицированных  $x_j \in D$ ; // уменьшить вес
11 end
12 return  $M(x) = \sum_{t=1}^T \alpha_t M_t(x)$ 
```

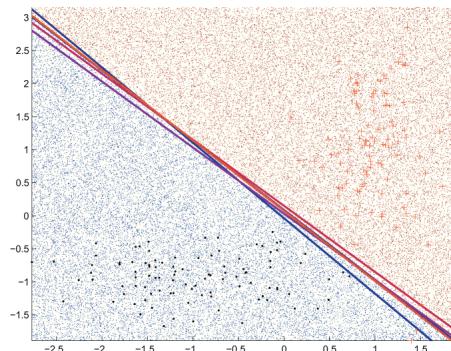
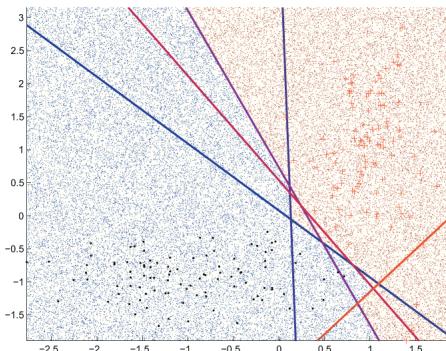


Рис. 11.2. (Слева) Ансамбль из пяти усиленных базовых линейных классификаторов с мажоритарным голосованием. Линейные классификаторы обучались в порядке от **синего** до **красного**, ни один из них в отдельности не достигает нулевой ошибки обучения, а ансамбль достигает. **(Справа)** Применение баггинга приводит к гораздо более однородному ансамблю, доказывая, что усиленные выборки различаются слабо

Перейду к вопросу о том, чем объясняется выбор коэффициента α_t в уравнении (11.1). Во-первых, я покажу, что модификации веса для неправильно и правильно классифицированных примеров можно записать в виде взаимно обратных членов δ_t и $1/\delta_t$, нормированных на некоторую величину Z_t :

$$\frac{1}{2\epsilon_t} = \frac{\delta_t}{Z_t}; \quad \frac{1}{2(1-\epsilon_t)} = \frac{1/\delta_t}{Z_t}.$$

Второе выражение дает $\delta_t = 2(1 - \epsilon_t)/Z_t$; подставляя это в первое выражение, получаем:

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}; \quad \delta_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = \exp(\alpha_t). \quad (11.2)$$

Таким образом, модификация веса для неправильно классифицированных примеров составляет $\exp(\alpha_t)/Z_t$, а для правильно классифицированных $\exp(-\alpha_t)/Z_t$. Пользуясь тем фактом, что $y_i M_t(x_i) = +1$ для объектов, правильно классифицированных моделью M_t , и -1 в противном случае, мы можем записать модификацию веса в виде:

$$w_{(t+1)i} = w_{ti} \frac{\exp(-\alpha_t y_i M_t(x_i))}{Z_t},$$

и именно это выражение часто встречается в литературе.

Теперь вернемся на шаг назад и сделаем вид, что мы еще не решили, какими должны быть значения α_t в каждом раунде. Так как модификации весов мультиплексивны, то имеем

$$w_{(T+1)i} = w_{1i} \prod_{t=1}^T \frac{\exp(-\alpha_t y_i M_t(x_i))}{Z_t} = \frac{1}{|D|} \frac{\exp(-y_i M(x_i))}{\prod_{t=1}^T Z_t},$$

где $M(x_i) = \sum_{t=1}^T \alpha_t M_t(x_i)$ – модель, представленная усиленным ансамблем. Сумма весов по всему пространству объектов всегда равна 1, поэтому

$$\prod_{t=1}^T Z_t = \frac{1}{|D|} \sum_{i=1}^{|D|} \exp(-y_i M(x_i)).$$

Отметим, что $\exp(-y_i M(x_i))$ всегда положительно и не меньше 1, если $-y_i M(x_i)$ положительно, как бывает, когда объект x_i неправильно классифицирован ансамблем (то есть $\text{sign}(M(x_i)) \neq y_i$). Таким образом, правая часть этого выражения не меньше ошибки обучения усиленного ансамбля, и $\prod_{t=1}^T Z_t$ – верхняя граница ошибки обучения. Следовательно, в качестве простой эвристики можно взять жадную минимизацию величины

$$Z_t = \sum_{i=1}^{|D|} w_{ti} \exp(-\alpha_t y_i M_t(x_i)) \quad (11.3)$$

в каждом раунде усиления. Сумма весов объектов, неправильно классифицированных моделью M_t , равна ϵ_t , и потому

$$Z_t = \epsilon_t \exp(\alpha_t) + (1 - \epsilon_t) \exp(-\alpha_t).$$

Приравнивая к нулю производную α_t и решая получившееся уравнение относительно α_t , находим α_t в виде, приведенном в формуле (11.1), и Z_t – в виде (11.2).

Отметим, что из уравнения (11.3) следует, что функция потерь, минимизированная алгоритмом усиления, представляет собой *экспоненциальную функцию потерь* $\exp(-y\hat{s}(x))$, с которой мы уже встречались на рис. 2.6 на стр. 77. Отметим далее, что минимизация Z_t в соответствии с (11.2) означает также минимизацию $2\sqrt{\epsilon_t(1 - \epsilon_t)}$. Вероятно, вы узнаете в этом выражении меру нечистоты Джини, которую мы изучали в главе 5. Там мы видели, что этот критерий разделения нечувствителен к изменениям в распределении по классам (см. рис. 5.7 на стр. 159). Здесь он возникает в основном из-за способа модификации весов в алгоритме усиления.

Обучение усиленных правил

Интересный вариант усиления возникает, когда наши базовые модели являются частичными классификаторами, которые иногда воздерживаются от предсказания. Предположим, к примеру, что наши базовые классификаторы – это конъюнктивные правила, заголовок которых фиксирован и всегда предсказывает положительный класс. Тогда отдельное правило либо предсказывает положительный класс для тех объектов, которые покрывает, либо воздерживается от предсказания. Мы можем воспользоваться усилением, чтобы обучить ансамбль таких правил, который возвращает взвешенный результат голосования среди своих членов.

В уравнения усиления необходимо внести показанные ниже небольшие корректизы. Отметим, что ϵ_t^0 – взвешенная ошибка t -го базового классификатора. Поскольку наши правила всегда предсказывают положительный класс для покрытых объектов, эти ошибки относятся только к покрытым отрицательным объектам, которые мы будем обозначать ϵ_t^\ominus . Аналогично взвешенную сумму покрытых положительных объектов обозначим ϵ_t^\oplus , она будет играть ту же роль, что $1 - \epsilon_t^0$. Однако при наличии воздерживающихся правил есть еще третий компонент, обозначаемый ϵ_t^0 , – взвешенная сумма объектов, не покрытых правилом ($\epsilon_t^0 + \epsilon_t^\ominus + \epsilon_t^\oplus = 1$). Тогда имеем:

$$Z_t = \epsilon_t^0 + \epsilon_t^\ominus \exp(\alpha_t) + \epsilon_t^\oplus \exp(-\alpha_t).$$

Это выражение достигает максимума при следующем значении α_t :

$$\alpha_t = \frac{1}{2} \ln \frac{\epsilon_t^\oplus}{\epsilon_t^\ominus} = \ln \sqrt{\frac{\epsilon_t^\oplus}{\epsilon_t^\ominus}}, \quad (11.4)$$

откуда получаем

$$Z_t = \epsilon_t^0 = 2\sqrt{\epsilon_t^\oplus \epsilon_t^\ominus} = 1 - \epsilon_t^\oplus - \epsilon_t^\ominus + 2\sqrt{\epsilon_t^\oplus \epsilon_t^\ominus} = 1 - \left(\sqrt{\epsilon_t^\oplus} - \sqrt{\epsilon_t^\ominus} \right)^2.$$

Это означает, что в каждом раунде усиления мы конструируем правило, которое максимизирует величину $\left| \sqrt{\epsilon_t^\oplus} - \sqrt{\epsilon_t^\ominus} \right|$, и задаем для него коэффициент доверия α_t ,

показанный в формуле (11.4). Чтобы получить от ансамбля предсказание для некоторого объекта, мы складываем коэффициенты доверия всех покрывающих его правил. Отметим, что эти коэффициенты доверия отрицательны, если $\epsilon_t^{\oplus} < \epsilon_t^{\ominus}$, откуда следует, что правило коррелирует с отрицательным классом; само по себе это не проблема, но тем не менее корреляцию можно устраниить, изменив целевую функцию для отдельных правил на $\sqrt{\epsilon_t^{\oplus}} - \sqrt{\epsilon_t^{\ominus}}$.

Модификации весов после каждой итерации усиления такие же, как и раньше, с тем отличием, что веса примеров, не покрытых правилом, не изменяются. Таким образом, обучение усиленных правил аналогично алгоритму построения [«взвешенного покрытия»](#) (алгоритм 6.5 на стр. 194) для выявления подгрупп. Различие в том, что там мы хотели поощрить перекрытие правил безотносительно к классу и потому уменьшали веса всех покрытых примеров, тогда как здесь мы уменьшаем веса покрытых положительных примеров и увеличиваем веса покрытых отрицательных примеров.

11.3 Карта ансамблевого ландшафта

Теперь, после того как мы ближе познакомились с часто используемыми методами построения ансамблей, поговорим о том, чем объяснить различия в их качестве, а затем уже обратимся к некоторым из многочисленных ансамблевых методов, описанных в литературе.

Смещение, дисперсия и зазоры

Методы построения ансамблей – хороший способ лучше разобраться в [«дилемме смещения-дисперсии»](#), которую мы рассматривали в разделе 3.2 в контексте регрессии. Вообще говоря, есть три причины неверной классификации тестового примера моделью. Во-первых, это просто неизбежно, если в данном пространстве признаков объекты разных классов описываются одинаковыми признаками. В вероятностном контексте это случается, когда условные распределения $P(X|Y)$ перекрываются, так что у некоторого объекта вероятности принадлежности к нескольким классам отличны от нуля. В такой ситуации лучшее, на что можно надеяться, – это аппроксимация целевого концепта.

Вторая причина ошибок классификации – недостаточная выразительность модели для представления целевого концепта. Например, если данные не являются линейно разделимыми, то даже самый лучший линейный классификатор будет делать ошибки. Это смещение классификатора, между ним и выразительностью существует обратная зависимость. И хотя не существует общепринятого способа измерения выразительности или смещения классификатора¹, интуитивно понят-

¹ Хотя квадратичная потеря хорошо раскладывается на квадратичное смещение и дисперсию, как видно из уравнения (3.2) на стр. 107, функции потерь, применяемые в классификации, например функция потерь типа 0–1, допускают различные разложения.

но, что, скажем, у гиперболической решающей границы смещение меньше, чем у линейной. Ясно также, что у древовидных моделей смещение наименьшее из возможных, поскольку их листовые узлы можно сделать настолько мелкими, что они будут покрывать только по одному объекту.

Может показаться, что модели с низким смещением в общем случае предпочтительнее. Однако в практике применения машинного обучения действует эвристическое правило: *у моделей с низким смещением обычно высокая дисперсия, и наоборот*. Дисперсия – это третий источник ошибок классификации. Модель имеет высокую дисперсию, если ее решающая граница сильно зависит от обучающих данных. Например, в случае классификатора по ближайшему соседу сегменты пространства объектов определяются одной точкой обучающего набора, поэтому если я сдвину точку в сегменте, примыкающем к решающей границе, то сдвинется и сама граница. У древовидных моделей дисперсия высока по другой причине: если изменить обучающие данные настолько сильно, что в корне дерева изменится выбранный для разделения признак, то, скорее всего, и все дерево станет другим. Примером модели с низкой дисперсией может служить базовый линейный классификатор, поскольку он производит усреднение по всем точкам класса.

Взгляните теперь на рис. 11.1. Ансамбль базовых линейных классификаторов, построенный методом баггинга, обучил кусочно-линейную решающую границу, которая по выразительности превосходит любой отдельно взятый линейный классификатор. Это говорит о том, что баггинг, как и всякий ансамблевый метод, способен уменьшить смещение базовой модели, которая изначально характеризуется высоким смещением, например линейного классификатора. Однако если сравнить это с результатами метода усиления, показанными на рис. 11.2, то мы увидим, что уменьшение смещения, получающееся в результате баггинга, гораздо меньше, чем в результате усиления. Фактически *баггинг – прежде всего техника уменьшения дисперсии, тогда как усиление – преимущественно метод уменьшения смещения*. Это объясняет, почему баггинг часто применяется в сочетании с моделями, имеющими высокую дисперсию, например древовидными («случайные леса» в алгоритме 11.2), а усиление – с моделями, имеющими высокое смещение, например линейными классификаторами или одномерными решающими деревьями (которые называются также *решающими пнями*).

По-другому усиление можно интерпретировать в терминах зазоров. Интуитивно представляется, что зазор – это расстояние со знаком до решающей границы, причем знак показывает, по правильную ли сторону от границы мы оказались. Экспериментально было отмечено, что усиление дает хороший эффект в плане увеличения зазоров примеров, даже если они уже расположены по правильную сторону от решающей границы. Улучшение качества на тестовом наборе в результате усиления может продолжаться даже после того, как ошибка обучения свелась к нулю. Если учесть, что усиление первоначально возникло в контексте РАС-обучения, не рассчитанном на увеличение зазоров, то этот результат покажется удивительным.

Другие ансамблевые методы

Существует много других методов построения ансамблей, помимо багтинга и усиления. Основные отличия связаны с тем, как комбинируются предсказания базовых моделей. Отметим, что этот вопрос и сам по себе можно было бы определить как проблему обучения: рассматривая предсказания некоторых базовых классификаторов как признаки, обучить *метамодель*, которая будет комбинировать их наилучшим способом. Например, в методе усиления мы могли бы обучить веса α_i , а не выводить их из частот ошибок каждой отдельной базовой модели. Обучение линейной метамодели называется *укладкой* (stacking). Существует несколько вариаций на эту тему, например в качестве метамоделей применялись решающие деревья. Можно также комбинировать различные базовые модели в гетерогенный ансамбль, при этом разнообразие достигается за счет того, что базовые модели обучаются различными алгоритмами, поэтому можно использовать один и тот же обучающий набор. Для некоторых базовых моделей можно задавать различные параметры, например ансамбль может включать несколько машин опорных векторов с разными значениями параметра сложности, который определяет, в какой мере терпимы ошибки зазора.

Итак, в общем случае ансамбль моделей состоит из множества базовых моделей и метамодели, которая обучена решать, как именно следует комбинировать предсказания базовых моделей. Обучение метамодели неявно подразумевает оценку качества каждой базовой модели, например если метамодель линейна, как в случае укладки, то вес, близкий к нулю, означает, что соответствующий базовый классификатор не дает большого вклада в ансамбль. Можно даже допустить, что базовый классификатор получает отрицательный вес, тогда в контексте остальных базовых моделей его предсказания следует инвертировать. Можно было бы пойти еще дальше и попытаться *предсказать* ожидаемое качество базовой модели еще до ее обучения! Сформулировав это как проблему обучения на метауровне, мы вступаем в область метаобучения.

Метаобучение

Прежде всего метаобучение подразумевает обучение ряда моделей на большой совокупности наборов данных. Цель состоит в том, чтобы сконструировать модель, которая поможет отвечать на такие вопросы:

- ☞ в каких случаях решающее дерево, скорее всего, окажется лучше метода опорных векторов?
- ☞ когда от линейного классификатора следует ожидать плохого качества?
- ☞ можно ли использовать данные для рекомендования конкретных параметров?

Ключевой вопрос метаобучения – как спроектировать признаки, на основе которых строится метамодель? Эти признаки должны сочетать в себе характеристики набора данных и релевантных аспектов обученной модели. Характеристики набора данных отнюдь не должны ограничиваться простым перечислением

количества и вида признаков и количества объектов, потому что маловероятно, что на основе только лишь этой информации можно будет предсказать что-то содержательное о качестве модели. Например, мы можем попытаться оценить уровень зашумленности данных, измерив размер обученного решающего дерева до и после редукции. Обучение на наборе данных простых моделей типа решающих пней и последующее измерение их качества также дает полезную информацию.

В замечании 1.1 на стр. 31 мы упоминали теорему о бесплатных завтраках, которая утверждает, что никакой алгоритм обучения не может быть лучше всех остальных алгоритмов обучения на множестве всех возможных проблем обучения. Отсюда следует, что попытка метаобучения на всех возможных проблемах обучения тщетна, – иначе мы могли бы построить одну гибридную модель, которая использует метамодель, чтобы сообщить, у какой базовой модели качество будет превосходить случайное на конкретном наборе данных. А это означает, что мы можем лишь надеяться, что метаобучение окажется полезным на проблемах обучения с неравномерным распределением.

11.4 Ансамбли моделей: итоги и литература для дальнейшего чтения

В этой короткой главе мы обсудили некоторые фундаментальные идеи методов построения ансамблей. У всех таких методов есть общая черта: они строят несколько базовых моделей на основе модифицированных обучающих данных, а затем применяют тот или иной способ комбинирования предсказаний или оценок отдельных базовых моделей для получения предсказания всего ансамбля. Мы рассмотрели два широко распространенных ансамблевых метода: баггинг и усиление. Хорошее введение в ансамбли моделей имеется в работе Brown (2010). Стандартный справочник по комбинированию классификаторов – работа Kuncheva (2004), а более современный обзор имеется в работе Zhou (2012).

- ☞ В разделе 11.1 обсудили баггинг и случайные леса. По методу баггинга на выборках из обучающих данных обучаются различающиеся модели, он впервые был описан в работе Breiman (1996а). Случайные леса, обычно приписываемые работе Breiman (2001), объединяют обученные по методу баггинга решающие деревья со случайными подпространствами; аналогичные идеи были развиты в работах Ho (1995) и Amit, Geman (1997). Эти методы особенно полезны для уменьшения дисперсии моделей с низким смещением, например решающих деревьев.
- ☞ В разделе 11.2 обсуждался метод усиления. Его основная идея – обучить различающиеся модели путем увеличения веса примеров, которые раньше были классифицированы неправильно. Это позволяет уменьшить смещение устойчивых методов обучения, каковыми являются, например, линейные классификаторы и решающие пни. Доступный обзор данной тематики имеется в работе Schapire (2003). В работах Kearns, Valiant (1989, 1994)

- был поставлен вопрос, верно ли, что слабый алгоритм обучения, который работает лишь немного лучше случайного угадывания, можно усилить до алгоритма обучения с произвольно высокой точностью. В работе Schapire (1990) дано теоретическое определение усиления, позволившее доказать эквивалентность слабой и сильной обучаемости. Алгоритм AdaBoost, лежащий в основе алгоритма 11.3, впервые был обнародован в работе Freund, Schapire (1997). В работе Schapire, Singer (1999) приведены обобщения AdaBoost на случаи нескольких классов и нескольких меток. Ранжирующий вариант AdaBoost был предложен в работе Freund et al. (2003). Подход к обучению усиленных правил, допускающий классификаторы, которые могут воздерживаться от предсказания, основан на алгоритме Slipper (Cohen, Singer, 1999), усиленном варианте алгоритма Ripper (Cohen, 1995).
- ☞ В разделе 11.3 мы обсудили баггинг и усиление с точки зрения смещения и дисперсии. В работе Schapire, Freund, Bartlett, Lee (1998) приведен детальный теоретический и экспериментальный анализ усиления в терминах улучшения маргинального распределения. Я упомянул также некоторые другие ансамблевые методы, предназначенные для обучения метамодели, комбинирующющей результаты базовых моделей. В методе укладки применяется линейная метамодель, он был введен в работе Wolpert (1992) для классификации и распространен на регрессию в работе Breiman (1996b). Решающие метадеревья впервые описаны в работе Todorovski, Dzeroski (2003).
 - ☞ Мы также обсудили метаобучение – технику, позволяющую предсказывать качество алгоритмов обучения. Эта направление зародилось в раннем эмпирическом исследовании, документированном в работе Michie et al. (1994). Из недавних работ следует отметить Brazdil et al. (2009, 2010). Редуцированные и нередуцированные решающие деревья использовались для получения характеристик наборов данных в работе Peng et al. (2002). Идея обучения простых моделей для получения дополнительных характеристик данных известна под названием межевания (landmarking) (Pfahringer et al. 2000).



Эксперименты в машинном обучении

Машинное обучение – дисциплина, в такой же степени практическая, как и вычислительная. В некоторых случаях мы можем доказать, что конкретный алгоритм сходится к теоретически оптимальной модели при определенных предположениях, но все равно необходимы реальные данные, например для того, чтобы исследовать, в какой мере эти предположения удовлетворяются в рассматриваемой предметной области или достаточно ли велика скорость сходимости, чтобы алгоритм имел практическую ценность. Поэтому мы запускаем конкретные модели или алгоритмы обучения на одном или нескольких наборах данных, выполняем измерения и используем их результаты для ответа на интересующие нас вопросы. Вся эта деятельность называется *экспериментами* в машинном обучении.

В естественных науках эксперимент можно рассматривать как адресованный природе вопрос о некоторой научной теории. Например, в знаменитом эксперименте Артура Эддингтона, поставленном в 1919 году для проверки эйнштейновской общей теории относительности, был задан вопрос: отклоняются ли лучи света в гравитационных полях массивных космических тел, например Солнца? Для ответа на этот вопрос наблюдаемое положение звезд было зафиксировано при различных условиях, в том числе при полном солнечном затмении. Эддингтон сумел доказать, что результаты измерений не могли быть объяснены законами ньютонаской физики, но согласовывались с общей теорией относительности.

И хотя для проведения экспериментов в машинном обучении вам не придется отправляться на остров Принципи, они все же похожи на физические эксперименты в том отношении, что *эксперименты в машинном обучении ставят вопросы о моделях, на которые мы пытаемся получить ответы путем измерения данных*. Ниже перечислены некоторые типичные вопросы, которые могут нас интересовать.

- ☞ Каково качество модели m на данных из предметной области \mathcal{D} ?
- ☞ Какие из предъявленных моделей показывают наилучшие результаты на данных из предметной области \mathcal{D} ?
- ☞ Как модели, порожденные алгоритмом обучения \mathcal{A} , ведут себя на данных из предметной области \mathcal{D} ?
- ☞ Какие из предъявленных алгоритмов обучения порождают наилучшую модель на данных из предметной области \mathcal{D} ?

В предположении, что у нас имеется доступ к данным из предметной области \mathcal{D} , мы используем эти данные для проведения измерения наших моделей

с целью получить ответы на поставленные вопросы¹. В математической статистике существует обширная литература по техническим деталям экспериментов с данными, и очень легко за деревьями не увидеть леса. Я хочу сказать, что в литературе по машинному обучению есть тенденция подходить к экспериментам по стандартному клише: привести некий набор измерений и критерии значимости (деревья), почти не уделив внимания вопросу, на который ищется ответ (лес). В этой коротенькой главе я постараюсь не погрязнуть в технических деталях, а дать представление о важности отбора тех измерений, которые лучше всего отвечают цели эксперимента (раздел 12.1). А в разделах 12.2 и 12.3 мы ближе познакомимся с тем, как проводить измерения и интерпретировать их результаты.

12.1 Что измерять

Неплохой отправной точкой для измерений могут стать *показатели качества*, приведенные в табл. 2.3 на стр. 71. Однако результаты измерений не обязаны быть скалярами: кривая РХП или покрытия в этом контексте также может рассматриваться как результат измерения. Адекватность измерений нашим целям зависит от того, как определить качество относительно вопроса, на который призван ответить эксперимент: назовем это *целью эксперимента*. Важно не путать показатели качества с целью эксперимента: первое – это то, что мы измеряем, второе – то, что нас в действительности интересует. Между ними часто имеется расхождение. Например, в психологии целью эксперимента может быть определение уровня интеллекта человека, а измеряется коэффициент IQ на стандартных тестах; и хотя коэффициент IQ может коррелировать с уровнем интеллекта, очевидно, что это не одно и то же.

В машинном обучении ситуация обычно более конкретна, и цель эксперимента – скажем, верность – это нечто, в принципе допускающее измерение или, по крайней мере, оценку (поскольку в общем случае нас интересует верность на ранее не предъявлявшихся данных). Однако могут существовать неизвестные факторы, которые необходимо учитывать. Например, модели, возможно, предстоит работать в других *рабочих контекстах*, с иным распределением по классам. В таком случае мы можем рассматривать верность на будущих данных как случайную величину и брать ее математическое ожидание в предположении некоторого распределения вероятностей доли положительных объектов *pos*. Поскольку $acc = pos \cdot tpr + (1 - pos) \cdot tnr$, то в предположении, что мы можем измерить частоты истинно положительных и истинно отрицательных результатов независимо от распределения по классам, имеем (в случае равномерного распределения *pos*):

¹ Гораздо труднее ответить на вопросы о новой предметной области, имея результаты измерений, относящиеся к другим областям, хотя именно это и представляет основной интерес.

$$\begin{aligned}\mathbb{E}[acc] &= \mathbb{E}[pos \cdot tpr + (1 - pos) \cdot tnr] = \mathbb{E}[pos]tpr + \mathbb{E}[1 - pos]tnr = \\ &= tpr/2 + tnr/2 = avg-rec.\end{aligned}$$

Иными словами, хотя целью нашего эксперимента является оценка верности в будущих контекстах, из того, что мы пытаемся предвосхитить максимально широкий диапазон возможных распределений по классам, следует, что на роль показателя качества следует выбрать не верность на тестовых данных, а среднюю полноту.

Пример 12.1. (ожидаемая верность для неизвестных распределений по классам). Допустим, что наш классификатор показывает такой результат на тестовом наборе данных:

	Предсказано \oplus	Предсказано \ominus	
Фактически \oplus	60	20	80
Фактически \ominus	0	20	20
	60	40	100

Это дает $tpr = 0.75$, $tnr = 1.00$ и $acc = 0.80$. Однако эти результаты получены при условии, что положительных примеров в четыре раза больше, чем отрицательных. Если взять математическое ожидание, когда pos равномерно распределено в единичном интервале, то ожидаемая верность возрастет до $(tpr + tnr)/2 = 0.88 = avg-rec$. Она больше, потому что тестовые данные недооценивают хорошее качество классификатора на отрицательных примерах.

Предположим, что другой классификатор на том же тестовом наборе показал такие результаты:

	Предсказано \oplus	Предсказано \ominus	
Фактически \oplus	75	5	80
Фактически \ominus	10	10	20
	85	15	100

Это дает $tpr = 0.94$, $tnr = 0.50$, $acc = 0.85$ и $avg-rec = 0.72$. Эти экспериментальные результаты говорят, что вторая модель лучше, если распределение по классам в тестовом наборе репрезентативно, но если мы не располагаем априорной информацией о распределении по классам в рабочем контексте, то следует выбрать первую модель.

Как следует из примера, принимая верность за показатель качества, мы делаем неявное предположение о том, что распределение по классам в тестовом наборе репрезентативно для рабочего контекста, в котором модель будет развернута. Кроме того, если в экспериментах мы измеряли только верность, то не сможем перейти на среднюю полноту впоследствии, когда поймем, что нужно учитывать изменяющееся распределение по классам. Поэтому рекомендуется запоминать достаточно информации, чтобы при необходимости можно было реконструировать таблицу сопряженности. Таким достаточным набором измерений можно считать частоту истинно положительных результатов, частоту истинно отрицательных (или ложноположительных) результатов, распределение по классам и размер тестового набора.

В качестве второго примера того, как выбор показателей качества может заключать в себе неявные предположения, рассмотрим точность и полноту, часто упоминаемые в литературе по информационному поиску.

Пример 12.2 (точность и полнота как показатели качества). Во второй таблице сопряженности в примере 12.1 точность равна $prec = 75/85 = 0.88$, а полнота $rec = 75/80 = 0.94$ (напомним, что полнота и частота истинно положительных результатов – разные названия одного и того же показателя). F-мера определяется как среднее гармоническое точности и полноты (см. замечание 10.1), равна 0.91.

Теперь рассмотрим такую таблицу сопряженности:

	Предсказано \oplus	Предсказано \ominus	
Фактически \oplus	75	5	80
Фактически \ominus	10	910	920
	85	915	1000

Здесь количество истинно отрицательных результатов гораздо больше, а потому гораздо выше частота истинно отрицательных результатов и верность (обе после округления равны 0.99). С другой стороны, частота истинно положительных результатов (полнота) и F-мера почти не изменились.

Этот пример доказывает, что *комбинация точности и полноты, а значит, и F-мера, нечувствительна к количеству истинно отрицательных результатов*. Это нельзя считать недостатком F-меры, совсем наоборот, это свойство весьма полезно в предметных областях, где отрицательные результаты преобладают, и потому было бы очень просто добиться высокой верности, всегда предсказывая отрицательный класс. Примеры таких предметных областей – поисковые системы (большинство документов не является ответом на большинство запросов) и предсказание связей в сетях (большинство пар узлов не связано). Важно уяснить, что если в качестве показателя качества мы выбираем F-меру, то делаем неявное предположение о том, что истинно отрицательные результаты в данном рабочем контексте несущественны.

Наконец, я хотел бы обратить внимание на показатель качества, которым часто пренебрегают, хотя он имеет практическое значение. Это *предсказанная частота положительных результатов*, то есть отношение количества положительных предсказаний (сумма чисел в левом столбце таблицы сопряженности) к общему количеству объектов:

$$ppr = \frac{TP + FP}{Pos + Neg} = pos \cdot tpr + (1 - pos) \cdot spr.$$

Хотя предсказанная частота положительных результатов мало говорит о качестве классификации, она несет информацию о том, как классификатор оценивает распределение по классам. Кроме того, этот показатель обычно контроли-

руется ранжировщиком или оценивающим классификатором, если задан весь тестовый набор: например, для достижения предсказанной частоты положительных результатов $1/2$ нужно просто задать пороговое значение так, чтобы ранжирование разделяло набор на две равные части. Это наводит на мысль о связи между верностью классификации и верностью ранжирования; например, можно показать, что если при ранжировании n объектов случайно и равномерно выбирать одну из $n + 1$ возможных точек разделения, то математическое ожидание верности будет равно

$$\mathbb{E}[acc] = \frac{n}{n+1} \frac{2\text{AUC}-1}{4} + 1/2.$$

Пример 12.3 (ожидаемая верность и AUC). Предположим, что ранжировщик получает следующее ранжирование на небольшом тестовом наборе: $\oplus\oplus\ominus\ominus\oplus\ominus$. Это соответствует двум ошибкам ранжирования из девяти возможных, поэтому $\text{AUC} = 7/9$. Существуют семь потенциальных точек разделения, которым соответствуют такие предсказанные частоты положительных результатов (слева направо): $0, 1/6, \dots, 5/6, 1$ – и верности: $3/6, 4/6, 5/6, 4/6, 3/6, 4/6, 3/6$. Математическое ожидание верности по всем девяти точкам разделения равно $(3 + 4 + 5 + 4 + 3 + 4 + 3) / (6 \cdot 7) = 26/42$. С другой стороны, $(2\text{AUC} - 1)/4 = 5/36$ и, следовательно, $(^n/_n+1)(2\text{AUC} - 1)/4 + 1/2 = 5/42 + 1/2 = 26/42$.

Это обсуждение призвано подчеркнуть две вещи. Во-первых, если модель является хорошим ранжировщиком, но выдаваемые ей оценки вероятностей плохо откалиброваны, то, возможно, будет разумно выбрать такой порог принятия решения, при котором достигается конкретная предсказанная частота положительных результатов (например, $ppr = pos$), а не использовать решающее правило апостериорного максимума. Во-вторых, в таких ситуациях площадь под кривой (AUC) является хорошим показателем качества, потому что линейно связана с ожидаемой верностью.

Короче говоря, выбор показателей качества должен отражать предположения о цели эксперимента, а также возможные контексты, в которых будут работать модели. Мы рассмотрели следующие случаи:

- ☞ верность является хорошим показателем качества, если распределение по классам в тестовом наборе репрезентативно для рабочего контекста;
- ☞ средняя полнота является хорошим показателем качества, если все распределения по классам равновероятны;
- ☞ точность и полнота смешают акцент с верности классификации на анализ качества при условии, что можно игнорировать истинно отрицательные результаты;
- ☞ предсказанная частота положительных результатов и AUC – подходящие показатели качества в контексте ранжирования.

В следующем разделе мы рассмотрим, как оценивать эти показатели качества на основе имеющихся данных.

12.2 Как измерять

Все показатели качества, рассмотренные в предыдущем разделе, вычисляются по таблице сопряженности. Поэтому вопрос «как измерять» кажется очень простым; построить таблицу сопряженности по тестовому набору и произвести необходимые вычисления. Однако есть две тонкости: (i) на основе каких данных производить измерения и (ii) как оценить неизбежную неопределенность, присущую каждому измерению. В этом разделе мы займемся первым вопросом, а второй отложим до следующего раздела.

Когда мы измеряем что-то – скажем, рост человека – несколько раз, мы ожидаем, что каждое следующее измерение будет немного отличаться от предыдущего. Это неотъемлемая особенность процесса измерения: быть может, во втором измерении вы чуть сильнее растянули мерительную ленту или, считывая результат, смотрели на нее немного под другим углом¹. Эту вариацию можно смоделировать, рассматривая измерение как случайную величину, характеризуемую своим средним – тем значением, которое мы пытаемся измерить, – и дисперсией σ^2 ; то и другое неизвестно, но может быть оценено. Из этой модели следует, что если измерить рост человека много раз, то выборочная дисперсия измеренных значений стремится к σ^2 . Стандартный прием заключается в усреднении k измерений, поскольку это уменьшает дисперсию оценки до σ^2/k . А это значит, что если повторить усреднение для многих наборов из k измерений, то выборочная дисперсия результатов усреднения будет равна σ^2/k . Подчеркну – здесь предполагается, что все измерения независимы: если вы внесете систематическую ошибку, например воспользуетесь неправильной мерительной лентой, то усреднение не поможет!

Теперь предположим, что мы измеряем верность классификатора (или частоту истинно положительных результатов, или предсказанную частоту положительных результатов, или еще какой-то из обсуждавшихся выше показателей качества), а не рост человека. Напрашивается модель, согласно которой каждый тестовый объект представляет испытание Бернулли с вероятностью успеха a – истинной, но неизвестной нам верностью классификатора. Мы оцениваем a , подсчитывая количество правильно классифицированных тестовых объектов A и полагая $\hat{a} = A/n$; отметим, что A имеет биномиальное распределение. Дисперсия одного испытания Бернулли равна $a(1-a)$; после усреднения по n тестовым объектам получается $a(1 - a)/n$ – в предположении, что тестовые объекты выбраны независимо. Мы можем оценить дисперсию, подставив сюда нашу оценку a ; как мы увидим в следующем разделе, это поможет оценить неопределенность \hat{a} . При некоторых условиях оценку можно улучшить, усреднив k независимых оценок

¹ Вариация может быть обусловлена и тем фактом, что, только встав с постели утром, человек обычно выше, чем в конце дня, в течение которого он много раз садился и вставал, но этот нюанс я игнорирую и считаю, что существует однозначное истинное значение, которое мы пытаемся измерить.

\hat{a}_i , и взять их выборочную дисперсию $\frac{1}{k-1} \sum_{i=1}^k (\hat{a}_i - \bar{a})^2$, где $\bar{a} = \frac{1}{k} \sum_{i=1}^k \hat{a}_i$ – выборочное среднее¹.

Как получить k независимых оценок a ? Если данных достаточно много, то можно выбрать k независимых тестовых наборов размера n и оценить a на каждом из них. Отметим, что если оцениваем качество алгоритма обучения, а не конкретной модели, то необходимо разделить обучающие и тестовые данные. Если же данных недостаточно, то часто применяется следующая процедура *перекрестной проверки*: данные случайным образом разбиваются на k частей, или групп, одна группа резервируется для тестирования, модель обучается на остальных $k - 1$ группах, а ее качество оценивается на зарезервированной тестовой группе. Этот процесс повторяется k раз, чтобы каждая группа один раз побывала в роли тестовой. На первый взгляд, это может показаться странным, потому что мы оцениваем качество k моделей, а не одной, однако смысл в этом есть, если оценивается качество алгоритма обучения (когда выходом является модель, так что мы производим усреднение по моделям), а не одной конкретной модели (когда выходом являются метки объектов, по которым и производится усреднение). Производя усреднение по обучающим наборам, мы получаем представление о дисперсии алгоритма обучения (то есть его зависимости от вариаций обучающих данных), хотя следует отметить, что обучающие наборы в процедуре перекрестной проверки значительно перекрываются и уж никак не могут считаться независимыми. Если мы остались довольны качеством алгоритма обучения, то можем применить его ко всему набору данных для получения конкретной модели.

В перекрестной проверке традиционно полагают $k = 10$, хотя это соглашение произвольно. Эвристическое правило гласит, что каждая группа должна содержать не менее 30 экземпляров, поскольку это позволит аппроксимировать биномиальное распределение числа правильно классифицированных объектов в группе нормальным. Следовательно, если в нашем распоряжении имеется меньше 300 объектов, то k нужно соответственно подправить. Можно вместо этого положить $k = n$ и производить обучение на всех объектах, кроме тестового, повторив его n раз; это называется перекрестной проверкой с исключением по одному (или критерием складного ножа в статистике). Это означает, что в каждой группе из одного объекта оценка верности равна 0 или 1, но усреднение по n группам даст приблизительно нормальное распределение – согласно центральной предельной теореме. Если мы ожидаем, что алгоритм обучения будет чувствителен к распределению по классам, то должны применить *послойную перекрестную проверку*: ее цель – добиться примерно одинакового распределения по классам в каждой группе. Перекрестную проверку можно повторять для разных случайных разбиений на группы, и результаты снова усредняются, чтобы еще уменьшить дисперсию оценок; такая процедура называется, например, 10-кратная 10-групповая

¹ Заметьте, что в выражении для выборочной дисперсии мы делим на $k - 1$, а не на k , чтобы учесть неопределенность в оценке выборочного среднего.

перекрестная проверка. Следует помнить, что при этом предположение о независимости нарушается все сильнее – если зайти достаточно далеко, то оценка верности будет переобучена на имеющихся данных, а для новых данных окажется непрепрезентативной.

Пример 12.4 (перекрестная проверка). В следующей таблице приведен один из возможных результатов оценки качества трех алгоритмов обучения на наборе данных с помощью 10-групповой перекрестной проверки:

Группа	Наивный байесовский	Решающее дерево	Ближайший сосед
1	0.6809	0.7524	0.7164
2	0.7017	0.8964	0.8883
3	0.7012	0.6803	0.8410
4	0.6913	0.9102	0.6825
5	0.6333	0.7758	0.7599
6	0.6415	0.8154	0.8479
7	0.7216	0.6224	0.7012
8	0.7214	0.7585	0.4959
9	0.6578	0.9380	0.9279
10	0.7865	0.7524	0.7455
avg	0.6937	0.7902	0.7606
stdev	0.0448	0.1014	0.1248

В последних двух строках приведены среднее и стандартное отклонения, вычисленные по всем десяти группам. Как видим, у алгоритма классификации по ближайшему соседу стандартное отклонение наибольшее. Очевидно, что решающее дерево дает наилучший результат, но нужно ли вовсе отказываться от алгоритма ближайшего соседа?

Перекрестную проверку можно применить и к кривым РХП, полученным от оценивающего классификатора. Объясняется это тем, что каждый объект участвует ровно в одной тестовой группе и получает оценку от соответствующей модели. Поэтому мы можем просто объединить все тестовые группы и получить единственное ранжирование.

12.3 Как интерпретировать

Имея оценки относящихся к делу показателей качества для наших моделей или алгоритмов обучения, мы можем воспользоваться ими для выбора лучшего представителя. Фундаментальная проблема – как быть с присущей этим оценкам неопределенностью? Мы обсудим два ключевых понятия: доверительные интервалы и критерии значимости. Понимать их необходимо, если вы хотите разобраться в современных подходах к интерпретации результатов экспериментов

в машинном обучении; однако не следует забывать, что современные подходы подвергаются критическому анализу. Отметим также, что описанные здесь методы – лишь малая толика широчайшего спектра возможностей.

Допустим, что наша оценка \hat{a} имеет нормальное распределение с истинным значением a в качестве среднего и стандартным отклонением σ . Если ненадолго предположить, что эти параметры нам известны, то для любого интервала мы сможем вычислить выраженную в процентах вероятность попадания оценки в этот интервал; для этого нужно лишь посчитать площадь под кривой плотности нормального распределения на этом интервале. Например, вероятность, что оценка находится от среднего на расстоянии ± 1 стандартное отклонение, составляет 68%. Следовательно, если получить 100 оценок по независимым тестовым наборам, то можно ожидать, что 68 из них окажутся не дальше одного стандартного отклонения от среднего – по ту или другую сторону. Ту же мысль можно выразить иначе: можно ожидать, что истинное среднее в 68 случаях окажется не дальше одного стандартного отклонения от оценки – с одной или с другой стороны. Это называется 68%-ным *доверительным интервалом* оценки. Для двух стандартных отклонений доверительный уровень составляет 95% – эти значения можно найти в таблицах вероятностей или вычислить с помощью таких статистических пакетов программ, как Matlab или R. Отметим, что доверительные интервалы для оценок с нормальным распределением симметричны, потому что симметрично само нормальное распределение, но в общем случае это не так: например, биномиальное распределение асимметрично (за исключением случая $p = 1/2$). Отметим также, что в симметричном случае интервал легко сделать односторонним; например, мы ожидаем, что среднее *превышает* оценку более чем на одно стандартное отклонение в 16 случаях из 100, что дает односторонний 84%-ный доверительный интервал от минус бесконечности до среднего плюс одно стандартное отклонение.

Более общо, для построения доверительных интервалов нам нужно знать (*i*) выборочное распределение оценок и (*ii*) параметры этого распределения. Выше мы видели, что верность, оцененная по одному тестовому набору из n объектов, имеет масштабированное биномиальное распределение с дисперсией $\hat{a}(1 - \hat{a})/n$. Это привело бы к асимметричным доверительным интервалам, но асимметрия биномиального распределения становится заметной, только если $na(1 - a) < 5$; в противном случае биномиальное распределение хорошо аппроксимируется нормальным. Таким образом, мы используем биномиальное выражение для дисперсии и нормальное распределение для построения доверительных интервалов.

Пример 12.5 (доверительный интервал). Предположим, что 80 из 100 тестовых объектов классифицированы правильно. Тогда $\hat{a} = 0.80$ с оценкой дисперсии $\hat{a}(1 - \hat{a})/n = 0.0016$ или стандартного отклонения $\sqrt{\hat{a}(1 - \hat{a})}/n = 0.04$. Отметим, что $na^*(1 - \hat{a}) = 16 \geq 5$, поэтому 68%-ный доверительный интервал в соответствии с нормальным распределением можно оценить как $[0.76, 0.84]$, а 95%-ный – как $[0.72, 0.88]$.

Если уменьшить размер тестового набора до 50 и при этом 40 объектов будут классифицированы правильно, то стандартное отклонение уменьшится до 0.06, а 95%-ный доверительный интервал расширится до [0.68, 0.92]. Если размер тестового набора станет меньше 30, то придется строить асимметричный доверительный интервал, пользуясь таблицами для биномиального распределения.

Отметим, что *доверительные интервалы – это утверждения об оценках, а не об истинном значении показателя качества*. Утверждение «в предположении, что истинная верность равна 0.80, вероятность того, что результат измерения попадает в интервал [0.72, 0.88], равна 0.95» правильно, но мы не можем обратить его, сказав «в предположении, что результат измерения $m = 0.80$, вероятность того, что истинная верность попадает в интервал [0.72, 0.88], равна 0.95». Чтобы вывести $P(a \in [0.72, 0.88] | m = 0.80)$ из $P(m \in [0.72, 0.88] | a = 0.80)$, мы должны каким-то образом применить формулу Байеса, но это требует знания осмысленных априорных распределений истинных значений верности и результатов измерений, чего у нас обычно нет.

Однако мы можем воспользоваться похожим рассуждением для проверки конкретной *нулевой гипотезы* об a . Например, предположим, что нулевая гипотеза состоит в том, что истинная верность равна 0.5 и что стандартное отклонение, выведенное на основе биномиального распределения, поэтому равно $\sqrt{0.5(1 - 0.5)/100} = 0.05$. Имея оценку 0.80, мы затем вычисляем *p-значение* – вероятность получить результат измерения не ниже 0.80 при условии нулевой гипотезы. Далее *p*-значение сравнивается с предопределенным уровнем значимости, скажем $\alpha = 0.05$, он соответствует доверительному уровню 95%. Нулевая гипотеза отвергается, если *p*-значение меньше α ; в нашем случае так оно и есть, потому что $p = 1.9732 \cdot 10^{-9}$.

Эту идею *проверки значимости* можно распространить на алгоритмы обучения, качество которых оценивается в ходе перекрестной проверки. Для пары алгоритмов мы вычисляем разность значений верности для каждой группы. Разность двух нормально распределенных случайных величин также имеет нормальное распределение. Наша нулевая гипотеза состоит в том, что истинная разность равна 0, так что любые отклонения качества приписываются влиянию случая. Мы вычисляем *p*-значение, пользуясь нормальным распределением, и отвергаем нулевую гипотезу, если *p*-значение меньше уровня значимости α . Есть, правда, одно осложнение: у нас нет доступа к истинному стандартному отклонению разностей, так что его придется оценивать. Это вносит в процесс дополнительную неопределенность, и, следовательно, выборочное распределение будет колоколообразным, как нормальное, но с более медленно убывающими хвостами. Это распределение называется *t*-распределением Стьюдента, или просто *t-распределением*¹.

¹ Его опубликовал Уильям Сили Госсет в 1908 году под псевдонимом «Стьюдент», потому что его работодатель, пивоварня Гиннесса в Дублине, не хотел, чтобы конкуренты знали, что компания пользуется услугами статистиков.

Насколько хвосты t -распределения более «тяжеловесны», по сравнению с нормальным распределением, регулируется количеством *степеней свободы*; в нашем случае оно на 1 меньше количества групп (поскольку последняя группа однозначно определена всеми остальными). Процедура в целом известна под названием *парный t -критерий*.

Пример 12.6 (парный t -критерий). В следующей таблице продемонстрировано вычисление парного t -критерия для результатов примера 12.4. Числа – это попарные разности в каждой группе. Нулевая гипотеза в каждом случае состоит в том, что разности подчиняются нормальному распределению со средним 0 и неизвестным стандартным отклонением.

Группа	НБ–РД	НБ–БС	РД–БС
1	−0.0715	−0.0355	0.0361
2	−0.1947	−0.1866	0.0081
3	0.0209	−0.1398	−0.1607
4	−0.2189	0.0088	0.2277
5	−0.1424	−0.1265	0.0159
6	−0.1739	−0.2065	−0.0325
7	0.0992	0.0204	−0.0788
8	−0.0371	0.2255	0.2626
9	−0.2802	−0.2700	0.0102
10	0.0341	0.0410	0.0069
avg	−0.0965	−0.0669	0.0295
stdev	0.1246	0.1473	0.1278
<i>p</i> -значение	0.0369	0.1848	0.4833

p-значение в последней строке таблицы вычисляется согласно t -распределению с $k - 1 = 9$ степенями свободы, и лишь разность между наивным байесовским алгоритмом и решающим деревом значима при уровне $\alpha = 0.05$.

Интерпретация результатов, полученных на нескольких наборах данных

t -критерий можно применить для сравнения двух алгоритмов обучения на одном наборе данных, обычно с использованием результатов, полученных при перекрестной проверке. Для нескольких наборов данных он не подходит, потому что показатели качества невозможно сравнивать на разных данных (они «несоизмеримы»). Чтобы сравнить два алгоритма обучения на нескольких наборах данных, необходим критерий, специально предназначенный для этой цели, например *критерий знаковых рангов Уилкоксона*. Идея заключается в том, чтобы ранжировать разности показателей качества по абсолютному значению от наименьшего (ранг 1) к наибольшему (ранг n). Затем мы вычисляем по отдельности суммы рангов для положительных и отрицательных разностей и наименьшую сумму принимаем в качестве статистического критерия. Если наборов данных

много (не менее 25), то эту статистику можно преобразовать в имеющую приблизительно нормальное распределение, в противном случае *критическое значение* (значение статистики, при котором p -значение равно α) можно найти в статистической таблице.

Пример 12.7 (критерий знаковых рангов Уилкоксона). Мы будем использовать разности между качеством наивного байесовского алгоритма и решающего дерева из предыдущего примера, но теперь предположим, что они были получены на 10 различных наборах данных.

Набор данных	НБ – РД	Ранг
1	–0.0715	4
2	–0.1947	8
3	0.0209	1
4	–0.2189	9
5	–0.1424	6
6	–0.1739	7
7	0.0992	5
8	–0.0371	3
9	–0.2802	10
10	0.0341	2

Сумма рангов для положительных разностей равна $1 + 5 + 2 = 8$, а для отрицательных $4 + 8 + 9 + 6 + 7 + 3 + 10 = 47$. Критическое значение для 10 наборов данных при $\alpha = 0.05$ равно, то есть если меньшая из двух сумм рангов меньше или равна 8, то нулевую гипотезу, состоящую в том, что ранги распределены одинаково для положительных и отрицательных разностей, можно отвергнуть. В данном случае так оно и есть, поэтому мы делаем вывод, что разность между показателями качества наивного байесовского алгоритма и решающих деревьев значима согласно критерию знаковых рангов Уилкоксона (как было и в случае парного t -критерия в примере 12.6).

В критерии Уилкоксона предполагается, что большие разности показателей качества лучше, чем меньшие, но никаких других предположений об их созмеримости не делается. Иными словами, разности рассматриваются как порядковые, а не вещественновзначимые величины. Кроме того, не предполагается, что эти разности имеют нормальное распределение¹, а это среди прочего означает, что критерий не слишком чувствителен к выбросам.

Если мы хотим сравнить k алгоритмов на n наборах данных, то должны использовать специализированные критерии значимости, чтобы избежать снижения доверительного уровня при каждом дополнительном попарном сравнении алгоритмов. *Критерий Фридмана* разработан именно с этой целью. Как и критерий Уилкоксона, он основан на ранжированных, а не абсолютных значениях

¹ В математической статистике говорят, что этот критерий «непараметрический» – в отличие от параметрических критерии, например t -критерия, в которых предполагается определенное распределение. В общем случае параметрические критерии более эффективны, если гипотетическое распределение хорошо описывает ситуацию, но в противном случае могут вводить в заблуждение.

показателей и потому не делает никаких предположений относительно распределения результатов их измерений¹. Идея заключается в том, чтобы ранжировать показатели всех k алгоритмов для каждого набора данных – от наилучшего (ранг 1) до наихудшего (ранг k). Обозначим R_{ij} ранг j -го алгоритма на i -м наборе данных, и пусть $R_j = (\sum_i R_{ij})/n$ – средний ранг j -го алгоритма. Если верна нулевая гипотеза, согласно которой качество всех алгоритмов одинаково, то средние ранги R_j должны быть равны. Чтобы проверить это, мы вычисляем следующие величины:

- 1) средний ранг $\bar{R} = \frac{1}{nk} \sum_{ij} R_{ij} = \frac{k+1}{2}$;
- 2) сумму квадратов разностей $n \sum_j (R_j - \bar{R})^2$;
- 3) сумму квадратов разностей $\frac{1}{n(k-1)} \sum_{ij} (R_{ij} - \bar{R})^2$.

Здесь наблюдается аналогия с кластеризацией в том смысле, что вторая величина измеряет разброс между «центроидами» рангов – мы хотим, чтобы он был велик, а третья – разброс между всеми рангами. Статистика Фридмана – это отношение первой величины ко второй.

Пример 12.8 (критерий Фридмана). Мы будем пользоваться данными из примера 12.4, предполагая, что они получены на разных наборах данных, а не путем перекрестной проверки. В следующей таблице в скобках приведены ранги.

Набор данных	Наивный байесовский	Решающее дерево	Ближайший сосед
1	0.6809 (3)	0.7524 (1)	0.7164 (2)
2	0.7017 (3)	0.8964 (1)	0.8883 (2)
3	0.7012 (2)	0.6803 (3)	0.8410 (1)
4	0.6913 (2)	0.9102 (1)	0.6825 (3)
5	0.6333 (3)	0.7758 (1)	0.7599 (2)
6	0.6415 (3)	0.8154 (2)	0.8479 (1)
7	0.7216 (1)	0.6224 (3)	0.7012 (2)
8	0.7214 (2)	0.7585 (1)	0.4959 (3)
9	0.6578 (3)	0.9380 (1)	0.9279 (2)
10	0.7865 (1)	0.7524 (2)	0.7455 (3)
avg rank	2.3	1.6	2.1

Имеем $\bar{R} = 2$, $n \sum_j (R_j - \bar{R})^2 = 2.6$ и $\frac{1}{n(k-1)} \sum_{ij} (R_{ij} - \bar{R})^2 = 1$, поэтому статистика Фридмана равна 2.6. Критическое значение для $k = 3$, $n = 10$ при уровне $\alpha = 0.05$ равно 7.8, поэтому мы не можем отвергнуть нулевую гипотезу, согласно которой качество всех алгоритмов одинаково. Но если бы средние ранги были равны 2.7, 1.3 и 2.0, то при таком уровне значимости нулевую гипотезу следовало бы отвергнуть.

¹ Хорошо известная параметрическая альтернатива критерию Фридмана – *дисперсионный анализ* (ANOVA).

Критерий Фридмана говорит, свидетельствуют ли средние ранги в целом о значимых различиях, но необходим дальнейший анализ на уровне попарных сравнений. Если критерий Фридмана показывает значимость, то далее применяется *апостериорный критерий*. Его идея заключается в том, чтобы вычислить *критическую разность* (*KP*), с которой сравнивается разность между средними рангами двух алгоритмов. В *критерии Немени* критическая разность вычисляется следующим образом:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}}, \quad (12.1)$$

где q_α зависит от уровня значимости α и от k : при $\alpha = 0.05$ и $k = 3$ эта величина равна 2.343, следовательно, в нашем примере критическая разность равна 1.047. Если средние ранги равны 2.7, 1.3 и 2.0, то только разность между показателями первого и второго алгоритмов превосходит критическую. На рис. 12.1 сверху показано полезное наглядное представление результатов апостериорного критерия Немени.

Вариант критерия Немени – *критерий Бонферрони-Данна* – применим, когда сравнения производятся только с контрольным алгоритмом. Критическая разность вычисляется почти так же, только q_α модифицируется, чтобы учесть тот факт, что число попарных сравнений равно $k - 1$, а не $k(k - 1)/2$. Например, для $\alpha = 0.05$ и $k = 3$ имеем $q_\alpha = 2.241$, то есть немного меньше, чем в случае критерия Немени, и, значит, критическая разность оказывается более жесткой. На рис. 12.1 снизу показано графическое представление апостериорного критерия Бонферрони-Данна.

12.4 Эксперименты в машинном обучении: итоги и литература для дальнейшего чтения

В этой главе мы обсудили, как воспользоваться данными для ответов на вопросы о качестве моделей и алгоритмов обучения. «Экспериментатору» необходимо рассмотреть три вопроса: (i) что измерять, (ii) как измерять и (iii) как интерпретировать результаты измерения. Отличным источником – по крайней мере, по двум последним вопросам – является работа Japkowicz, Shah (2011).

- ☞ Для того чтобы решить, что измерять, мы сначала должны четко сформулировать цель эксперимента. Необходимо также учесть рабочий контекст: те аспекты качества, которые могут измениться при использовании модели. Например, рабочий контекст может определяться распределением по классам, но у нас может не быть априорных знаний о том, какие распределения более, а какие – менее вероятны. В примере 12.1 было показано, что в таком случае средняя полнота может оказаться более подходящим

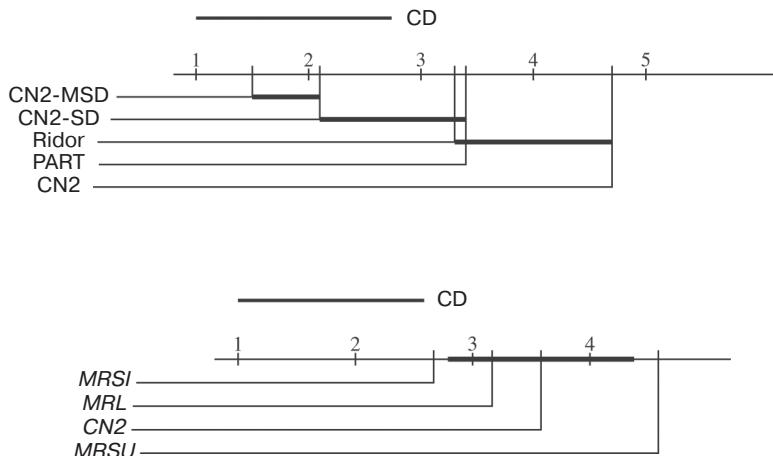


Рис. 12.1. (Сверху) Диаграмма критических разностей для попарного критерия Немени. На вещественной оси отложены средние ранги каждого алгоритма. Критическая разность изображена полосой над рисунком, а любая группа алгоритмов с соседними рангами – такая, что алгоритмы, находящиеся на разных концах, отстоят друг от друга на величину, меньшую критической разности, – соединена жирной горизонтальной линией. Эта диаграмма показывает, например, что качество алгоритма с наибольшим рангом существенно лучше, чем у трех нижних. **(Снизу)** Диаграмма критических разностей для критерия Бон-Феррони-Данна с контрольным алгоритмом CN2. Теперь критические разности нарисованы симметрично относительно среднего ранга контрольного алгоритма. Качество алгоритма с наибольшим рангом существенно лучше, чем у контрольного, а качество алгоритма с наименьшим рангом – существенно хуже. (Рисунок приведен с любезного разрешения Тарека Абудавуда (Tarek Abudawood (2011))

показателем качества, даже если цель эксперимента – верность. Мы также рассмотрели различие между анализом точность–полнота, при котором игнорируются истинно отрицательные результаты, и анализом частоты истинно и ложноположительных результатов, при котором они учитываются; более полно эта тема освещена в работе Davis, Goadrich (2006). Связь между верностью как целью эксперимента и AUC как показателем качества исследуется в работе Hernandez-Orallo *et al.* (2011).

- ☞ Решив, что измерять, мы должны определить порядок измерений. Чаще всего применяется процедура k -группой перекрестной проверки, при которой набор данных разбивается на k групп и последовательно производится обучение на $k - 1$ группах и тестирование на оставшейся. Исключительно важно, чтобы не было информационной утечки между обучающими данными, используемыми для обучения модели, и тестовыми данными для оценки ее качества. Типичная ошибка – использовать перекрестную проверку для нахождения оптимальных значений одного или нескольких параметров алгоритма обучения, скажем параметра сложности метода опор-

ных векторов. Это методологически неправильно, потому что настройка параметров должна производиться в ходе процесса обучения, без доступа к тестовым данным. Методологически верное решение – применить *внутреннюю перекрестную проверку*, зарезервировав контрольную группу в каждом раунде перекрестной проверки исключительно для настройки параметров. Результаты экспериментальных исследований перекрестной проверки приведены в работах Dietterich (1998) и Bouckaert, Frank (2004): в первой рекомендуется пятикратная проверка с двумя группами, во второй – десятикратная с десятью группами. Можно нарисовать кривые РХП перекрестной проверки, потому что каждый объект встречается в тестовой группе ровно один раз, так что есть возможность собрать оценки по всем тестовым группам. В работе Fawcett (2006) рассматриваются альтернативы, в том числе горизонтальное и вертикальное усреднения.

- ☞ Что касается интерпретации результатов эксперимента, то мы рассмотрели доверительные интервалы и критерии значимости. У доверительных интервалов имеется четкая статистическая интерпретация: они количественно выражают вероятность попадания результата измерения в определенный интервал в предположении определенного истинного значения. Критерии значимости обобщают эту идею на рассуждение об определенной нулевой гипотезе, например: «у этих алгоритмов обучения одинаковое качество на этих наборах данных». В зависимости от порядка измерений применяются те или иные критерии значимости: *t*-критерий можно использовать для оценки двух алгоритмов обучения на двух наборах данных; критерий знаковых рангов Уилкоксона – для сравнения двух алгоритмов на нескольких наборах данных; критерий Фридмана (или дисперсионный анализ) – для сравнения нескольких алгоритмов на нескольких наборах данных. Прекрасное обсуждение этих и родственных критериев имеется в работе Demšar (2006).
- ☞ Следует отметить, что на тему использования критериев значимости в машинном обучении и шире – машинного обучения как экспериментальной науки – ведутся оживленные дискуссии. Важность экспериментов в машинном обучении подчеркнута уже Пэтром Лэнгли в двух оказавших заметное влияние работах (Langley, 1988; Kibler, Langley, 1988); однако впоследствии он подверг критике экспериментальные методики в машинном обучении, ставшие, по его мнению, недостаточно гибкими (Langley, 2011). Из других авторов, критикующих современное положение вещей в этой области, отметим Драммонда (Drummond 2006) и Демшара (Demšar 2008).



Эпилог: что дальше?

Вот мы и подошли к концу нашего путешествия, посвященного «извлечению смысла из данных». Мы видели, как машинное обучение позволяет строить по признакам модели для решения различных задач, касающихся данных. Мы видели, что модели бывают прогностическими и дескриптивными; что обучение может быть с учителем и без него; что есть модели логические, геометрические и вероятностные, а также ансамбли таких моделей. Я вооружил вас базовыми знаниями, необходимыми для чтения литературы, и теперь перед вами открыт целый мир, ожидающий исследования. Поэтому будет уместно с моей стороны упомянуть о некоторых областях, которые вы, возможно, захотите изучить далее.

В этой книге мы предполагали, что данные поступают в виде, пригодном для решаемой задачи. Например, если задача состоит в том, чтобы классифицировать почту, то кто-то любезно подготовил предварительно размеченные сообщения, на которых можно обучить классификатор. Для таких задач, как оценивание вероятностей классов, я ввел пространство выходов (для модели), отличное от пространства меток (для данных), поскольку выходы модели (оценки вероятностей классов) непосредственно в данных не наблюдаются и должны реконструироваться. Существует область, в которой различие между данными и выходом модели выражено гораздо отчетливее, – *обучение с подкреплением*. Представьте, что ваша задача – научить, как стать хорошим шахматистом. Эту задачу можно рассматривать как задачу классификации, но тогда необходим учитель, который будет оценивать каждый ход. На практике обучаемый время от времени получает поощрение или наказание, например выигрывает партию или теряет одну из своих фигур. Проблема в том, чтобы назначить отдельным ходам такую положительную или отрицательную оценку, которая вела бы к означенным поощрениям и наказаниям. Обучение с подкреплением – принципиальный способ обучить тактике принятия решений о том, какое действие предпринять в данной ситуации или состоянии. Сейчас это одна из наиболее активно развивающихся ветвей машинного обучения. Стандартным учебником является книга Sutton, Barto (1998), но не составит труда найти более современные труды конференций или статьи в специальных журналах.

Существует много других задач, для решения которых необходимо ослабить некоторые предположения. Например, в многоклассовой классификации мы предполагаем, что классы являются взаимно исключающими. А в задаче *многометочной классификации* это предположение опускается, так что один объект может быть помечен произвольным числом меток. Эта задача естественно возникает, например, при разметке сетевых материалов, скажем, статей в блогах. Зависимость между метками – дополнительный источник информации; например, знание о применимости метки «машинаное обучение» делает применимость метки «обучение с подкреплением» более вероятной. Цель многометочного обуче-

ния – воспользоваться этой информацией для того, чтобы вывести зависимость между метками, а также построить отображение между признаками и индивидуальными метками. Из работ в этой области упомяну, например, Tsoumakas *et al.* (2012). Сюда же примыкает *обучение предпочтений* (preference learning), цель которого – вывести зависящие от объектов предпочтения между метками классов (Furnkranz, Hullermeier, 2010). Следуя далее по пути усложнения выходов модели, мы вступаем в область *структурного предсказания выхода* (Bakir *et al.*, 2007). Если же вернуться к многометочному обучению, то можно заметить, что, несмотря на то что каждая метка определяет отдельную задачу бинарной классификации, наша цель – избежать обучения не связанных между собой моделей для каждой такой задачи. По существу, это частный случай так называемого *многозадачного обучения*. Например, задачей может быть предсказание одной вещественнозначной целевой переменной на одном и том же пространстве объектов, а цель обучаемого – скажем, исследовать корреляции между этими переменными. С этим тесно связан *перенос обучения* (transfer learning) – дисциплина, в которой изучается перенос моделей между задачами. Обе эти темы обсуждаются в работе Silver, Bennett (2008).

Еще одно предположение, заслуживающее пристального анализа, – доступность данных в виде единого пакета. В *оперативном обучении*, которое также называется инкрементным, модель необходимо обновлять при поступлении каждого нового элемента данных. Одно из применений этой идеи – *предсказание последовательности* (Cesa-Bianchi, Lugosi, 2006). По мере распространения датчиков такая постановка задачи приобретает все большую важность, о чем свидетельствует стремительно развивающееся направление *обучения по потокам данных* (Gama, Gaber, 2007). Иногда удобно наделить обучаемого более активной ролью в сборе данных, например разрешить ему запрашивать примеры, которые должны быть размечены учителем. Это предмет *активного обучения* (Settles, 2011).

В конечном итоге машинное обучение является – и, по всей видимости, останется – областью на стыке двух направлений исследования. С одной стороны, общепризнано, что способность к обучению и самообучению – необходимое условие для любой формы машинного интеллекта. Та часть машинного обучения, которая посвящена этой тематике, называется *глубинным обучением* и ставит своей целью работу с иерархиями автономно конструируемых признаков (Bengio, 2009). С другой стороны, машинное обучение – незаменимый инструмент борьбы с затоплением данными. Построение моделей машинного обучения – важнейший шаг процесса добычи данных, с которым связаны свои специфические проблемы, например подходы к работе с «большими данными» и платформами облачных вычислений. Надеюсь, что эта книга пробудила в вас интерес к этим захватывающим исследованиям.



Что нужно запомнить

Машинным обучением называется систематическое обучение алгоритмов и систем, в результате которого их знания или качество работы возрастают по мере накопления опыта.	16
Задачи решаются с помощью моделей, а проблемы обучения – алгоритмами обучения, которые порождают модели.	23
Предметом машинного обучения является использование нужных признаков для построения моделей, подходящих для решения правильно поставленных задач.	23
Модели обеспечивают разнообразие предмета машинного обучения, тогда как задачи и признаки придают ему единство.	25
Пользуйтесь правдоподобием, если желательно игнорировать априорное распределение или предположить, что оно равномерно, и апостериорными вероятностями – в противном случае.	40
Все следует делать настолько просто, насколько возможно, но не проще.	42
На графике покрытия классификаторы с одинаковой верностью соединены отрезками прямых с угловым коэффициентом 1.	73
На нормированном графике покрытия отрезки прямых с угловым коэффициентом 1 соединяют классификаторы с одинаковой средней полнотой.	74
Площадь под кривой РХП равна верности ранжирования.	82
В случае группирующей модели количество отрезков в кривой РХП совпадает с количеством сегментов пространства объектов в модели, а в случае ранжирующей модели имеется по одному отрезку для каждого примера в наборе данных.	83
Уменьшая уточнение модели, мы иногда можем достичь лучшего качества ранжирования.	83
Вогнутости кривых РХП можно устраниТЬ, объединив отрезки с одинаковыми оценками.	91
Чтобы избежать переобучения, количество параметров, оцениваемых на основе выборки данных, должно быть значительно меньше объема имеющейся выборки.	106
При дескриптивном обучении задача и проблема обучения совпадают.	109
Наименьшее обобщение – это самое консервативное обобщение, которое можно вывести из данных.	120
Каждый концепт между наименее общим и одним из наиболее общих также является допустимой гипотезой.	124
Путь вверх в пространстве гипотез соответствует кривой покрытия.	127
Решающие деревья строго более выразительны, чем конъюнктивные концепты.	144

Один из способов избежать переобучения и способствовать обучению состоит в том, чтобы сознательно выбирать ограничительный язык гипотез.	144
Ранжирование, полученное из эмпирических вероятностей в листьях решающего дерева, дает выпуклую кривую РХП на обучающих данных.	151
Энтропия и индекс Джини чувствительны к флуктуациям распределения по классам, корень из индекса Джини – нет.	160
Списки правил подобны решающим деревьям в том смысле, что эмпирические вероятности, ассоциированные с каждым правилом, порождают выпуклые кривые РХП и покрытия на обучающих данных.	178
$(\mathbf{X}^T \mathbf{X})^{-1}$ играет роль преобразования, которое устраняет корреляцию признаков, а также центрирует и нормирует их.	213
Предположение о некоррелированности признаков, по существу, позволяет разложить задачу многомерной регрессии на d одномерных задач.	215
Общий способ построения линейного классификатора с решающей границей $\mathbf{w} \cdot \mathbf{x} = t$ заключается в том, чтобы конструировать \mathbf{w} в виде $\mathbf{M}^{-1}(n^\oplus \boldsymbol{\mu}^\oplus - n^\ominus \boldsymbol{\mu}^\ominus)$	218
В двойственной, основанной на объектах форме линейной классификации мы ищем веса объектов α_i , а не веса признаков w_j .	221
Классификатор минимальной сложности с мягким зазором сводит классы к их средним – почти так же, как в случае базового линейного классификатора.	230
С метрической точки зрения, базовый линейный классификатор можно интерпретировать как построение эталонов, минимизирующих сумму квадратов расстояний в пределах каждого класса, и последующее применение решающего правила ближайшего эталона.	250
Вероятности необязательно интерпретировать как оценки относительных частот, они могут нести и более общий смысл: степень доверия (возможно, субъективная).	275
Для некоррелированных гауссовых признаков с единичной дисперсией базовый линейный классификатор является оптимальным по Байесу.	282
Взятый со знаком минус логарифм гауссова правдоподобия можно интерпретировать как квадрат расстояния.	282
Хороший вероятностный подход к проблеме машинного обучения позволяет достичь баланса между солидным теоретическим основанием и прагматизмом, необходимым для получения рабочего решения.	284
Часто забывают о том, что для таких неоткалиброванных оценок вероятностей, какие порождаются наивным байесовским классификатором, решающие правила ML и MAP становятся неадекватными.	289
Древовидные модели игнорируют шкалу количественных признаков, трактуя их как порядковые.	316

Подгонку данных к фиксированной линейной решающей границе в пространстве логарифмических шансов с помощью калибровки признаков можно интерпретировать как обучение наивной байесовской модели.	329
У моделей с низким смещением обычно высокая дисперсия, и наоборот.	351
Баггинг – прежде всего техника уменьшения дисперсии, тогда как усиление – преимущественно метод уменьшения смещения.	351
Эксперименты в машинном обучении ставят вопросы о моделях, на которые мы пытаемся получить ответы путем измерения данных.	355
Комбинация точности и полноты, а значит, и F-мера, нечувствительна к количеству истинно отрицательных результатов.	358
Доверительные интервалы – это утверждения об оценках, а не об истинном значении показателя качества.	364

Библиография

- Abudawood, T. (2011).** Multi-class subgroup discovery: Heuristics, algorithms and predictiveness. Ph.D. thesis, University of Bristol, Department of Computer Science, Faculty of Engineering.
- Abudawood, T., Flach, P. A. (2009).** Evaluation measures for multi-class subgroup discovery. In W. L. Buntine, M. Grobelnik, D. Mladenić and J. Shawe-Taylor (eds.), *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2009), Part I, LNCS*, volume 5781, pp. 35–50. Springer.
- Agrawal, R., Imielinski, T., Swami, A. N. (1993).** Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia (eds.), *Proceedings of the ACM International Conference on Management of Data (SIGMOD 1993)*, pp. 207–216. ACM Press.
- Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A. I. (1996).** Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pp. 307–328. AAAI/MIT Press.
- Allwein, E. L., Schapire, R.E., Singer, Y. (2000).** Reducing multiclass to binary: A unifying approach for margin classifiers. In P. Langley (ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 9–16. Morgan Kaufmann.
- Amit, Y., Geman, D. (1997).** Shape quantization and recognition with randomized trees. *Neural Computation* 9(7): 1545–1588.
- Angluin, D., Frazier, M., Pitt, L. (1992).** Learning conjunctions of Horn clauses. *Machine Learning* 9: 147–164.
- Bakir, G., Hofmann, T., Scholkopf, B., Smola, A. J., Taskar, B., Vishwanathan, S. V. N. (2007).** *Predicting Structured Data*. MIT Press.
- Banerji, R. B. (1980).** *Artificial Intelligence: A Theoretical Approach*. Elsevier Science.
- Bengio, Y. (2009).** Learning deep architectures for AI. *Foundations and Trends in Machine Learning* 2(1): 1–127.
- Best, M. J., Chakravarti, N. (1990).** Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming* 47(1): 425–439.
- Blockeel, H. (2010a).** Hypothesis language. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 507–511. Springer.
- Blockeel, H. (2010b).** Hypothesis space. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 511–513. Springer.
- Blockeel, H., De Raedt, L., Ramon, J. (1998).** Top-down induction of clustering trees. In J. W. Shavlik (ed.), *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, pp. 55–63. Morgan Kaufmann.
- Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M. K. (1989).** Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM* 36(4): 929–965.

- Boser, B. E., Guyon, I., Vapnik, V. (1992).** A training algorithm for optimal margin classifiers. In *Proceedings of the International Conference on Computational Learning Theory (COLT 1992)*, pp. 144–152.
- Bouckaert, R., Frank, E. (2004).** Evaluating the replicability of significance tests for comparing learning algorithms. In H. Dai, R. Srikant and C. Zhang (eds.), *Advances in Knowledge Discovery and Data Mining, LNCS*, volume 3056, pp. 3–12. Springer.
- Boulle, M. (2004).** Khiops: A statistical discretizationmethod of continuous attributes. *Machine Learning* 55(1): 53–69.
- Boulle, M. (2006).** MODL: A Bayes optimal discretization method for continuous attributes. *Machine Learning* 65(1): 131–165.
- Bourke, C., Deng, K., Scott, S. D., Schapire, R. E., Vinodchandran, N. V. (2008).** On reoptimizing multi-class classifiers. *Machine Learning* 71 (2–3): 219–242.
- Brazdil, P., Giraud-Carrier, C. G., Soares, C., Vilalta, R. (2009).** *Metalearning – Applications to DataMining*. Springer.
- Brazdil, P., Vilalta, R., Giraud-Carrier, C. G., Soares, C. (2010).** Metalearning. In C. Sammut and G. I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 662–666. Springer.
- Breiman, L. (1996a).** Bagging predictors. *Machine Learning* 24 (2): 123–140.
- Breiman, L. (1996b).** Stacked regressions. *Machine Learning* 24 (1): 49–64.
- Breiman, L. (2001).** Random forests. *Machine Learning* 45 (1): 5–32.
- Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. (1984).** *Classification and Regression Trees*. Wadsworth.
- Brier, G. W. (1950).** Verification of forecasts expressed in terms of probability. *Monthly Weather Review* 78 (1): 1–3.
- Brown, G. (2010).** Ensemble learning. In C. Sammut and G. I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 312–320. Springer.
- Bruner, J. S., Goodnow, J. J., Austin, G. A. (1956).** *A Study of Thinking*. Science Editions. 2nd edn 1986.
- Cesa-Bianchi, N., Lugosi, G. (2006).** *Prediction, Learning, and Games*. Cambridge University Press.
- Cestnik, B. (1990).** Estimating probabilities: A crucial task in machine learning. In *Proceedings of the European Conference on Artificial Intelligence (ECAI 1990)*, pp. 147–149.
- Clark, P., Boswell, R. (1991).** Rule induction with CN2: Some recent improvements. In Y. Kodratoff (ed.), *Proceedings of the European Working Session on Learning (EWSL 1991), LNCS*, volume 482, pp. 151–163. Springer.
- Clark, P., Niblett, T. (1989).** The CN2 induction algorithm. *Machine Learning* 3: 261–283.
- Cohen, W. W. (1995).** Fast effective rule induction. In A. Prieditis and S. J. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)*, pp. 115–123. Morgan Kaufmann.
- Cohen, W. W., Singer, Y. (1999).** A simple, fast, and effictive rule learner. In J. Hendler and D. Subramanian (eds.), *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI 1999)*, pp. 335–342. AAAI Press / MIT Press.

- Cohn, D. (2010).** Active learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 10–14. Springer.
- Cortes, C., Vapnik, V. (1995).** Support-vector networks. *Machine Learning* 20 (3): 273–297.
- Cover, T., Hart, P. (1967).** Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13 (1): 21–27.
- Cristianini, N., Shawe-Taylor, J. (2000).** *An Introduction to Support Vector Machines*. Cambridge University Press.
- Dasgupta, S. (2010).** Active learning theory. In C. Sammut and G. I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 14–19. Springer.
- Davis, J., Goadrich, M. (2006).** The relationship between precision-recall and ROC curves. In W. W. Cohen and A. Moore (eds.), *Proceedings of the Twenty-Third International Conference on Machine Learning (ICML 2006)*, pp. 233–240. ACM Press.
- De Raedt, L. (1997).** Logical settings for concept-learning. *Artificial Intelligence* 95 (1): 187–201.
- De Raedt, L. (2008).** *Logical and Relational Learning*. Springer.
- De Raedt, L. (2010).** Logic of generality. In C. Sammut and G. I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 624–631. Springer.
- De Raedt, L., Kersting, K. (2010).** Statistical relational learning. In C. Sammut and G. I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 916–924. Springer.
- Dempster, A. P., Laird, N. M., Rubin, D. B. (1977).** Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)* pp. 1–38.
- Demšar, J. (2008).** On the appropriateness of statistical tests in machine learning. In *Proceedings of the ICML08 Workshop on Evaluation Methods for Machine Learning*.
- Demšar, J. (2006).** Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7: 1–30.
- Dietterich, T. G. (1998).** Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation* 10 (7): 1895–1923.
- Dietterich, T. G., Bakiri, G. (1995).** Solving multiclass learning problems via error correcting output codes. *Journal of Artificial Intelligence Research* 2: 263–286.
- Dietterich, T. G., Kearns, M. J., Mansour, Y. (1996).** Applying the weak learning framework to understand and improve c4.5. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 96–104.
- Ding, C. H. Q., He, X. (2004).** K-means clustering via principal component analysis. In C. E. Brodley (ed.), *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004)*. ACM Press.
- Domingos, P., Pazzani, M. (1997).** On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning* 29 (2): 103–130.
- Donoho, S. K., Rendell, L. A. (1995).** Rerrepresenting and restructuring domain theories: A constructive induction approach. *Journal of Artificial Intelligence Research* 2: 411–446.

- Drummond, C. (2006).** Machine learning as an experimental science (revisited). In *Proceedings of the AAAI'06 Workshop on Evaluation Methods for Machine Learning*.
- Drummond, C., Holte, R. C. (2000).** Exploiting the cost (in)sensitivity of decision tree splitting criteria. In P. Langley (ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 239–246. Morgan Kaufmann.
- Egan, J. P. (1975).** *Signal Detection Theory and ROC Analysis*. Academic Press.
- Fawcett, T. (2006).** An introduction to ROC analysis. *Pattern Recognition Letters* 27 (8): 861–874.
- Fawcett, T., Niculescu-Mizil, A. (2007).** PAV and the ROC convex hull. *Machine Learning* 68 (1): 97–106.
- Fayyad, U. M., Irani, K. B. (1993).** Multi-interval discretization of continuous valued attributes for classification learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 1993)*, pp. 1022–1029.
- Ferri, C., Flach, P. A., Hernandez-Orallo, J. (2002).** Learning decision trees using the area under the ROC curve. In C. Sammut and A.G. Hoffmann (eds.), *Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002)*, pp. 139–146. Morgan Kaufmann.
- Ferri, C., Flach, P. A., Hernandez-Orallo, J. (2003).** Improving the AUC of probabilistic estimation trees. In N. Lavrač, D. Gamberger, L. Todorovski and H. Blockeel (eds.), *Proceedings of the European Conference on Machine Learning (ECML 2003), LNCS*, volume 2837, pp. 121–132. Springer.
- Fix, E., Hodges, J. L. (1951).** Discriminatory analysis. Nonparametric discrimination: Consistency properties. Technical report, USAF School of Aviation Medicine, Texas: Randolph Field. Report Number 4, Project Number 21-49-004.
- Flach, P. A. (1994).** *Simply Logical – Intelligent Reasoning by Example*. Wiley.
- Flach, P. A. (2003).** The geometry of ROC space: Understanding machine learning metrics through ROC isometrics. In T. Fawcett and N. Mishra (eds.), *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pp. 194–201. AAAI Press.
- Flach, P. A. (2010a).** First-order logic. In C. Sammut and G. I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 410–415. Springer.
- Flach, P. A. (2010b).** ROC analysis. In C. Sammut and G. I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 869–875. Springer.
- Flach, P. A., Lachiche, N. (2001).** Confirmation-guided discovery of first-order rules with Tertius. *Machine Learning* 42 (1/2): 61–95.
- Flach, P. A., Matsubara, E. T. (2007).** A simple lexicographic ranker and probability estimator. In J. N. Kok, J. Koronacki, R.L. de Mantaras, S. Matwin, D. Mladenic and A. Skowron (eds.), *Proceedings of the Eighteenth European Conference on Machine Learning (ECML 2007), LNCS*, volume 4701, pp. 575–582. Springer.
- Freund, Y., Iyer, R. D., Schapire, R. E., Singer, Y. (2003).** An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research* 4: 933–969.

- Freund, Y., Schapire, R. E. (1997).** A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* 55 (1): 119–139.
- Furnkranz, J. (1999).** Separate-and-conquer rule learning. *Artificial Intelligence Review* 13 (1): 3–54.
- Furnkranz, J. (2010).** Rule learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 875–879. Springer.
- Furnkranz, J., Flach, P. A. (2003).** An analysis of rule evaluation metrics. In T. Fawcett and N. Mishra (eds.), *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pp. 202–209. AAAI Press.
- Furnkranz, J., Flach, P. A. (2005).** ROC 'n' Rule learning – towards a better understanding of covering algorithms. *Machine Learning* 58 (1): 39–77.
- Furnkranz, J., Gamberger, D., Lavrač, N. (2012).** *Foundations of Rule Learning*. Springer.
- Furnkranz, J., Hullermeier, E. (eds.) (2010).** *Preference Learning*. Springer.
- Furnkranz, J., Widmer, G. (1994).** Incremental reduced error pruning. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML 1994)*, pp. 70–77.
- Gama, J., Gaber, M. M. (eds.) (2007).** *Learning from Data Streams: Processing Techniques in Sensor Networks*. Springer.
- Ganter, B., Wille, R. (1999).** *Formal Concept Analysis: Mathematical Foundations*. Springer.
- Garriga, G. C., Kralj, P., Lavrač, N. (2008).** Closed sets for labeled data. *Journal of Machine Learning Research* 9: 559–580.
- Gartner, T. (2009).** *Kernels for Structured Data*. World Scientific.
- Grunwald, P. D. (2007).** *The Minimum Description Length Principle*. MIT Press.
- Guyon, I., Elisseeff, A. (2003).** An introduction to variable and feature selection. *Journal of Machine Learning Research* 3: 1157–1182.
- Hall, M. A. (1999).** Correlation-based feature selection for machine learning. Ph. D. thesis, University of Waikato.
- Han, J., Cheng, H., Xin, D., Yan, X. (2007).** Frequent pattern mining: Current status and future directions. *Data Mining and Knowledge Discovery* 15 (1): 55–86.
- Hand, D. J., Till, R. J. (2001).** A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning* 45 (2): 171–186.
- Haussler, D. (1988).** Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence* 36 (2): 177–221.
- Hernandez-Orallo, J., Flach, P. A., Ferri, C. (2011).** Threshold choice methods: The missing link. Available online at <http://arxiv.org/abs/1112.2640>.
- Ho, T. K. (1995).** Random decision forests. In *Proceedings of the International Conference on Document Analysis and Recognition*, p. 278. IEEE Computer Society, Los Alamitos, CA, USA.
- Hoerl, A. E., Kennard, R. W. (1970).** Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* pp. 55–67.

- Hofmann, T. (1999).** Probabilistic latent semantic indexing. In *Proceedings of the Twenty-Second Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR 1999)*, pp. 50–57. ACM Press.
- Hunt, E. B., Marin, J., Stone, P. J. (1966).** *Experiments in Induction*. Academic Press.
- Jain, A. K., Murty, M. N., Flynn, P. J. (1999).** Data clustering: A review. *ACM Computing Surveys* 31 (3): 264–323.
- Japkowicz, N., Shah, M. (2011).** *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press.
- Jebara, T. (2004).** *Machine Learning: Discriminative and Generative*. Springer.
- John, G. H., Langley, P. (1995).** Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI 1995)*, pp. 338–345. Morgan Kaufmann.
- Kaufman, L., Rousseeuw, P. J. (1990).** *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley.
- Kearns, M. J., Valiant, L. G. (1989).** Cryptographic limitations on learning Boolean formulae and finite automata. In D. S. Johnson (ed.), *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing (STOC 1989)*, pp. 433–444. ACM Press.
- Kearns, M. J., Valiant, L. G. (1994).** Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM* 41 (1): 67–95.
- Kerber, R. (1992).** Chimerge: Discretization of numeric attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI 1992)*, pp. 123–128. AAAI Press.
- Kibler, D. F., Langley, P. (1988).** Machine learning as an experimental science. In *Proceedings of the European Working Session on Learning (EWSL 1988)*, pp. 81–92.
- King, R. D., Srinivasan, A., Dehaspe, L. (2001).** Warmr: A data mining tool for chemical data. *Journal of Computer-Aided Molecular Design* 15 (2): 173–181.
- Kira, K., Rendell, L. A. (1992).** The feature selection problem: Traditional methods and a new algorithm. In W.R. Swartout (ed.), *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI 1992)*, pp. 129–134. AAAI Press / MIT Press.
- Klosgen, W. (1996).** Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pp. 249–271. MIT Press.
- Kohavi, R., John, G. H. (1997).** Wrappers for feature subset selection. *Artificial Intelligence* 97 (1–2): 273–324.
- Koren, Y., Bell, R., Volinsky, C. (2009).** Matrix factorization techniques for recommender systems. *IEEE Computer* 42 (8): 30–37.
- Kramer, S. (1996).** Structural regression trees. In *Proceedings of the National Conference on Artificial Intelligence (AAAI 1996)*, pp. 812–819.
- Kramer, S., Lavrač, N., Flach, P. A. (2000).** Propositionalization approaches to relational data mining. In S. Džeroski and N. Lavrač (eds.), *Relational Data Mining*, pp. 262–286. Springer.

- Krogel, M. A., Rawles, S., Zelezny, F., Flach, P. A., Lavrač, N., Wrobel, S. (2003).** Comparative evaluation of approaches to propositionalization. In T. Horvath (ed.), *Proceedings of the Thirteenth International Conference on Inductive Logic Programming (ILP 2003)*, LNCS, volume 2835, pp. 197–214. Springer.
- Kuncheva, L. I. (2004).** *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley and Sons.
- Lachiche, N. (2010).** Propositionalization. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 812–817. Springer.
- Lachiche, N., Flach, P. A. (2003).** Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. In T. Fawcett and N. Mishra (eds.), *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pp. 416–423. AAAI Press.
- Lafferty, J. D., McCallum, A., Pereira, F. C. N. (2001).** Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In C. E. Brodley and A. P. Danyluk (eds.), *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pp. 282–289. Morgan Kaufmann.
- Langley, P. (1988).** Machine learning as an experimental science. *Machine Learning* 3: 5–8.
- Langley, P. (1994).** *Elements of Machine Learning*. Morgan Kaufmann.
- Langley, P. (2011).** The changing science of machine learning. *Machine Learning* 82 (3): 275–279.
- Lavrač, N., Kavšek, B., Flach, P. A., Todorovski, L. (2004).** Subgroup discovery with CN2-SD. *Journal of Machine Learning Research* 5: 153–188.
- Lee, D. D., Seung, H. S. et al. (1999).** Learning the parts of objects by non-negative matrix factorization. *Nature* 401 (6755): 788–791.
- Leman, D., Feelders, A., Knobbe, A. J. (2008).** Exceptional model mining. In W. Daelemans, B. Goethals and K. Morik (eds.), *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2008), Part II*, LNCS, volume 5212, pp. 1–16. Springer.
- Lewis, D. (1998).** Naive Bayes at forty: The independence assumption in information retrieval. In *Proceedings of the Tenth European Conference on Machine Learning (ECML 1998)*, pp. 4–15. Springer.
- Li, W., Han, J., Pei, J. (2001).** CMAR: Accurate and efficient classification based on multiple class-association rules. In N. Cercone, T. Y. Lin and X. Wu (eds.), *Proceedings of the IEEE International Conference on Data Mining (ICDM 2001)*, pp. 369–376. IEEE Computer Society.
- Little, R. J. A., Rubin, D. B. (1987).** *Statistical Analysis with Missing Data*. Wiley.
- Liu, B., Hsu, W., Ma, Y. (1998).** Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD 1998)*, pp. 80–86. AAAI Press.
- Lloyd, J. W. (2003).** *Logic for Learning – Learning Comprehensible Theories from Structured Data*. Springer.

- Lloyd, S. (1982).** Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28 (2): 129–137.
- Mahalanobis, P. C. (1936).** On the generalised distance in statistics. *Proceedings of the National Institute of Science, India* 2 (1): 49–55.
- Mahoney, M. W., Drineas, P. (2009).** CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences* 106 (3): 697.
- McCallum, A., Nigam, K. (1998).** A comparison of event models for naive Bayes text classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, pp. 41–48.
- Michalski, R. S. (1973).** Discovering classification rules using variable-valued logic system VL1. In *Proceedings of the Third International Joint Conference on Artificial Intelligence*, pp. 162–172. Morgan Kaufmann Publishers.
- Michalski, R. S. (1975).** Synthesis of optimal and quasi-optimal variable-valued logic formulas. In *Proceedings of the 1975 International Symposium on Multiple-Valued Logic*, pp. 76–87.
- Michie, D., Spiegelhalter, D. J., Taylor, C. C. (1994).** *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.
- Miettinen, P. (2009).** Matrix decomposition methods for data mining: Computational complexity and algorithms. Ph.D. thesis, University of Helsinki.
- Minsky, M., Papert, S. (1969).** *Perceptrons: An Introduction to Computational Geometry*. MIT Press.
- Mitchell, T. M. (1977).** Version spaces: A candidate elimination approach to rule learning. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pp. 305–310. Morgan Kaufmann Publishers.
- Mitchell, T. M. (1997).** *Machine Learning*. McGraw-Hill.
- Muggleton, S. (1995).** Inverse entailment and Progol. *New Generation Computing* 13 (3–4): 245–286.
- Muggleton, S., DeRaedt, L., Poole, D., Bratko, I., Flach, P. A., Inoue, K., Srinivasan, A. (2012).** ILP turns 20 – biography and future challenges. *Machine Learning* 86 (1): 3–23.
- Muggleton, S., Feng, C. (1990).** Efficient induction of logic programs. In *Proceedings of the International Conference on Algorithmic Learning Theory (ALT 1990)*, pp. 368–381.
- Murphy, A. H., Winkler, R. L. (1984).** Probability forecasting in meteorology. *Journal of the American Statistical Association* pp. 489–500.
- Nelder, J. A., Wedderburn, R. W. M. (1972).** Generalized linear models. *Journal of the Royal Statistical Society, Series A (General)* pp. 370–384.
- Novikoff, A. B. (1962).** On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pp. 615–622. Polytechnic Institute of Brooklyn, New York.
- Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L. (1999).** Discovering frequent closed itemsets for association rules. In *Proceedings of the International Conference on Database Theory (ICDT 1999)*, pp. 398–416. Springer.

- Peng, Y., Flach, P. A., Soares, C., Brazdil, P. (2002).** Improved dataset characterisation for meta-learning. In S. Lange, K. Satoh and C.H. Smith (eds.), *Proceedings of the Fifth International Conference on Discovery Science (DS 2002)*, LNCS, volume 2534, pp. 141–152. Springer.
- Pfahringer, B., Bensusan, H., Giraud-Carrier, C. G. (2000).** Meta-learning by landmarking various learning algorithms. In P. Langley (ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 743–750. Morgan Kaufmann.
- Platt, J. C. (1998).** Using analytic QP and sparseness to speed training of support vector machines. In M. J. Kearns, S. A. Solla and D. A. Cohn (eds.), *Advances in Neural Information Processing Systems 11 (NIPS 1998)*, pp. 557–563. MIT Press.
- Plotkin, G. D. (1971).** Automatic methods of inductive inference. Ph.D. thesis, University of Edinburgh.
- Provost, F. J., Domingos, P. (2003).** Tree induction for probability-based ranking. *Machine Learning* 52 (3): 199–215.
- Provost, F. J., Fawcett, T. (2001).** Robust classification for imprecise environments. *Machine Learning* 42 (3): 203–231.
- Quinlan, J. R. (1986).** Induction of decision trees. *Machine Learning* 1 (1): 81–106.
- Quinlan, J. R. (1990).** Learning logical definitions from relations. *Machine Learning* 5: 239–266.
- Quinlan, J. R. (1993).** *C4.5: Programs for Machine Learning*. Morgan Kaufmann.
- Ragavan, H., Rendell, L. A. (1993).** Lookahead feature construction for learning hard concepts. In *Proceedings of the Tenth International Conference on Machine Learning (ICML 1993)*, pp. 252–259. Morgan Kaufmann.
- Rajnarayan, D. G., Wolpert, D. (2010).** Bias-variance trade-offs: Novel applications. In C. Sammut and G. I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 101–110. Springer.
- Rissanen, J. (1978).** Modeling by shortest data description. *Automatica* 14(5): 465–471.
- Rivest, R. L. (1987).** Learning decision lists. *Machine Learning* 2 (3): 229–246.
- Robnik-Sikonja, M., Kononenko, I. (2003).** Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning* 53 (1–2): 23–69.
- Rosenblatt, F. (1958).** The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65 (6): 386.
- Rousseeuw, P. J. (1987).** Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* 20 (0): 53–65.
- Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986).** Learning representations by back-propagating errors. *Nature* 323 (6088): 533–536.
- Schapire, R. E. (1990).** The strength of weak learnability. *Machine Learning* 5: 197–227.
- Schapire, R. E. (2003).** The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification*, pp. 149–172. Springer.

- Schapire, R. E., Freund, Y., Bartlett, P., Lee, W. S. (1998).** Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics* 26 (5): 1651–1686.
- Schapire, R. E., Singer, Y. (1999).** Improved boosting algorithms using confidence rated predictions. *Machine Learning* 37 (3): 297–336.
- Settles, B. (2011).** *Active Learning*. Morgan & Claypool.
- Shawe-Taylor, J., Cristianini, N. (2004).** *Kernel Methods for Pattern Analysis*. Cambridge University Press.
- Shotton, J., Fitzgibbon, A. W., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A. (2011).** Real-time human pose recognition in parts from single depth images. In *Proceedings of the Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, pp. 1297–1304.
- Silver, D., Bennett, K. (2008).** Guest editor's introduction: special issue on inductive transfer learning. *Machine Learning* 73 (3): 215–220.
- Solomonoff, R. J. (1964a).** A formal theory of inductive inference: Part I. *Information and Control* 7 (1): 1–22.
- Solomonoff, R. J. (1964b).** A formal theory of inductive inference: Part II. *Information and Control* 7 (2): 224–254.
- Srinivasan, A. (2007).** The Aleph manual, version 4 and above. Available online at www.cs.ox.ac.uk/activities/machlearn/Aleph/.
- Stevens, S. S. (1946).** On the theory of scales of measurement. *Science* 103 (2684): 677–680.
- Sutton, R. S., Barto, A. G. (1998).** *Reinforcement Learning: An Introduction*. MIT Press.
- Tibshirani, R. (1996).** Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B (Methodological)* pp. 267–288.
- Todorovski, L., Dzeroski, S. (2003).** Combining classifiers with meta decision trees. *Machine Learning* 50 (3): 223–249.
- Tsoumakas, G., Zhang, M. L., Zhou, Z. H. (2012).** Introduction to the special issue on learning from multi-label data. *Machine Learning* 88 (1–2): 1–4.
- Tukey, J. W. (1977).** *Exploratory Data Analysis*. Addison-Wesley.
- Valiant, L. G. (1984).** A theory of the learnable. *Communications of the ACM* 27 (11): 1134–1142.
- Vapnik, V. N., Chervonenkis, A. Y. (1971).** On uniform convergence of the frequencies of events to their probabilities. *Teoriya Veroyatnostei I Ee Primeneniya* 16 (2): 264–279.
- Vere, S. A. (1975).** Induction of concepts in the predicate calculus. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pp. 281–287.
- von Hippel, P. T. (2005).** Mean, median, and skew: Correcting a textbook rule. *Journal of Statistics Education* 13 (2).
- Wallace, C. S., Boulton, D. M. (1968).** An information measure for classification. *Computer Journal* 11 (2): 185–194.

- Webb, G. I. (1995).** Opus: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research* 3: 431–465.
- Webb, G. I., Boughton, J. R., Wang, Z. (2005).** Not so naive Bayes: Aggregating one dependence estimators. *Machine Learning* 58 (1): 5–24.
- Winston, P. H. (1970).** Learning structural descriptions from examples. Technical report, MIT Artificial Intelligence Lab. AITR-231.
- Wojtusiak, J., Michalski, R. S., Kaufman, K. A., Pietrzykowski, J. (2006).** The AQ21 natural induction program for pattern discovery: Initial version and its novel features. In *Proceedings of the Eighteenth IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2006)*, pp. 523–526.
- Wolpert, D. H. (1992).** Stacked generalization. *Neural Networks* 5 (2): 241–259.
- Zadrozny, B., Elkan, C. (2002).** Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2002)*, pp. 694–699. ACM Press.
- Zeugmann, T. (2010).** PAC learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 745–753. Springer.
- Zhou, Z. H. (2012).** *Ensemble Methods: Foundations and Algorithms*. Taylor & Francis.

Предметный указатель

- 0-норма, 245
1-норма, 245, 250
2-норма, 245, 250
 χ^2 статистика, 115, 324, 335
- AggloMerge(S, f, Q)
алгоритм 10.2, 324
- AssociationRules(D, f_0, c_0)
алгоритм 6.7, 198
- AUC, (площадь под кривой), 82
многоклассовая
пример 3.5, 102
- Bagging(D, T, \mathcal{A})
алгоритм 11.1, 344
- BestSplit-Class(D, F)
алгоритм 5.2, 150
- Boosting(D, T, \mathcal{A})
алгоритм 11.3, 347
- DKM, метод, 169
- DualPerceptron(D)
алгоритм 7.2, 221
- d-штрих, 327
- EM-алгоритм (Expectation-Maximization), 111, 300, 334
- FrequentItems(D, f_0)
алгоритм 6.6, 196
- F-мера, 113, 312
нечувствительность к истинно отрицательным результатам, 358
- GrowTree(D, F)
алгоритм 5.1, 145
- HAC(D, L)
алгоритм 8.4, 266
- Horn(Mb, Eq)
алгоритм 4.5, 133
- ILP-системы
Aleph, 204
FOIL, 204
Golen, 204
Progol, 48, 204
- Kernel-KMeans(D, K)
алгоритм 8.5, 270
- KernelPerceptron(D, κ)
алгоритм 7.4, 238
- Kinect, датчик движения, 142, 168
- KMeans(D, K)
алгоритм 8.1, 260
- KMedoids(D, K, Dis)
алгоритм 8.2, 261
- k ближайших соседей, 255
- К медоидов, 261, 321
- К средних, 37, 110, 111, 259, 271, 300, 321
проблема, 257, 259
связь с гауссовой смесевой моделью, 303
- lasso, 216
- LearnRule(D)
алгоритм 6.2, 177
- LearnRuleForClass(D, C_i)
алгоритм 6.4, 184
- LearnRuleList(D)
алгоритм 6.1, 176
- LearnRuleSet(D)
алгоритм 6.3, 184

LGG. *См.* Наименьшее обобщение
 LGG-Conj-ID(x, y)
 алгоритм 4.3, 124
 LGG-Conj(x, y)
 алгоритм 4.2, 124
 LGG-Set(D)
 алгоритм 4.1, 122

MAP. *См.* Апостериорный максимум
 MGConsistent(C, N)
 алгоритм 4.4, 129
 m-оценка, 90, 154, 160, 290

n-грамма, 334

PAC (probably approximately correct). *См.* Почти корректная
 PAM, 261. *См.* Разбиение по медоидам
 PAM(D, K, Dis)
 алгоритм 8.3, 262
 PCA. *См.* Метод главных компонент
 Perceptron(D, η)
 алгоритм 7.1, 220
 PerceptronRegression(D, T)
 алгоритм 7.3, 222
 PruneTree(T, D)
 алгоритм 5.3, 157
 p-значение, 364
 p-норма, 245

RandomForest(D, T, d)
 алгоритм 11.2, 345
 RecPart(S, f, Q)
 алгоритм 10.1, 323

SpamAssassin, 14, 75

t-распределение, 364

VC-размерность, 138
 линейного классификатора, 138

WeightedCovering(D)
 алгоритм 6.5, 194

z-оценка, 278, 325, 327

Абстракция, 317
 Агломеративное объединение, 323
 Агрегирование, в структурированных признаках, 318
 Активное обучение, 134, 372
 Алгоритм построения покрытия, 175
 Ансамбль моделей, 342
 Антиунификация, 136
 Апостериорная вероятность, 38, 273
 Апостериорный критерий, 368
 Апостериорный максимум, 40
 Апостериорный шанс, 40
 пример 1.3, 40
 Априорная вероятность, 39
 Априорный шанс, 40
 Асимметрия, 315
 пример 10.3, 315
 Ассоциативное правило, 27
 алгоритмы обучения
 Apriori, 204
 Warmr, 205
 выявление
 пример 3.12, 115
 Атрибут. *См.* Признак
 Аффинное преобразование, 207

Баггинг, 169, 343
 Базовый линейный
 классификатор, 33, 218, 250, 253,
 281, 329, 344, 345, 351, 374
 оптимальность по Байесу, 282
 триюк с ядром, 56
 Байеса правило, 39
 Бернулли
 испытание, 161, 285
 распределение, 161, 285

- многомерное, 285
Якоб, 58, 285
Биграмма, 334
Биномиальное распределение, 285
Битовый вектор, 285
Большие данные, 372
Бонферрони-Данна критерий, 368
Брайера оценка, 88
Бритва Оккама, 42
- Вапник Владимир, 138
Вектор счетчиков, 287
Верность, 31, 67, 70
 как взвешенное среднее
 пример 2.1, 69
ожидаемая
 пример 12.1, 357
 пример 12.3, 359
ранжирования
 пример 2.3, 79
- Вероятностная модель
 дискриминантная, 273
 порождающая, 273
Вероятностное пространство, 328
Взвешенная относительная
верность, 191
Взвешенное покрытие, 350
 пример 6.7, 192
Взвешивание расстояний, 256
Внутренняя дизъюнкция, 122, 175
Вогнутость, 91
Вороного диаграмма, 112, 252
Восполнение отсутствующих
элементов матрицы, 339
Вращение, 36
Выборка подпространства, 344
Выбор модели, 276
Выборочная дисперсия, 58
Выборочная ковариация, 58
Выборочная сложность, 137
- Выборочное среднее, 58
Выброс, 210, 249, 313
 пример 7.2, 210
Выпуклая, 225, 252
 кривая РХП, 91, 151
множество, 126, 194
оболочка, 92
 нижняя, 320
функция потерь, 77
Выходные коды, 97
- Гармоническое среднее, 113
Гауссиана, 277
Гауссова смесовая модель, 277, 300
 двумерная
 пример 9.3, 279
одномерная
 пример 9.2, 279
связь с методом К средних, 303
Гауссово ядро, 238
 полоса пропускания, 238
Геометрическая медиана, 249
Гиперплоскость, 33
Гистограмма, 313
Глубинное обучение, 372
Голосование один против одного
 пример 3.2, 99
Градиент, 224, 249
Грама матрица, 221, 226, 337
График покрытия, 70
График процентиелей, 313
Гребневая регрессия, 216
Группирующая модель, 65, 106
- Дедукция, 31
Декартово произведение, 64
Декодирование, 98
 на основе потерь, 100
Декодирование на основе потерь
 пример 3.3, 100

- Дендрограмма, 264
 Дерева регрессии
 пример 5.4, 163
 Дерево оценивания
 вероятностей, 154, 160, 273, 276
 Дерево правил, 187
 пример 6.5, 187
 Дерево признаков, 45, 168
 определение, 145
 полное, 45
 разметка
 пример 1.5, 45
 рост
 пример 5.2, 152
 Дескриптивная
 кластеризация, 110, 257
 Дескриптивная модель, 29
 Дециль, 313
 Джини индекс, 147
 Джини коэффициент, 147
 Дизъюнкт, 119
 Дизъюнктивная нормальная форма
 (ДНФ), 119
 Дизъюнкция, 46, 118
 Дilemma смещения-дисперсии,
 107, 350
 Дискретизация, 168
 агломеративная, 321
 агломеративное объединение
 пример 10.7, 323
 восходящая, 321
 дивизивная, 321
 нисходящая, 321
 рекурсивное разбиение, 322
 с равной частотой, 321
 с равной шириной, 321
 Дисперсия, 58, 107, 162, 209, 212,
 312, 315
 индекс Джини как, 161
 Добыча данных, 194, 372
 Добыча моделей исключений, 117
 Доверительный интервал
 пример 12.5, 363
 Доминировать, 72
 Дополнение, 190
 Евклидово расстояние, 245
 Жадный алгоритм, 146
 Жаккара коэффициент, 27
 Задание порога
 пример 10.5, 320
 Задача, 25
 Зазор
 линейного классификатора, 34
 примера, 76, 223, 351
 решающей границы, 223
 Закон больших чисел, 58
 Запрос, 318
 Иерархическая агломеративная
 кластеризация, 325
 Изолинии
 верности, 75, 84, 92, 129
 индекса Джини, 159
 корня из индекса Джини, 158
 критериев разделения, 158
 нечистоты, 172
 средней полноты, 74, 86
 точности, 181
 точности (с поправкой Лапласа), 185
 энтропии, 159
 Импликация, 118
 Индикаторная функция, 67
 Индуктивное смещение, 144
 Индукция, 31
 Интервал, 321
 Информационный
 поиск, 113, 312, 339, 358

- Исключение по одному, 361
Исправляющие ошибки выходные коды, 116
Истинно отрицательный, 68
Истинно положительный, 68
- Калибровка, 232
изотонная, 93, 234, 296, 330
логистическая, 297, 327
отображение, 93
потери на, 90
Каруша-Куна-Таккера условия, 225
Категориальное распределение, 285
Квадратичная ошибка, 88
Квантиль, 313
Квартиль, 313
Класс
дисбаланс
пример 2.4, 82
метка, 65
отношение, 76, 81, 85, 86, 154, 156, 232
Классификатор, 65
Классификатор по ближайшему соседу, 35, 253
Классификатор по методу наименьших квадратов, 219, 222
одномерный
пример 7.4, 217
Классификатор с максимальным зазором
мягкий зазор
пример 7.6, 229
пример 7.5, 226
Классификация
бинарная, 65
многоклассовая, 26, 96
Классификация текста, 19, 23, 33
Класс эквивалентности, 64
Кластеризация, 26
- агломеративная, 266, 321
дескриптивная, 29
оценивание качества
пример 3.10, 113
представления
пример 3.9, 112
прогностическая, 29
стационарные состояния
пример 8.5, 260
Кластеризующее дерево, 264
использование евклидова расстояния
пример 5.6, 167
использование матрицы расхождений
пример 5.5, 165
Ковариационная матрица, 211, 213, 215, 248, 278, 280, 281
Ковариация, 58, 209
Компонент, 277
Контрпример, 133
Концепт, 170
замкнутый, 130, 195
конъюнктивный
пример 4.1, 119
непротиворечивый, 126
Концептуальное обучение, 65
отрицательные примеры
пример 4.2, 122
Конъюнктивная нормальная форма (КНФ), 119, 130
Конъюнктивная отделимость, 128
пример 4.4, 128
Конъюнкция, 46, 118
Корреляция, 165
Косинусоидальная мера сходства, 270
Коэффициент корреляции, 58, 278, 335
Коэффициент правдоподобия, 40
Коэффициент регрессии, 209
Кривая покрытия, 79

- Критерий значимости, 364
 Критерий остановки, 176, 321
 Критерий разделения
 чувствительность к затратам
 пример 5.3, 157
 Критическая разность, 368
 Критическое значение, 366
 Куртозис, 315
 Кусочно-линейная аппроксимация, 207
 Кусочно-линейная функция
 потерь, 77, 229
- Лапласа поправка, 90, 154, 160, 181, 286, 290, 298, 327
- Латентная переменная. См. Скрытая переменная
- Латентное семантическое индексирование, 339
- Левенштейна расстояние, 246
- Линейная аппроксимация, 207
- Линейная комбинация, 206
- Линейная модель, 206
- Линейная разделимость, 218
- Линейная регрессия, 106, 164
 двумерная
 пример 7.3, 214
 одномерная
 пример 7.1, 208
- Линейная функция, 206
- Линейное преобразование, 207
- Линейный классификатор, 18, 33, 49, 53, 56, 96, 219, 276, 293, 325, 345
- VC-размерность, 138
- геометрическая интерпретация, 232
- зазор, 34
- кривая покрытия, 81
- Логистическая калибровка
 пример 7.7, 233
- общий вид, 218, 374
- пример 1, 15
- Литерал, 118
 Ллойда алгоритм, 259
 Логарифмически линейные модели, 234
 Логарифмическое правдоподобие, 282
 Логика первого порядка, 135
 Логистическая регрессия, 234, 293
 одномерная
 пример 9.6, 296
- Логистическая функция, 233
- Ложноотрицательный, 68
- Ложноположительный, 68
- Локальные переменные, 201, 318
- Лучевой поиск, 182
- Мажоритарный класс, 45, 47, 65, 69
- Максимальное правдоподобие, 40
- Манхэттенское расстояние, 245
- Маргинал, 66, 198
- Маргинальное правдоподобие, 41
 пример 1.4, 42
- Масштабирование, 36
 пропорциональное, 36
- Масштабирующая матрица, 213
- Математическое ожидание, 58, 278
- Матрица
 диагональная, 213
 обратная, 278
 ранг, 338
- Матрица неточностей, 66
- Матрица разбиения, 111
- Махалонобиса расстояние, 248, 282
- Машинное обучение
 одномерное, 65
 определение, 16
- Машинный интеллект, 372
- Медиана, 278, 311
- Медоид, 166, 249
- Межевание, 354
- Межквартильный размах, 313, 325
- Меры нечистоты, 171

- индекс Джини, 147, 148, 157, 160
как дисперсия, 161
корень из индекса Джини, 147, 158, 160, 169, 349
Миноритарный класс, 147, 148, 171
энтропия, 147, 148, 149, 157, 160, 305
Метамодель, 352
Метод главных компонент, 36, 254, 336
Метод наименьших квадратов, 208
обобщенный, 211, 284
обычный, 211
Метод опорных векторов, 34, 77, 223, 317
параметр сложности, 228
Метрика
определение, 246, 317
Мешок слов, 54
Минимальная длина описания
определение 9.1, 306
Минковского расстояние
определение 8.1, 245
ММО набор данных
иерархическая кластеризация
пример 8.6, 264
кластеризация
пример 8.4, 259
пример 1.7, 51
Многозадачное обучение, 372
Многоклассовые вероятности
пример 3.7, 104
Многоклассовые оценки, 101
повторное взвешивание
пример 3.6, 104
Многоклассовых классификаторов
качество
пример 3.1, 97
Многомерная линейная
регрессия, 288
Многомерная наивная байесовская
модель
разложение на одномерные, 44
Многомерная регрессия
разложение на одномерные
задачи, 215, 374
Многомерное нормальное
распределение, 301
Многометочная классификации, 371
Многообразие, 254
Множество, 63
дизъюнктные, 64
дополнение, 64
кардинальное число, 64
объединение, 64
пересечение, 64
разность, 64
Множители Лагранжа, 225
Мода, 278, 311
Модель, 25, 62
вероятностная, 37
геометрическая, 33
группирующая, 49
декларативная, 48
логическая, 44
одномерная, 53
параметрическая, 207
ранжирующая, 49
Модельное дерево, 164
Модель обучения, 136
Монотонность, 194, 316
Мультиномиальное распределение, 285
Мягкий зазор, 228
Наивный байесовский, 42, 45, 215, 290, 329, 334
варианты, 291
диагональная ковариационная
матрица, 292
категориальные признаки, 316

- линейность в пространстве логарифмических шансов, 288
- многоклассовые оценки, 101
- обучение
 - [пример 9.5](#), 290
- предположение, 286
- предсказание
 - [пример 9.4](#), 287
- разложение, 288, 292
- шотландский
 - классификатор, 44, 293
- Наименьшее обобщение, 120, 124, 144
- Нарушители монотонности, 91
- Начальный пример, 181
- Невязка, 106, 208
- Нейронная сеть, 219
- Немени критерий, 368
- Неравенство треугольника, 247
- Несправедливость, 64
- Неустойчивость, 216
- Нечистота
 - относительная, 157
 - [пример 5.1](#), 148
- Нормальное распределение, 277, 316
 - многомерное, 278
 - многомерное стандартное, 278
 - стандартное, 278, 280
- Нормальный вектор, 207
- Нормировка, 210
 - по строкам, 104
- Нулевая гипотеза, 364
- Облачные вычисления, 372
- Обобщенная линейная модель, 308
- Обучаемость, 136
- Обучающий набор, 26, 63
- Обучение без учителя, 26, 29, 59
- Обучение по импликации, 140
- Обучение по интерпретациям, 140
- Обучение предпочтений, 372
- Обучение с подкреплением, 371
- Обучение с учителем, 26, 29, 59
- Обучение с частичным привлечением учителя, 29
- Объект, 62
 - помеченный, 63
- Объем информации, 304
 - [пример 9.7](#), 305
- Объяснение, 48
- Один против всех, 97
- Один против одного, 97
- Однородные координаты, 17, 36, 207, 212
- Оперативное обучение, 372
- Опора, 194
- Опорный вектор, 223
- Оптимальность по Байесу, 41, 42, 276
 - базовый линейный классификатор, 282
- Оптимизация
 - двойственная задача, 225
 - квадратичная, 224
 - Основная задача, 225
 - с ограничениями, 224, 225
- Оракул членства, 133
- Оракул эквивалентности, 133
- Ослабляющая переменная, 228, 306
- Отвержение, 98
- Отделяй и властуй, 48, 173, 175
- Отношение, 64
 - антисимметричное, 64
 - полное, 64
 - рефлексивное, 64
 - симметричное, 64
 - транзитивное, 64
 - эквивалентности, 64
- Отношение затрат, 86, 154
- Отрицание, 118
- Отсутствующие значения, 333
 - [пример 1.2](#), 38

- Оценивание вероятностей классов, 87, 371
дерево, 154
квадратичная ошибка
пример 2.6, 89
- Оценивающий классификатор, 75
- Оценка, 58
- Оценка максимального правдоподобия, 282, 298
в линейной регрессии, 210
- Оценочная функция, 105
- Ошибка зазора, 228
- Ошибка ранжирования, 78
- Параллельный перенос, 36
- Парный t -критерий, 365
- Перебор с возвратом, 146
- Перекрестная проверка, 31, 361
внутренняя, 370
послойная, 361
пример 12.4, 362
- Перенос обучения, 372
- Переобучение, 30, 45, 63, 105, 106, 111, 144, 164, 207, 222, 296, 335
пример 2, 18
- Перцептрон, 218, 221
динамический, 219
- Пирсон Карл, 311
- Подгруппа, 114
выявление, 29
пример 3.11, 114
- оценка качества
пример 6.6, 191
- расширение, 114
- Подмножество, 63
- Подстановка, 333
- Подъем, 197
- Поиск ближайшего соседа, 255
- Поиск в ширину, 195
- Поисковая эвристика, 77
- Показатели качества, 356
классификаторов, 71
- Полнота, 70, 113, 312
средняя, 74, 191
- Полный порядок, 64
- Понижение размерности, 338
- Порог принятия решения
настройка
пример 2.5, 85
- Порождающая модель, 41
- Порядковое число, 310
- Последовательная минимальная оптимизация, 241
- Постобработка, 197
- Потоки данных, 372
- Поурневый обход дерева, 195
- Почти корректная, 136
- Правила де Моргана, 118, 143
- Правило, 119
заголовок, 170
неполнота, 47
- перекрытие
пример 1.6, 47
- противоречивость, 47
- тело, 170
- Правило по умолчанию, 47, 174
- Предикаты. См. Логика первого порядка
- Предметный набор, 194
замкнутый, 195
частый, 194
- Предсказание
последовательности, 372
- Предсказанная частота положительных результатов, 358
- Преобразование ранжировщиков в классификаторы, 289
- Признак, 25, 63, 273
бинаризация, 319
булев, 316

- два способа использования
пример 1.8, 53
- дискретизация, 54, 321
- задание порога, 319
- калибровка, 288, 325
изотонная, 330, 332
категориальных, 326
логистическая, 328
категориальный, 167, 316
- нормировка, 213, 248, 281, 325
- область значений, 50, 63
- общий, 58
- отбор, 254
Relief, 335
обертка, 336
обратное исключение, 336
прямой отбор, 336
фильтрация, 335
- Преобразование, 318
- Пространство, 236
- разупорядочение, 319
- список, 45
устранение корреляции, 213, 248, 281
- Пример, 63
- Прирост информации, 149, 321, 334
- Прогностическая
кластеризация, 110, 257, 301
- Прогностическая модель, 29
- Проекция, 231
- Проклятие размерности, 254
- Пропозиционализация, 318
- Пропозициональная логика, 135
- Пространство версий, 126
- Пространство входов, 236
- Пространство выходов, 62, 371
- Пространство гипотез, 120, 199
- Пространство логарифмических
шансов, 288, 328
- Пространство меток, 62, 371
- Пространство объектов, 33, 50, 53, 62
сегмент, 45, 64, 118, 145
- Пространство признаков, 236
- Процентиль, 313
пример 10.1, 313
- Псевдометрика, 247
- Псевдосчетчики, 90, 185, 286, 289
- Рабочий контекст, 356
- Разбиение, 64
- Разбиение по медоидам, 261, 321
- Разброс, 111
внутриклusterный, 111
матрица, 211, 257, 337
внутриклusterного, 257
межклusterного, 257
уменьшение путем разбиения, 258
- Разделение, 145
двоичное, 53
- Разделение множества объектов
пример 4.7, 138
- Разделяй и властвуй, 47, 145, 150, 173
- Разложение матрицы, 29, 111, 336
неотрицательное, 341
с ограничениями, 338
- Размах, 312
- Разреженность, 33, 216
- Ранжирование, 78
верность
пример 2.3, 79
пример 2.2, 78
- Ранжирующая модель, 65, 106
- Распределение вероятностей
кумулятивное, 313
скошенное вправо, 314
- Расстояние, 35
евклидово, 35, 317
манхэттенское, 37
эллиптическое
пример 8.1, 248
- Расхождение, 110

- кластерное, 165
разделения, 165
Расширение, 119
Регрессия, 26, 78
изотонная, 93
многомерная, 213
одномерная, 208
пример 3.8, 105
Регуляризация, 216, 228, 306
Редакционное расстояние, 246
Редукция, 45, 155
Редукция с уменьшением ошибки, 155, 161, 164
инкрементная, 203
Редуцирующий набор, 155
Режим работы, 86
Рекурсивное разбиение, 321
Решающая граница, 17, 26
Решающее дерево, 45, 65, 115, 325
Решающее правило откалиброванного правдоподобия, 289
Решающее правило, 38
Решающий пень, 351
Решающий список, 45, 203
Решение задачи линейной регрессии по методу наименьших квадратов, 283
Решка, 120, 194, 199
РХП
вершина, 84, 158
график, 74
кривая, 81
Рэнда индекс, 113

Свободный член, 207, 209
Сглаживание вероятностей, 90
Сегмент, 49
Сигмоид, 233
Силуэт, 263
Сингулярное разложение матрицы, 336

Системы обучения деревьев
C4.5, 168
CART, 168
ID3, 168
Системы обучения правил, 203
AQ, 203
CN2, 203, 369
Opus, 204
Ripper, 203, 354
Slipper, 354
Tertius, 204
Складного ножа критерий, 361
Скорость обучения, 219
Скрытая переменная, 27, 299
Слабая обучаемость, 342
Словарь, 19
Случайная величина, 57
Случайный лес, 142, 345, 351
Смесовая модель, 277
Смещение, 107
Сопряженное априорное распределение, 275
Сосед, 249
Спектральное разложение, 337
Специфичность, 68
Список правил, 151, 170
как ранжировщик
пример 6.4, 184
пример 6.1, 172
Сравнимость, 64
Среднее, 278, 311
арифметическое, 311
гармоническое, 311
геометрическое, 312
Среднее арифметическое
минимизирует сумму квадратов расстояний, 249, 303
Среднее значение размаха, 312
Среднее по генеральной совокупности, 58

Среднеквадратичная ошибка, 88
 Стандартное отклонение, 312
 Статистика
 разброса, 311
 формы, 311, 315
 центральной тенденции, 311
 Степенное множество, 64
 Степень свободы
 в t -распределении, 365
 в таблице сопряженности, 70
 Стоп-слова, 291
 Структурное предсказание выхода, 372
 Стягивание, 216
 Субаддитивность, 247
 Сходство, 87
 пример 1.1, 27
 Счетчики покрытия, 101
 в качестве оценок
 пример 3.4, 101

 Таблица сопряженности, 66
 Теорема о бесплатных завтраках, 32, 353
 Теория вычислительного обучения, 136
 Теория обнаружения сигналов, 327
 Теория прогнозирования, 88
 Тестовый набор, 31, 63
 Точная верхняя грань, 120
 Точная нижняя грань, 120
 Точность, 70, 113, 179, 198, 312
 с поправкой Лапласа, 181
 Триграмма, 334

 Угловой коэффициент, 207
 Уилкоксона критерий знаковых рангов, 365
 пример 12.7, 366
 Уилкоксона-Манна-Уитни критерий, 95
 Укладка, 352

Универсальное множество, 64, 119
 Унigramма, 334
 Унификация, 136
 пример 4.6, 136
 Упорядочение по общности, 119
 Уровень доверия, 196
 Усиление, 77, 345
 изменение весов
 пример 11.1, 346
 Усиливающая выборка, 343
 Условная независимость, 292
 Условное правдоподобие, 295
 Условное случайное поле, 308
 Уточнение, 83
 потери на, 90

 Фридмана критерий, 366
 пример 12.8, 367
 Функтор, 199
 Функция потерь, 76, 106
 Функция потерь типа 0–1, 76
 Функция правдоподобия, 39, 316
 Функция связи
 монотонность, 267
 определение 8.5, 264
 пример 8.7, 266

 Характеристики набора данных, 352
 Характеристическая функция, 64
 Хорна дизъюнкт, 119, 132, 199
 Хорна теория, 132
 обучение
 пример 4.5, 134
 Хэмминга расстояние, 246, 317

 Целевая переменная, 37, 273
 Целевая функция, 77, 224
 Цель эксперимента, 356
 Центральная предельная теорема, 361
 Центральный момент, 315

- Центрирование относительно нуля, 210, 211, 214, 336
Центр масс, 36
Центроид, 112, 249
- Частичный порядок, 64
Частота истинно отрицательных результатов, 68
Частота истинно положительных результатов, 68
Частота ложноотрицательных результатов, 68
Частота ложноположительных результатов, 68
Частота ложных тревог, 68
Частота ошибок, 67
Частота ошибок ранжирования, 78
Чебышева неравенство, 312
Чебышева расстояние, 245
Червоненкис Алексей, 138
Чистота, 47, 170
Чистый, 146
Чувствительность, 68
- Шкала, 310
обратная, 312
Шум, 63
меточный, 63
объектный, 63
- Эксперимент, 355
Экспоненциальная функция потерь, 77, 349
Эксцесс, 315
Эмпирическая вероятность, 89, 146, 148, 151
Энтропия, 171, 305
Эталон, 36, 145, 249
- Ядро, 55, 334
квадратичное
пример 7.8, 236
- Языки запросов
SQL, 318
Пролог, 199, 204, 318

Книги издательства «ДМК Пресс» можно заказать
в торгово-издательском холдинге «Планета Альянс» наложенным платежом,
выслав открытку или письмо по почтовому адресу:
115487, г. Москва, 2-й Нагатинский пр-д, д. 6А.

При оформлении заказа следует указать адрес (полностью), по которому должны быть
высланы книги; фамилию, имя и отчество получателя.

Желательно также указать свой телефон и электронный адрес.

Эти книги вы можете заказать и в интернет-магазине: www.aliants-kniga.ru.

Оптовые закупки: тел. (499) 782-38-89.

Электронный адрес: books@aliants-kniga.ru.

Петер Флах

Машинное обучение

**Наука и искусство построения алгоритмов,
которые извлекают знания из данных**

Главный редактор *Мовчан Д. А.*

dmkpress@gmail.com

Перевод *Слинкин А. А.*

Корректор *Синяева Г. И.*

Верстка *Чаннова А. А.*

Дизайн обложки *Мовчан А. Г.*

Формат 70×100 1/16.

Гарнитура «Петербург». Печать офсетная.

Усл. печ. л. 25. Тираж 100 экз.

Веб-сайт издательства: www.dmk.ru