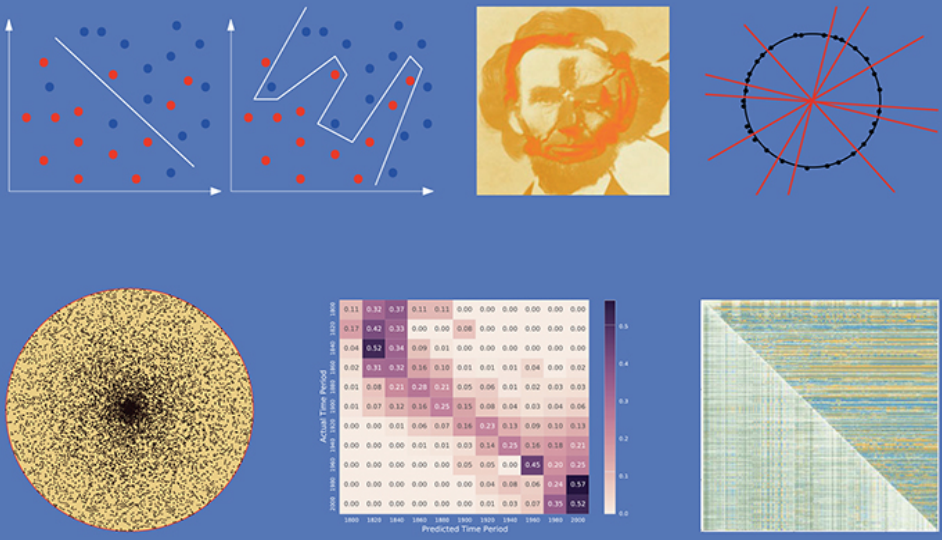
СЕРИЯ КНИГ О КОМПЬЮТЕРНЫХ НАУКАХ

Наука о данных

УЧЕБНЫЙ КУРС



Стивен С. Скиена

ДИАЛЕКТИКА Springer

Steven S. Skiena

The Data Science Design Manual

Springer

Стивен С. Скиена

Наука о данных

УЧЕБНЫЙ КУРС

Москва • Санкт-Петербург
2020

ББК 32.972.134
С42
УДК 004.652

ООО "Диалектика"
Зав. редакцией *С.Н. Тригуб*
Перевод с английского и редакция *В.А. Коваленко*

По общим вопросам обращайтесь в издательство "Диалектика" по адресу:
info@dialektika.com, http://www.dialektika.com

Скиена, Стивен С.
С42 Наука о данных: учебный курс. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 544 с. : ил. — Парал. тит. англ.

ISBN 978-5-907144-74-3 (рус.)

ББК 32.972.134

Научно-популярное издание

Стивен С. Скиена

Наука о данных: учебный курс

Подписано в печать 31.01.2020.
Формат 70x100/16. Гарнитура Times.
Усл. печ. л. 44,2. Уч.-изд. л. 30,8.
Тираж 300 экз. Заказ № 905.

Отпечатано в АО "Первая Образцовая типография"
Филиал "Чеховский Печатный Двор"
142300, Московская область, г. Чехов, ул. Полиграфистов, д. 1
Сайт: www.chpd.ru, E-mail: sales@chpd.ru, тел. 8 (499) 270-73-59

ООО "Диалектика", 195027, Санкт-Петербург, Магнитогорская ул., д. 30, лит. А, пом. 848

ISBN 978-5-907144-74-3 (рус.)
ISBN 978-3-319-55443-3 (англ.)

Оглавление

Содержание

Введение

Понимание мира вокруг нас требует сбора и анализа данных об окружающей среде. Объединение последних технологических тенденций предоставляет новые возможности для применения анализа данных к более сложным задачам, чем когда-либо прежде.

Емкость компьютерных хранилищ увеличивается экспоненциально; хранение данных сейчас стало настолько дешевым, что компьютерным системам почти невозможно ничего забыть. Сенсорные устройства все шире и шире контролируют все, за чем только можно наблюдать: потоки видео, действия в социальных сетях и местоположение всего, что перемещается. Сетевая вычислительная среда позволяет использовать огромные количества машин для манипулирования этими данными. Каждый раз, когда вы осуществляете поиск в Google, задействуются сотни компьютеров, тщательно исследующие все ваши предыдущие действия, только для того, чтобы решить, какая реклама является наилучшей для демонстрации именно вам.

Результатом всего этого стало рождение *науки о данных* (data science) — новой области, посвященной максимизации значения обширных коллекций информации. Как дисциплина наука о данных находится где-то на пересечении статистики, информатики и машинного обучения, но стоит она отдельно, как самостоятельный персонаж. Эта книга служит введением в науку о данных, сосредоточиваясь на навыках и принципах, необходимых для построения систем, предназначенных для анализа и интерпретации данных.

Моя профессиональная практика как исследователя и преподавателя убедила меня в том, что одной из главных сложностей науки о данных является то, что она значительно сложней, чем выглядит. Любой студент, когда-либо вычислявший свой *средний балл успеваемости* (grade point average — GPA), может сказать, что выполнял элементарный статистический расчет, а рисование простого графика разброса позволит вам добавить в свое резюме упоминание о наличии опыта в визуализации данных. Однако реальный анализ и интерпретация данных требуют и технических знаний, и мудрости. Основами обладает очень много людей, но не техническими знаниями, что и вдохновило меня на написание этой книги.

Для кого написана эта книга

Меня порадовал теплый прием моей предыдущей книги, *The Algorithm Design Manual* [1], впервые опубликованной в 1997 году. Она была признана уникальным руководством по использованию алгоритмических методик для решения задач, которые нередко возникают на практике. Книга, которую вы держите в руках, — это совершенно другой материал, но предназначенный для решения тех же задач.

В частности, здесь я подчеркиваю следующие принципы, необходимые, чтобы стать хорошим аналитиком данных.

- *Главное — делать простые вещи правильно*. Наука о данных — это не ракетостроение. Студенты и практики зачастую теряются в технологических вопросах, пытаясь применить наиболее передовые методы машинного обучения, новейшие библиотеки программного обеспечения, реализации с открытым исходным кодом или шикарные методики визуализации. Однако наука о данных призвана делать правильно простые вещи: понимать область применения, очищать и интегрировать корректные источники данных, а также доходчиво представлять ваши результаты другим.

  Как бы то ни было, просто — не значит легко. Действительно, нужна существенная проницательность и опыт, чтобы задавать правильные вопросы, и ум, чтобы двигаться к правильным ответам и действенным решениям. Я противостою искушению излишне углубиться в чисто технический материал только потому, что он и так доступен. Есть много других книг, которые рассматривают сложности алгоритмов машинного обучения или статистической проверки гипотез. Моя цель — заложить основы того, что действительно имеет значение в анализе данных.

- *Думайте как программист, но действуйте как статистик*. Наука о данных подобна зонтику, объединяющему программистов, статистиков и специалистов в некой области. Однако у каждого сообщества есть его собственные специфические стили мышления и действия, которые становятся шаблонами для его членов.

  В этой книге я излагаю подходы, которые наиболее естественны для программистов, в частности алгоритмы манипулирования данными, использование машинного обучения и мастерство масштабирования. Но я также пытаюсь передавать основы статистического рассуждения: потребность понимать область применения, надлежащая оценка малого, поиск значения и жажда исследования.

> Ни у какой дисциплины нет монополии на правду. Лучшие аналитики данных объединяют инструментальные средства из нескольких областей, и эта книга представляет относительно нейтральную территорию, где конкурирующие философии вполне могут обсуждаться вместе.

Не менее важным является то, чего вы не найдете в этой книге. Я не рассматриваю специфические языки или комплекты инструментальных средств анализа данных. Вместо этого здесь обсуждаются важные принципы проектирования. Я постараюсь остаться скорее на концептуальном уровне, чем на техническом. Задача этого руководства в том, чтобы направить вас по правильному пути как можно быстрее, вне зависимости от конкретных программных инструментальных средств, которые вы находите наиболее подходящими для себя.

Для преподавателей

Эта книга содержит достаточно материала курса *Введение в науку о данных* для студентов младших или даже старших курсов. Я надеюсь, что читатель закончил по крайней мере один курс программирования и имеет хоть немного предварительного опыта работы с вероятностью и статистикой, но всегда лучше больше, чем меньше.

Я подготовил полный набор слайдов для лекций, чтобы сделать этот курс более доступным для дистанционного обучения, см. `http://www.data-manual.com`. Там же приводятся ресурсы для домашних заданий и проектов. Кроме того, в сети доступны мои видеолекции курса науки о данных для полного семестра.

Эта книга включает следующие педагогические средства.

- *Случай из жизни*. Чтобы продемонстрировать, как методы науки о данных применяются в реальном мире, я включил в книгу разделы “Случай из жизни”, содержащие рассказы о нашей практике с реальными проблемами. Мораль этих историй в том, что рассматриваемые методы — не просто теория, а полезные инструменты.
- *Фальстарты*. Большинство методов в этой книге представлены в готовом виде, за кадром остались идеи, возникавшие в ходе их разработки, и причины, по которым другие подходы потерпели неудачу. Правдивые истории иллюстрируют процесс рассуждений при решении некоторых практических задач и описывают материал, лежащий в их основе.
- *На заметку*. Фрагменты выделенного этим заголовком текста встречаются в каждой главе и подчеркивают наиболее общие концепции, которые стоит усвоить при изучении данной главы.

- *Домашняя работа*. Я предоставляю широкий диапазон различных упражнений для домашней работы и самообучения. Многие задачи представлены в традиционном стиле, но есть также задания на крупномасштабную реализацию и вопросы меньшего масштаба, как на интервью при поиске работы студентами. Всем задачам присвоена степень сложности.

  Вместо ключей к ответам, были установлены Solution Wiki, где решения всех пронумерованных задач будут востребованы краудсорсингом. Подобная система в моей книге *Algorithm Design Manual* позволила выработать сбалансированные решения, по крайне мере мне так говорят. Я отказываюсь смотреть их из принципа, так что покупателю стоит остерегаться.
- *Конкурсы Kaggle*. Kaggle (`www.kaggle.com`) поддерживает форум для аналитиков данных, позволяя им соревноваться в решении сложных реальных задач на великолепных наборах данных и проверять, насколько хороша ваша модель относительно других. Упражнения каждой главы включают три подходящих конкурса Kaggle, способных послужить источником вдохновения, материалом для самообучения, а также данными для других проектов и исследований.
- *Телевидение науки о данных*. Для широкой общественности наука о данных остается таинственной и даже зловещей. Любительское телевизионное шоу *The Quant Shop* демонстрирует то, чем наука о данных должна быть в действительности. Студенческие группы занимаются разнообразными задачами прогнозирования реальных проблем и пытаются предсказывать результат будущих событий. Ознакомьтесь с этим по адресу `http://www.quant-shop.com`.

  Была подготовлена серия из восьми 30-минутных эпизодов, каждый из которых посвящен конкретной реальной проблеме прогнозирования. Рассматриваются цены произведений искусства на аукционе, выбор победителя на конкурсе Мисс Вселенная и предсказание продолжительности жизни знаменитостей. В каждом случае мы наблюдаем, как группа студентов справляется с задачей, и вместе с ними учимся строить модель прогноза. Они делают свои прогнозы, а мы, глядя на них, видим, оказываются они правы или нет.

  В этой книге я использую передачу *The Quant Shop* для предоставления конкретных примеров сложностей прогнозирования и обсуждения науки о данных, моделируя последовательность от сбора данных до вычисления оценки. Надеюсь, что вы найдете эти примеры интересными и они сподвигнут вас на создание своих собственных конкурсов по моделированию.

- *Дополнительная информация*. Каждая глава книги завершается разделом кратких заметок, рекомендующих читателям основные первоисточники и дополнительные ссылки.

Посвящение

Моим умным и любящим дочерям Бонни и Эбби, теперь уже настоящим подросткам, в том смысле, что они не всегда рассматривают статистическое доказательство с такой живостью, как мне бы хотелось. Я посвящаю эту книгу им в надежде, что их аналитические навыки улучшатся до такой степени, что они всегда будут только соглашаться со мной.

Кроме того, я посвящаю эту книгу своей прекрасной жене Рене, которая соглашается со мной, даже когда она не согласна, и любит меня без всяких заслуживающих доверия доказательств.

Благодарности

Мой список заслуживающих благодарности людей достаточно велик, хотя некоторых я, вероятно, пропустил. Я попытаюсь перечислить их систематически, чтобы минимизировать пропуски, но заранее прошу прощения у тех, кого не упомянул по невнимательности.

В первую очередь, я благодарю тех, кто оказал мне конкретную помощь в компоновке этой книги. *Есыль Ли* (Yeseul Lee) была на этом проекте практикантом, она помогала мне с рисунками, упражнениями и другим на протяжении лета 2016 года. Вы будете видеть доказательство работы ее рук почти на каждой странице, и я очень ценю ее помощь. Аакрити Миттал (Aakriti Mittal) и Джек Зхенг (Jack Zheng) также поспособствовали созданию нескольких рисунков.

Студенты моего курса *Introduction to Data Science* (CSE 519) помогли откорректировать эту рукопись, а также нашли множество проблемных мест. Я особенно благодарю Ребекку Сифорд (Rebecca Siford), которая предложила более ста исправлений. Мои друзья и коллеги по науке о данных, Аншул Ганди (Anshul Gandhi), Ю Фан Ху (Yifan Hu), Клаус Мюллер (Klaus Mueller), Франческо Орабона (Francesco Orabona), Энди Шварц (Andy Schwartz) и Чарльз Уорд (Charles Ward), отрецензировали несколько моих глав, и я благодарю их за усилия.

Я признателен всем студентам *The Quant Shop* из Fall 2015, видео и усилия которых по моделированию и так видны на экране. Большое спасибо Жанне (Дине) Дискин-Циммерман (Jan (Dini) Diskin-Zimmerman), чьи редакторские усилия не

входили в ее служебные обязанности, я чувствовал себя просто преступником за то, что позволил ей делать это.

Как обычно, работать с моими редакторами из Springer — Уэйном Виллером (Wayne Wheeler) и Саймоном Риисом (Simon Rees) — было просто удовольствием. Я также благодарю весь производственный и маркетинговый персонал, который помог предоставить эту книгу вам, особенно Адриана Пьерона (Adrian Pieron) и Аннетт Анлауф (Annette Anlauf).

Некоторые упражнения были предложены коллегами или созданы по мотивам других источников. Восстановить первоначальные источники через несколько лет очень сложно, но благодарности за каждую задачу (по моей памяти) приведены на веб-сайте.

Большая часть моих знаний о науке о данных была получена в ходе работы с другими людьми. К ним относятся мои аспиранты, в частности Рами аль-Рфоу (Rami al-Rfou), Михаил Баутин (Mikhail Bautin), Чи-Хао Чен (Haochen Chen), Яньцин Чен (Yanqing Chen), Вивек Кулкарни (Vivek Kulkarni), Левон Ллойд (Levon Lloyd), Андрей Мехлер (Andrew Mehler), Брайан Пероззи (Bryan Perozzi), Йингтао Тьян (Yingtao Tian), Юн Тинг Йе (Junting Ye), Венбин Чжан (Wenbin Zhang) и докторант Чарльз Уорд. Я с нежностью вспоминаю всех студентов моего проекта Lydia за эти годы и напоминаю, что предложенный мною приз первому, кто назовет свою дочь Лидией, остается невостребованным. Я благодарю других своих сотрудников за рассказанные истории, в частности Брюса Фатчера (Bruce Futcher), Джастина Гардина (Justin Gardin), Арно ван де Рийта (Arnout van de Rijt) и Алексея Старова (Oleksii Starov).

Я помню всех членов мира General Sentiment/Canrock, особенно Марка Фасциано (Mark Fasciano), с которым я разделил мечту о новом и испытал то, что случается, когда данные попадают в реальный мир. Я благодарю коллег по Yahoo Labs/Research, работавших со мной в 2015-2016 годах, когда была задумана б*о*льшая часть этой книги. Я вспоминаю Аманду Стент (Amanda Stent), благодаря которой мне довелось быть в компании Yahoo на протяжении того особенно трудного года в ее истории. Я узнал очень ценные вещи от других людей, которые вели курсы, связанные с наукой о данных, включая Эндрю Ына (Andrew Ng) и Ханса-Петера Фистера (Hans-Peter Pfister). Я благодарю их всех за помощь.

Если у вас есть процедура с десятью параметрами, то некоторые вы, вероятно, пропустили.

— Алан Перлис (Alan Perlis)

Предупреждения

Для автора традиционно великодушно принимать на себя вину за любые оставшиеся неточности. Я не таков. Любые ошибки, неточности или проблемы в этой книге — это чья-то ошибка, и я оценил бы информацию о них, чтобы определить, кто виноват.

Стивен С. Скиена
Факультет информатики
Университет штата Нью-Йорк в Стоуни-Брук
http://www.cs.stonybrook.edu/~skiena
skiena@data-manual.com
Май 2017

Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересны любые ваши замечания в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш веб-сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится ли вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Отправляя письмо или сообщение, не забудьте указать название книги и ее авторов, а также свой обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию новых книг.

Актуальность ссылок не гарантируется.

Наши электронные адреса:

E-mail: info@dialektika.com
WWW: http://www.dialektika.com

Глава 1

Что такое наука о данных?

> Целью вычисления является проникновение в суть, а не числа.
>
> — Ричард Уэсли Хэмминг (Richard W. Hamming)

Что такое наука о данных? Подобно любой новой области, она еще не определена полностью, но вы уже знаете о ней достаточно, чтобы заинтересоваться, иначе вы не читали бы эту книгу.

Я считаю науку о данных местом на пересечении информатики, статистики и независимых областей применения. От информатики исходит машинное обучение и технологии высокопроизводительных вычислений, чтобы справляться с масштабом. От статистики исходит давняя традиция исследовательского анализа данных, проверка достоверности и визуализация. От областей применения в бизнесе и науках исходят задачи, ст*о*ящие сломанных копий, а также стандарты вычислений, позволяющие оценить, когда победа будет окончательно достигнута.

Но все эти области уже хорошо известны. Почему наука о данных, и почему теперь? Я вижу три причины для этого внезапного пика активности.

- Новая технология позволяет уловить, аннотировать и сохранить обширные объемы данных из социальных сети, регистрируя также данные сенсоров. После того как вы накопили все эти данные, вы начинаете задаваться вопросом, что с ними можно сделать.
- Компьютерные достижения позволяют анализировать данные новыми способами и в невиданных ранее масштабах. Архитектуры сетевой вычислительной среды предоставляют некоторый доступ к своей огромной мощи даже простым парням, когда они нуждаются в ней. Новые подходы к машинному обучению приводят к удивительным достижениям в решении давнишних проблем, подобных компьютерному зрению и обработке текстов на естественном языке.
- Ведущие технологические компании (такие как Google и Facebook) и многомиллиардные фонды (такие как Renaissance Technologies и TwoSigma) уже доказали мощь современной аналитики данных. Удачные случаи

применения данных в таких разных областях, как менеджмент в спорте (*Moneyball* [2]) и прогнозирование результатов выборов (Нейт Силвер (Nate Silver) [3]), послужили образцами для подражания и вынесли науку о данных на большую публику.

У этой вводной главы три цели. Во-первых, я попытаюсь объяснять, как думают хорошие аналитики данных и чем их мышление отличается от такового у традиционных программистов и программных разработчиков. Во-вторых, мы рассмотрим наборы данных с точки зрения их потенциала для практического применения и научимся задавать более общие вопросы, на которые они способны давать ответы. И наконец, я представлю наборы задач по анализу данных, которые будут использоваться повсюду в этой книге в качестве мотивации примеров.

1.1. Информатика, наука о данных и реальная наука

Программисты не уважают данные. Их традиционно учили, что алгоритм — это вещь, а данные — это только суть, которая будет проходить через мясорубку.

Прежде чем стать эффективным аналитиком данных, необходимо научиться думать как настоящий ученый. Настоящие ученые стараются понять естественный мир, являющийся сложным и грязным местом. В отличие от них, программисты склонны создавать собственные чистые и организованные виртуальные миры и с удобствами жить в них. Ученые сосредоточиваются на поиске неких вещей, в то время как программисты изобретают, а не находят.

Человеческое сознание четко определяет, как оно мыслит и действует, вызывая недоразумения, когда мы пытаемся общаться с представителями других научных школ. Эти пристрастия столь фундаментальны, что мы зачастую не сознаем их наличия. Вот несколько примеров культурных различий между информатикой и реальной наукой.

- *Данные против ориентированности на метод.* Ученые ориентируются на данные, а программисты — на алгоритмы. Реальные ученые тратят огромные усилия на сбор данных, чтобы ответить на интересующий их вопрос. Они изобретают причудливые измерительные приборы, не спят ночами, ставят эксперименты и думают в основном о том, как получить необходимые им данные.

  В отличие от них, программисты сосредоточиваются на методах: какой алгоритм лучше другого и чем, какой язык программирования лучше для

работы, какая программа лучше и чем. Подробности о наборах данных, с которыми они работают, кажутся сравнительно неинтересными.

- *Забота о результатах*. Реальных ученых заботят ответы. Они анализируют данные, чтобы выяснить что-то о том, как устроен мир. Хороших ученых заботит наличие у результатов смысла, поскольку им нужны ответы.

  В отличие от них, плохие программисты заботятся о создании чисел, выглядящих так, как надо. Как только числа прекращают выглядеть совершенно неправильными, они полагают себя правыми. Это связано с тем, что их личный вклад невелик в то, что может быть изучено в результате вычислений, в отличие от того, что они должны быть выполнены быстро и эффективно.

- *Достоверность*. Реальные ученые привычны к идее, что в данных есть ошибки, а программисты — обычно нет. Ученые много думают о возможных источниках неточностей или ошибок в их данных, и эти возможные проблемы могут повлиять на делаемые ими заключения. Хорошие программисты используют мощные методологии типизации и анализа данных, чтобы справиться с ошибками форматирования, но проблемы эти разнятся.

  Знание о том, что в данных могут быть ошибки, мобилизует. В ответ на критику программисты поют защитную мантру "мусор на входе, мусор на выходе", как способ сказать, что *это не моя проблема*. Реальные ученые достаточно близки к своим данным, чтобы почувствовать их запах и на нюх определить, являются ли они мусором.

- *Точность*. Ничто в науке никогда не является совершенно истинным или ложным, в то время как в информатике либо математике *все* или истинно, или ложно.

  По правде говоря, программисты имеют счастливую возможность выводить числа с плавающей запятой с таким количеством цифр, с каким захочется: 8/13 = 0,61538461538. Реальные ученые будут использовать только две значащие цифры: $8/13 \approx 0{,}62$. Программистов заботит, что за число, в то время как реальных ученых заботит, что оно означает.

Аналитики данных должны стремиться думать как реальные ученые. Ваша задача — постараться превратить числа в смысл. Важно понять, *почему* столько, а не *как*.

Справедливости ради, реальным ученым также полезно было бы думать как аналитики данных. Новые экспериментальные технологии обеспечивают измерительным системам значительно более грандиозный масштаб, чем когда-либо

было возможно прежде, подобно технологии упорядочения полного генома в биологии и отчетам о телескопических наблюдениях всего неба в астрономии. За новой широтой взглядов следуют новые уровни зрения.

Традиционная *управляемая гипотезами* наука основана на формулировании конкретных вопросов об окружающем мире с последующим сбором определенных данных, которые должны подтвердить или опровергнуть ее. Теперь это приросло наукой, *управляемой данными*, которая вместо этого сосредоточивается на создании данных неслыханного ранее масштаба или разрешения в уверенности, что появятся новые изыскания, как только все будут в состоянии посмотреть на них. Оба мышления будут важны для нас.

- Дана задача, какие доступные данные помогут нам решить ее?
- Дан набор данных, к каким интересным задачам мы можем их применить?

Есть и другой способ уловить это фундаментальное различие между разработкой программного обеспечения и наукой о данных. Разработчики программного обеспечения заняты построением систем, в то время как аналитики данных заняты проникновением в суть.

Это может быть камнем преткновения для некоторых разработчиков. Существует важный класс инженеров, которые разрабатывают огромные распределенные инфраструктуры, необходимые для хранения и анализа, скажем, финансовых транзакций или всех данных социальных сетей уровня Facebook или Twitter. Я посвящу главу 12 отличительным особенностям больших инфраструктур данных. Эти инженеры строят инструментальные средства и системы для поддержки науки о данных, даже при том, что они могут и не сами добывать данные, используемые ими. Можно ли квалифицировать их как аналитиков данных?

Это справедливый вопрос, но я ловко обойду его, чтобы максимизировать количество потенциальных читателей этой книги. Но в действительности я полагаю, что чем лучше такие инженеры понимают полный процесс анализа данных, тем более вероятно они будут в состоянии построить мощные инструментальные средства, способные к поддержанию важных исследований. Главная задача этой книги — обеспечить инженеров больших данных интеллектуальными инструментами, которые позволят им думать как аналитикам больших данных.

1.2. Формирование интересных вопросов для данных

Хорошие аналитики данных вырабатывают врожденное любопытство о мире вокруг них, особенно в областях, связанных с темами, над которыми они работают. Они любят говорить на профессиональные темы с теми людьми, с данными которых они работают. Они задают им вопросы: что наихудшее в вашем деле? Почему вы им заинтересовались? Что вы надеетесь узнать из анализа этого набора данных? Аналитики данных всегда задают вопросы.

У хороших аналитиков данных всесторонние интересы. Они читают газету каждый день, чтобы иметь более широкую точку зрения, это хорошо. Они понимают, что мир — интересное место. Даже небольшие знания обо всем, что их окружает, позволяет им держаться на равных с другими людьми. Они достаточно храбры, чтобы немного выйти из своих зон комфорта, и целенаправленны, чтобы узнавать побольше, побывав там.

Разработчикам программного обеспечения нет смысла задать вопросы, в отличие от аналитиков данных. Мы задаем вопросы так.

- Что вы могли бы узнать из данного набора данных?
- Что вы (или ваши люди) действительно хотят узнать об окружающем мире?
- Что это будет означать для вас, как только вы это узнаете?

Программисты действительно традиционно не ценят данные. Они думают о способе экспериментального измерения производительности алгоритма. Обычно программа выполняется на "случайных данных", чтобы увидеть, как долго она получает результат. Они даже редко смотрят на результаты вычисления, кроме проверки их правильности или эффективности. Поскольку "данные" бессмысленны, результаты не могут быть важны. Реальные наборы данных, напротив, являются ценным ресурсом, для получения которого требуется тяжелая работа и воображение.

Чтобы стать аналитиком данных, нужно научиться задавать вопросы о данных, поэтому давайте попрактикуемся. Каждый из разделов ниже знакомит с интересным набором данных. После того как вы понимаете, какая информация доступна, попытайтесь придумывать, скажем, пять интересных вопросов, которые вы могли бы задать и исследовать, имея доступ к этому набору данных.

Главное, думать широко: ответы на большие, общие вопросы зачастую скрыты в высоко специфических наборах данных, которые ни в коем случае не были задуманы для того, чтобы содержать их.

1.2.1. Бейсбольная энциклопедия

В мире науки о данных у бейсбола довольно долго была важность куда выше средней. Этот спорт был и остается национальным времяпрепровождением Соединенных Штатов Америки; действительно, французский историк Жак Барзен (Jacques Barzun) заметил: "Кто бы ни захотел узнать сердце и ум Америки, тот должен изучать бейсбол". Я понимаю, что многие читатели не являются американцами и даже, являясь ими, они, могут не очень интересоваться спортивными состязаниями. Но будьте со мной хоть на некоторое время.

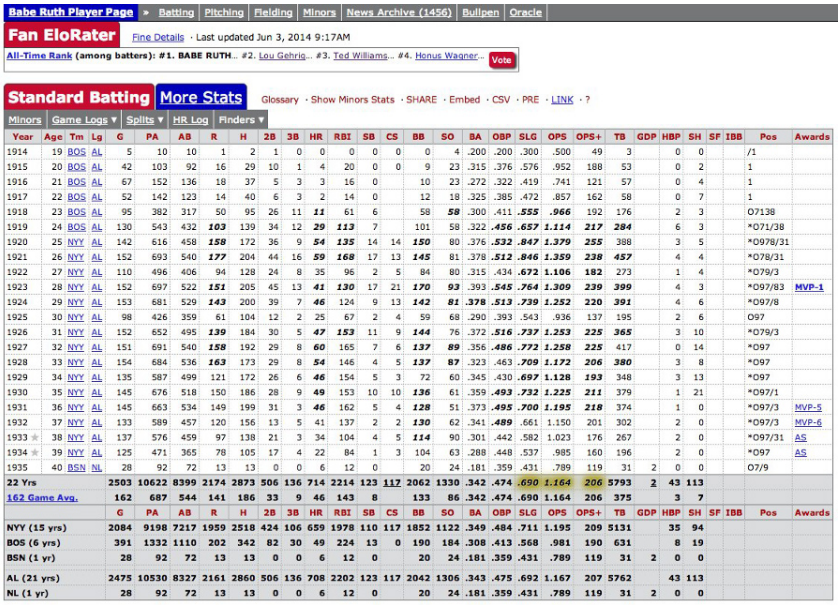
Что делает бейсбол важным для науки о данных — это его обширные статистические записи об играх на протяжении более чем ста последних лет. Бейсбол — это спорт отдельных событий: питчеры бросают мячи, а бэттеры пытаются отбить их, что, естественно, предоставляет место для информативной статистики. Болельщики увлекаются этой статистикой с детства, проявляя интуицию о расстановке сил и выполняя элементарный количественный анализ. Некоторые из этих детей, повзрослев, стали аналитиками данных. Успех статистически продуманной бейсбольной команды Брэда Питта в фильме *Человек, который изменил всё* (Moneyball) остается самым ярким контактом американской общественности с наукой о данных.

Эта бейсбольная историческая запись доступна по адресу `http://www.baseball-reference.com`. Там вы найдете полные статистические данные по результативности каждого игрока, который хоть раз вышел на поле. Они включают обобщенную итоговую статистику отбоев, подач и пробежек по каждому сезону, а также информацию о командах и наградах, как показано на рис. 1.1.

Но есть не только статистика, есть метаданные о жизни и карьере всех людей, которые когда-либо играли в бейсбол в высшей лиге, как показано на рис. 1.2. Доступна статистика о жизни каждого игрока (рост, вес, правша или левша), их возраст (даты и места их рождения и смерти). Мы также получаем информацию о зарплате (сколько каждому игроку заплатили в какой сезон) и транзакционные данные (как они стали собственностью каждой команды, за которую играли).

Теперь предположим, что большинство из вас не имеет ни малейшего представления о бейсболе и не интересуется им. Этот спорт несколько напоминает крикет, если это вам поможет. Но помните, что как аналитик данных вы должны интересоваться миром вокруг вас. Считайте это шансом узнать что-то новое.

Так на какие интересные вопросы могут дать ответ эти наборы бейсбольных данных? Попробуйте написать пять вопросов, прежде чем продолжать. Не волнуйтесь, я подожду, пока вы закончите.



Babe Ruth Player Page » Batting | Pitching | Fielding | Minors | News Archive (1456) | Bullpen | Oracle

Fan EloRater Fine Details · Last updated Jun 3, 2014 9:17AM

All-Time Rank (among batters): #1. BABE RUTH... #2. Lou Gehrig... #3. Ted Williams... #4. Honus Wagner... Vote

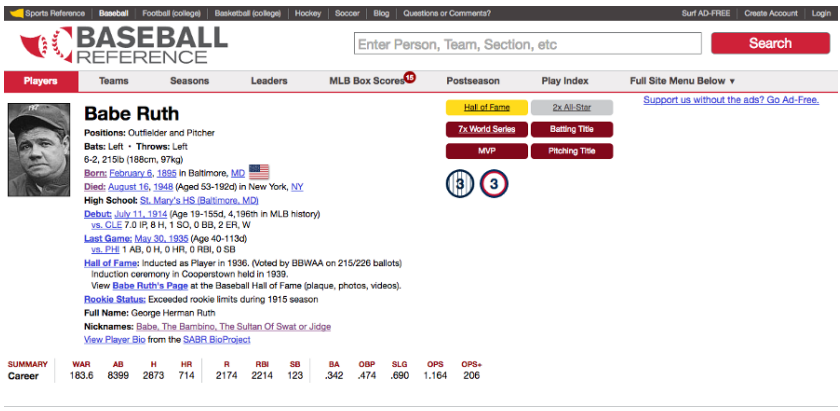
Standard Batting More Stats Glossary · Show Minors Stats · SHARE · Embed · CSV · PRE · LINK · ?

Minors | Game Logs ▾ | Splits ▾ | HR Log | Finders ▾

Year	Age	Tm	Lg	G	PA	AB	R	H	2B	3B	HR	RBI	SB	CS	BB	SO	BA	OBP	SLG	OPS	OPS+	TB	GDP	HBP	SH	SF	IBB	Pos	Awards
1914	19	BOS	AL	5	10	10	1	2	1	0	0	0	0	0	0	4	.200	.200	.300	.500	49	3		0	0			/1	
1915	20	BOS	AL	42	103	92	16	29	10	1	4	20	0	0	9	23	.315	.376	.576	.952	188	53		0	2			1	
1916	21	BOS	AL	67	152	136	18	37	5	3	3	16	0		10	23	.272	.322	.419	.741	121	57		0	4			1	
1917	22	BOS	AL	52	142	123	14	40	6	3	2	14	0		12	18	.325	.385	.472	.857	162	58		0	7			1	
1918	23	BOS	AL	95	382	317	50	95	26	11	***11***	61	6		58	***58***	.300	.411	***.555***	***.966***	192	176		2	3			O7138	
1919	24	BOS	AL	130	543	432	***103***	139	34	12	***29***	***113***	7		101	58	.322	***.456***	***.657***	***1.114***	***217***	***284***		6	3			*O71/38	
1920	25	NYY	AL	142	616	458	***158***	172	36	9	***54***	***135***	14	14	***150***	80	.376	***.532***	***.847***	***1.379***	***255***	388		3	5			*O978/31	
1921	26	NYY	AL	152	693	540	***177***	204	44	16	***59***	***168***	17	13	***145***	81	.378	***.512***	***.846***	***1.359***	***238***	***457***		4	4			*O78/31	
1922	27	NYY	AL	110	496	406	94	128	24	8	35	96	2	5	84	80	.315	.434	***.672***	***1.106***	***182***	273		1	4			*O79/3	
1923	28	NYY	AL	152	697	522	***151***	205	45	13	***41***	***130***	17	21	***170***	***93***	.393	***.545***	***.764***	***1.309***	***239***	***399***		4	3			*O97/83	MVP-1
1924	29	NYY	AL	153	681	529	***143***	200	39	7	***46***	124	9	13	***142***	***81***	***.378***	***.513***	***.739***	***1.252***	***220***	***391***		4	6			*O97/8	
1925	30	NYY	AL	98	426	359	61	104	12	2	25	67	2	4	59	68	.290	.393	.543	.936	137	195		2	6			O97	
1926	31	NYY	AL	152	652	495	***139***	184	30	5	***47***	***153***	11	9	***144***	76	.372	***.516***	***.737***	***1.253***	***225***	***365***		3	10			*O79/3	
1927	32	NYY	AL	151	691	540	***158***	192	29	8	***60***	165	7	6	***137***	***89***	.356	***.486***	***.772***	***1.258***	***225***	417		0	14			*O97	
1928	33	NYY	AL	154	684	536	***163***	173	29	8	***54***	146	4	5	***137***	***87***	.323	.463	***.709***	***1.172***	***206***	***380***		3	8			*O97	
1929	34	NYY	AL	135	587	499	121	172	26	6	***46***	154	5	3	72	60	.345	.430	***.697***	***1.128***	***193***	348		3	13			*O97	
1930	35	NYY	AL	145	676	518	150	186	28	9	***49***	153	10	10	***136***	61	.359	***.493***	***.732***	***1.225***	***211***	379		1	21			*O97/1	
1931	36	NYY	AL	145	663	534	149	199	31	3	***46***	162	5	4	***128***	51	.373	***.495***	***.700***	***1.195***	***218***	374		1	0			*O97/3	MVP-5
1932	37	NYY	AL	133	589	457	120	156	13	5	41	137	2	2	***130***	62	.341	***.489***	.661	1.150	201	302		2	0			*O97/3	MVP-6
1933 ★	38	NYY	AL	137	576	459	97	138	21	3	34	104	4	5	***114***	90	.301	.442	.582	1.023	176	267		2	0			*O97/31	AS
1934 ★	39	NYY	AL	125	471	365	78	105	17	4	22	84	1	3	104	63	.288	.448	.537	.985	160	196		2	0			*O97	AS
1935	40	BSN	NL	28	92	72	13	13	0	0	6	12	0		20	24	.181	.359	.431	.789	119	31	2	0	0			O7/9	
22 Yrs				**2503**	**10622**	**8399**	**2174**	**2873**	**506**	**136**	**714**	**2214**	**123**	**117**	**2062**	**1330**	**.342**	**.474**	***.690***	***1.164***	***206***	**5793**	**2**	**43**	**113**				
162 Game Avg.				**162**	**687**	**544**	**141**	**186**	**33**	**9**	**46**	**143**	**8**		**133**	**86**	**.342**	**.474**	**.690**	**1.164**	**206**	**375**		**3**	**7**				

	G	PA	AB	R	H	2B	3B	HR	RBI	SB	CS	BB	SO	BA	OBP	SLG	OPS	OPS+	TB	GDP	HBP	SH	SF	IBB	Pos	Awards
NYY (15 yrs)	**2084**	**9198**	**7217**	**1959**	**2518**	**424**	**106**	**659**	**1978**	**110**	**117**	**1852**	**1122**	**.349**	**.484**	**.711**	**1.195**	**209**	**5131**		**35**	**94**				
BOS (6 yrs)	**391**	**1332**	**1110**	**202**	**342**	**82**	**30**	**49**	**224**	**13**	**0**	**190**	**184**	**.308**	**.413**	**.568**	**.981**	**190**	**631**		**8**	**19**				
BSN (1 yr)	**28**	**92**	**72**	**13**	**13**	**0**	**0**	**6**	**12**	**0**		**20**	**24**	**.181**	**.359**	**.431**	**.789**	**119**	**31**	**2**	**0**	**0**				
AL (21 yrs)	**2475**	**10530**	**8327**	**2161**	**2860**	**506**	**136**	**708**	**2202**	**123**	**117**	**2042**	**1306**	**.343**	**.475**	**.692**	**1.167**	**207**	**5762**		**43**	**113**				
NL (1 yr)	**28**	**92**	**72**	**13**	**13**	**0**	**0**	**6**	**12**	**0**		**20**	**24**	**.181**	**.359**	**.431**	**.789**	**119**	**31**	**2**	**0**	**0**				

Рис. 1.1. Статистическая информация о результативности Бейба Рута (Babe Ruth) находится по адресу `http://www.baseball-reference.com`



Sports Reference | Baseball | Football (college) | Basketball (college) | Hockey | Soccer | Blog | Questions or Comments? | Surf AD-FREE | Create Account | Login

BASEBALL REFERENCE Enter Person, Team, Section, etc Search

Players | Teams | Seasons | Leaders | MLB Box Scores | Postseason | Play Index | Full Site Menu Below ▾

Support us without the ads? Go Ad-Free.

Babe Ruth

Positions: Outfielder and Pitcher
Bats: Left · **Throws:** Left
6-2, 215lb (188cm, 97kg)
Born: February 6, 1895 in Baltimore, MD
Died: August 16, 1948 (Aged 53-192d) in New York, NY
High School: St. Mary's HS (Baltimore, MD)
Debut: July 11, 1914 (Age 19-155d, 4,196th in MLB history)
vs. CLE 7.0 IP, 8 H, 1 SO, 0 BB, 2 ER, W
Last Game: May 30, 1935 (Age 40-113d)
vs. PHI 1 AB, 0 H, 0 HR, 0 RBI, 0 SB
Hall of Fame: Inducted as Player in 1936. (Voted by BBWAA on 215/226 ballots)
Induction ceremony in Cooperstown held in 1939.
View Babe Ruth's Page at the Baseball Hall of Fame (plaque, photos, videos).
Rookie Status: Exceeded rookie limits during 1915 season
Full Name: George Herman Ruth
Nicknames: Babe, The Bambino, The Sultan Of Swat or Jidge
View Player Bio from the SABR BioProject

Hall of Fame | 2x All-Star | 7x World Series | Batting Title | MVP | Pitching Title

SUMMARY	WAR	AB	H	HR	R	RBI	SB	BA	OBP	SLG	OPS	OPS+
Career	183.6	8399	2873	714	2174	2214	123	.342	.474	.690	1.164	206

Рис. 1.2. Личная информация каждого бейсбольного игрока высшей лиги доступна по адресу `http://www.baseball-reference.com`

Для этих данных наиболее очевидны те типы вопросов, ответы на которые непосредственно связаны с бейсболом.

- Как лучше всего измерить навык отдельного игрока или его значение?
- Насколько справедливы сделки между командами в целом?

- Какова общая тенденция уровня результативности игрока с возрастом?
- До какой степени результативность бэттера коррелирует с позицией, в которой он играет? Например, действительно ли аутфилдеры (outfielder) лучшие нападающие, чем инфилдеры (infielder)?

Это интересные вопросы. Но куда интересней вопросы о демографических и социальных проблемах. Почти 20 000 бейсболистов высшей лиги вышли на поле за прошлые 150 лет, представив собой большую, тщательно задокументированную когорту мужчин, которые могут послужить представителями еще большего, но менее хорошо задокументированного населения. Мы можем использовать эти данные об игроках в бейсбол для ответа на такие вопросы.

- На самом ли деле продолжительность жизни левшей короче, чем у правшей? Информация о ведущей руке отсутствует в большинстве демографических наборов данных, но здесь она собрана крайне тщательно. Анализ этого набора данных показывает, что правши живут дольше, чем левши [4]!
- Как часто люди возвращаются жить в то же самое место, где они родились? Места рождения и смерти были тщательно записаны в этом наборе данных. Далее, почти все эти люди играли, по крайней мере часть времени, далеко от дома, предоставляя таким образом себя более широкому миру в критически важный период своей юности.
- Зарплаты игрока отражают в общем прошлую, нынешнюю или будущую результативность?
- До какой степени рост и вес населения увеличились в целом?

Здесь есть две специфические темы. Во-первых, идентификаторы и ссылочные метки (т.е. метаданные) зачастую оказываются более интересными в наборе данных, чем тот материал, который, как мы предполагаем, содержит статистическая запись игры.

Во-вторых, идея *статистического прокси-объекта* (statistical proxy): используя набор данных, вы должны заменить его тем, который действительно нужен. Набор данных вашей мечты, вероятно, не существует, или он может быть заперт внутри корпоративных стен, даже если он и есть. Хороший аналитик данных является прагматиком, хорошо видящим то, что он может сделать с тем, что он имеет, а не оплакивающим то, чего он не может заполучить в свои руки.

1.2.2. Интернет-база кинофильмов (IMDb)

Все любят фильмы. Интернет-база кинофильмов (Internet Movie Database — IMDb) предоставляет подвергшиеся краудсорсингу организованные данные

обо всех аспектах кинопромышленности по адресу `www.imdb.com`. База IMDb содержит в настоящее время данные о более чем 3,3 млн фильмов и телевизионных программ. По каждому фильму база IMDb предоставляет его название, продолжительность, жанр, дату выпуска, а также полный список исполнителей и участников. Есть финансовые данные о каждом проекте, включая бюджет создания фильма и величину его кассового успеха.

И наконец, есть обширные оценки каждого фильма как от зрителей, так и от критиков. Данные рейтингов состоят из баллов (от нуля до десяти звезд), собранных в таблицу в соответствии со средним возрастом и полом. Обзоры нередко включают объяснения, почему данный критик присвоил такое количество звезд. Между фильмами есть также ссылки: например, указания других фильмов, чаще всего просматриваемых зрителями, скажем, фильма *Эта прекрасная жизнь* (It's a Wonderful Life).

Каждый исполнитель, директор, продюсер и член съемочной группы, связанный с фильмом, заслуживает записи в базе IMDb, которая теперь содержит данные о 6,5 млн человек.

Рис. 1.3. Данные о фильме в базе IMDb

Они вполне могут включать моего коллегу, кузена и невестку. Каждый актер связывается с каждым фильмом, в котором он участвовал, с описанием роли и порядке в титрах. Данные о каждой личности включают даты рождения/смерти, рост, заслуги и семейное положение.

Так на какие вопросы можно найти ответ по этими данным о фильмах?

Рис. 1.4. Данные об актере в базе IMDb

Вероятно, наиболее естественным вопросом, задаваемым базе IMDb, будет поиск экстремальных значений среди фильмов и актеров.

- Какие актеры участвовали в наибольшем количестве фильмов? Заработали больше всех денег? Участвовали в фильмах с самым низким рейтингом? Имели самую долгую карьеру или самую короткую жизнь?
- Какой фильм имел самый высокий рейтинг в конкретном году или был наилучшим в своем жанре? Какие фильмы потеряли больше всех денег, имели наиболее мощный актерский состав или получили наименее благоприятные отзывы?

Далее следуют крупномасштабные вопросы, которые можно поставить о характере самого кинобизнеса.

- Насколько хорошо кассовый сбор фильма коррелирует с рейтингами зрителей или наградами? Неужели клиенты инстинктивно слетаются на мусор или добродетель со стороны творческой команды должным образом вознаграждается?
- Насколько Голливудские фильмы сравнимы с Болливудскими в терминах рейтингов, бюджета и кассового сбора? Американские фильмы получаются лучше, чем иностранные, и как это различается американскими и неамериканскими рецензентами?

- Каково распределение актеров и актрис по возрасту в фильмах? Насколько чаще в среднем молодые актрисы играют жену, чем актеры играют мужа? Со временем это различие увеличивается или уменьшается?
- Жить быстро, умереть молодым и оставить красивый труп? Кинозвезды живут дольше или меньше, чем обычные актеры или широкая публика?

С учетом, что сотрудничающие на фильме люди знакомы друг с другом, список актеров и данные о команде применимы для построения социальной сети кинобизнеса. Как выглядит социальная сеть исполнителей? Сайт Oracle of Bacon (`https://oracleofbacon.org/`) полагает, что Кевин Бейкон (Kevin Bacon) является центром вселенной Голливуда и создает самый короткий путь к Бейкону от любого другого актера. Однако некоторые другие актеры, такие как Сэмюэл Л. Джексон (Samuel L. Jackson), оказались еще ближе к центру.

Можем ли мы проанализировать эти данные так, чтобы определить вероятность того, что кому-то данный фильм понравится? Методика *совместной фильтрации* (collaborative filtering) способна находить людей, которым нравятся те же фильмы, что и мне, и рекомендовать другие фильмы, которые нравятся *им*, а следовательно, являются хорошими кандидатами для меня. Приз 2007 Netflix Prize составляет $1,000,000 и предлагается за механизм оценок рейтингов, который на 10% лучше, чем собственная система Netflix. Абсолютный победитель этого конкурса, BellKor, использовал множество источников данных и методик, включая анализ каналов связи [5].

1.2.3. N-граммы Google

Печатанные книги были первым хранилищем человеческого знания, начиная с изобретения Гутенбергом наборных шрифтов в 1439 году. Физические объекты живут в сложной обстановке в современном цифровом мире, но технология имеет способ свести все к данным. В качестве части своей миссии по организации информации в мире компания Google предприняла усилие по просмотру всех опубликованных книг в мире. Работа еще далеко не закончена, но 30 миллионов книг к настоящему времени уже оцифровано, что составляет более 20% всех когда-либо опубликованных книг.

Компания Google использует эти данные для улучшения результатов поиска и предоставления быстрого доступа к новейшим книгам. Но, вероятно, наиболее интересным продуктом является удивительный ресурс для контроля изменений в современной культуре — *N-граммы Google*. Он представляет частоту, с которой короткие фразы встречаются в книгах, публикуемых каждый год. Каждая фраза должна встретиться по крайней мере сорок раз в просмотренных книгах. Это позволяет устранять неясные слова и фразы, но также оставляет два миллиарда временны*х* рядов, доступных для анализа.

Этот богатый набор данных демонстрирует, как используемый язык изменился за прошлые 200 лет, и широко применяется в анализе тенденций в культуре [6]. На рис. 1.5 эти данные используются для демонстрации того, как слово *данные* теряло популярность при описании компьютерных вычислений. Термин *обработка данных* (data processing) был популярен в связанных с вычислениями областях во времена перфокарт и бобин с магнитной лентой в 1950-х годах. Данные N-грамм демонстрируют быстрое повышение популярности термина *информатика* (computer science) в 1980-х годах. Даже сегодня термин *наука о данных* (data science) остается почти незаметным на этом фоне.

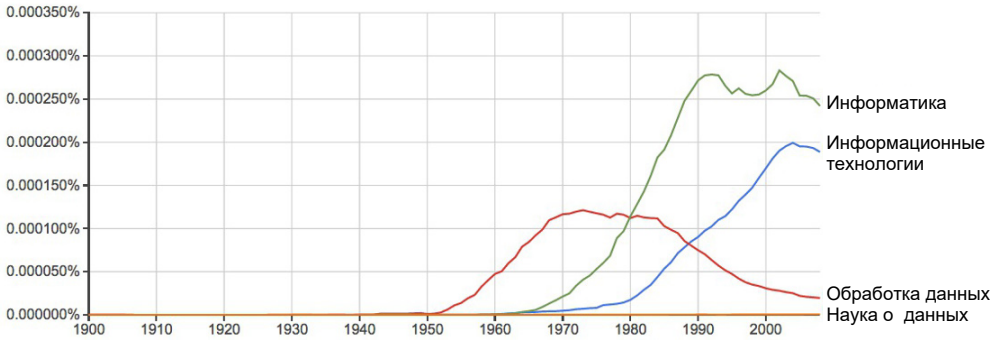


Рис. 1.5. Взлеты и падения обработки данных, засвидетельствованые N-граммами Google

Посмотрите N-граммы Google по адресу `http://books.google.com/ngrams`. Я обещаю, вам понравится играть с ними. Сравните *hot dog* (хот-дог) с *tofu* (тофу), *science* (наука) с *religion* (религия), *freedom* (свобода) с *justice* (правосудие), а также *sex* (секс) и *marriage* (брак), чтобы лучше понять этот фантастический телескоп для изучения прошлого.

Однако, закончив играть, подумайте о вещах, которые вы могли бы сделать, если бы эти данные оказались в ваших руках. Предположим, что у вас есть доступ к годовым сводкам по количествам *всех* слов и фраз, опубликованных в книгах за прошлые 200 лет.

Google делает эти данные общедоступными. Итак, что вы собираетесь делать с ними?

Обзор временных рядов, связанных с конкретными словами, прекрасно осуществляется с использованием Ngrams Viewer. Но более сложные исторические тенденции могут быть выявлены при объединении нескольких временных рядов. Мне кажутся особенно интересными следующие типы вопросов.

- Как со временем меняется объем ругательств? Использование трехбуквенных слов, по моим личным представлениям, началось, кажется, взры-

вообразно с 1960-х годов, хотя, возможно, это менее однозначно отражает увеличение количества сквернословий или снижение стандартов на публикации.

- Как часто новые слова появляются и становятся популярными? Эти слова имеют тенденцию оставаться в применении или они быстро исчезают? Можем ли мы определить, когда слово изменило свой смысл со временем, как, например, слово *gay* сменило смысл *happy* (счастливый) на *homosexual* (гомосексуалист)?
- Стандарты проверки правописания улучшаются или ухудшаются со временем, особенно теперь, в эру автоматизированной проверки правописания? Редко встречающиеся слова, отличающиеся только одним символом, добавленным к общепринятому слову, являются вероятными кандидатами в орфографические ошибки (например, слова *algorithm* и *algorthm*). Количество таких ошибок, связанных с множеством орфографических форм, растет или уменьшается?

Вы можете также использовать эту совокупность N-грамм, чтобы построить языковую модель, которая улавливает смысл и применение слов на данном языке. В разделе 11.6.3 мы обсудим внедрение слов, что является мощным инструментом познания языковых моделей. Подсчеты частот демонстрируют, какие слова являются самыми популярными. Частота пар слов, встречающихся вместе, применяется для улучшения систем распознавания речи, помогая различить, сказал ли говорящий *that's too bad* (очень жаль) или *that's to bad* (это плохо). Эти миллионы книг предоставляют вполне достаточный набор данных для построения солидных моделей.

1.2.4. Записи нью-йоркских такси

Любая финансовая транзакция оставляет сегодня след в данных. Следуя этими путями, можно придти к интересным выводам.

Такси являются важной частью городской транспортной сети. Они перемещаются по улицам города в поисках клиентов, а затем везут их к месту назначения за плату, пропорциональную длине поездки. Каждое такси содержит измеряющее устройство, чтобы вычислить стоимость поездки как функцию времени. Этот измеритель служит устройством хранения записей, а также механизмом, гарантирующим получение водителем надлежащей суммы за каждую поездку.

Таксометры, в настоящее время используемые в нью-йоркских такси, могут делать много вещей, кроме вычисления платы. Они действуют как терминалы кредитных карточек, позволяя клиентам расплатиться за поездку без наличных

денег. Они интегрированы с системами глобального позиционирования (GPS), регистрируя точное место каждой посадки и высадки. И наконец, поскольку они подключены к беспроводной сети, эти устройства способны передавать все эти данные на центральный сервер.

Результат — база данных, документирующая каждую конкретную поездку всех такси в одном из самых больших городов в мире, небольшая часть записей которой представлена на рис. 1.6. Поскольку New York Taxi and Limousine Commission является открытым агентством, его данные не секретны и доступны для всех согласно *закону о свободе информации* (Freedom of Information Act — FOA).

Vendor ID	passenger _count	trip_ distance	pickup_ longitude	pickup_ latitude	dropoff_ longitude	dropoff_ latitude	payment _type	tip_ amount	total_ amount
2	1	7.22	-73.9998	40.74334	-73.9428	40.80662	2	0	30.8
1	1	2.3	-73.977	40.7749	-73.9783	40.74986	1	2.93	16.23
1	1	1.5	-73.9591	40.77513	-73.9804	40.78231	1	1.65	9.95
1	1	0.9	-73.9766	40.78075	-73.9706	40.78885	1	1.45	8.75
2	1	2.44	-73.9786	40.78592	-73.9974	40.7563	1	2	16.3
2	1	3.36	-73.9764	40.78589	-73.9424	40.82209	1	3.58	17.88
2	2	2.34	-73.9862	40.76087	-73.9569	40.77156	1	1	13.8
2	1	10.19	-73.79	40.64406	-73.9312	40.67588	2	0	32.8
1	2	3.3	-73.9937	40.72738	-73.9982	40.7641	1	2	21.3
1	1	1.8	-73.9949	40.74006	-73.9767	40.74934	1	1.85	11.15

Рис. 1.6. Отображаемые поля базы данных нью-йоркского такси: пункты посадки и высадки, а также расстояния и тарифы

Каждая поездка создает две записи: одну с данными о поездке, другую с деталями оплаты. Каждая поездка является ключевой к номеру (лицензии) каждого автомобиля наравне с идентификатором каждого водителя. Для каждой поездки мы получаем время/дату посадки и высадки, так же как координаты GPS (долгота и широта) пунктов отправления и назначения. Мы не получаем данных GPS о маршруте между этими пунктами, но до некоторой степени можем предположить кратчайший путь между ними.

Что касается данных оплаты, мы получаем измеренную стоимость каждой поездки, включающую налог, дополнительный налог и пошлины. Существует традиция платить водителю чаевые за службу, объем которых также записывается в данных.

Говорю вам, эти данные о такси совершенно доступны, в них записи о более чем 80 млн поездок за несколько прошлых лет. Что вы собираетесь сделать с этим?

Любой интересный набор данных применим для ответов на вопросы разных масштабов. Данные о тарифах такси могут помочь нам не только лучше понять транспортную отрасль, но даже и то, как работает город и как мы могли бы улучшить его работу. Вполне естественны следующие вопросы об индустрии такси.

- Сколько денег водители получают каждую ночь в среднем? Каково их распределение? Водители зарабатывают больше в солнечные дни или в дождливые?
- Где наиболее прибыльные места в городе? Как их положение изменяется в течение дня?
- Как далеко водители ездят во время ночной смены? Мы не можем ответить на этот вопрос, точно используя данный набор данных, поскольку он не включает данные GPS о маршрутах. Но мы знаем последнее место высадки, следующее место посадки и сколько времени прошло между ними. Вместе это должно предоставить достаточно информации, чтобы сделать общую оценку.
- Какие водители подбирают ничего не подозревающих иногородних пассажиров и "катают" их, наматывая счетчик, когда поездка могла бы быть намного короче и дешевле?
- Сколько водителям дают чаевых и почему? Получают ли более быстрые водители больше чаевых? Как чаевые варьируются в зависимости от района, и оказываются ли богатые кварталы более щедрыми, чем бедные?

Признаюсь, мы сделали анализ всего этого, и я напишу об этом в разделе "Случай из жизни" главы 9. Мы нашли множество интересных шаблонов [7]. На рис. 1.7 показано, что Манхеттен скуповат по сравнению с Бруклином, Квинсом и Стейтен-Айлендом, где поездки длинней, а уличные такси — явление редкое, но приятное.

Однако основные вопросы относятся к пониманию транспорта в городе. Мы можем использовать время поездки такси как индикатор уровня трафика в городе на общем уровне. Насколько медленней движение в часы пик, чем в другие времена, и где пробки самые страшные? Выявление проблемных областей является первым этапом в предложении решений, при смене шаблонов синхронизации светофоров, при пуске большего количества автобусов или при создании полос только высокой пропускной способности.

Точно так же мы можем использовать данные о такси, чтобы измерить транспортные пути через город. Куда люди едут в разное время дня? Это говорит нам намного больше, чем просто заторы. Глядя на данные о такси, мы должны быть в состоянии видеть, что туристы направляются от гостиниц к

достопримечательностям, служащие едут из пригородов на Уолл-стрит, а пьянчуги возвращаются домой из ночных клубов после попойки.

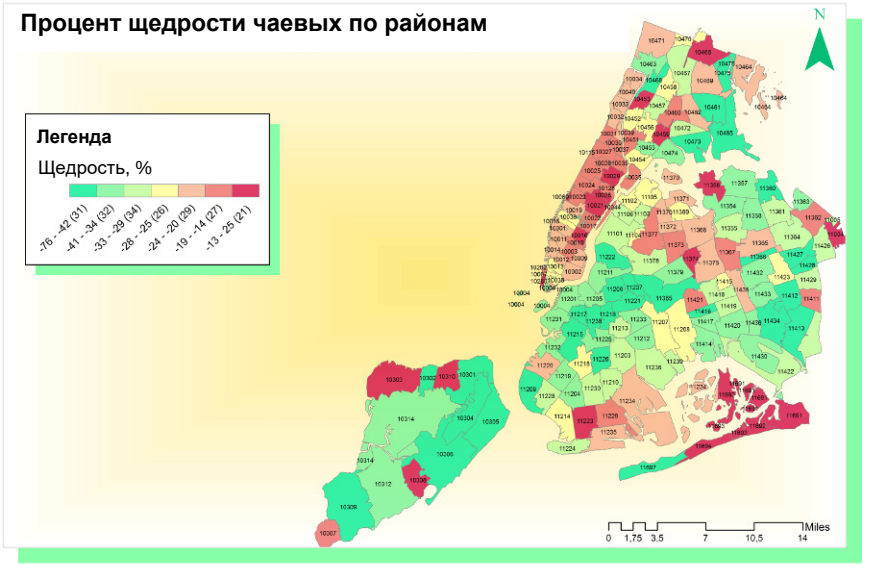


Рис. 1.7. Какие окрестности Нью-Йорка щедрей на чаевые? Относительно отдаленные внешние районы Бруклина и Квинса, где поездки являются самыми длинными, а предложение относительно невелико

Данные важны для разработки лучших транспортных систем. Они чересчур расточительны для одиночного ездока, желающего проехать из точки a в точку b, когда есть другой ездок в точке $a + \varepsilon$, также желающий того же. Анализ данных такси позволяет точно смоделировать систему совместного использования поездок, поэтому мы можем точно оценить требования и сократить затраты на такую услугу.

1.3. Свойства данных

Эта книга о методиках анализа данных. Но каков основной материал, который мы будем изучать? Данный раздел знакомит с краткой таксономией свойств данных, чтобы мы могли лучше оценить и понять то, над чем мы будем работать.

1.3.1. Структурированные или неструктурированные данные

Некоторые наборы данных прекрасно структурируются, как таблицы в базе данных или программе электронных таблиц. Другие фиксируют информацию о состоянии окружающего мира, но куда более гетерогенным способом. Возможно, это большая совокупность текстов с изображениями и ссылками, как Википедия, или сложная смесь из заметок и результатов анализов, как медицинские записи пациента.

По правде говоря, эта книга сосредоточится на том, как работать со структурированными данными. Данные зачастую приводятся в виде *матрицы* (matrix), где ряды представляют отдельные элементы или записи, а столбцы — отдельные свойства этих элементов. Например, набор данных о городах Америки мог бы содержать по одному ряду для каждого города со столбцами, представляющими штат, население и площадь.

Сталкиваясь с неструктурированным источником данных, таким как коллекция записей из Twitter, первым делом обычно строят матрицу для их структурирования. Модели *набора слов* (bag of words) строят матрицу с рядами для каждой записи Twitter со столбцами для каждого часто используемого словарного слова. Теперь матричная запись $M[i, j]$ означает, что количество i записей Twitter содержит слово j. Такие матричные формулировки мотивируют наше обсуждение линейной алгебры в главе 8.

1.3.2. Количественные данные или качественные

Количественные данные (quantitative data) состоят из таких числовых значений, как рост и вес. Эти данные могут быть непосредственно включены в алгебраические формулы и математические модели либо отображены в обычных графиках и диаграммах.

В отличие от них, *качественные данные* (categorical data) состоят из меток, описывающих такие свойства исследуемых объектов, как пол, цвет волос и профессия. Эта описательная информация может быть не менее точной и осмысленной, чем числовые данные, но для ее обработки требуются иные методики.

Категориальные данные обычно могут быть закодированы численно. Например, пол может быть представлен как *male* = 0 или *female* = 1 (мужской = 0 или женский = 1). Но все усложняется, когда объект содержит более двух персонажей, особенно если между ними нет неявного порядка. Мы можем закодировать цвета волос в виде чисел, назначив каждому оттенку отдельное значение, например седые волосы = 0, рыжие волосы = 1 и светлые волосы = 2. Однако мы не можем по-настоящему трактовать эти значения как числа для чего-либо,

кроме простого описания личности. Есть ли смысл говорить о максимальном или минимальном цвете волос? Какова интерпретация моего цвета волос минус ваш цвет волос?

Большая часть того, что мы делаем в этой книге, будет вращаться вокруг числовых данных. Но следите за качественными характеристиками и методами, которые на них работают. Методы классификации и кластеризации можно рассматривать как генерирование категориальных меток из числовых данных, и они будут в центре внимания в этой книге.

1.3.3. Большие данные или небольшие

Наука о данных попала на глаза широкой публике с появлением *больших данных* (big data), анализом массивных наборов данных, получаемых из компьютерных журналов регистрации и сенсоров устройств. В принципе, наличие большего количества данных всегда лучше, чем их недостаток, поскольку в случае необходимости всегда можно отбросить их часть или сделать выборку, чтобы получить меньший набор.

Большие данные — это захватывающее явление, и мы обсудим их в главе 12. Однако на практике в работе с большими наборами данных возникают трудности. Повсюду в этой книге мы будем рассматривать алгоритмы и полезные советы по анализу данных. Все становится куда более трудным, как только объем данных становится слишком большим. К проблемам больших данных относятся следующие.

- *Рост размера данных замедляет их анализ.* Вычислительные операции с наборами данных занимают все больше времени по мере увеличения их объемов. Маленькие электронные таблицы предоставляют ответ мгновенно, позволяя вам экспериментировать и играть во *что если*? Но большие электронные таблицы могут быть очень медленными и неповоротливыми, обработка достаточно массивных наборов данных может занимать часы или дни.

  Хитрые алгоритмы могут позволить делать удивительные вещи с большими данными, но анализ и исследование маленьких наборов обычно куда быстрее.

- *Большие наборы данных сложней визуализировать.* Графики с миллионами точек невозможно отобразить на экране компьютера или распечатать, не говоря уже о понимании концепций. Как мы можем надеяться что-либо действительно понять, когда мы не можем это видеть?

- *Простые модели не требуют ни больших данных, ни вычислений.* Типичная задача науки о данных могла бы заключаться в принятии решения

(скажем, должен ли я предложить страхование жизни?) на основе небольшого количества переменных: например возраст, пол, рост, вес и наличие или отсутствие определенных медицинских показаний.

Если у меня есть эти данные по 1 миллиону человек с соответствующими результатами продолжительности жизни, то я смог бы построить хорошую общую модель риска. Это, вероятно, не помогло бы мне создать существенно лучшую модель, если бы у меня были подобные данные по сотням миллионов людей. Критерии принятия решения только по нескольким переменным (например, возраст и боевой статус) не могут быть слишком сложными и должны быть надежными для большого числа претендентов. Любое наблюдение, которое настолько тонко, что требует обработки массированных данных, окажется неподходящим для крупного бизнеса.

Большие данные иногда называют *плохими данными*. Их сбор зачастую оказывается побочным продуктом некой системы или процедуры, а не результатом целенаправленного сбора для ответа на ваш насущный вопрос. В результате нам, возможно, придется пойти на героические усилия, чтобы извлечь хоть какой-то смысл из того, что имеется.

Рассмотрите проблему получения тенденций в предпочтениях избирателей среди кандидатов в президенты. Подход больших данных мог бы подразумевать анализ множества высказываний в Twitter или Facebook, а также интерпретацию намеков на их мнения в тексте. Подход малых данных мог бы подразумевать проведение опроса среди нескольких сотен человек, ответы которых на некий вопрос сводятся в таблицу результатов. Так какой подход, по-вашему, окажется более точным? Правильный набор данных — это тот, который более других относится непосредственно к вашей задаче, а не самый большой набор данных.

> *На заметку*. Не стремитесь безоглядно анализировать большие наборы данных. Ищите *правильные* данные, отвечающие на заданный вопрос, а не самые большие, которые вы можете заполучить в свои руки.

1.4. Классификация и регрессия

Два типа проблем регулярно возникают в традиционной науке о данных и приложениях распознавания образов — это трудность классификации и регрессии. По мере написания книги я добавлял обсуждения алгоритмических подходов к решению этих проблем в последующие главы, поэтому они позволят

извлечь пользу из четкого понимания базового материала о манипулировании данными, статистике, визуализации и математическом моделировании.

Однако я буду упоминать проблемы, связанные с классификацией и регрессией, по мере их возникновения, поэтому имеет смысл сделать паузу для быстрого рассмотрения этих проблем, чтобы помочь вам распознать их, когда вы их встретите.

- *Классификация*. Мы нередко пытаемся поставить метку на элемент из дискретного набора возможностей. Такие задачи, как предсказание победителя конкретного спортивного соревнования (команда *A* или команда *B*?) или выбор жанра некого фильма (комедия, драма или анимация?), являются *задачами классификации* (classification problem), поскольку каждая подразумевает выбор метки из нескольких возможных.
- *Регрессия*. Другая наиболее популярная задача заключается в том, чтобы предсказать некое числовое значение. Прогноз веса человека или количества снежинок, выпавших в этом году, является *задачей регрессии* (regression problem), когда мы предсказываем будущее значение числовой функции в терминах предыдущих значений и других подходящих средств.

Возможно, наилучший способ демонстрации различия заключается в том, чтобы рассмотреть множество задач науки о данных и пометить (классифицировать) их как регрессию или классификацию. Для решения задач этих двух типов используются разные алгоритмические методы, хотя к тем же самым вопросам можно нередко подходить таким образом.

- Будет цена некой акции выше или ниже завтра? (классификация)
- Что будет с ценой некой акции завтра? (регрессия)
- Действительно ли этому человеку имеет смысл продать страховой полис? (классификация)
- Как долго этот человек, по нашим оценкам, будет жить? (регрессия)

Держите свои глаза открытыми для задач классификации и регрессии, поскольку вы встретитесь с ними и в жизни, и в этой книге.

1.5. Видеоматериал: The Quant Shop

Я полагаю, что для усвоения основных принципов необходим практический опыт. Таким образом, когда я преподаю науку о данных, мне нравится задавать каждой студенческой группе интересную, но сложную задачу по прогнозированию, которая требует от них построения и вычисления прогнозирующей модели.

Эти конкурсы по прогнозированию связаны с событиями, где студенты могут проверить свои прогнозы. Они начинают с самого начала: поиск подходящих наборов данных, создание собственных вычислительных сред и разработка своих моделей. И наконец, я заставляю их отследить событие и его ход, чтобы засвидетельствовать правоту или ложность их прогноза.

В качестве эксперимента мы документировали развитие проекта каждой группы на видео осенью 2014 года. После профессионального редактирования получился видеоматериал *The Quant Shop* в стиле телесериала, посвященный науке о данных для общей аудитории. Восемь эпизодов этого первого сезона доступны по адресу `http://www.quant-shop.com` и включают следующее.

- *Finding Miss Universe* (Прогноз победителя конкурса Мисс мира). Ежегодный конкурса Мисс мира стремится выявить самую большую красавицу в мире. Могут ли вычислительные модели спрогнозировать победителя конкурса красоты? Красота субъективна, или алгоритмы могут предсказать, кто самый красивый из всех?
- *Modeling the Movies* (Моделирование фильмов). Кинобизнес задействует множество важных анализов данных. Мы можем строить модели для прогнозирования, какой фильм получит на Рождество "грязными" больше всех? Какие актеры получат награды за свое усердие?
- *Winning the Baby Pool* (Детский клуб). Вес при рождении — важнейший фактор в оценке здоровья новорожденного ребенка. Но как мы можем точно спрогнозировать вес ребенка до фактического рождения? Как данные могут разъяснить экологические риски в развитии беременности?
- *The Art of the Auction* (Аукцион произведений искусств). Самые ценные картины в мире продают на аукционах лицу, предлагающему самую высокую цену. Но можем ли мы спрогнозировать, за сколько миллионов будет продано конкретное полотно Уильяма Тернера? Могут ли компьютеры разобрать художественную ценность покупки?
- *White Christmas* (Белое Рождество). Прогноз погоды — это, вероятно, самая знакомая область прогнозирующего моделирования. Краткосрочные прогнозы обычно точны, но как насчет долгосрочного прогноза? Какие места осчастливит снежное Рождество в этом году? И сможете ли вы предсказать это за один месяц заранее?
- *Predicting the Playoffs* (Предсказание решающих встреч). В спортивных событиях есть победители и проигравшие, а букмекеры счастливы взять ваши ставки на результат любого матча. Способна ли статистика помочь спрогнозировать, какая футбольная команда выиграет чемпионат Супербоул? Может ли алгоритм Google PageRank выбирать победителей на поле точно так же, как он это делает в вебе?

- *The Ghoul Pool* (Клуб вампиров). Смерть настигает всех людей, но когда? Можем ли мы применить страховые модели к знаменитостям и решить, кто следующим умрет? Подобный анализ лежит в основе работы систем страхования жизни, где точные прогнозы продолжительности жизни необходимы для установки страховых взносов, которые и справедливы, и недороги.
- *Playing the Market* (Игры на рынке). Хедж-фонды богатеют, когда правильно угадывают завтрашние цены, и беднеют, когда ошибаются. Насколько точно мы можем прогнозировать будущие цены на золото и нефть, используя историю данных о ценах? Какая другая информация связана с построением успешной модели цены?

Рис. 1.8. Захватывающие сцены из видеоматериала The Quant Shop

Я предлагаю смотреть определенные эпизоды сериала *The Quant Shop* при чтении этой книги. Мы пытаемся сделать это развлечением, хотя я уверен, что вы найдете там много важных вещей. Каждый эпизод длится тридцать минут и, возможно, вдохновит вас заниматься собственным конкурсом прогноза.

Данная программа, конечно, даст вам больше понимания об этих восьми конкурсах. Я буду использовать эти проекты повсюду в книге для иллюстрации важных уроков о том, как делать исследования в науке о данных, а также в качестве положительных и отрицательных примеров. Эти проекты представляют собой лабораторию, позволяющую увидеть, как интеллектуальные, но неопытные люди, не существенно отличающиеся от вас, непосредственно обдумывают задачу науки о данных, и что происходит затем.

1.5.1. Конкурсы Kaggle

Другой источник вдохновения — конкурсы от Kaggle (`www.kaggle.com`), где предоставляется соревновательный форум для аналитиков данных. Там регулярно регистрируются новые конкурсы, отражающие конкретные проблемы, есть учебные данные и функция рейтингов по скрытым оценочным данным. Список лидеров отображает баллы самых сильных участников, таким образом, вы можете видеть, насколько хорошо ваша модель двигается по списку вверх по сравнению с вашими конкурентами. Победители излагают свои тайны моделирования во время интервью после конкурса, чтобы помочь вам улучшить свои навыки моделирования.

Хороший рейтинг на конкурсах Kaggle — это превосходный сертификат для вашего профессионального резюме, чтобы получить хорошую работу как аналитика данных. Потенциальные работодатели разыщут вас, если вы станете реальной звездой Kaggle. Однако настоящая причина участия в конкурсах состоит в том, что его задачи интересны и занимательны, а практика помогает сделать вас лучшим аналитиком данных.

Упражнения в конце каждой главы указывают на прошедшие конкурсы Kaggle, тесно связанные с материалом данной главы. Имейте в виду, что Kaggle демонстрирует вводящий в заблуждение гламурный взгляд на науку о данных как приложение машинного обучения, поскольку он представляет чрезвычайно четко определенные проблемы, подразумевающие тяжелую работу по сбору и очистке данных, которая уже была выполнена за вас. Тем не менее я призываю вас ознакомиться с этим для вдохновения и в качестве источника данных для новых проектов.

1.6. О случаях из жизни

Гений и мудрость — это два разных интеллектуальных дара. *Гений* проявляется в поиске правильного ответа, создании образных умственных скачков, которые преодолевают препятствия и выигрывают конкурсы. *Мудрость* проявляется в избегании препятствий, она обеспечивает чувство направления или луч света, который заставляет нас двигаться в правильном направлении.

Гений проявляется в технической силе и глубине, способности видеть и делать такие вещи, которые другие люди не могут. Мудрость, напротив, возникает из опыта и общего знания. Она возникает, когда слушаешь других. Мудрость происходит из терпеливого наблюдения, из того, как часто вы оказывались неправы в прошлом, и выяснения, почему вы были неправы, чтобы лучше понять признаки ловушек в будущем и избегать их.

Наука о данных, как и большинство вещей в жизни, извлекает больше пользы из мудрости, чем из гения. В этой книге я пытаюсь передать вам мудрость, которую я скопил на трудном пути через многие *случаи из жизни*, встречавшиеся в разнообразных проектах, над которыми я работал.

- *Крупномасштабная текстовая аналитика и NLP.* Моя лаборатория науки о данных в Университете штата Нью-Йорк в Стоуни-Брук работает над множеством проектов больших данных, включая анализ отношения в социальных сетях, анализ исторических тенденций, подходы глубокого обучения к *обработке текстов на естественном языке* (Natural Language Processing — NLP) и извлечение признаков из сетей.
- *Новые компании.* Я был соучредителем и руководителем исследовательских работ в двух компаниях по анализу данных: General Sentiment и Thrivemetrics. General Sentiment анализировали крупномасштабные текстовые потоки из новостей, блогов и социальных сетей, чтобы выявить тенденции в отношении (положительные или отрицательные), связанные с людьми, местами и вещами. Компания Thrivemetrics применяла анализ этого типа к внутренним корпоративным коммуникациям, системе обмена сообщениями и электронной почте.

  Ни одно из этих предприятий не сделало меня достаточно богатым, чтобы воздержаться от авторского гонорара за эту книгу, но они действительно снабдили меня практическим опытом в основанных на облаке вычислительных системах и понимании того, как данные используются в промышленности.
- *Сотрудничество с реальными учеными.* У меня было несколько интересных случаев сотрудничества с биологами и социологами, которые помогли сформировать мое понимание сложностей работы с реальными данными. Экспериментальные данные ужасно неточны и полны ошибок, но все же вы должны сделать все, что только можете, с тем, что вы имеете, чтобы узнать, как устроен мир.
- *Создание игорных систем.* Особенно забавным был проект по построению системы для прогнозирования результатов матчей хай-алай (jai-alai), чтобы можно было делать ставки на них. Это было изложено в моей книге *Calculated Bets: Computers, Gambling, and Mathematical Modeling to Win* [8]. Наша система полагалась на веб для сбора данных, статистический анализ, симуляцию, моделирование и тщательную оценку. Мы также разработали и рассчитали прогнозирующие модели для кассовых сборов по фильмам [9], курсам акций [10] и футбольным играм [11], используя анализ социальной сети.

- *Ранжирование исторических фигур.* При анализе Википедии для извлечения значащих переменных для более чем 800 000 исторических фигур мы разработали функцию выигрыша, чтобы ранжировать их на основании упоминаний. Это ранжирование — громадная работа, отделяющая самых великих из великих (Иисус, Наполеон, Шекспир, Магомет и Линкольн — это только первые пять) от меньших смертных, послужившая основой для нашей книги *Who's Bigger?: Where Historical Figures Really Rank* [12].

Весь этот опыт управляет тем, чему я учу в данной книге, особенно рассказы, где я описываю случаи из реальной жизни. *Каждые из этих случаев — истина.* Конечно, истории немного улучшены в пересказе, чтобы сделать их более интересными для чтения. Тем не менее я попытался честно пересказать процесс перехода от первоначальной проблемы к решению, чтобы вы могли понять, как это происходило.

1.7. Случай из жизни: ответ на правильный вопрос

Наша исследовательская группа в университете штата Нью-Йорк в Стоуни-Брук разработала NLP-ориентированную систему для анализа миллионов новостей, блогов и сообщений социальных сетей и сведения этих текстов к тенденциям относительно всех обсуждаемых сущностей. Подсчет количества упоминаний каждого имени в текстовом потоке относительно прост. Определить, положительна коннотация конкретной ссылки или отрицательна (анализ отношения), трудно. Но наша система выполняла довольно хорошую работу, особенно когда объединяла большое количество ссылок.

Эта технология послужила основанием для компании General Sentiment по анализу социальных сетей. Она удачно пережила этап становления, преодолев проблемы поиска денег, найма персонала и разработки новых продуктов.

Но, вероятно, самой большой проблемой, с которой мы столкнулись, оказался ответ на правильный вопрос. Система General Sentiment фиксировала тенденции в отношении и объемах упоминаний для *каждого* человека, места и вещи, которые когда-либо упоминалась в новостях, блогах и социальных сетях: более 20 млн отдельных сущностей. Мы контролировали репутацию знаменитостей и политических деятелей. Мы контролировали судьбы компаний и товаров. Мы отслеживали результативность спортивных команд и слухи о фильмах. Мы могли сделать что угодно!

Однако за *что угодно* никто не платит. Платят всегда за *нечто*, за решение конкретной задачи или устранение определенного момента, который мешает бизнесу. Оказывается, быть способным сделать что угодно — это плохая коммерческая стратегия, поскольку она требует от вас поиска новых возможностей заново для каждого клиента.

Facebook не был открыт миру до сентября 2006 года. Таким образом, когда компания General Sentiment начала работу в 2008 году, мы были в самом начале эры социальных сетей. К нам был большой интерес со стороны крупных брендов и рекламных агентств, которые *знали* о грядущем взрыве социальных сетей. Они *знали*, что эта новомодная вещь важна и что они должны быть там. Они *знали*, что надлежащий анализ данных социальных сетей может дать им новое представление о том, что думают их клиенты. Но они не знали точно, что именно они хотели знать.

Один производитель авиадвигателей очень заинтересовался тем, что дети говорят о них в Facebook. Мы вынуждены были разочаровать его, поскольку ответ был нулевым. Другие потенциальные клиенты требовали доказательств, что наши оценки были точней телевизионных рейтингов Nielsen. Ну, конечно, если вы хотите рейтинги от Nielsen, вы должны покупать их у Nielsen. Наша система предоставляла иные возможности проникновения в суть совершенно иного мира. Но вы должны были знать то, как вы хотели бы использовать их.

Нам удалось получить существенные контракты от разных групп клиентов, включая такие потребительские бренды, как Toyota и Blackberry, правительственные организации, такие как офис туризма на Гавайях, и даже президентскую кампанию республиканского кандидата Митта Ромни (Mitt Romney) в 2012 году. Наши аналитики предоставили им информацию о широком спектре деловых вопросов.

- Что люди думают о Гавайях? (Ответ: они думают, что это очень хорошее место для посещений.)
- Как быстро репутация Тойоты восстановится после новостей о серьезных проблемах с тормозами на их автомобилях? (Ответ: приблизительно через шесть месяцев.)
- Что люди думают о новых моделях телефонов Blackberry? (Ответ: им куда больше нравится iPhone.)
- Как быстро репутация Ромни восстановится после оскорбления 47% электората в записанной речи? (Ответ: никогда.)

Но каждая продажа требовала вникнуть в новую вселенную, задействовав значительные усилия и воображение со стороны нашего коммерческого штата и аналитиков. Нам ни разу не удалось заполучить двух клиентов из одной отрасли, что позволило бы нам извлечь выгоду из масштаба и накопленной мудрости.

Конечно, клиент всегда прав. Это была наша ошибка, что мы не могли объяснить им наилучший способ использования нашей технологии. Отсюда урок: мир не протопчет тропу к вашей двери только для получения нового источника данных. Вы должны научиться задавать правильные вопросы, прежде чем сможете превратить данные в деньги.

1.8. Дополнительная информация

Идея использования исторических записей об игроках в бейсбол показала, что у левшей жизнь короче, согласно Халперн и Корен [4, 13], но их заключение остается спорным. Процент левшей в населении быстро растет, и наблюдаемые тенденции могут быть следствием разброса [14]. Так что, левши, не вешайте нос! Откровенно говоря, я один из вас.

Дисциплину количественного анализа бейсбола иногда называют *саберметрикой* (sabermetrics), а ее ведущим специалистом является парень по имени Билл Джеймс (Bill James). Я рекомендую начинающим аналитикам данных прочитать его книгу *Historical Baseball Abstract* [15] в качестве отличного примера того, как человек превращает числа в знания и понимание. Журнал *Time Magazine* однажды сказал о Джеймсе: "Особое удовольствие от его чтения исходит из экстравагантного спектакля первоклассного ума, израсходованного на бейсбол". Я благодарю `http://sports-reference.com` за разрешение использовать изображения с их веб-сайта в этой книге. А также Amazon, владельца IMDb.

Потенциал систем совместного использования автомобилей в Нью-Йорке был изучен Паоло Санти (Paolo Santi) и др. [16], он показал, что почти 95% поездок могут быть совместными при задержках не более пяти минут на поездку.

Система Lydia для анализа настроений описана в публикации [17]. В публикации [18] описаны способы определения изменений в значении слова по результатам анализа исторических текстов подобно N-граммам Google, как сообщается в публикации [18].

1.9. Упражнения

Идентификация наборов данных

1.1. *[3]* Укажите, где находятся в вебе интересные наборы данных, подходящие для следующих областей.
(a) Книги.
(b) Скачки.
(c) Курсы акций.
(d) Риски болезней.
(e) Колледжи и университеты.
(f) Индексы преступности.
(g) Наблюдение за птицами.
Для каждого из этих источников данных объясните, что следует сделать, чтобы превратить данные в формат, пригодный для использования на вашем компьютере для анализа.

1.2. *[3]* Предложите подходящие источники данных для следующих конкурсов по прогнозированию для передачи *The Quant Shop*. Укажите, какие источники данных, вы уверены, должны *у кого-то* быть, и какие данные вполне определенно доступны для вас.
(a) *Miss Universe*.
(b) *Movie gross*.
(c) *Baby weight*.
(d) *Art auction price*.
(e) *White Christmas*.
(f) *Football champions*.
(g) *Ghoul pool*.
(h) *Gold/oil prices*.

1.3. *[3]* Посетите сайт `http://data.gov` и выявите пять наборов данных, которые покажутся вам интересными. Для каждого сделайте краткое описание и предложите три интересные вещи, которые вы могли бы сделать с ними.

Задайте вопросы

1.4. *[3]* Для каждого из следующих источников данных предложите три интересных вопроса, на которые вы сможете ответить, проанализировав их.
(a) Данные расчетов по кредитным карточкам.
(b) Данные о щелчках на `http://www.amazon.com`.
(c) Справочники адресов и телефонов.

1.5. *[5]* Посетите портал Национального центра биотехнологической информации (NCBI). Посмотрите, какие источники данных доступны, особенно ресурсы Pubmed и Genome. Предложите три интересных проекта исследования каждого из них.

1.6. *[5]* Вы хотели бы провести эксперимент и установить, предпочитают ли ваши друзья вкус обычной кока-колы или диетической. Коротко набросайте проект такого исследования.

1.7. *[5]* Вы хотели бы провести эксперимент и установить, учатся ли студенты лучше, если они делают это без музыки, с инструментальной музыкой или с лирическими песнями. Коротко набросайте проект такого исследования.

1.8. *[5]* Традиционные опросы, в частности Gallup, используют такую процедуру, как звонок по случайному номеру, который состоит из строки случайных цифр, а не выбирается из телефонной книги. Укажите, почему такие опросы проводятся с использованием случайных наборов цифр в номере.

Реализация проектов

1.9. *[5]* Напишите программу для очистки списка книжных бестселлеров на Amazon.com. Используйте ее для составления графика рангов всех книг Скиены за все время. Какая из этих книг должна быть следующей вашей покупкой? Если у вас есть друзья, которым вы желаете добра, то не хотите ли вы сделать им ценный подарок?

1.10. *[5]* Создайте для вашего любимого вида спорта (бейсбол, американский футбол, баскетбол, крикет или футбол) набор данных с историческими статистическими записями обо всех главных участниках. Разработайте и реализуйте систему рангов, чтобы выявить наилучшего игрока в каждой позиции.

Вопросы на интервью

1.11. *[3]* Для каждого из следующих вопросов: (1) сделайте быстрое предположение на основании только вашего понимания мира, а затем (2) используйте Google для поиска приемлемых чисел, чтобы получить более принципиальную оценку. Насколько отличаются ваши две оценки?

(a) Сколько настройщиков фортепьяно есть во всем мире?

(b) Сколько весит лед на хоккейном поле?

(c) Сколько бензоколонок в Соединенных Штатах?

(d) Сколько людей входит и выходит из аэропорта Ла Гуардия каждый день?

(e) Сколько галлонов (3,78 литра) мороженого продается в Соединенных Штатах каждый год?

(f) Сколько баскетболистов покупает Национальная баскетбольная ассоциация (NBA) каждый год?
(g) Сколько рыб плавает во всех океанах в мире?
(h) Сколько людей во всем мире летит в воздухе прямо сейчас?
(i) Сколько теннисных мячей может вместить большой коммерческий самолет?
(j) Сколько миль (1609,34 метра) мощеных дорог в вашей любимой стране?
(k) Сколько долларов находится в бумажниках всех людей в университете Стоуни-Брук?
(l) Сколько галлонов бензина продает в день типичная бензоколонка?
(m) Сколько слов в этой книге?
(n) Сколько котов живет в Нью-Йорке?
(o) Сколько стоило бы заполнить бензобак типичного автомобиля кофе из Starbuck?
(p) Сколько чая в Китае?
(q) Сколько текущих счетов в Соединенных Штатах?

1.12. *[3]* В чем разница между регрессией и классификацией?
1.13. *[8]* Как вы построили бы управляемую данными систему рекомендаций? Каковы ограничения этого подхода?
1.14. *[3]* Как вы заинтересовались наукой о данных?
1.15. *[3]* Как, по-вашему, наука о данных — это искусство или наука?

Конкурсы Kaggle

1.16. Кто пережил кораблекрушение "Титаника"?
`https://www.kaggle.com/c/titanic`
1.17. Куда какое такси едет?
`https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i`
1.18. Сколько времени займет конкретная поездка на такси?
`https://www.kaggle.com/c/pkdd-15-taxi-trip-time-prediction-ii`

Глава 2

Математические основы

> Аналитик данных — это тот, кто знает о статистике больше, чем программист, и больше об информатике, чем статистик.
>
> — Джош Блюменсток (Josh Blumenstock)

Вы должны научиться ходить, прежде чем сможете бегать. Аналогично существует определенный уровень математической зрелости, достичь которого необходимо прежде, чем вам станут доверять выполнение чего-либо значимого с числовыми данными.

Создавая эту книгу, я предполагал, что у читателей есть некоторый уровень знаний в областях вероятности и статистики, линейной алгебры и непрерывной математики. Я также предполагал, что они, вероятно, забыли большинство из этого или, возможно, не всегда видят лес (почему вещи важны, и как их использовать) за деревьями (все подробности определений, доказательства и операции).

Эта глава будет пытаться обновить ваше понимание определенных элементарных математических концепций. Следуйте за мной и вытаскивайте ваши старые учебники на случай, если понадобится справка. Более сложные концепции будут представлены в книге по мере необходимости.

2.1. Вероятность

Теория вероятности служит формальной основой для рассуждения о вероятности событий. Поскольку это формальная дисциплина, существует множество взаимосвязанных определений, позволяющих точно определить, о чем мы рассуждаем.

- *Эксперимент* (experiment) — это процедура, приводящая к одному из нескольких возможных результатов. В качестве примера будем рассматривать эксперимент, в ходе которого бросают две шестигранные игральные кости (кубика), одна красного, другая синего цвета, на каждой грани которых расположены целые числа $\{1, \ldots, 6\}$.

- *Выборочное пространство* (sample space) S — это набор возможных результатов эксперимента. В нашем примере игральных костей есть 36 возможных результатов, а именно

  $S = \{(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6), (3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6), (4, 1), (4, 2), (4, 3), (4, 4), (4, 5), (4, 6), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 6), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6)\}$.

- *Событие* (event) E является определенным подмножеством результатов эксперимента. Событие, когда сумма игральных костей равна 7 или 11 (условия победы в костях при первом броске), является подмножеством $E = \{(1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1), (5, 6), (6, 5)\}$.
- *Вероятность результата* (probability of an outcome) s обозначается как $p(s)$ и является числом со следующими двумя свойствами.
- Для каждого результата s в выборочном пространстве S, $0 \le p(s) \le 1$.
- Сумма вероятностей всех результатов равна единице: $\sum_{s \in S} p(s) = 1$.

Если мы используем две разные настоящие игральные кости, то вероятность $p(s) = (1/6)(1/6) = 1/36$ для всех результатов $s \in S$.

- *Вероятность события* (probability of an event) E является суммой вероятностей результатов эксперимента. Таким образом,

  $$p(E) = \sum_{s \in E} p(s).$$

  Дополнительная формулировка в терминах *дополнения* (complement) события $\overline{E}$ — это случай, когда E не происходит. Таким образом,

  $$P(E) = 1 - P(\overline{E}).$$

  Это весьма полезно, поскольку зачастую бывает проще проанализировать $P(\overline{E})$, чем непосредственно $P(E)$.

- *Случайная переменная* (random variable) V является числовой функцией в пространстве вероятности результата. Функция "суммирования значений двух игральных костей" ($V(a, b) = a + b$) дает целочисленный результат от 2 до 12. Это подразумевает *распределение вероятностей* (probability distribution) значений случайной переменной. Вероятность $P(V(s) = 7) = 1/6$, как было показано ранее, в то время как $P(V(s) = 12) = 1/36$.

- *Ожидаемое значение* (expected value) случайной переменной V, определенной в выборочном пространстве S, т.е. $E(V)$ определяется как

  $$E(V) = \sum_{s \in S} p(s) V(s).$$

Все это вы, по-видимому, уже видели прежде. Но это язык, который мы будем использовать для связи между вероятностью и статистикой. Данные, которые мы обычно видим, поступают из измеряемых свойств наблюдаемых событий. Теория вероятности и статистика предоставляют инструментальные средства для анализа этих данных.

2.1.1. Вероятность против статистики

Вероятность и статистика — это связанные области математики, занимающиеся анализом относительной частоты событий. Однако есть фундаментальные различия в способе, которым они видят мир.

- *Вероятность* занимается предсказанием вероятности будущих событий, в то время как *статистика* задействует анализ частоты прошлых событий.
- *Вероятность* — это прежде всего теоретическая ветвь математики, которая изучает следствия математических определений. *Статистика* — это прежде всего прикладная ветвь математики, которая пытается извлечь смысл из наблюдений в реальном мире.

Обе темы важны, актуальны и полезны. Но это разные темы, и понимание их различий крайне важно для надлежащей интерпретации значимости математического доказательства. Многие игроки сошли в холодную одинокую могилу из-за того, что были не в состоянии понять различие между вероятностью и статистикой.

Возможно, это различие станет более ясным, если мы проследим мыслительный процесс математика, впервые севшего играть в кости.

- Если бы этот математик был *специалистом в области теории вероятности* (probabilist), то, глядя на игральную кость, он бы подумал: "Шестисторонняя игральная кость? Каждая грань кости, по-видимому, будет выпадать с одинаковой вероятностью. Теперь, *предположив*, что каждая грань выпадает с вероятностью 1/6, я могу убедиться, что мои шансы невелики".
- Если бы на его месте оказался статистик, он, глядя на игральную кость, думал бы так: "Как мне *узнать*, что они без грузиков? Я послежу некоторое время и увижу, как часто выпадает каждое число. Затем я смогу решить, совместимы ли мои наблюдения с предположением о равной вероятности граней. Однажды я стану достаточно уверен в том, что кости беспристрастны, и тогда я вызову специалиста в области теории вероятности, чтобы он подсказал мне, как делать ставки".

Таким образом, теория вероятности позволяет нам находить последствия данного идеального мира, в то время как статистическая теория позволяет измерить степень, до которой наш мир идеален. Это постоянное напряжение между теорией и практикой заключается в том, что статистики оказываются угнетенной группой людей по сравнению со счастливыми специалистами в области теории вероятности.

Современная теория вероятностей впервые появилась из таблиц с костями во Франции в 1654 году. Французский дворянин Антуан Гомбо, шевалье де Мере (Chevalier de Méré), задался вопросом, имеет ли игрок или дом (house) преимущество в конкретной азартной игре.[1] В базовой версии игрок бросает четыре кости, и побеждает при условии, что ни одна из них не выпадает на 6. Дом забирает деньги за ставку, если выпадает хотя бы одна шестерка.

Де Мере привлек к этой проблеме внимание французских математиков Блеза Паскаля (Blaise Pascal) и Пьера Ферма (Pierre de Fermat), сформулировавшего известную Великую теорему Ферма. Совместно эти мужи выработали основы теории вероятности, заодно установив, что дом имеет в этой игре преимущество с вероятностью $p = 1 - (5/6)^4 \approx 0{,}517$, где вероятность $p = 0{,}5$ означала бы беспристрастную игру, где дом выигрывает точно половину раз.

2.1.2. Составные события и независимость

Нас интересуют *сложные события* (complex event), вычисляемые из простых событий A и B на том же самом наборе результатов. Возможно, событие A — это то, что по крайней мере на одной из двух игральных костей выпадет четное число, в то время как событие B обозначает выпадение в сумме 7 или 11. Обратите внимание, что есть определенные результаты A, которые не являются результатами B, а именно

$$A - B = \{(1,2),(1,4),(2,1),(2,2),(2,3),(2,4),(2,6),(3,2),(3,6),(4,1),(4,2),(4,4),(4,5),(4,6),(5,4),(6,2),(6,3),(6,4),(6,6)\}.$$

Это операция *разности множеств*. Заметьте, что здесь $B - A = \{\}$, поскольку каждая пара, имеющая в сумме 7 или 11, должна обязательно содержать одно нечетное и одно четное число.

Результаты, общие для обоих событий A и B, являются *пересечением* и обозначаются как $A \cap B$. Это можно записать как

$$A \cap B = A - (S - B).$$

[1] В действительности ему не стоило задаваться вопросом. Дом всегда имеет преимущество.

Результаты, которые принадлежат либо A, либо B, являются *объединением* и обозначаются как $A \cup B$. Операция дополнения $\overline{A} = S - A$ предоставляет богатейший язык для комбинирования событий, представленных на рис. 2.1. Мы легко можем вычислить вероятность любого из этих наборов, сложив вероятности результатов в определенных наборах.

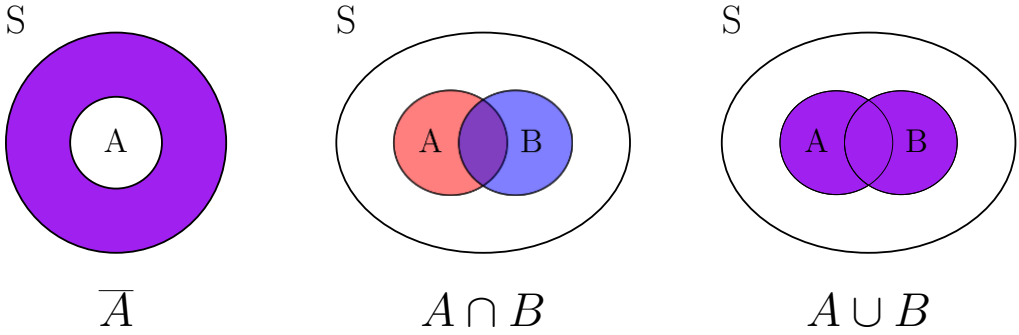


Рис. 2.1. Диаграммы Венна, иллюстрирующие разность множеств (слева), пересечение (середина) и объединение (справа)

События A и B независимы, если и только если

$$P(A \cap B) = P(A)\,P(B).$$

Это значит, что нет никакой специальной структуры результатов, общих для событий A и B. Если предположить, что половина студентов в моей группе — женщины и половина студентов в моей группе выше среднего роста, то мы могли бы ожидать, что четверть моих студентов — это женщины выше среднего роста, если эти события независимы.

Теоретики вероятности любят независимые события, поскольку это упрощает их вычисления. Но аналитики данных обычно так не поступают. При построении моделей для прогнозирования вероятности некоего будущего события B, при знании о некоем предыдущем событии A мы хотим иметь максимально сильную зависимость B от A.

Предположим, что я всегда использую зонтик, если и только если идет дождь. Предположим, что вероятность, при которой идет дождь (событие B), скажем, $p = 1/5$. Следовательно, вероятность, что я беру свой зонтик (событие A), составляет $q = 1/5$. Но даже больше того, если вы знаете, идет ли дождь, вы знаете точно, есть ли у меня зонтик. Эти два события отлично *коррелируются*.

Предположим теперь, что эти события стали независимыми. Тогда

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A),$$

и наличие дождя не оказывает абсолютно никакого влияния на то, беру ли я с собой свое защитное устройство.

Корреляции — это движущая сила прогнозирующих моделей, поэтому мы обсудим, как их измерить и что они означают в разделе 2.3.

2.1.3. Условная вероятность

Когда два события коррелируются, между ними есть зависимость, которая существенно затрудняет вычисления. *Условная вероятность* A для данного B, $P(A|B)$ определяется так:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Вспомним событие броска игральной кости из раздела 2.1.2.

- Событие A — это выпадение четного числа на по крайней мере одной из двух игральных костей.
- Событие B является суммой 7 или 11 на двух игральных костях.

Обратите внимание, что $P(A|B) = 1$, поскольку *любая* нечетная сумма броска должна состоять из одного четного и одного нечетного числа. Таким образом, $A \cap B = B$, аналогично случаю с зонтиком выше. Для $P(B|A)$ обратите внимание на то, что $P(A \cap B) = 9/36$ и $P(A) = 25/36$, следовательно, $P(B|A) = 9/25$.

Условная вероятность будет важна для нас потому, что мы интересуемся вероятностью события A (возможно, что некая часть электронной почты является спамом) как функцией от некоторого доказательства B (возможно, распределения слов в пределах документа). Задачи классификации вообще сводятся к вычислению условных вероятностей, так или иначе.

Нашим главным инструментом вычисления условной вероятности будет *теорема Байеса*, которая меняет направление зависимостей:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

Зачастую оказывается куда проще вычислить вероятности в одном направлении, чем в другом, как в этой задаче. Теорема Байеса $P(B|A) = (1 \cdot 9/36) = (25/36) = 9/25$ дает точно то, что мы получили прежде. Мы вернемся к теореме Байеса в разделе 5.6, где она заложит фундамент вычислительных вероятностей перед лицом доказательства.

2.1.4. Вероятностные распределения

Случайные переменные — это числовые функции, где значения ассоциируются с вероятностями происхождения. В нашем примере, где $V(s)$ — это сумма двух брошенных игральных костей, функция дает целое число между 2 и 12. Вероятность конкретного значения $V(s) = X$ является суммой вероятностей всех результатов, которые составляют в целом X.

Такие случайные переменные могут быть представлены их *функцией плотности вероятностей* (probability density function — pdf). Это график, где ось x представляет диапазон возможных значений случайной переменной, а ось y — вероятность данного значения. На рис. 2.2 (слева) представлена функция pdf суммы двух беспристрастных игральных костей. Обратите внимание, что пик $X = 7$ соответствует самой частой сумме игральных костей с вероятностью 1/6.

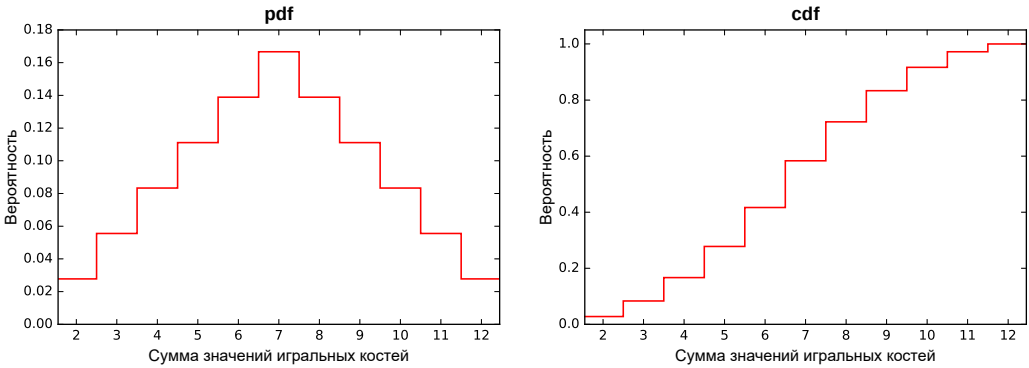


Рис. 2.2. Функция плотности вероятностей (pdf) для суммы двух игральных костей содержит точно ту же самую информацию, что и кумулятивная функция плотности (cdf), но выглядит совсем иначе

У таких графиков pdf есть тесная связь с гистограммами частот данных, где ось x также представляет диапазон значений, но y теперь представляет частоту точного количества событий, наблюдаемых для каждого конкретного значения X. Преобразование гистограммы в функцию pdf может быть выполнено при делении каждой ячейки полной частотой на все ячейки. Сумма элементов должна составить 1, так мы получаем распределение вероятностей.

Гистограммы являются статистическими: они отражают фактические наблюдения за результатами. Функции pdf, напротив, являются вероятностными: они демонстрируют шанс, что у следующего наблюдения будет значение X. Мы зачастую используем гистограмму наблюдений $h(x)$ на практике, чтобы оценить вероятности[2] при нормализации подсчетов по сумме наблюдений.

[2] Такая методика, как *дисконтирование*, предоставляет лучший способ оценки частоты редких событий, она будет обсуждаться в разделе 11.1.2.

$$P(k = X) = \frac{h(k = X)}{\sum_x h(x = X)}.$$

Есть и другой способ представления случайных переменных, который зачастую оказывается полезным, — это *кумулятивная функция плотности* (cumulative density function — cdf). Функция cdf — это текущая сумма (нарастающий итог) вероятностей в функции pdf; как функция от k, она отражает вероятность того, что $X \leq k$ вместо вероятности, что $X = k$. Рис. 2.2 (справа) демонстрирует функцию cdf распределения суммы игральных костей. Значения монотонно увеличиваются слева направо, поскольку каждый член получается из добавления положительной вероятности к предыдущему общему количеству. Крайнее справа значение — это 1, поскольку все результаты дают значение, не превосходящее максимум.

Важно понимать, что функции pdf $P(V)$ и cdf $C(V)$ для данной случайной переменной V содержат *точно* ту же самую информацию. Мы легко можем перемещаться между ними, поскольку

$$P(k = X) = C(X \leq k + \sigma) - C(X \leq k),$$

где $\sigma = 1$ для целочисленных распределений. Функция cdf — это текущая сумма pdf, таким образом,

$$C(X \leq k) = \sum_{x \leq k} P(X = x).$$

Только не забывайте, какое распределение вы рассматриваете. Графики кумулятивного распределения всегда проходят выше, поскольку двигаясь вправо, мы достигаем максимума с вероятностью $C(X \leq \infty) = 1$. В отличие от него, общая площадь под кривой функции pdf равна 1, таким образом, вероятность в любой точке распределения обычно существенно меньше.

Забавный пример различия между кумулятивным и возрастающим распределениями представлен на рис. 2.3. Оба распределения представляют точно те же самые данные по продажам Apple iPhone, но какую кривую генеральный директор Apple Тим Кук (Tim Cook) выбрал бы для представления на общем событии акционеров? Кумулятивное распределение (красный график) демонстрирует взрыв продаж, не так ли? Но это представление обманчиво отображает темпы роста, поскольку стремительный рост является свойством этой функции, а темпы роста трудноразличимы. Действительный график поквартальных продаж (синий) демонстрирует, что частота продаж iPhone фактически снижалась на протяжении последних двух периодов перед демонстрацией.

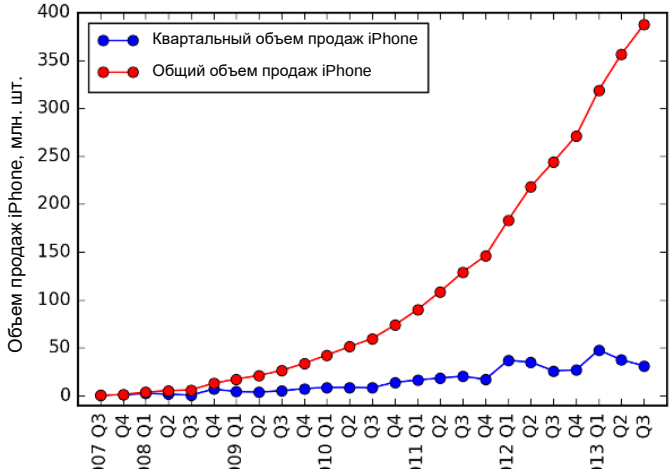


Рис. 2.3. Данные о квартальных объемах продаж iPhone, представленные как кумулятивные и возрастающие (квартальные) распределения. Какую кривую президент Apple Тим Кук выберет для демонстрации?

2.2. Описательная статистика

Описательная статистика предоставляет способы фиксации свойства некого набора данных, или выборки. Они суммируют наблюдаемые данные и предоставляют язык для их обсуждения. Представляя группу элементов новым производным элементом, таким как среднее, минимум, счет или сумма, сводят большой набор данных к меньшей обобщенной статистической величине: т.е. объединение как уплотнение данных.

Такая статистика сама по себе может стать средством получения натуральных групп или кластеров в полном наборе данных. Есть два основных типа описательной статистики.

- *Поиск центра тенденций*, фиксирующий центр, вокруг которого распределяются данные.
- *Вариация* или *поиск дисперсии*, описывающий распределение данных, т.е. насколько далеко показатели располагаются от центра.

Совместно они говорят нам очень много о нашем распределении.

2.2.1. Поиск центра

Первый элемент статистики, которой мы изучаем в школе, это элементарные действия по поиску центра: среднее, медиана и мода. Это правильное место для начала размышлений о едином числе, характеризующем набор данных.

- *Среднее*. Вы, вероятно, вполне привычны к использованию *среднего арифметического*, когда мы суммируем значения и делим на количество наблюдений:

$$\mu_X = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

  Мы можем легко поддерживать среднее значение в потоке вставок и удалений, сохраняя сумму значений отдельно от числа частот и деля только по требованию.

  Среднее значение весьма емко характеризует симметричные распределения без выбросов, такие как рост и вес. Симметричность означает, что количество элементов выше среднего должно быть примерно таким же, как и количество ниже. Отсутствие выбросов означает, что диапазон значений является достаточно монотонным. Обратите внимание, что один MAXINT, попавший в отличную группу, отбрасывает среднее значение прочь. Медиана — это показатель центральности, который более уместен при таких неблагополучных распределениях.

- *Среднее геометрическое*. Это n-й корень произведения n значений:

$$\left(\prod_{i=1}^{n} a_i\right)^{1/n} = \sqrt[n]{a_1 a_2 \ldots a_n}.$$

  Среднее геометрическое всегда меньше или равно среднему арифметическому. Например, средние геометрическое от суммы 36 бросков кости составляет 6,5201, в отличие от среднего арифметического 7. Оно очень чувствительно к значениям, близким к нулю. Единственное нулевое значение гробит среднее геометрическое: независимо от любых других значений в данных, вы получаете нуль. Это несколько похоже на наличие остатков 1 в среднем арифметическом.

  Однако среднее геометрические доказало свою ценность при усреднении коэффициентов. Среднее геометрическое значение 1/2 и 2/1 равно 1, тогда как среднее значение равно 1,25. Есть и менее доступная "комната" для коэффициентов менее 1, чем для коэффициентов выше 1, что создает асимметрию, которая завышает среднее арифметическое. В этих случаях среднее геометрическое имеет большее значение, равно как и среднее арифметическое *логарифмов* коэффициентов.

- *Медиана*. Это срединное значение в наборе данных; выше медианы находится столько же элементов, сколько и ниже. Существует оговорка о том, что взять в качестве медианы, когда у вас есть четное количество элементов. Вы можете взять любой из двух центральных кандидатов: в любом разумном наборе данных эти два значения должны быть примерно одинаковыми. Действительно, в примере с игральными костями, оба являются 7.

  Хорошим свойством медианы по определению является то, что она должна быть подлинным значением исходного потока данных. На самом деле некто вполне может иметь медианный рост, и его пример можно привести, но вряд ли кто в мире имеет *точно* средний рост. Вы теряете это свойство, когда усредняете два центральных элемента.

  Какой показатель центральности лучше всего подходит для применения? Медиана обычно лежит довольно близко к среднему арифметическому в симметричных распределениях, но зачастую бывает интересно увидеть, насколько далеко они расположены друг от друга, и с какой стороны среднего находится медиана.

  Медиана обычно оказывается лучшей статистическая величиной для смещенных распределений или данных с выбросами: таких как богатство и доход. Билл Гейтс добавляет 250 долл. к среднему уровню благосостояния на душу населения в Соединенных Штатах, но не к медианному. Если он заставит вас лично почувствовать себя богаче, тогда идите и используйте среднее. Но медиана здесь является более информативной статистической величиной, поскольку она подходит для любого экспоненциального распределения.

- *Мода*. Это самый частый элемент в наборе данных. Это 7 в нашем примере с игральными костями, поскольку он встречается шесть раз из тридцати шести элементов. Откровенно, я никогда не видел, чтобы мода обеспечивала понимание как показатель центральности, поскольку она зачастую никак не близка к центру. У выборок, измеренных по большому диапазону, должно быть очень немного повторных элементов или коллизий в любом конкретном значении. Это делает моду вопросом случайности. Действительно, наиболее часто встречающиеся элементы зачастую обнаруживают артефакты или аномалии в наборе данных, такие как стандартные значения или коды ошибки, которые на самом деле не представляют собой элементы основного распределения.

  Связанная концепция пика в *плотности распределения* (frequency distribution) (или гистограмме) имеет смысл, но интересные пики обнаруживаются только при правильном балансировании. Текущий пик годового распределения заработной платы в Соединенных Штатах составляет от 30 до 40 тыс. долл. в год, хотя мода, по-видимому, находится на нуле.

2.2.2. Поиск дисперсии

Наиболее распространенной мерой дисперсии является *стандартное отклонение* σ, вычисляемое как сумма квадратов разниц между отдельными элементами и средним:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(a_i - \overline{a})^2}{n-1}}.$$

Связанная с ней статистическая величина, *дисперсия* V, является квадратом стандартного отклонения, т.е. $V = \sigma^2$. Но иногда удобней говорить о дисперсии, чем о стандартном отклонении, поскольку этот термин короче на несколько символов. Но измеряют они точно то же самое. В качестве примера рассмотрим скромную лампочку, которая обычно имеет срок службы, скажем, μ = 3000 часов, происходящий от некоего базового распределения, представленного на рис. 2.4. У обычной лампы шанс продержаться дольше μ тот же самый, что и сгореть быстрее, и эту степень неопределенности измеряет σ. Теперь вообразите "лампу картриджа принтера", где зловредный изготовитель применяет очень надежные лампы, но также ставит и счетчик, чтобы они не могли светиться после 3000 часов использования. Здесь μ = 3000 и σ = 0. У обоих распределений будет одинаковое среднее, но существенно разная дисперсия.

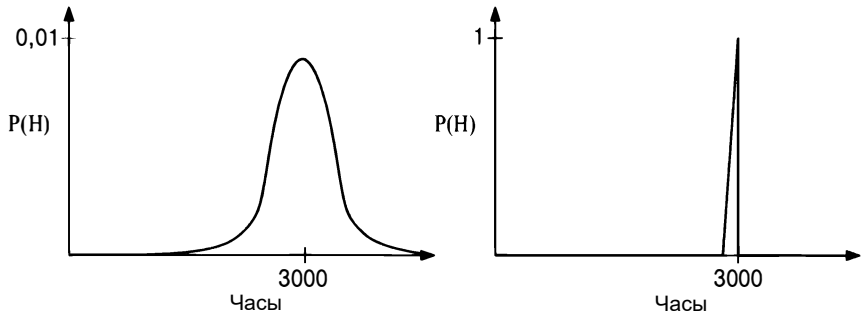


Рис. 2.4. Два разных вероятностных распределения при μ = 3000 для продолжительности работы лампочек: нормальное (слева) и с нулевой дисперсией (справа)

Сумма квадратов в формуле для σ означает, что одно-единственное значение d из среднего вносит в дисперсию столько же, сколько и d^2 из среднего, поэтому дисперсия очень чувствительна к выбросам.

Зачастую вопросы возникают по поводу знаменателя в формуле для стандартного отклонения. Мы должны делить на n или $n - 1$? Различие чисто техническое. Стандартное отклонение всей *совокупности* (population) делится на n, тогда как стандартное отклонение *выборки* (sample) делится на $n - 1$.

Проблема заключается в том, что выборка — это только одна точка, абсолютно ничего не говорящая нам о базовой дисперсии в любой совокупности, о которой совершенно разумно сказать, что дисперсия в весовом коэффициенте среди населения острова с одним человеком нулевая. Но для разумных по размеру наборов данных $n \approx (n-1)$, поэтому это действительно не имеет значения.

2.2.3. Интерпретация дисперсии

Регулярные наблюдения за тем же самым явлением не всегда приводят к тем же самым результатам из-за случайных помех или ошибок. *Ошибки выборки* (sampling error) происходят, когда наши наблюдения фиксируют нетипичные обстоятельства, как, например, измерение объемов трафика в часы пик по выходным и в течение рабочих дней недели. *Ошибки измерения* (measurement error) отражают пределы точности, присущие любому измерительному устройству. Понятие *отношения сигнала к шуму* (signal to noise ratio) обусловливает степень, в которой серия наблюдений отражает представляющее интерес значение, в отличие от дисперсии данных. Как аналитиков данных, нас заботят изменения в сигнале, а не помехи, и такая вариация зачастую делает решение проблемы удивительно трудным.

Я считаю дисперсию неотъемлемым свойством вселенной, таким же, как скорость света или как деньги — эквивалент времени. Каждое утро взвешиваясь на весах, вы гарантированно получаете разное значение и размышляете, что бы это значило, может быть, плотно поел (ошибка выборки), может быть, кривой пол или весы износились (ошибки измерения), либо на самом деле вес изменился (фактическая вариация). Так, каков ваш реальный вес?

Каждая измеряемая величина подвержена некоторой дисперсии (разбросу), однако причина явления намного глубже, чем на первый взгляд. Большая часть происходящего в мире является только случайными флуктуациями или произвольной случайностью, вызывающей дисперсию, даже когда ситуация неизменна. Аналитики данных пытаются объяснять мир через данные, но, к сожалению, зачастую нет никаких реальных явлений для объяснения, есть только призрак, созданный дисперсией. Вот примеры.

- *Фондовый рынок*. Рассмотрите проблему измерения "уровня" различных инвесторов фондового рынка. Известно, что Уоррен Баффетт (Warren Buffet) намного лучше разбирается в инвестировании, чем мы с вами. Но очень немного профессиональных инвесторов оказываются существенно лучше, чем другие. Определенные инвестиционные средства значительно опережают рынок в любой заданный период времени. Тем не менее горячий фонд (hot fund) на один год обычно отстает от рынка в течение года, чего не должно происходить, если такая выдающаяся результативность была обусловлена умением, а не удачей.

Руководители фондов сами по себе быстро кредитуют в выгодные годы, полагаясь на свой гений, но несут потери при непредвиденных обстоятельствах. Тем не менее некоторые исследования показали, что эффективность профессиональных инвесторов, по сути, является случайной удачей, а это значит, что в их умении мало реальной разницы. Большинство инвесторов платят менеджерам за предыдущую удачу. Так почему же этим предсказателям платят так много денег?

- *Спортивная результативность.* У студентов бывают хорошие семестры и плохие семестры, как свидетельствует их средний балл успеваемости (GPA). У атлетов бывают хорошие и плохие сезоны, как отражает их результативность и статистика. Такие изменения отражают подлинные различия в усилиях и способностях или являются только результатом дисперсии?

  В бейсболе хитеры .300 (hitter) (игроки, отбивающие с результативностью 30%) демонстрируют постоянство в течение всего сезона. Результативность .275 отбоев — это не лучший сезон, а .300 — это хит, и вы звезда. С результативностью .325 отбоев вы, скорее всего, будете чемпионом среди игроков.

  На рис. 2.5 приведены результаты простого моделирования, где случайные числа использовались для определения результата каждого выхода на биту из 500 выходов на биту за сезон. Наш синтезированный игрок — это *реальный* хитер .300, поскольку мы запрограммировали его так, чтобы сообщить о хите с вероятностью 300/1000 (0,3). Результаты показывают, что реальный хитер .300 имеет 10%-ный шанс случайно получить результат .275 или ниже. Такой сезон, как правило, объясняется травмами или, возможно, неизбежным воздействием возраста на спортивные результаты. Но это может быть просто естественная дисперсия. Разумные владельцы команды стараются приобретать хороших нападающих после паршивого сезона, когда цена на них падает, пытаясь использовать в своих интересах эту дисперсию.

  У наших нападающих .300 есть также 10%-ный шанс на отбой выше .325, но вы можете быть вполне уверены, что они припишут такой блестящий сезон созданию улучшенных условий или новым учебным методам, а не тому факту, что им просто повезло. Хороший или плохой сезон, удачный или неудачный: довольно трудно сказать, сигнал это или шум.

- *Эффективность моделей.* Как аналитики данных, мы будем обычно разрабатывать и обсчитывать по несколько моделей для каждой задачи прогнозирования. Модели могут располагаться от очень простой до очень сложной и различаться по своим условиям или параметрам.

```
In[28]:= Season[p_Real, n_Integer] :=
           Count[Table[If[RandomReal[1] ≤ p, 1, 0], {n}], 1] / (1.0 * n)

In[29]:= Histogram[d = Table[Season[0.300, 500], {100 000}], 100]
```

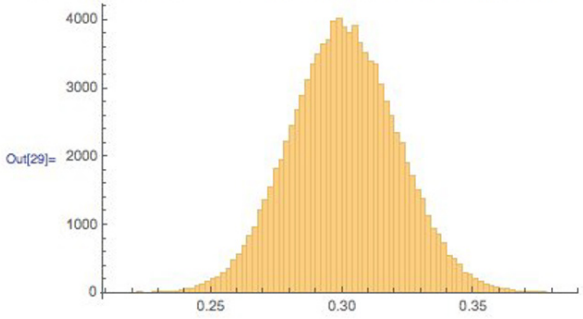


Рис. 2.5. Пример дисперсии у нападающих с реальным 30%-ным коэффициентом успеха приводит к широкому диапазону наблюдаемой результативности даже при более чем 500 попытках за сезон

Как правило, модель с наилучшей точностью и учебной ценностью будет торжественно выставлена напоказ перед миром как правильная. Но небольшие различия в производительности между моделями, скорее всего, объясняются простой дисперсией, а не мудростью: какие пары обучения и оценки были выбраны, насколько хорошо были оптимизированы параметры и т.д.

Помните об этом, когда дело доходит до учебных моделей машинного обучения. Действительно, когда вас просят выбрать между моделями с небольшими различиями в эффективности, я, скорее всего, приведу доводы в пользу самой простой модели, чем таковой с самым высоким баллом. Попросите сто человек предсказать результаты бросков монеты, и один из них гарантированно даст больше правильных ответов, чем другие. Однако нет никакой разумной причины полагать, что у этого человека лучшие способности к предсказанию, чем у остальных.

2.2.4. Характеристика распределений

У распределений не обязательно будет наибольшая масса вероятности точно в среднем. Рассмотрим, как выглядели бы ваши финансы после того, как вы займете 100 млн долл., а затем поставите их все на один бросок монеты в орлянку. Вначале у вас было 100 млн долга, теперь 100 млн выигрыша. Посередине ваше ожидаемое богатство равно нулю, но это среднее значение мало что говорит вам о форме распределения вашего финансового состояния.

Однако взятые вместе среднее и стандартное отклонение вполне прилично характеризуют *любое* распределение. Даже относительно небольшой объем массы, расположенный далеко от среднего, добавил бы довольно много к стандартному отклонению, таким образом, небольшое значение s подразумевает, что больший объем массы должен быть ближе к средине.

Чтобы быть точным, независимо от того, как распределяются ваши данные, по крайней мере $(1 - (1/k^2))$-я часть массы, должна лечь в пределах $\pm k$ стандартного отклонения от среднего. Это означает, что по крайней мере 75% всех данных должны лечь в пределах 2σ от среднего и почти 89% в пределах 3σ для любого распределения.

Мы увидим, что границы куда жестче, когда мы знаем, что распределение хорошо укладывается в Гауссово или нормальное распределение. Именно поэтому желательно всегда указывать и μ, и σ, когда речь идет о среднем значении. Средний рост взрослых женщин в Соединенных Штатах составляет 63,7 ± 2,7 дюйма (160,02 ± 5,08 см), а значит, $\mu = 63{,}7$ и $\sigma = 2{,}7$. Средняя температура в Орландо, штат Флорида, составляет 60,3 градуса по Фаренгейту (15,55 градуса по Цельсию). Тем не менее в Мире Диснея было куда больше 100-градусных дней, чем посетительниц ростом 100 дюймов (8,33 фута, или 254 см), на загляденье всем.

> *На заметку.* Среднее значение и стандартное отклонение, характеризующие ваше распределение, записывают как $\mu \pm \sigma$.

2.3. Корреляционный анализ

Предположим, что дано две переменные x и y, представленные выборкой из n точек в форме (x_i, y_i), для $1 \leq i \leq n$. Мы говорим, что переменные x и y *коррелируют*, когда значения x имеют некую степень прогнозирования значения y.

Коэффициент корреляции $r(X, Y)$ является статистической величиной, выражающей степень, в которой переменная Y является функцией от X и наоборот. Значение коэффициента корреляции варьируется от –1 до 1, где 1 означает полную корреляцию, а 0 подразумевает полное отсутствие взаимосвязи или независимые переменные. Отрицательные корреляции означают, что переменные *антикоррелируют*, т.е., когда значение переменной X повышается, значение переменной Y понижается.

У абсолютно антикоррелирующих переменных корреляция составляет –1. Обратите внимание, что отрицательные корреляции столь же хороши в целях прогнозирования, что и положительные. То, что вы менее подвержены риску

безработицы, если у вас выше квалификация, является примером отрицательной корреляции, поскольку уровень образования действительно может помочь предсказать статус трудоустройства. Корреляции, близкие к 0, бесполезны для прогнозирования.

Наблюдаемые корреляции управляют большинством прогнозирующих моделей, которые мы строим в науке о данных. К типичным примерам корреляций относят такие вопросы.

- Являются ли более высокие люди более склонными к худобе? Наблюдаемая корреляция между ростом и ИМТ (индекс массы тела) составляет $r = -0{,}711$, поэтому рост действительно отрицательно коррелирует с индексом массы тела.[3]
- Прогнозируют ли стандартизированные тесты результативность студентов в колледже? Наблюдаемая корреляция между баллами SAT и GPA абитуриентов составляет $r = 0{,}47$, так что некоторая степень прогнозирования есть. Но социально-экономический статус также жестко коррелирует с баллами SAT ($r = 0{,}42$).[4]
- Влияет ли финансовое состояние на состояние здоровья? Наблюдаемая корреляция между доходом и распространенностью болезни коронарной артерии составляет $r = -0{,}717$, таким образом, есть жесткая отрицательная корреляция. Действительно, чем вы богаче, тем ниже риск сердечного приступа.[5]
- Курение влияет на здоровье? Наблюдаемая корреляция между склонностью группы к курению и их смертностью составляет $r = 0{,}716$, следовательно, лучше не курить.[6]
- Повышают ли агрессивные видеоигры агрессивность? Наблюдаемая корреляция между игрой и насилием составляет $r = 0{,}19$, таким образом, корреляция слабая, но она есть.[7]

Этот раздел знакомит с первичными показателями корреляции. Мы узнаем, как определять силу и степень любой наблюдаемой корреляции, чтобы помочь вам понять, когда взаимосвязь между переменными на самом деле реальна.

---

[3] https://onlinecourses.science.psu.edu/stat500/node/60.

[4] https://research.collegeboard.org/sites/default/files/publications/2012/9/researchreport-2009-1-socioeconomic-status-sat-freshman-gpa-analysis-data.pdf.

[5] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3457990/.

[6] http://lib.stat.cmu.edu/DASL/Stories/SmokingandCancer.html.

[7] http://webspace.pugetsound.edu/facultypages/cjones/chidev/Paper/Articles/Anderson-Aggression.pdf.

2.3.1. Коэффициенты корреляции Пирсона и Спирмена

Фактически есть два основных статистических коэффициента, используемых для измерения корреляции. К счастью, оба работают в тех же самых масштабах от –1 до 1, хотя измеряют они несколько разные вещи. Эти статистические показатели являются более подходящими в разных ситуациях, таким образом, необходимо знать об обоих.

Коэффициент корреляции Пирсона

Наиболее заметной из двух является корреляция *Пирсона* (Pearson), определяемая как

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} = \frac{Cov(X,Y)}{\sigma(X)\sigma(Y)}.$$

Давайте проанализируем это уравнение. Предположим, что X и Y строго коррелируют. Далее мы ожидаем, что когда X_i больше, чем среднее $\bar{X}$, то и Y_i должно быть больше, чем его среднее $\bar{Y}$. Когда X_i ниже, чем его среднее, то и Y_i должен быть таковым. Теперь рассмотрим числитель. Знак каждого члена положителен, когда оба значения выше $1 \cdot 1$ или ниже $-1 \cdot 1$, чем их соответствующее среднее. Знак каждого члена отрицателен ($-1 \cdot 1$ или $1 \cdot (-1)$), если они перемещаются в противоположных направлениях, предлагая отрицательную корреляцию. Если бы X и Y не коррелировали, то положительные и отрицательные члены должны были встречаться с равной частотой, смещая друг друга и сводя значение к нулю.

Работа числителя, определяющая знак корреляции, настолько полезна, что мы присваиваем ей имя, *ковариация* (covariance), и вычисляем ее как

$$Cov(X,Y) = \sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y}).$$

Запомните ковариацию: мы вернемся к ней снова в разделе 8.2.3.

Знаменатель формулы Пирсона отражает объем дисперсии в этих двух переменных, измеряемый по их среднеквадратичным отклонениям. Ковариация между X и Y потенциально возрастает с дисперсией этих переменных, и этот знаменатель является магическим значением для его разделения и приведения корреляции к масштабу от –1 до 1.

Коэффициент ранговой корреляции Спирмена

Коэффициент корреляции Пирсона определяет степень, в которой линейный предиктор в форме $y = mx + b$ может соответствовать наблюдаемым данным.

Он обычно хорошо подходит для замера сходства между переменными, однако вполне возможно привести примеры, где коэффициент корреляции между X и Y является нулевым, но Y все же полностью зависит (а следовательно, совершенно предсказуем) от X.

Рассмотрим точки в форме $(x, |x|)$, где x однородно (или симметрично) выбирается из интервала $[-1, 1]$, как показано на рис. 2.6. Корреляция будет нулевой, поскольку для каждой точки (x, x) будет точка смещения $(-x, x)$, но все же $y = |x|$ является прекрасным предиктором. Корреляция Пирсона является мерой того, насколько хорошо могут работать наилучшие *линейные предикторы*, но он ничего не говорит о более сложных функциях, таких как абсолютная величина.

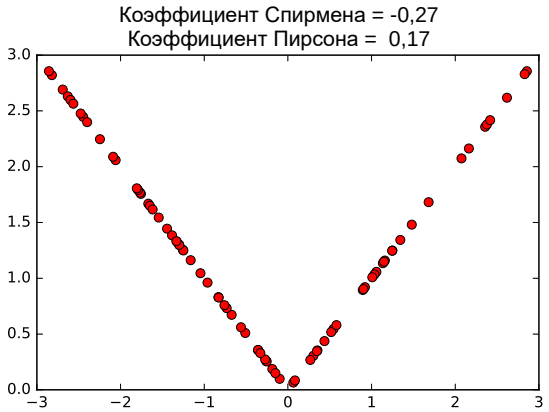


Рис. 2.6. Функция $y = |x|$ не имеет линейной модели, однако кажется, что она должна быть легко установлена, несмотря на слабые корреляции

Коэффициент ранговой корреляции Спирмена, по существу, подсчитывает количество пар входных точек, которые находятся вне порядка. Предположим, что наш набор данных содержит точки (x_1, y_1) и (x_2, y_2), где $x_1 < x_2$ и $y_1 < y_2$. Это свидетельство положительной корреляции значений, тогда как свидетельством отрицательной корреляции было бы $y_2 < y_1$.

Сумма всех пар точек и правильная нормализация дает нам ранговую корреляцию Спирмена. Предположим, что $rank(x_i)$ является позицией ранга x_i в отсортированной последовательности среди всех x_i, таким образом, ранг наименьшего значения — это 1, а наибольшего значения — n. Теперь

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)},$$

где $d_i = rank(x_i) - rank(y_i)$.

Отношения между нашими двумя коэффициентами лучше описывает пример на рис. 2.7. Кроме предоставления высоких оценок нелинейным, но монотонным функциям, корреляция Спирмена менее чувствительна к экстремальным выбросам, чем корреляция Пирсона. Пусть $p = (x_1, y_{max})$ является точкой на графике с наибольшим значением y в данном наборе данных. Предположим, что мы заменяем p на $p' = (x_1, \infty)$. Корреляция Пирсона сойдет с ума, поскольку наилучшим образом теперь подходит вертикальная линия $x = x_1$. Но корреляция Спирмена останется неизменной, поскольку все точки были под p, так же как теперь они под p'.

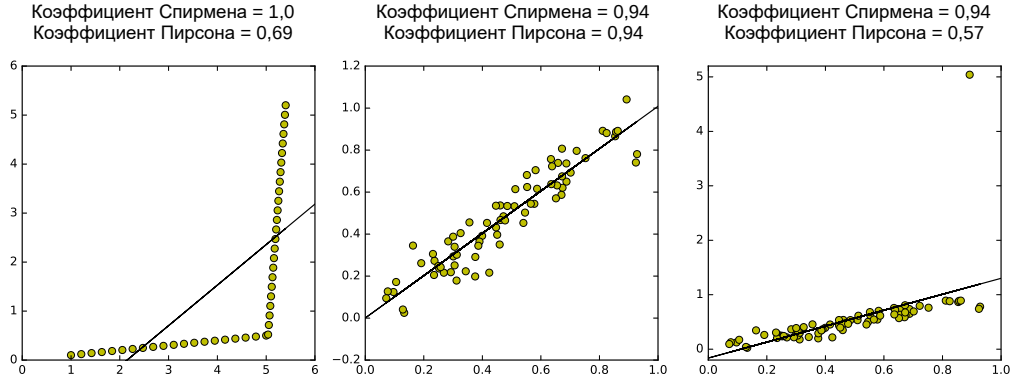


Рис. 2.7. Монотонный, но не линейный точечный набор имеет коэффициент Спирмена $r = 1$, хотя он не имеет хорошего линейного соответствия (слева). Последовательности с высокой корреляцией распознаются обоими коэффициентами (в центре), но коэффициент Пирсона гораздо более чувствителен к выбросам (справа)

2.3.2. Сила и значение корреляции

Коэффициент корреляции r отражает степень, в которой x применим для прогнозирования y в данной выборке точек S. Пока $|r| \rightarrow 1$, эти прогнозы становятся все лучше и лучше.

Однако реальный вопрос заключается в том, как эта корреляция будет поддерживаться в реальном мире, вне выборки. Более сильные корреляции имеют больший $|r|$, но также и задействуют выборки с достаточным количеством точек, чтобы быть существенными. Есть саркастическое высказывание, что если вы хотите поместить ваши данные на прямой линии, то лучше проводить ее только через две точки. Ваша корреляция будет становиться более внушительной, чем на большем количестве точек она основана.

Статистические пределы в интерпретации корреляций, представленных на рис. 2.8, основаны на силе и размере.

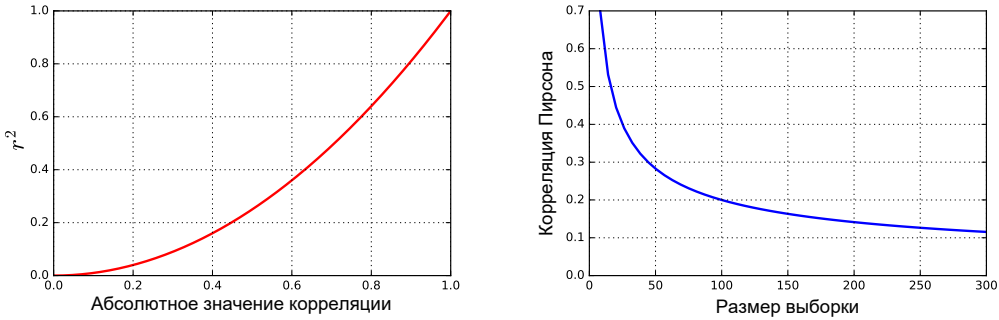


Рис. 2.8. Пределы в интерпретации значимости. Значение r^2 демонстрирует, что слабые корреляции объясняют только малую долю дисперсии (слева). Уровень корреляции, необходимый для статистической значимости, быстро уменьшается с размером выборки n (справа)

- *Сила корреляции*: R^2. Квадрат коэффициента корреляции r^2 выборки оценивает долю дисперсии в Y, которая объясняется X в простой линейной регрессии. Корреляция между ростом и весом составляет примерно 0,8, т.е. она объясняет примерно две трети дисперсии.

  На рис. 2.8 (слева) показано, как быстро r^2 уменьшается с r. Есть глубокий предел тому, как мы должны заботиться об установлении слабой корреляции. Корреляция 0,5 обладает только 25% максимальной силы прогнозирования, а корреляция $r = 0{,}1$ составляет всего 1%. Таким образом, прогностическое значение корреляций быстро уменьшается с r.

  Что мы понимаем под "объяснением дисперсии"? Пусть $f(x) = mx + c$ будет прогностическим значением y от x, причем параметры m и c соответствуют наилучшему возможному подходу. У *остаточных* (residual) значений $r_i = y_i - f(x_i)$ среднее будет нулевым, как показано на рис. 2.9. Кроме того, дисперсия полного набора данных $V(Y)$ должна быть намного большей, чем $V(r)$, если существует хороший *подбор прямой* (linear fit) $f(x)$. Если x и y отлично коррелируют, не должно быть никакой остаточной ошибки и $V(r) = 0$. Если x и y совершенно не коррелируют, соответствие не должно внести ничего и $V(y) \approx V(r)$. По правде говоря, $1 - r^2 = V(r)/V(y)$.

  Рассмотрим рис. 2.9, демонстрирующий набор точек (слева), допускающих хороший подбор прямой, с корреляцией $r = 0{,}94$. Соответствующие остатки $r_i = y_i - f(x_i)$ изображены справа. Дисперсия значений y слева $V(y) = 0{,}056$ существенно больше, чем дисперсия $V(r) = 0{,}0065$ справа. Действительно,

$$1 - r^2 = 0{,}116 \leftrightarrow V(r)/V(y) = 0{,}116.$$

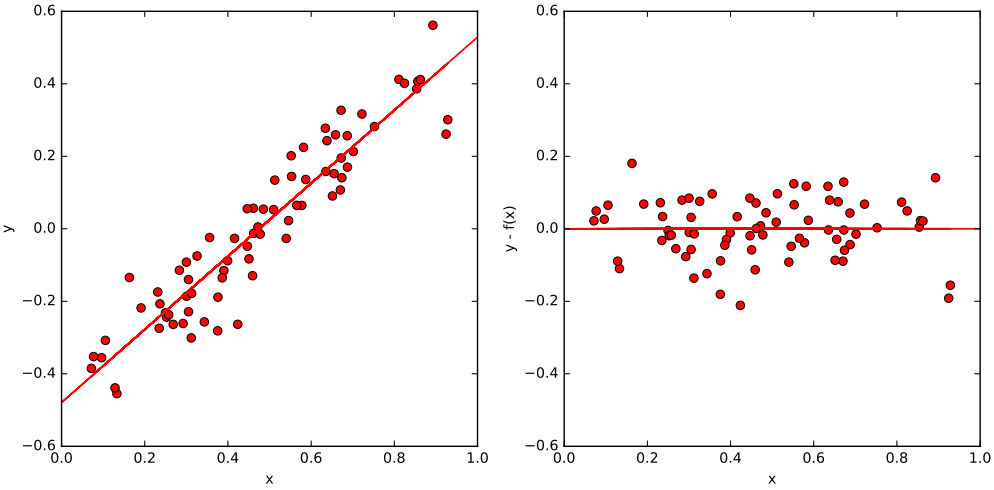


Рис. 2.9. График $= y - f(x_i)$ демонстрирует, что остаточные значения имеют более низкую дисперсию и не значат ничего. Первоначальные точки на графике слева с соответствующими остатками справа

- *Статистическая значимость.* Статистическая значимость корреляции зависит от размера выборки n так же, как и r. По традиции мы говорим, что корреляция точек n *существенна*, если есть $\alpha \leq 1/20 = 0{,}05$ шанса, что мы наблюдали бы корреляцию столь же сильную, как r, в любом случайном наборе из n точек.

  Это не особенно сильный стандарт. Даже небольшие корреляции становятся существенными на уровне 0,05 при достаточно больших размерах выборки, как показано на рис. 2.8 (справа). Корреляция $r = 0{,}1$ становится существенной при $\alpha = 0{,}05$ и примерно $n = 300$, даже при том, что такой фактор объясняет только 1% дисперсии.

Слабые, но значимые корреляции могут иметь ценность в больших моделях данных, задействующих большие количества функций. Любая отдельная функция/корреляция может объяснить/предсказать только малые эффекты, но взятые вместе многие слабые, но независимые корреляции могут иметь большую прогностическую силу. *Возможно.* Мы обсудим значение этого снова подробней в разделе 5.3.

2.3.3. Корреляция не означает причину!

Возможно, вы уже слышали выражение: корреляция не означает причину.

- Количество действующих в районе полицейских сильно коррелирует с местным уровнем преступности, но полиция не является причиной преступлений.

- Количество принимающих лекарства людей коррелирует с вероятностью того, что они больны, но лекарства не вызывают болезни.

Выражение работает в лучшем случае только в одном направлении. Но многие наблюдаемые корреляции полностью ложны, причем ни одна переменная не имеет никакого реального влияния на другие.

Однако *корреляция означает причину* — это наиболее распространенная ошибка во взглядах, даже среди тех, кто мыслит логически. По правде говоря, доступно немного статистических инструментов, позволяющих выяснить, действительно ли A является причиной B. Мы можем проводить контролируемые эксперименты, если можем манипулировать одной из переменных и наблюдать за воздействием этого на другую. Например, тот факт, что мы можем посадить людей на диету, которая заставит их потерять вес, но не сделать ниже, является убедительным доказательством того, что вес *не влияет* на рост. Но зачастую бывает куда сложнее поставить эксперимент другим способом, например, нет никого разумного способа сделать людей ниже, кроме как сломать им конечности.

Рис. 2.10. Корреляция не означает причину (источник: https://www.xkcd.com/552)

2.3.4. Обнаружение автокорреляцией периодичности

Предположим, что инопланетянина наняли для анализа продаж игрушек в США. Вместо приятной гладкой функции, демонстрирующей постоянную тенденцию, он бы с удивлением обнаружил гигантский пик каждый двенадцатый месяц каждый год. Так инопланетянин открыл бы явление Рождества.

Сезонные тенденции отражают циклы фиксированной продолжительности с регулярными взлетами и падениями. Многие виды человеческой деятельности происходят по семидневному циклу, связанному с рабочей неделей. Большие популяции насекомых некого типа, называемого *цикадой*, появляются с 13- или 17-летним циклом, чтобы хищники не научились их есть.

Как мы можем распознать такие циклические шаблоны в последовательности S? Предположим, что мы коррелируем значения S_i с S_{i+p} для всех $1 \le i \le n-p$. Если значения синхронизируются для периода p определенной длительности, то эта корреляция с самим собой будет необычно высокой по сравнению с другими возможными значениями. Сравнение последовательности с самой собой называется *автокорреляцией*, а прогрессия корреляций для всех $1 \le k \le n-1$ является *функцией автокорреляции*. На рис. 2.11 представлены временные ряды ежедневных продаж и соответствующая функция автокорреляции для этих данных. Пик каждые семь дней (и каждый раз семь дней) свидетельствует, что в продажах есть еженедельная периодичность: по выходным продается больше товаров.

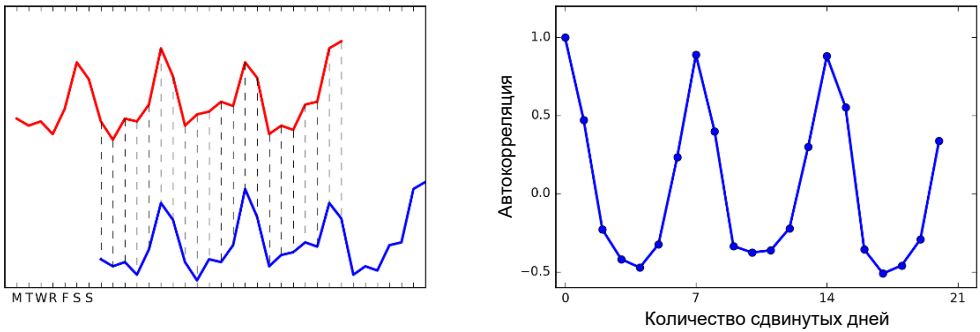


Рис. 2.11. Циклические тенденции во временном ряду (слева) выявляются при корреляции его с самим собой при сдвиге (справа)

Автокорреляция — важная концепция в прогнозировании будущих событий, поскольку она означает, что мы можем использовать предыдущие наблюдения как функцию в модели. Эвристика, что завтрашняя погода будет похожа на сегодняшнюю, основана на автокорреляции с отставанием в $p = 1$ день. Разумеется, мы ожидаем, что такая модель будет более точной, чем прогнозы, сделанные по метеорологическим данным за полгода назад (отставание $p = 180$ дней).

По правде говоря, функция автокорреляции для многих величин имеет тенденцию быть самой высокой для очень коротких задержек. Вот почему долгосрочные прогнозы менее точны, чем краткосрочные: автокорреляции, как правило, значительно слабее. Но периодические циклы иногда растягиваются намного дольше. Действительно, прогноз погоды, основанный на задержке $p = 365$ дней, будет намного лучше, чем таковой при $p = 180$, из-за сезонных эффектов.

Вычисление полной функции автокорреляции требует вычисления $n-1$ различных корреляций в точках временных рядов, что может обойтись весьма дорого для большого n. К счастью, есть эффективный алгоритм на основании *быстрого преобразования Фурье* (Fast Fourier Transform — FFT), который позволяет построить функцию автокорреляции даже для очень длинных последовательностей.

2.4. Логарифмы

Логарифм — это обратная экспоненциальная функция $y = b^x$, ее уравнение может быть переписано как $x = \log_b y$. Это определение эквивалентно следующему:

$$b^{\log_b y} = y.$$

Экспоненциальные функции растут с очень высокой скоростью: например $b = \{2^1, 2^2, 2^3, 2^4, ...\}$. И наоборот, логарифмы растут очень медленно: это только показатели предыдущей прогрессии {1, 2, 3, 4...}. Они связаны с любым процессом, в котором мы неоднократно умножаем на некоторое значение b или многократно делим на b. Просто запомните определение:

$$y = \log_b x \leftrightarrow b^y = x.$$

Логарифмы весьма полезны и нередко используются при анализе данных. Здесь я подробно описываю три важные роли, которые играют логарифмы в науке о данных. Удивительно, но только одна из них связана с семью алгоритмическими приложениями логарифмов, которые я представляю в книге *The Algorithm Design Manual* [1]. Логарифмы действительно очень полезны.

2.4.1. Логарифмы и умножение вероятностей

Логарифмы были впервые изобретены в качестве помощника для вычислений, снизив сложность задачи умножения до задачи сложения. В частности, для вычисления произведения $p = xy$ мы могли бы вычислить сумму логарифмов $s = \log_b x + \log_b y$, а затем взять обратный логарифм (т.е. возвести b в степень s), чтобы получить p, поскольку:

$$p = xy = b^{(\log_b x + \log_b y)}.$$

Этот трюк приводил в действие механические логарифмические линейки, которые использовали во времена до появления карманных калькуляторов.

Однако эта идея остается важной и сегодня, особенно при умножении длинных цепочек вероятностей. Вероятности — это небольшие числа. Таким образом, умножение длинных цепочек вероятностей дает *очень* малые числа, которые определяют шансы очень редких событий. Существуют серьезные проблемы с численной стабильностью при умножении чисел с плавающей запятой на реальных компьютерах. Вкрадывающиеся числовые ошибки в конечном итоге уничтожат истинный смысл достаточно малых чисел.

Суммирование логарифмов вероятностей гораздо более устойчиво к числам, чем их умножение, но дает эквивалентный результат, поскольку:

$$\prod_{i=1}^{n} p_i = b^{P}, \text{ где } P = \sum_{i=1}^{n} \log_b(p_i).$$

Мы можем привести нашу сумму в экспоненциальную форму, если нам нужна реальная вероятность, но обычно это необязательно. Когда нам просто нужно сравнить две вероятности, чтобы решить, какая из них больше, мы можем спокойно оставаться в мире логарифмов, поскольку б*о*льшие логарифмы соответствуют б*о*льшей вероятности.

Есть еще одна особенность, о которой нужно знать. Напомним, что $\log_2(1/2) = -1$. Логарифмы вероятностей всегда являются отрицательными числами, кроме $\log(1) = 0$. Именно поэтому уравнения с логарифмами вероятностей зачастую имеют знак минус в странных местах. Вы их еще встретите.

2.4.2. Логарифмы и соотношения

Соотношения (ratio), или коэффициенты, — это величины в форме a/b. Они часто встречаются в наборах данных либо как элементарные функции, либо как значения, полученные из пар функций. Соотношения естественным образом происходят при нормализации данных для условий (т.е. вес после некого лечения по сравнению с начальным весом) или времени (т.е. сегодняшняя цена по сравнению с вчерашней).

Но соотношения ведут себя по-разному, когда они отражают увеличение и уменьшение. Соотношение 200/100 на 200% выше базового уровня, но 100/200 лишь на 50% ниже, несмотря на то, что изменение величин аналогично. Таким образом, такие вещи, как *усредняющие коэффициенты* (averaging ratio), грешат против статистики. Вы действительно хотите, чтобы удвоение с последующим делением на два оказалось в среднем увеличением, а не отсутствием изменения?

Одно из решений здесь заключалось бы в использовании среднего геометрического. Но лучше для этих соотношений брать логарифм, поскольку они дают равное смещение, так как $\log_2 2 = 1$ и $\log_2(1/2) = -1$. Мы получаем дополнительный бонус, поскольку единичное соотношение равно нулю, поэтому положительные и отрицательные числа соответствуют неправильным и правильным коэффициентам соответственно.

Мои студенты нередко делают ошибку новичка, составляя график значений коэффициентов вместо их логарифмов. На рис. 2.12 (слева) приведен график из студенческой работы, демонстрирующий отношение нового счета по сравнению со старым по данным за 24 часа (каждая красная точка — это показатель, измеряемый в течение одного часа) из четырех разных наборов данных

(каждая в своем ряду). Сплошная черная линия отмечает единичное соотношение, где оба показателя дают одинаковый результат. Теперь попробуйте прочитать этот график: это непросто, поскольку точки слева от линии плотно укладываются в узкую полосу. То, что выходит за нее — это выбросы. Конечно, новый алгоритм ужасно работает для 7UM917 в верхней строке: точка справа — это реальный выброс.

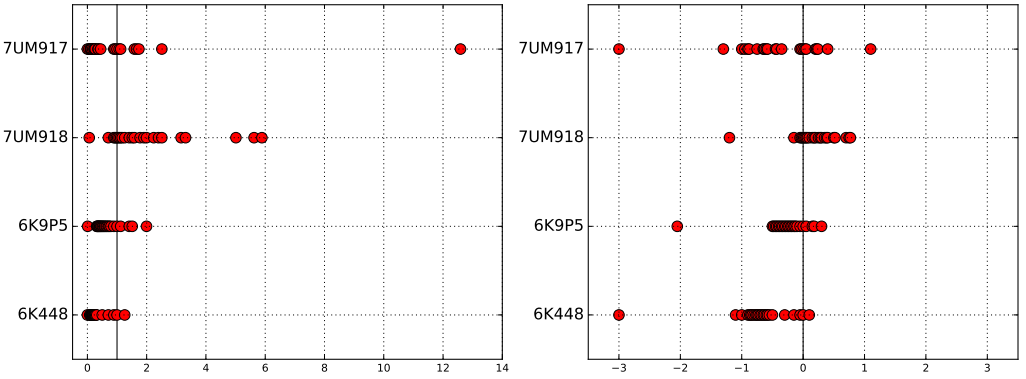


Рис. 2.12. График соотношений в масштабе сужает пространство, выделенное для небольших соотношений, относительно больших (слева). График логарифмов соотношений лучше отражает основные данные (справа)

Кроме того, это не так. Теперь посмотрим на рис. 2.12 (справа), где приведен график логарифмов соотношений. Пространство слева и справа от черной линии теперь может быть равным. И это доказывает, что данная точка не была *действительно* выбросом. Величина отклонения точек слева намного больше, чем у точек справа. Этот график доказывает, что новый алгоритм, как правило, улучшает работу только потому, что мы демонстрируем логарифмы коэффициентов, а не сами коэффициенты.

2.4.3. Логарифмы и нормализация асимметричных распределений

Переменные, которые следуют симметричным, колоколообразным распределениям, как правило, хороши как функции в моделях. Они показывают существенные вариации, поэтому их можно использовать для различения вещей, но не в таком широком диапазоне, чтобы выбросы являлись ошеломляющими.

Однако не каждое распределение симметрично. Рассмотрим рис. 2.13 (слева).

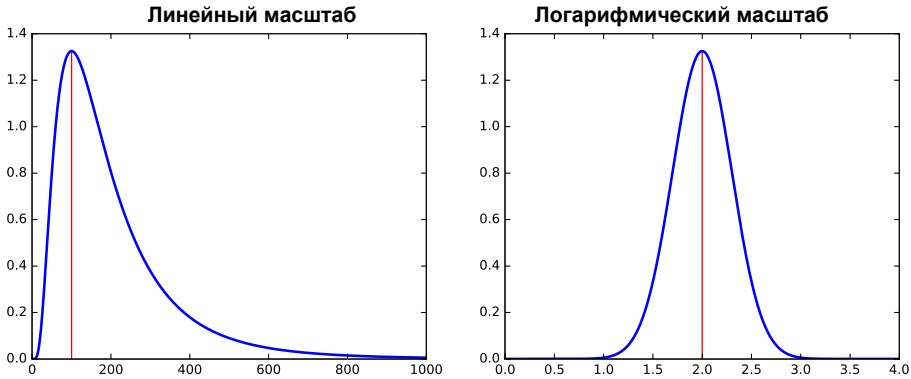


Рис. 2.13. Перевод асимметричного распределения (слева) в логарифмическую форму зачастую дает более симметричную форму распределения (справа)

Хвост справа намного длиннее, чем хвост слева. И нам предстоит увидеть гораздо более ассиметричные распределения, когда мы будем обсуждать степенные законы в разделе 5.1.5. Примером такого распределения является благосостояние, когда у бедняков нулевое или, возможно, отрицательное богатство, средний класс (в лучшем случае) владеет тысячами долларов, а Билл Гейтс владел 100 млрд долл. на момент написания этой книги.

Для преобразования таких распределений требуется нормализация, чтобы с ними проще было иметь дело. Чтобы ударить в колокол *распределения по степенному закону* (power law distribution), необходимо нечто нелинейное, что сводит большие значения к непропорциональному уровню, по сравнению с более скромными значениями.

Логарифм является подходящим преобразованием для переменных степенного закона. Подвергните свое длиннохвостое распределение логарифмированию, и, как правило, будет вам счастье. Распределение на рис. 2.13 оказалось *логарифмически нормальным* (log normal), так что логарифмирование дало идеальную колоколообразную кривую справа. Логарифмирование переменных с распределением по степенному закону приводит их к более традиционным распределениям. Например, по логарифмической шкале мое финансовое состояние, как профессионала класса выше среднего, отстоит на такое же расстояние от моих голодных студентов, как и я от Билла Гейтса!

Иногда использование логарифма оказывается слишком резким, и менее драматичное нелинейное преобразование, такое как квадратный корень, оказывается лучше для нормализации распределения. В качестве пробного камня построим график плотности распределения преобразованных значений и посмотрим, будет ли он выглядеть колоколообразно: примерно симметричным, с выпуклостью посередине. Следовательно, тогда вы узнаете, что у вас есть правильная функция.

2.5. Случай из жизни: поиск дизайнерских генов

Биоинформатика (bioinformatics) — это наука о жизни, указывающая "аналитикам данных" практиковать новую дисциплину и изучать огромные коллекции данных последовательностей ДНК в поисках шаблонов. Данные последовательности очень интересны для работы, и я участвовал в биоинформационных исследованиях с самого начала проекта исследования генома человека.

Последовательности ДНК — это строки из четырех алфавитных символов $\{A, C, G, T\}$. Белки формируют материал, из которого мы физически состоим, и состоят из строк 20 различных типов молекулярных единиц, называемых аминокислотами. *Гены* представляют собой последовательности ДНК, которые точно описывают, как создавать конкретные белки, причем единицы, из которых они состоят, описываются триплетами $\{A, C, G, T\}$, называемыми *кодонами*.

Для наших целей достаточно знать, что существует огромное количество возможных последовательностей ДНК, описывающих гены, которые *могут* кодировать любую конкретную желаемую последовательность белков. Но *используется* только один из них. Мои соавторы-биологи и я хотели знать почему.

Первоначально предполагалось, что все эти различные синонимические кодировки были, по существу, идентичны, но статистика, собранная по данным последовательности, дала понять, что некоторые кодоны используются чаще, чем другие. Биологическое заключение "кодоны имеют значение" и есть хорошая биологическая причина, по которой так и должно быть.

Нас интересовало, "имеют ли значение соседние пары кодонов". Возможно, некоторые пары триплетов подобны маслу и воде, они никогда не смешиваются. У определенных пар символов в английском языке есть преимущества в предпочтениях: биграмма *gh* встречается гораздо чаще, чем *hg*. Возможно, это верно и для ДНК? Если это так, то существуют ли пары триплетов, которые должны быть недостаточно представлены в данных последовательности ДНК.

Чтобы проверить это, нам понадобился балл, сравнивающий количество раз, когда встречается определенный триплет (скажем, $x = CAT$) рядом с другим конкретным триплетом (скажем, $y = GAG$), с тем, что ожидалось бы случайно. Пусть $F(xy)$ — это частота xy, т.е. количество раз, когда мы фактически видим кодон x, за которым следует кодон y в базе данных последовательности ДНК. Эти кодоны кодируют определенные аминокислоты, скажем a и b соответственно. Для аминокислоты a вероятность того, что она будет кодироваться x, составляет $P(x) = F(x)/F(a)$, и аналогично $P(y) = F(y)/F(b)$. Тогда ожидаемое количество случаев xy составляет

$$Expected(xy) = \frac{F(x)}{F(a)} \cdot \frac{F(y)}{F(b)} F(ab).$$

На основании этого мы можем вычислить балл пары кодонов для любого заданного гексамера xy следующим образом:

$$CPS(xy) = \ln\left(\frac{Observed(xy)}{Expected(xy)}\right) = \ln\left(\frac{F(xy)}{\frac{F(x)F(y)}{F(a)F(b)}F(ab)}\right).$$

Логарифмирование этого соотношения дало очень интересные свойства. Самое главное, что знак балла, отличает часто встречающиеся пары от редко встречающихся пар. Поскольку величины были симметричными (+1 был столь же впечатляющим, как −1), мы могли бы сложить или усреднить эти баллы разумным способом, чтобы дать оценку для каждого гена. Мы использовали эти баллы для разработки генов, которые должны были быть вредными для вирусов, что дало бы новую захватывающую технологию для изготовления вакцин. Более подробная информация по этой теме приведена в разделе 2.6.

Знание того, что некоторые пары кодонов были вредны, никак не объясняло, *почему* они были вредны. Но вычисление двух смежных показателей (детали несущественны) и сортировка триплетов на их основе, как показано на рис. 2.14, позволили выявить некоторые шаблоны. Вы замечаете шаблоны? Все вредные последовательности слева содержат триплет *TAG*, который оказывается специальным кодоном, указывающим гену остановиться. И все вредные последовательности справа состоят из *C* и *G* в очень простых повторяющихся последовательностях. Они объясняют биологически, почему шаблоны избегают эволюции, т.е. мы обнаружили нечто очень значимое в жизни.

Fr. Dep.	Score		Fr. Ind.	Score
CA**TAG**G	-1.74		GGGGGG	-1.01
TC**TAG**C	-1.61		CCCCCC	-0.95
GT**TAG**G	-1.58		GGCGCC	-0.66
GC**TAG**T	-1.48		GGGGGT	-0.63
CC**TAG**T	-1.44		CGGGGG	-0.59
GG**TAG**G	-1.41		AGGGGG	-0.58
CT**TAG**G	-1.40		CACGTG	-0.58
AC**TAG**C	-1.38		ACCCCC	-0.56
GC**TAG**C	-1.37		GGGCCC	-0.56
GC**TAG**A	-1.36		CCCCCT	-0.53
CC**TAG**C	-1.35		CGCCCC	-0.52
GA**TAG**G	-1.35		CCCCCG	-0.51

Рис.2.14. Шаблоны в последовательностях ДНК с наименьшим количеством пар кодонов становятся очевидными при просмотре. При интерпретации во фрейме символ остановки TAG существенно обеднен (слева). При интерпретации в двух других фреймах наиболее редкие шаблоны имеют очень низкую сложность, как пробежка к одной базе (справа)

Из этой истории есть два урока. Во-первых, разработка числовых функций подсчета, которые выделяют конкретные аспекты элементов, может быть очень полезна для выявления шаблонов. В главе 4 основное внимание будет уделено разработке таких систем. Во-вторых, логарифмирование таких величин может сделать их еще более полезными, позволяя нам увидеть лес за деревьями.

2.6. Дополнительная информация

Существует множество отличных и доступных введений в теорию вероятности, в том числе [19, 20]. То же самое относится к элементарной статистике с хорошими вводными курсами, включая [21, 22]. Краткая история теории вероятностей в этой главе основана на Weaver [23].

В своей самой сильной форме эффективная рыночная гипотеза гласит, что фондовый рынок, по существу, непредсказуем при использовании открытой информации. Мой личный совет заключается в том, что вам следует инвестировать в индексные фонды, которые не пытаются активно прогнозировать направление рынка. Книга Малкиела (Malkiel) *A Random Walk Down Wall Street* [24] является превосходным введением в такое инвестиционное мышление.

Быстрое преобразование Фурье (FFT) обеспечивает $O(n \log n)$ алгоритм времени для вычисления полной автокорреляционной функции для последовательности из n элементов, где прямое вычисление n корреляций получает $O(n^2)$. Книги Брейсуэлла (Bracewell) [25] и Бригама (Brigham) [26] — это отличные введения в преобразования Фурье и FFT. (См. также выставку в Press et.al. [27].)

Комические карикатуры на рис. 2.10 представлены веб-коммюнике *xkcd* Рэндалла Манро (Randall Munroe), в частности `https://xkcd.com/552`, и перепечатаны с разрешения.

Случай из жизни в разделе 2.5 взят из нашей работы о том, как феномен смещения пары кодонов влияет на трансляцию генов. Рис. 2.14 принадлежит моему сотруднику Джастину Гардину (Justin Gardin). См. [28, 29, 30] для обсуждения того, как мы использовали привязку пары кодонов для разработки вакцин против вирусных заболеваний, таких как полиомиелит и грипп.

2.7. Упражнения

Вероятность

2.1. *[3]* Предположим, что 80% людей любят арахисовое масло, 89% любят желе и 78% любят и то, и другое. Учитывая, что случайно выбранный человек любит арахисовое масло, какова вероятность того, что ему также нравится желе?

2.2. *[3]* Предположим, что $P(A) = 0{,}3$ и $P(B) = 0{,}7$.
   (a) Можно ли вычислить $P(A$ И $B)$, если вы знаете только $P(A)$ и $P(B)$?
   (b) Предположим, что события A и B являются результатом независимых вероятностных процессов.
   - Каково $P(A$ И $B)$?
   - Каково $P(A$ ИЛИ $B)$?
   - Каково $P(A \mid B)$?

2.3. *[3]* Рассмотрим игру, где ваш счет — это максимальное значение от двух игральных костей. Вычислите вероятность каждого события от $\{1, \ldots, 6\}$.

2.4. *[8]* Докажите, что функция кумулятивного распределения максимума пары значений, взятых из случайной величины X, является квадратом исходной функции кумулятивного распределения X.

2.5. *[5]* Если две двоичные случайные величины X и Y независимы, то будут ли $\overline{X}$ (дополнение X) и Y также независимы? Приведите доказательство или контрпример.

Статистика

2.6. *[3]* Сравните каждую пару распределений, чтобы решить, какое из них имеет большее среднее значение и большее стандартное отклонение. Вы не должны вычислять фактические значения μ и σ, только выяснить, как они сравниваются друг с другом.
   (a) i. 3, 5, 5, 5, 8, 11, 11, 11, 13.
   ii. 3, 5, 5, 5, 8, 11, 11, 11, 20.
   (b) i. –20, 0, 0, 0, 15, 25, 30, 30.
   ii. –40, 0, 0, 0, 15, 25, 30, 30.
   (c) i. 0, 2, 4, 6, 8, 10.
   ii. 20, 22, 24, 26, 28, 30.
   (d) i. 100, 200, 300, 400, 500.
   ii. 0, 50, 300, 550, 600.

2.7. *[3]* Постройте распределение вероятностей, где ни одна из масс не лежит в пределах одного s от среднего.

2.8. *[3]* Как арифметическое и геометрическое средние сравниваются у случайных целых чисел?

2.9. *[3]* Покажите, что среднее арифметическое равно среднему геометрическому, когда все члены одинаковы.

Корреляционный анализ

2.10. *[3]* Истина или ложь: коэффициент корреляции –0,9 означает более сильные линейные соотношения, чем коэффициент корреляции 0,5. Объясните почему.

2.11. *[3]* Каков будет коэффициент корреляции между годовыми зарплатами выпускников колледжей и выпускников средней школы в данной компании, если по каждой возможной профессии выпускники колледжа всегда получали:
(a) на 5000 долл. больше, чем выпускники средней школы?
(b) на 25% больше, чем выпускники средней школы?
(c) на 15% меньше, чем выпускники средней школы?

2.12. *[3]* Какова была бы корреляция между возрастами мужей и жен, если бы мужчины всегда были женаты на женщинах, которые были:
(a) на три года моложе?
(b) на два года старше?
(c) вдвое моложе, чем они?

2.13. *[5]* Используйте данные или литературу, найденные в Google, чтобы оценить (измерить) силу корреляции между:
(a) отбоями и пробежками, зарегистрированными для хиттеров в бейсболе;
(b) отбоями и пробежками, допущенными питчерами в бейсболе.

2.14. *[5]* Вычислите коэффициенты корреляции Пирсона и Спирмена для равномерно расставленных выборок точек (x, x_k). Как эти значения изменяются в зависимости от увеличения k?

Логарифмы

2.15. *[3]* Покажите, что логарифм любого числа меньше 1 отрицателен.

2.16. *[3]* Покажите, что логарифм нуля не определен.

2.17. *[5]* Докажите, что

$$xy = b^{(\log_b x + \log_b y)}.$$

2.18. *[5]* Докажите правильность формулы для замены логарифма по основанию b на логарифм по основанию a

$$\log_a(x) = \log_b(x) \Big/ \log_b(a).$$

Реализация проектов

2.19. *[3]* Найдите некие интересные наборы данных и сравните сходство их средних значений и медиан. Каковы распределения, в которых среднее и медиана наиболее различаются?

2.20. *[3]* Найдите некие интересные наборы данных и определите в них все пары интересных корреляций. Можно начать с того, что доступно по адресу `http://www.data-manual.com/data`. Что вы нашли?

Вопросы на интервью

2.21. *[3]* Какова вероятность получить ровно k орлов при n бросках, когда у монеты есть вероятность p выпадения орла при каждом броске? Как насчет k или большего количества орлов?

2.22. *[5]* Предположим, что вероятность выпадения орла при i-м броске постоянно меняющейся монеты составляет $f(i)$. Как эффективно вычислить вероятность получения точно k орлов при n бросках?

2.23. *[5]* В полупериод баскетбольной игры вам предлагаются две возможные задачи.

(a) Сделайте три броска и забросьте по крайней мере два из них.

(b) Сделайте восемь бросков и забросьте по крайней мере пять из них.

Какую задачу вам следует выбрать, чтобы получить лучший шанс на победу в игре?

2.24. *[3]* Бросив монету десять раз, вы получили восемь орлов и две решки. Как бы вы выяснили беспристрастность монеты? Каково p-значение?

2.25. *[5]* Дан поток из n чисел, покажите, как случайным образом выбрать уникальное число, равномерно распределенное по множеству. Что делать, если вы не знаете n заранее?

2.26. *[5]* k страйков начинаются с i-го броска в последовательности из n бросков монеты, когда результат i-го броска и следующие $k - 1$ бросков идентичны. Например, последовательность OPPPOO содержит 2 страйка, начиная со второго, третьего и пятого бросков. Каково ожидаемое количество k страйков при бросках абсолютно симметричной монеты (т.е. монеты, вероятность выпадения орла или решки которой равно точно 1/2).

2.27. *[5]* Человек случайным образом вводит восьмизначное число в карманный калькулятор. Какова вероятность того, что число будет выглядеть одинаково, даже если перевернуть калькулятор?

2.28. *[3]* Вы играете в кости, у вас есть два варианта.

(a) Бросить кости один раз и получить выигрыш с призом, равным числу результата (например, 3 долл. за номер 3), а затем остановить игру.

(b) Вы можете отказаться от первой награды в соответствии с ее результатом и бросить кубики во второй раз, чтобы получить вознаграждение таким же образом.

Какую стратегию вы должны выбрать, чтобы максимизировать свой выигрыш? Для каких результатов первого броска вы должны выбрать вторую

игру? Каково статистическое ожидание вознаграждения, если вы выберете вторую стратегию?

2.29. *[3]* Что такое A/B-тестирование и как оно работает?

2.30. *[3]* В чем разница между статистической независимостью и корреляцией?

2.31. *[3]* Мы часто говорим, что корреляция не означает причину. Что это значит?

2.32. *[5]* В чем разница между асимметричным распределением и единообразным?

Конкурсы Kaggle

2.33. Пары причинно-следственных связей: корреляция или причинность.
`https://www.kaggle.com/c/cause-effect-pairs`

2.34. Предскажите следующее "случайное число" в последовательности.
`https://www.kaggle.com/c/random-number-grand-challenge`

2.35. Предскажите судьбу животных в приюте для домашних животных.
`https://www.kaggle.com/c/shelter-animal-outcomes`

Глава 3

Манипулирование данными

> Меня дважды спрашивали: "Если помолясь, мистер Бэббидж, вы введете в машину неправильные цифры, будут ли получены правильные ответы?"
>
> ...Я не в состоянии понять, какую путаницу идей может спровоцировать такой вопрос.
>
> — Чарльз Бэббидж (Charles Babbage)

Большинство аналитиков данных тратит бо́льшую часть своего времени, очищая и форматируя данные. Остальные тратят бо́льшую часть своего времени, жалуясь на то, что у них нет подходящих данных, чтобы сделать то, что они хотят.

В этой главе мы рассмотрим некоторые основные принципы работы с данными. Не такие высокопарные вещи, как статистика или машинное обучение, а кропотливая работа по поиску и очистке данных, которая известна под названием *манипулирование данными* (data munging).

Хота такие практические вопросы, как "Какая библиотека или язык программирования лучше подходят?", безусловно, важны, ответы на них меняются так быстро, что подобная книга — это совсем не то место, где их можно найти. Поэтому я лучше буду придерживаться уровня общих принципов, а не строить книгу вокруг определенного набора программных инструментов. Тем не менее в этой главе мы обсудим несколько доступных ресурсов и ответим на вопросы: почему они существуют, что они делают и как их использовать лучше всего.

Первым этапом любого проекта в науке о данных является получение правильных данных. Но зачастую это очень тяжело. Эта глава познакомит с обширными охотничьими угодьями для источников данных, а затем представит методы очистки того, что вы подстрелите. Обработка данных, чтобы вы могли безопасно их анализировать, имеет решающее значение для достижения существенных результатов. Как мог бы сказать сам Бэббидж более кратко: "Мусор на входе, мусор на выходе".

3.1. Языки для науки о данных

В теории любой достаточно мощный язык программирования способен выразить любой серьезный алгоритм вычисления. Однако на практике некоторые языки программирования оказываются намного лучше, чем другие, для решения конкретных задачах. Лучше здесь может означать *проще для программиста* или, возможно, *эффективней при вычислении*, в зависимости от решаемой задачи.

Вот основные языки программирования, применяемые в науке о данных.

- *Python.* Это современный популярный язык программирования для науки о данных. Python содержит множество средств, облегчающих манипулирование базовыми данными, включая регулярные выражения. Это интерпретируемый язык, ускоряющий и упрощающий процесс разработки. Python поддерживает огромное разнообразие библиотек, способных выполнить все: от очистки до визуализации, включая линейную алгебру и машинное обучение.

  Возможно, наибольшим недостатком языка Python является его эффективность: интерпретирующие языки не могут конкурировать с компилирующими по скорости. Но существуют и компиляторы Python, которые по-своему обеспечивают компоновку с использованием эффективных библиотек языков C/assembly для задач, требующих больших вычислительных ресурсов. Таким образом, язык Python, вероятно, должен быть вашим основным инструментом при работе с материалом, который мы представляем в этой книге.
- *Perl.* Этот язык *использовался раньше* для манипулирования данными в Интернете, прежде чем Python съел его на обед. В индексе популярности языков программирования TIOBE (`http://www.tiobe.com/tiobe-index`) язык Python впервые превзошел Perl по популярности в 2008 году и первенства больше не уступал. Тому есть несколько причин, включая улучшенную поддержку объектно-ориентированного программирования и более доступные библиотеки, но суть в том, что на данный момент очень мало причин для запуска проектов на языке Perl. Однако не удивляйтесь, если вы встретитесь с ним в каком-либо антикварном проекте.
- *R.* Это язык программирования для статистиков с самыми развитыми библиотеками, доступными для анализа и визуализации данных. Мир науки о данных разделен на лагеря приверженцев языков R и Python, где язык R, возможно, больше подходит для исследований, а Python — для рабочего применения. Стиль взаимодействия с языком R довольно интересен, поэтому я рекомендую вам немного поэкспериментировать с ним, чтобы убедиться, не лучше ли он подходит для вас.

  Между языками R и Python существуют связи, поэтому вы вполне можете вызывать библиотечные функции языка R в коде Python. Он обеспечи-

вает доступ к расширенным статистическим методам, которые могут не поддерживаться базовыми библиотеками Python.

- *Matlab.* Часть Mat здесь означает *матрица*, поскольку Matlab — это язык, предназначенный для быстрого и эффективного манипулирования матрицами. Как мы увидим, многие алгоритмы машинного обучения сводятся к операциям с матрицами, что делает язык Matlab естественным выбором для инженеров, программирующих на высоком уровне абстракции. Matlab — это частная система. Однако бо́льшая часть его функций доступна в GNU Octave — альтернативной реализации с открытым исходным кодом.
- *Java и C/C++.* Это основные языки программирования для разработки больших важных систем и приложений с большими данными. Системы параллельной обработки, такие как Hadoop и Spark, написаны на языках Java и C++ соответственно. Если вы живете в мире распределенных вычислений, то вы живете в мире Java и C++, а не других перечисленных здесь языков.
- *Mathematica/Wolfram Alpha.* Mathematica — это запатентованная система, обеспечивающая вычислительную поддержку всех аспектов числовой и символической математики, построенная на базе менее частного языка программирования Wolfram. Это основа вычислительного механизма знаний Wolfram Alpha, который обрабатывает запросы на естественном языке с помощью смеси алгоритмов и предварительно обработанных источников данных. Узнайте больше по адресу `http://www.wolframalpha.com`.

  Признаюсь, Mathematica мне не безразлична.[1] Это именно то, к чему я стремлюсь, когда занимаюсь небольшим анализом данных или моделированием, но цена традиционно выводит его из диапазона доступных для многих пользователей. Возможно, выпуск языка Wolfram откроет его теперь для более широкой общественности.
- *Excel.* Программы для работы с электронными таблицами, такие как Excel, являются мощными инструментами для исследовательского анализа данных, например просмотра некого набора данных, чтобы увидеть, что именно он содержит. Такие приложения заслуживают нашего уважения.

  Полнофункциональные программы для работы с электронными таблицами содержат удивительное количество скрытых функций для опытных пользователей. Мой студент, который стал руководителем в Microsoft, сказал мне, что 25% всех запросов на новые функции для предлагаемых возможностей Excel уже присутствуют в нем. Специальные функции и функции манипулирования данными, которые вам нужны, вероятно, уже есть в Excel, если вы посмотрите достаточно внимательно, точно так же,

[1] Буду откровенен: я знаю Стивена Вольфрама (Stephen Wolfram) более тридцати лет. На самом деле мы изобрели iPad вместе [105, 106].

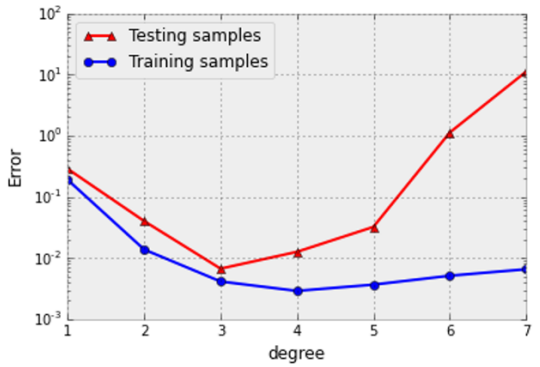
как библиотека Python, возможно, уже содержит то, что вам необходимо, нужно только найти, если хорошо поискать.

3.1.1. Важность окружения интерактивной оболочки

Основным результатом проекта науки о данных является не программа. Это даже не набор данных. Не результат выполнения программы для ваших данных и не просто письменный отчет.

Результатом каждого проекта в науке о данных должна быть *интерактивная оболочка* (computable notebook), увязывающая вместе код, данные, результаты вычислений и письменный анализ того, что вы узнали в процессе. На рис. 3.1 представлена выдержка из оболочки Jupyter/IPython (Jupyter/IPython notebook), демонстрирующая, как он интегрирует код, графику и документацию в осмысленный документ, который может быть выполнен как программа.

```
In [40]: degrees = range(1, 8)
         errors = np.array([regressor3(d) for d in degrees])
         plt.plot(degrees, errors[:, 0], marker='^', c='r', label='Testing samples
         plt.plot(degrees, errors[:, 1], marker='o', c='b', label='Training sample
         plt.yscale('log')
         plt.xlabel("degree"); plt.ylabel("Error")
           = plt.legend(loc='best')
```



Согласно степени мы рассмотрим две области производительности модели.

- *Недостаточное оснащение* (степень < 3). Характеризуется тем, что ошибка тестирования станет меньше, если мы увеличим емкость модели.
- *Переоснащение* (степень > 3). Характеризуется тем, что ошибка тестирования станет больше, если мы увеличим емкость модели. Обратите внимание, что ошибка обучения уменьшается или остается неизменной!

Рис. 3.1. Оболочка Jupyter/IPython объединяет код, результаты вычислений и документацию

Причина, по которой это так важно, заключается в том, что результаты вычислений являются результатом длинных последовательностей выбора параметров и проектных решений. Это создает несколько проблем, решаемых окружениями интерактивных оболочек.

- Вычисления должны быть *воспроизводимы*. Мы должны иметь возможность повторно запускать те же программы и получать точно те же результаты. Это значит, что конвейеры данных должны быть *завершенными*: должна быть возможность взять необработанный ввод и получить окончательный вывод. Это ужасная судьба — начинать с необработанного набора данных, выполнять некоторую обработку, редактировать/форматировать файлы данных вручную, а затем выполнять дополнительную обработку, поскольку то, что сделано вручную, нельзя легко выполнить снова для другого набора данных или отменить после того, как вы поймете, что, возможно, ошиблись.
- Вычисления должны быть *настраиваемыми*. Зачастую повторное вычисление или оценка приводят к изменению одного или нескольких параметров либо алгоритмов. Проведение нового вычисления требует перезапуска интерактивной оболочки. Нет ничего более обескураживающего, чем предоставить продукт с большими данными без истории происхождения и сказать, что это конечный результат, и вы ничего не можете изменить. Интерактивная оболочка никогда не останавливается, пока не будет завершен весь проект.
- Конвейеры данных должны быть *документированы*. Эти интерактивные оболочки позволяют интегрировать текст и результаты визуализации с вашим кодом, предоставляя великолепный способ рассказать о том, что вы делаете и почему так, как не могут позволить традиционные программные среды.

На заметку. Для создания и демонстрации результатов любого проекта по науке о данных используйте интерактивную оболочку, такую как IPython или Mathematica.

3.1.2. Стандартные форматы данных

Данные поступают из разных мест и в разных форматах. Какое из представлений окажется лучше всего, зависит от того, кто является конечным потребителем. Диаграммы и графики — это замечательные способы донести до людей значение числовых данных. Методам визуализации данных будет посвящена глава 6. Но эти изображения, по сути, бесполезны в качестве источника данных для вычислений. От рисованных карт до Google Maps еще очень далеко.

Лучшие форматы вычислительных данных имеют несколько полезных свойств.

- *Компьютерам легко их анализировать.* Данные, записанные в удобном формате, предназначены для повторного использования в другом месте. Сложные форматы данных зачастую поддерживаются API, которые управляют техническими деталями, обеспечивая надлежащий формат.
- *Они просты для чтения людьми.* Наглядность данных — немаловажный фактор во многих контекстах. Какой из файлов данных в этом каталоге подходит мне лучше? Что мы знаем о полях данных в этом файле? Каков общий диапазон значений для каждого конкретного поля?

  Эти варианты говорят об огромной ценности возможности открыть файл данных в текстовом редакторе, чтобы просмотреть его. Как правило, это означает представление данных в удобочитаемом текстовом формате с записями, разграниченными отдельными строками, и полями, разделенными символами-разделителями.
- *Они могут использоваться другими разнообразными инструментами и системами.* Стремление изобрести собственный стандарт данных бьется в сердце любой корпорации, и большинство разработчиков программного обеспечения предпочли бы поделиться скорее зубной щеткой, чем форматом файла. Но это побуждения, которых следует избегать. Мощность данных заключается в их смешении и сопоставлении с другими ресурсами данных, что лучше всего достигается за счет популярности стандартных форматов.

Одним из свойств, которое я пропустил в этом списке, является *краткость*, поскольку это, как правило, не является серьезной проблемой для большинства приложений, работающих в современных вычислительных системах. Стремление минимизировать затраты на хранение данных зачастую противоречит другим целям. Разумная упаковка нескольких полей в старшие биты целых чисел экономит место, но за счет того, что делает данные несовместимым и нечитаемыми.

Популярные утилиты сжатия, такие как gzip, прекрасно справляются с удалением избыточности удобного для чтения людьми формата. Ныне диски невероятно дешевы: когда я пишу это, вы можете купить диск емкостью 4 ТБ примерно за 100 долл., что значительно меньше стоимости одного часа работы, потраченного на разработку более жесткого формата. Если вы не работаете в масштабах Facebook или Google, краткость не имеет столь высокого значения, о котором вы, вероятно, думаете.[2]

[2] Мои друзья в Google утверждают, что нередко бывают небрежны в отношении объемов даже в масштабе петабайта.

Вот наиболее важные форматы и представления данных, о которых следует знать.

- *Файлы CSV* (Comma Separated Value — значения, разделенные запятыми).

  Эти файлы имеют самый простой и самый популярный формат для обмена данными между программами. Как можно заметить, каждая строка представляет одну запись с полями, разделенными запятыми. Однако проблемы создают специальные символы и текстовые строки: что, если ваши данные об именах содержат запятую, например "Thurston Howell, Jr". Формат CSV предоставляет специальные коды для таких символов, чтобы они не рассматривались как разделители, но это отнюдь не изящно. Лучшей альтернативой является использование более редкого символа-разделителя, как в файлах TSV (Tab Separated Value — значения, разделенные знаками табуляции).

  Наилучшей проверкой правильности форматирования файла CSV является Microsoft Excel или другая программа для работы с электронными таблицами, они должны читать его без проблем. Убедитесь, что результат каждого проекта проходит этот тест, если он представлен в формате CSV, чтобы избежать последующих проблем.

- *XML* (eXtensible Markup Language — расширяемый язык разметки). Структурированные, но не табличные данные зачастую записываются в виде текста с аннотациями. Естественный вывод тегов именованных объектов для текста — заключить соответствующие подстроки текста в скобки, обозначив человека, место или предмет. Я пишу эту книгу на языке форматирования LaTex с заключающимися в скобки командами, расположенными вокруг математических выражений и выделенного *курсивом текста*. Все веб-страницы написаны на HTML — языке разметки гипертекста, который организует документы с помощью таких команд, как `<b>` и `</b>`, заключающих текст, **выделенный полужирным шрифтом**.

  Язык XML применяется для написания спецификаций таких языков разметки. Собственная спецификация XML позволяет пользователю анализировать любой документ, соответствующий спецификации. Разработка таких спецификаций и их полное соблюдение требует дисциплины, но оно того стоит. В первой версии нашей системы анализа текста Lydia мы писали наши разметки на "псевдо-XML", читаемом специальными анализаторами, которые правильно обрабатывали 99% документов, но отказывали всякий раз, когда мы пытались их расширить. После болезненного перехода на язык XML все работало надежнее *и* эффективнее, поскольку мы могли установить более быстрые анализаторы XML с открытым исходным кодом, чтобы справиться со всей тяжелой работой по обеспечению соблюдения наших спецификаций.

- *Базы данных SQL* (Structured Query Language — язык структурированных запросов). Таблицы естественным образом структурированы вокруг отдельных таблиц данных. В отличие от них, реляционные базы данных отлично подходят для манипулирования множеством различных, но связанных таблиц с помощью языка SQL для предоставления неуклюжего, но мощного языка запросов.

  Любая рабочая система баз данных импортирует и экспортирует записи в виде файлов CSV или XML, а также использует внутренний дамп содержимого. Внутреннее представление в базах данных непрозрачно, поэтому описывать их как формат данных не совсем корректно. Тем не менее я обсуждаю их здесь потому, что базы данных SQL, как правило, оказываются лучшим и более мощным решением, чем произвольное управление несколькими файлами данных.

- *JSON* (JavaScript Object Notation — текстовый формат обмена данных JavaScript). Это формат для передачи объектов данных между программами. Это естественный способ передачи состояния переменных или структур данных из одной системы в другую. Он представляет собой в основном список пар "атрибут-значение", соответствующих именам переменных/полей и связанных с ними значений:

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

Поскольку библиотечные функции, которые поддерживают чтение и запись объектов JSON, легко доступны на всех современных языках программирования, это стало очень удобным способом хранения структур данных для последующего использования. Объекты JSON читаемы человеком, но выглядят довольно загроможденными, представляя массивы записей по сравнению с файлами CSV. Используйте их для сложных структурированных объектов, но не для простых таблиц данных.

- *Буферы протокола*. Это не зависящий от языка или платформы способ сериализации структурированных данных для передачи их между приложениями и хранения. По сути, это облегченные версии XML (где вы определяете формат структурированных данных), предназначенные для передачи небольших объемов данных между такими программами, как JSON. Этот формат данных используется для большей части межмашинного общения в Google. Apache Thrift — это подобный стандарт, используемый на Facebook.

3.2. Сбор данных

Важнейшей проблемой в любой модели или проекте науки о данных является поиск правильного набора данных. Определение подходящих источников данных — это искусство, которое вращается вокруг трех основных вопросов.

- У кого могут быть данные, которые мне на самом деле нужны?
- Почему он может предоставить их мне?
- Как они могут попасть мне в руки?

В данном разделе мы рассмотрим ответы на эти вопросы. Мы опишем наиболее популярные источники данных, и вы, вероятно, сможете понять почему. Затем мы рассмотрим основные механизмы получения доступа, в том числе интерфейсы API, очистку и регистрацию.

3.2.1. Охота на данные

У кого есть данные и как их получить? Некоторые из вероятных подозреваемых рассматриваются ниже.

Компании и частные источники данных

Крупные компании, такие как Facebook, Google, Amazon, American Express и Blue Cross, обладают огромным количеством интересных данных о пользователях и транзакциях. Эти данные можно было бы использовать, чтобы сделать мир лучше. Проблема в том, что получить к ним доступ извне обычно невозможно. Компании не хотят делиться данными по двум веским причинам.

- Бизнес-проблемы и страх помочь своим конкурентам.
- Проблемы конфиденциальности и страх обидеть своих клиентов.

Вот душераздирающая история о том, что может случиться при публикации корпоративных данных, которая произошла, когда компания AOL предоставила ученым набор данных о миллионах запросов в свою поисковую систему, тщательно устранив идентифицирующую информацию. Первое, что обнаружили ученые, было то, что наиболее частыми запросами были отчаянные попытки пользователей добраться до других поисковых систем, таких как Google. Это никак не способствовало повышению доверия общественности к качеству поискового механизма AOL.

Их второе открытие заключалось в том, что задача обеспечения анонимности поисковых запросов оказалось намного сложнее, чем предполагалось ранее. Конечно, вы можете заменить имена пользователей идентификационными номерами, однако не так уж и сложно определить, кто именно из парней с

Лонг-Айленда регулярно делал запросы по словам *Steven Skiena* и *Stony Brook*, а также обращался по адресу `https://twitter.com/search?q=Skiena&src=sprv`. В самом деле, как только стало известно, что в результате разглашения этих данных была раскрыта личность людей, ответственные за это лица были уволены, а набор данных исчез. Конфиденциальность пользователя важна, и этические вопросы, связанные с наукой о данных, будут обсуждаться в разделе 12.7.

Так что не думайте, что вам удастся заинтересовать компании разглашением конфиденциальных пользовательских данных. Однако многие ответственные компании, такие как *The New York Times*, Twitter, Facebook и Google, публикуют определенные данные, как правило, с помощью ограниченных интерфейсов прикладных программ (Application Program Interface — API). Для этого у них обычно есть две причины.

- Предоставление клиентам и третьим лицам данных, которые могут увеличить продажи. Например, публикация данных о частоте запросов и стоимости рекламы может побудить бо́льшее количество людей размещать рекламу на данной платформе.
- Как правило, для компании лучше предоставлять интерфейсы API с хорошей защитой, чем терпеть регулярные попытки взлома сайта.

Таким образом, ищите общедоступные интерфейсы API, прежде чем читать раздел 3.2.2. Вы не найдете именно то содержимое и в нужном объеме, о котором мечтаете, но, вероятно, этого будет достаточно для начала. Не забывайте об ограничениях и условиях использования.

Другие организации позволяют загружать массу интересных данных для автономного анализа, как в случае с N-граммами Google, IMDb и наборами данных о тарифах такси, рассматривавшихся в главе 1. Большие наборы данных зачастую содержат ценные метаданные, такие как названия книг, подписи к изображениям и отредактированные статьи, которые можно изменять при хорошем воображении.

И наконец, у большинства организаций есть внутренние наборы данных, которые имеют отношение к их бизнесу. Как сотрудник, вы должны быть в состоянии получить привилегированный доступ к ним, пока вы там работаете. Имейте в виду, что у компаний есть внутренние политики доступа к данным, поэтому на вас по-прежнему будут распространяться определенные ограничения. Нарушение условий этой политики — отличный способ стать бывшим сотрудником.

Правительственные источники данных

Сбор данных является одной из важнейших задач правительства. Требование о том, чтобы Соединенные Штаты проводили перепись населения, является обязательным условием Конституции США и проводится по расписанию каждые десять лет, начиная с 1790 года.

Городские власти, правительства штатов и федеральные власти все больше и больше стремятся открывать эти данные, облегчая их новые способы применения и улучшая работу правительства. Веб-сайт `http://Data.gov` является инициативой федерального правительства по централизованному сбору источников данных, и по последним подсчетам он насчитывает более 100 000 наборов данных!

Правительственные данные отличаются от промышленных данных тем, что в принципе они принадлежат народу. *Закон о свободе информации* (Freedom of Information Act — FOI) позволяет любому гражданину подать официальный запрос на любой государственный документ или набор данных. Такой запрос запускает процесс определения того, что может быть опубликовано без ущерба для национальных интересов или нарушения конфиденциальности.

Правительства штатов действуют согласно пятидесяти различным сводам законов, поэтому данные, которые строго секретны в одной юрисдикции, могут быть свободно доступны в других. В крупных городах, таких как Нью-Йорк, объем обработки данных больше, чем в некоторых штатах, опять же с учетом ограничений, которые зависят от местоположения.

Я рекомендую следующий способ мышления о государственных документах. Если после некоторых усилий вы не можете найти то, что вам нужно в Интернете, выясните, какое агентство, вероятно, обладает этой информацией. Позвоните им, чтобы узнать, могут ли они помочь вам найти то, что вам нужно. Но если вам не идут на встречу, не стесняйтесь напомнить о законе о свободе информации. Сохранение конфиденциальности, как правило, является самой большой проблемой при принятии решения о том, можно ли опубликовать конкретный набор правительственных данных.

Академические наборы данных

Существует огромный мир научных исследований, охватывающий все, что человечество считает полезным знать. Все большая часть научных исследований подразумевает создание больших массивов данных. Многие журналы в настоящее время требуют, чтобы исходные данные были доступны другим исследователям еще до публикации. Будьте уверены, что сможете найти огромное количество экономических, медицинских, демографических, исторических и других научных данных, если поищете достаточно внимательно.

Ключом к поиску этих наборов данных является отслеживание соответствующих документов. Существует академическая литература практически на любую интересующую вас тему. Google Scholar является наиболее доступным источником научных публикаций. Проводите поиск по теме, и, возможно, ключевым словам "Open Science" или "data". Исследовательские публикации, как правило, содержат указатели на то, где можно найти связанные с ними данные. Если нет, то обращение к автору напрямую с запросом должно быстро дать желаемый результат.

Самым большим преимуществом использования опубликованных наборов данных является то, что кто-то уже усердно работал над их анализом еще до того, как вы их получили, поэтому эти ранее обработанные данные могли бы стать источником новых интересных результатов. Постановка новых вопросов для старых данных обычно открывает новые возможности.

Зачастую интересные проекты в науке о данных подразумевают сотрудничество между исследователями из различных дисциплин, таких как социальные и естественные науки. Эти люди говорят на разных языках, что вам на первый взгляд может показаться пугающим. Однако сотрудничество, как правило, приветствуется, и, как только вы преодолеете проблемы терминологии, их проблемы обычно можно понять на вполне разумном уровне без специального обучения. Будьте уверены, что люди из других дисциплин, как правило, не умнее вас.

Трудовой вклад

Иногда вам придется работать над собственными данными, а не просто брать их у других. Многие исторические данные все еще существуют только в книгах или других бумажных документах, поэтому требуют ручного ввода и обработки. График или таблица могут содержать информацию, которая вам нужна, но может быть довольно сложно получить числа из графиков, упакованных в файле PDF (portable document format — формат переносимого документа).

Я заметил, что люди, ориентированные на вычисления, значительно переоценивают количество усилий, необходимых для ввода данных вручную. При одной записи в минуту вы можете легко ввести 1000 записей всего за два рабочих дня. Вместо этого ученые, как правило, прилагают огромные усилия, пытаясь избежать такой тяжелой работы. Они пытаются использовать системы оптического распознавания символов (OCR), которые создают беспорядок в файле, или тратят куда больше времени на очистку результатов сканирования, чем его потребуется, чтобы просто набрать данные заново.

Один из возможных вариантов — заплатить кому-то другому, чтобы он сделал тяжелую работу за вас. Такие краудсорсинговые платформы, как Amazon Turk и CrowdFlower, позволяют вам нанять армию людей, которые помогут вам

извлечь данные или даже осуществить их первичную обработку. Такие требующие человеческого вмешательства задачи, как маркировка изображений или ответы на опросы, особенно хороши для использования дистанционных сотрудников. Более подробно краудсорсинг будет обсуждаться в разделе 3.5.

Многие удивительные ресурсы открытых данных были созданы группами разработчиков, такими как Википедия, Freebase и IMDb. Однако следует помнить одну важную вещь: люди обычно работают лучше, когда вы им платите.

3.2.2. Скрепинг данных

Веб-страницы зачастую содержат ценные текстовые и числовые данные, которые мы хотели бы заполучить в свои руки. Например, в нашем проекте по созданию системы ставок в игре хай-алай нам нужно было представить системе результаты вчерашних матчей и расписание игр, которые проходили сегодня. Наше решение заключалось в том, чтобы извлечь данные о ставках в игре хай-алай с сайтов, размещающих эту информацию для своих поклонников.

Для этого мы использовали два разных этапа: спайдеринг и скрепинг.

- *Спайдеринг* (spidering) — процесс загрузки подходящего набора страниц для анализа.
- *Скрепинг* (scraping) — настоящее искусство извлечения содержимого из каждой страницы, чтобы подготовить его к анализу.

Дело в том, что веб-страницы обычно написаны на простых для понимания языках форматирования, таких как HTML и/или JavaScript. Ваш браузер знает эти языки и интерпретирует текст веб-страницы как программу, чтобы определить, как ее отобразить. При вызове функции, которая подражает/имитирует веб-браузер, ваша программа может загрузить любую веб-страницу и интерпретировать ее содержимое для анализа.

Традиционно программы скрепинга представляли собой специфические для сайта сценарии, созданные для вскрытия определенных шаблонов HTML, окружающих интересующее содержимое. Этот подход опирается на тот факт, что большинство страниц на определенных веб-сайтах сами генерируются программами, а следовательно, их формат очень предсказуем. Однако такие сценарии обычно уродливы и хрупки, они ломаются каждый раз, когда целевой веб-сайт меняет внутреннюю структуру своих страниц.

Сегодня библиотеки на таких языках, как Python (см. BeautifulSoup), упрощают написание надежных программ загрузки и скрепинга. Для каждого популярного веб-сайта кто-то, вероятно, *уже написал* такую программу и сделал ее доступной на SourceForge или Github, так что ищите, прежде чем программировать.

Одни случаи спайдеринга могут быть тривиальными, например посещение определенного URL (Uniform Resource Locator — унифицированный указатель ресурса) через регулярные промежутки времени. Такие шаблоны встречаются, в частности, при мониторинге рейтинга продаж этой книги на странице Amazon. Несколько более сложные случаи спайдеринга основаны на регулярности имен базовых URL. Если на всех страницах сайта указана дата или идентификационный номер продукта, например `http://www.amazon.com/gp/product/1107041376/`, перебор всего диапазона интересующих значений становится только вопросом подсчета.

Наиболее передовой формой спайдеринга является *веб-сканирование* (web crawling), при котором вы систематически рекурсивно просматриваете все исходящие ссылки с заданной корневой страницы, пока не посетите все страницы на целевом веб-сайте. Это то, что Google делает при индексации Интернета. Вы тоже можете сделать это, имея достаточно терпения или найдя библиотеки Python для веб-сканирования.

Пожалуйста, не забывайте из вежливости ограничивать частоту сканирования заданного веб-сайта. Считается невежливым посещать сайт чаще, чем раз в секунду, и действительно лучше не поступать так, поскольку провайдеры зачастую блокируют доступ тем людям, которые их "задалбывают" (hammering).

Каждый крупный веб-сайт содержит документ об *условиях использования*, который излагает ограничения на то, что вы можете законно делать с любыми связанными данными. По правде говоря, большинство сайтов оставит вас в покое при условии, что вы не будете их особо беспокоить или перераспространять данные, которые вы просматриваете. Поймите, что это только наблюдение, а не юридическое заключение. Лучше почитайте о деле Аарона Шварца (Aaron Schwartz), когда известный интернет-деятель был привлечен к уголовной ответственности за нарушение условий предоставления услуг в журнальных статьях, полученных поиском/очисткой, и буквально доведен до смерти. Если вы пытаетесь профессионально заняться веб-скрепингом, убедитесь, что руководство понимает условия предоставления услуг, прежде чем проявлять слишком творческий подход к чужой собственности.

3.2.3. Регистрация

Если вы имеете потенциальный источник данных, относитесь к нему как к собственному. Внутренний доступ к веб-службе, устройству связи или лабораторному прибору дает вам право и накладывает ответственность регистрировать все действия для последующего анализа.

Можно сделать удивительные вещи с помощью сбора данных из веб-блогов и сенсорных устройств, взрыв которых должен вскоре произойти с появлением

Интернета вещей (Internet of Things). Акселерометры в сотовых телефонах можно использовать для измерения силы землетрясений, а корреляции событий в регионе достаточно, чтобы отфильтровать людей, едущих по ухабистым дорогам или оставляющих свои телефоны в сушилке для одежды. Мониторинг данных GPS парка такси отслеживает пробки на улицах города. Компьютерный анализ потоков изображений и видео открывает двери для бесчисленных приложений. Еще одна замечательная идея заключается в использовании камер в качестве метеорологических инструментов — можно контролировать цвет неба на фоне миллионов фотографий, ежедневно загружаемых на сайты фотографий.

Основная причина, по которой ваша система собирает данные, заключается в том, что вы это можете. Вероятно, вы сейчас не знаете точно, что с ними делать, но любой правильно построенный набор данных станет полезным, как только его размер достигнет определенной критической массы.

Текущие затраты на хранение ясно показывают, насколько низок барьер для системы. Мой местный Costco продает в настоящее время диск на три терабайта по цене дешевле 100 долл., т.е. практически даром. Если каждая запись транзакции занимает 1 килобайт (одну тысячу символов), это устройство, в принципе, вмещает 3 млрд записей, примерно по одной на каждых двух человек в мире.

При проектировании любой системы регистрации важнейшими являются следующие соображения.

- Создавайте ее так, чтобы гарантировать минимум обслуживания. Установите ее и забудьте, предоставив ей достаточно памяти для неограниченного расширения и резервного копирования.
- Обеспечьте хранение всех возможных значений полей без проблем.
- Используйте удобочитаемый формат или базу данных транзакций, чтобы вы могли точно понимать, что там происходит, когда по прошествии месяцев или лет придет время сесть и проанализировать ваши данные.

3.3. Очистка данных

Принцип "мусор на входе, мусор на выходе" является фундаментальным в анализе данных. Путь от необработанных данных до очищенного, поддающегося анализу набора данных может быть очень долог.

При очистке данных для анализа может возникнуть множество потенциальных проблем. В этом разделе мы обсудим идентификацию артефактов обработки и интеграцию различных наборов данных. В центре нашего внимания находится обработка *перед тем*, как мы проведем настоящий анализ, чтобы гарантировать, что мусор никогда не попадет на первое место.

На заметку. Опытные реставраторы картин делают только то, что обратимо и не затрагивает оригинал. Они никогда не причиняют оригиналу вреда. Точно так же очистка данных всегда выполняется на копии исходных данных, в идеале с помощью такого конвейера, который вносит изменения систематическим и повторяемым способом.

3.3.1. Ошибки против артефактов

Согласно древнему еврейскому закону, если подозреваемый был единогласно признан виновным всеми судьями, то этот подозреваемый будет *оправдан*. Судьи заметили, что единодушное согласие зачастую свидетельствует о наличии системной ошибки в судебном процессе. Они рассуждали так, что, когда что-то кажется слишком хорошим, чтобы быть правдой, вероятно, где-то была допущена ошибка.

Если мы рассматриваем элементы данных как показатели некого аспекта реального мира, *ошибки* данных представляют информацию, которая в основном теряется при получении. *Гауссовский шум* (Gaussian noise), размывающий разрешение наших датчиков, представляет собой ошибку, постоянно ухудшающую точность измерений. Два часа пропуска в журналах регистрации, возникших из-за сбоя сервера, представляют собой ошибку данных: это информация, которую невозможно восстановить заново.

Артефакты, напротив, как правило, являются систематическими проблемами, возникающими в результате обработки исходной информации. Хорошая новость заключается в том, что артефакты обработки можно исправить, если исходный набор данных остается доступен. Плохая новость состоит в том, что эти артефакты следует обнаружить, прежде чем их можно будет исправить.

Ключом к обнаружению артефактов обработки является "проба на запах" (sniff test), при которой продукт анализируется достаточно внимательно, чтобы обнаружить неприятный запах. Что-то плохое обычно является чем-то неожиданным или удивительным, потому что люди, естественно, оптимисты. Удивительные наблюдения — это то, для чего живут аналитики данных. Действительно, такое понимание является основной причиной, по которой мы делаем то, что делаем. Но по моему опыту, большинство сюрпризов оказываются артефактами, поэтому мы должны смотреть на них скептически.

На рис. 3.2 представлены результаты расчетов проекта, в котором мы исследовали процесс научной публикации. Здесь показан временной ряд из 100 000 самых плодовитых авторов, сгруппированных по году их первой статьи, появившейся в Pubmed, по существу, чрезвычайно полной библиографии биомедицинской литературы.

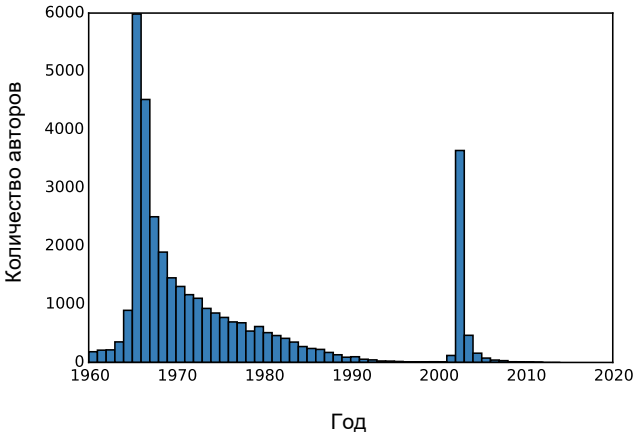


Рис. 3.2. Какие артефакты вы можете найти в этом временном ряду, подсчитав количество имен авторов, впервые появляющихся в научной литературе каждый год?

Внимательно изучите этот рисунок и посмотрите, сможете ли вы найти какие-либо артефакты, которые стоит прокомментировать. Я вижу по крайней мере два из них. Дополнительная награда ждет того, кто сможет выяснить причину проблемы.

Ключом к обнаружению артефактов является поиск аномалий в данных, которые противоречат тому, что вы ожидаете увидеть. Как *должно* выглядеть распределение по количеству новых авторов, и как оно должно меняться со временем? Для начала создайте предварительное распределение того, что вы ожидаете увидеть, чтобы затем суметь правильно оценить потенциальные аномалии по сравнению с этим.

Моя интуиция говорит, что распределение новых ведущих ученых должно быть довольно симпатичным, поскольку новые звезды рождаются с каждым последующим классом аспирантов. Я также предположил бы, что возможен постепенный сдвиг в сторону увеличения по мере роста совокупности людей, входящих в научное сообщество. Но это совсем не то, что я вижу на рис. 3.2. Итак, попробуйте перечислить аномалии и потенциальные артефакты...

Я вижу два больших пика, глядя на рис. 3.2: левый начинается приблизительно в 1965 году, а второй приходится на 2002 год. По здравом размышлении, крайний слева пик имеет смысл. Он приходится на год, когда Pubmed впервые начал систематически собирать библиографические записи. Хотя есть некоторые очень неполные данные за 1960–1964 годы, большинство ученых, которые публиковались на протяжении нескольких прошлых лет, "проявились" только с началом систематических записей в 1965 году. Таким образом, это объясняет пик слева, который затем снижается к 1970 году, что вполне похоже на ожидаемое нами распределение.

Но как насчет этого гигантского пика 2002 года? А снижение числа новых авторов практически до нуля в предшествующие ему годы? Аналогичное снижение также видно справа от большого пика. Всем главным ученым мира суждено было родиться в 2002 году?

Тщательная проверка записей в большом пике позволила выявить источник аномалии: имена. В первые дни Pubmed авторы идентифицировались по инициалам и фамилиям. Но в конце 2001 года *SS Skiena* стал *Steven S. Skiena*, поэтому он выглядел как новый автор, упавший с небес.

Но почему слева и справа от этого пика располагаются спады до нуля? Напомним, что мы ограничили это исследование 100 000 самых плодовитых ученых. Научная рок-звезда, появившаяся в 1998 году, вряд ли присутствует в этом рейтинге, поскольку ее имя было обречено измениться несколько лет спустя, не оставив ему времени для накопления достаточного количества публикаций. Подобные вещи случаются в крайней справа части распределения: ученые, возникшие только что, к 2010 году никак не смогут добиться полноценной карьеры всего за пару лет. Оба явления хорошо объясняются на основании имен авторов.

Очистка этих данных для объединения ссылок на имена и получения правильных результатов заняла несколько итераций. Даже после устранения пика 2002 года мы все же заметили значительный спад выдающихся ученых, начавших свою карьеру в середине 1990-х годов. Это было связано с тем, что многие люди, сделавшие половину карьеры перед переименованием и вторую половину после, не дотянули до порога отличной полноценной карьеры ни в один из этих периодов. Таким образом, мы должны были сопоставить все имена в полном объеме, *прежде чем* определить, кто был лучшим из 100 000 ученых.

На рис. 3.3 представлено наше окончательное распределение авторов, которое идеально соответствует ожидаемому распределению. Не спешите рационализировать то, как ваши данные выглядят на компьютере. Мои сотрудники в какой-то момент были готовы списать пик 2002 года на увеличение финансирования исследований или создание новых научных журналов. Всегда с подозрением относитесь к чистоте ваших данных и выясняйте, можно ли доверять им.

3.3.2. Совместимость данных

Мы говорим, что сравниваем "яблоки с яблоками", когда сравнение двух элементов корректно, т.е. эти элементы достаточно похожи, чтобы их можно было обоснованно противопоставить друг другу. Напротив, сравнения "яблок с апельсинами" в конечном итоге не имеют смысла.

- Нет смысла сравнивать вес 123,5 с весом 78,9, когда один представлен в фунтах, а другой в килограммах.

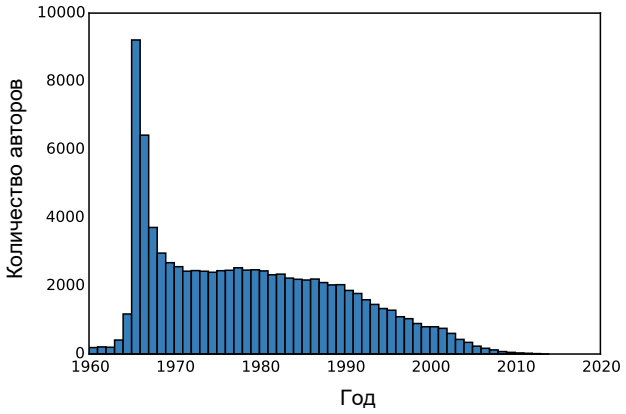


Рис. 3.3. В очищенных данных эти артефакты удалены, и итоговое распределение выглядит правильно

- Нет смысла непосредственно сравнивать фильм *Унесенные ветром* с фильмом *Аватар*, поскольку доллар 1939 года в 15,43 раза дороже доллара 2009 года.
- Нет смысла сравнивать цены на золото в полдень сегодня в Нью-Йорке и Лондоне, поскольку их разделяет пять часовых поясов, а на цены влияют происходящие события.
- Нет смысла сравнивать цену акций Microsoft на 17 февраля 2003 года с ценой 18 февраля 2003 года, потому что прошедшее дробление акций 2 к 1 сократило цену вдвое, но не отразилось на изменении реальной стоимости.

Эти типы проблем сопоставимости данных возникают при каждом объединении наборов данных. Здесь я надеюсь показать вам, насколько коварными могут быть такие проблемы сопоставимости, чтобы вы поняли, почему о них нужно знать. Кроме того, для некоторых важных классов преобразований я указываю способы их решения.

> *На заметку*. Проверяйте значение каждого из полей в любом наборе данных, с которым вы работаете. Если вы не понимаете, что там находится и каковы единицы измерения, у вас нет осмысленного способа использовать эти значения.

Преобразование единиц измерения

Количественная оценка наблюдений в физических системах требует стандартных единиц измерения. К сожалению, существует множество функционально эквивалентных, но несовместимых систем измерения. Моя 12-летняя дочь и я оба весим порядка 70, но один из нас в фунтах, а другой в килограммах.

Такие катастрофические вещи, как взрывы ракет, происходят, когда измерения вводятся в компьютерные системы с использованием неправильных единиц измерения. В частности, 23 сентября 1999 года НАСА потеряла космическую систему Mars Climate Orbiter за 125 млн долл. из-за проблемы преобразования английских единиц измерения.

Решать такие проблемы лучше всего, выбрав единую систему измерений и придерживаясь ее. Метрическая система предлагает несколько преимуществ по сравнению с традиционной английской системой. В частности, отдельные измерения естественным образом выражаются в виде единичных десятичных величин (например, 3,28 метра) вместо несопоставимых пар величин (5 футов, 8 дюймов). Эта же проблема возникает при измерении углов (радианы и градусы/секунды) и веса (килограммы и фунты/унции).

Соблюдение метрической системы само по себе не решает всех проблем сравнимости, поскольку ничто не мешает вам смешивать высоты в метрах и сантиметрах. Но это хорошее начало.

Как защитить себя от несовместимых единиц при объединении наборов данных? Главным оружием должна быть ваша бдительность. Убедитесь, что вы знаете предполагаемые единицы измерения для каждого числового столбца в своем наборе данных и проверьте их совместимость при объединении. Любой столбец, с которым не связаны единицы измерения или тип объекта, должен немедленно оказаться под подозрением.

При объединении записей из разных источников рекомендуется создать новое поле "origin" (происхождение) или "source" (источник), чтобы определить, откуда поступила каждая запись. Это дает, по крайней мере, надежду на то, что ошибки преобразования единиц измерения можно будет исправить впоследствии, систематически обрабатывая записи из проблемного источника.

Частично автоматизированная процедура обнаружения таких проблем может быть разработана на основе статистической значимости, что будет обсуждаться в разделе 5.3. Предположим, что мы должны были изобразить частоты роста людей в объединенном наборе данных в английских (футы) и метрических единицах. Мы увидим один пик в распределении около 1,8, а второй около 5,5. Наличие нескольких пиков в распределении должно вызвать у нас подозрения. *P*-значение, полученное в результате проверки значимости двух исходных групп, обеспечивает точный показатель степени, в которой наши подозрения подтверждаются.

Преобразования числовых представлений

Числовые средства легче всего включить в математические модели. Некоторые алгоритмы машинного обучения, такие как линейная регрессия и метод опорных векторов, работают только с данными в цифровой форме. Но даже

превращение чисел в числа может быть сложной задачей. Числовые поля могут быть представлены различными способами: в виде целых чисел (123), в виде десятичных чисел (123,5) или даже в виде дробей (123 1/2). Числа могут быть даже представлены в виде текста, что потребует преобразования "десяти миллионов" в 10000000 для числовой обработки.

Проблемы числового представления способны уничтожить ракету. Ракета Ariane 5 стоимостью 500 млн долл., запущенная 4 июня 1996 года, взорвалась через сорок секунд после взлета, и причина в конечном счете объяснялась неудачным преобразованием 64-разрядного числа с плавающей запятой в 16-разрядное целое число.

Различие между целыми числами и числами с плавающей запятой (вещественными числами) важно учитывать. Целые числа используются для подсчета: дискретные величины должны быть представлены как целые числа. Физически измеренные величины никогда не определяются количественно точно, потому что мы живем в не дискретном мире. Таким образом, все измерения должны быть представлены как вещественные числа. Целочисленные аппроксимации вещественных чисел иногда используются при неудачной попытке сэкономить место. Не делайте этого: количественные эффекты округления или усечения создают артефакты.

В одном особенно неуклюжем наборе данных, с которым мы столкнулись, вес ребенка был представлен в виде двух целочисленных полей (фунты и оставшиеся унции). Гораздо лучше было бы объединить их в одно десятичное поле.

Унификация имен

Интеграция записей из двух разных наборов данных требует, чтобы они имели общее ключевое поле. В качестве ключевых полей зачастую используются имена, но их нередко вносят неоднозначно. *José*, это тот же парень, что и *Jose* (Хосе)? Такие диакритические знаки запрещены в официальных записях о рождении в нескольких штатах США в отчаянной попытке обеспечить их однозначность.

Вот другой пример. Базы данных представляют меня как декартово произведение первого (*Steve*, *Steven* или *S.*), второго (*Sol*, *S.* или ничего) имени и фамилии (*Skiena*), что дает девять разных вариаций. Все ухудшается еще больше, если есть опечатки. Я могу найти себя в Google под именем *Stephen* и с фамилиями *Skienna* и *Skeina*.

Объединение записей по ключу — это очень сложная проблема, для которой нет волшебной палочки. Именно поэтому были изобретены идентификационные номера, по возможности используйте именно их в качестве ключей.

Наилучшим методом обычно является унификация: выполнение простых преобразований текста, чтобы свести каждое имя к одной канонической версии. Преобразование всех строк в нижний регистр увеличивает количество (обычно правильных) коллизий. Исключение вторых имен или по крайней мере их сокращение до аббревиатур создает еще больше совпадений/коллизий имен, как и сопоставление имен с каноническими версиями (например, превращение всех *Steve* в *Steven*).

Любая такая трансформация рискует создать людей-Франкенштейнов, отдельные записи, собранные из нескольких тел. Применимость подхода зависит от того, насколько велика опасность слишком агрессивного или слишком робкого слияния. Выясните, где именно находится ваша задача в этом диапазоне, и действуйте соответственно.

Важнейшей проблемой при объединении наборов данных является унификация *кодов символов* (character code). Символам в текстовых строках присваиваются числовые представления с сопоставлениями между символом и числом в соответствии со стандартом кода символов. К сожалению, существует несколько различных стандартов кодировки символов, а значит, то, что вы чистите с веб-страницы, может не совпадать с кодом символа в системе, которая предполагается для его обработки.

Исторически старый добрый стандарт 7-битового кода *ASCII* был расширен до 8-битового кода *IS0 8859-1 Latin*, в который добавлены символы и знаки препинания из нескольких европейских языков. Кодировка *UTF-8* — это все символы Unicode, использующие различные номера 8-битных блоков, что обратно совместимо с кодировкой ASCII. Это доминирующая кодировка для веб-страниц, хотя и другие системы остаются в использовании.

Корректная унификация кодов символов после слияния практически невозможна. У вас должна быть дисциплина, чтобы выбрать один код в качестве стандарта и проверять кодировку каждого входного файла при предварительной обработке, преобразовывая его в выбранный стандарт перед дальнейшей работой.

Унификация дат и времени

Метки даты и времени используются для определения относительного порядка событий и их группировки по относительной одновременности. Интеграция данных о событиях из нескольких источников требует тщательной очистки, чтобы обеспечить значимые результаты.

Сначала давайте рассмотрим вопросы измерения времени. Часы на двух компьютерах никогда не совпадают, поэтому точное выравнивание журналов из разных систем требует сочетания работы и догадок. Существуют также проблемы с часовыми поясами при работе с данными из разных регионов, а также различия в местных правилах, регулирующих переход на летнее время.

Правильный ответ здесь — привести все измерения времени в соответствие с *универсальным координированным временем* (Coordinated Universal Time — UTC) — современным стандартом, использующим традиционное *время по Гринвичу* (Greenwich Mean Time — GMT). Подходящим стандартом времени является *UNIX time*, в котором указывается точное время события с точки зрения количества секунд, прошедших с 00:00:00 UTC в четверг, 1 января 1970 года.

Григорианский календарь распространен во всем технологическом мире, хотя в разных странах используются многие другие календарные системы. Для преобразования между календарными системами должны использоваться специальные алгоритмы, как описано в издании [31]. Большая проблема унификации дат связана с правильной интерпретацией часовых поясов и международной линией дат.

Унификация временны*х* рядов нередко осложняется характером бизнес-календаря. Финансовые рынки закрыты в выходные и праздничные дни, что вызывает вопросы толкования, когда вы соотносите, скажем, цены на акции с местной температурой. Каков подходящий момент в выходные дни для измерения температуры, чтобы он соответствовал другим дням недели? Чтобы справиться с подобными проблемами, такие языки, как Python предоставляют обширные библиотеки для работы с временны*ми* рядами финансовых данных. Аналогичные проблемы возникают с ежемесячными данными, поскольку месяцы (и даже годы) имеют разную длину.

Финансовая унификация

Деньги заставляют мир вращаться, поэтому так много проектов науки о данных связано с финансовыми временны*ми* рядами. Но деньги могут быть грязными, поэтому эти данные требуют очистки.

Одним из вопросов является *конвертация валют* (currency conversion), представляющая международные цены с использованием стандартизованной финансовой единицы. Курсы обмена валют могут варьироваться на несколько процентов в течение текущего дня, поэтому некоторые приложения требуют зависящих от времени преобразований. Коэффициенты конвертации на самом деле не стандартизованы. На разных рынках будут разные ставки и *наценки* (spread), разница между ценами покупки и продажи, которые покрывают стоимость конвертации.

Еще одна важная коррекция касается инфляции. *Изменение стоимости денег с течением времени* (time value of money) подразумевает, что доллар сегодня, как правило, более ценен, чем доллар через год, а процентные ставки обеспечивают правильный способ дисконтирования долларов в будущем. Уровень инфляции оценивается за счет отслеживания изменений цен в товарных корзинах и позволяет стандартизировать покупательную способность доллара с течением времени.

Использовать в модели некорректируемые цены на протяжении нетривиальных периодов времени — просто напрашиваться на неприятности. Однажды группа моих учеников была очень взволнована сильной корреляцией, которая наблюдалась между ценами на акции и ценами на нефть в течение тридцатилетнего периода, а потому пыталась использовать цены на акции в модели прогнозирования товара. Но оба товара были оценены в долларах без какой-либо корректировки с учетом инфляции. Временны*е* ряды цен практически *любых* пар товаров будут сильно коррелировать со временем, если не учитывать инфляцию.

На самом деле наиболее значимым способом представления изменений цен во времени, вероятно, являются не различия, а *доходы*, которые нормализуют разницу по начальной цене:

$$r_i = \frac{p_{i+1} - p_i}{p_i}.$$

Это больше похоже на процентное изменение с преимуществом, заключающимся в том, что логарифм этого отношения становится симметричным для прибыли и убытков.

Финансовые временны*е* ряды содержат много других тонкостей, которые требуют очистки. Многие акции дают акционерам *дивиденды*, запланированные на определенную дату каждый год. Скажем, например, что компания Microsoft выплатит дивиденды в размере 2,50 долл. 16 января. Если вы владеете акцией Microsoft в начале финансового дня, вы получаете этот чек в тот же день, поэтому стоимость этой акции сразу же падает на 2,50 долл., поскольку дивиденды уже получены. Это снижение цены не отражает реальных убытков для акционера, но должным образом очищенные данные должны учитывать дивиденды в цене акций. Легко представить себе модель, обученную работе на нескорректированных ценовых данных, которая научилась продавать акции непосредственно перед выплатой дивидендов и ошибочно чувствует себя гордой за это.

3.3.3. Как справиться с отсутствующими значениями

Не все наборы данных полны. Важным аспектом очистки данных является определение полей, для которых нет данных, а затем их корректная компенсация.

- Каков год смерти живого человека?
- Что делать с ответом на вопрос, оставшийся пустым или заполненным явно необычным значением?
- Какова относительная частота событий, слишком редких, чтобы их можно было увидеть в выборке ограниченного размера?

Числовые наборы данных подразумевают значения для каждого элемента в матрице. Обнуление недостающих значений заманчиво, но, как правило, ошибочно, поскольку всегда есть некоторая двусмысленность относительно того, следует интерпретировать эти значения как данные или нет. У кого-то зарплата нулевая потому, что он безработный, или он просто не ответил на вопрос?

Опасность использования бессмысленных значений в качестве символов, не относящихся к данным, заключается в том, что они могут быть неверно истолкованы как данные, когда придет время строить модели. Модель линейной регрессии, обученная прогнозировать зарплату по возрасту, образованию и полу, будет иметь проблемы с людьми, которые отказались отвечать на вопрос.

Использование такого значения, как –1, в качестве символа отсутствия данных имеет те же недостатки, что и нуль. Будьте как тот математик, который боится отрицательных чисел: не останавливайтесь ни перед чем, чтобы избежать их.

На заметку. Держите исходные необработанные данные отдельно от их очищенной версии. Необработанные данные — это основополагающая истина, и их следует сохранять без изменений для последующего анализа. Очищенные данные могут быть улучшены с использованием добавлений для заполнения пропущенных значений. Но держите необработанные данные отдельно от очищенных, чтобы исследовать различные подходы к предположениям.

Так как же нам поступить с пропущенными значениями? Самый простой подход — удалить все записи, содержащие пропущенные значения. Это работает просто отлично, когда остается достаточно данных, при условии, что отсутствующие значения отсутствуют по несистематическим причинам. Если люди, отказывающиеся указывать свою зарплату, имеют ее, как правило, выше среднего, то удаление этих записей приведет к необъективным результатам.

Но обычно мы хотим использовать записи с пропущенными полями. Лучше как-то оценивать или заполнять пропущенные значения, а не оставлять их пустыми. Нам нужны общие методы для заполнения пропущенных значений. Кандидаты таковы.

- *Эвристическая замена* (heuristic-based imputation). С учетом достаточного знания базовой области применения мы должны быть в состоянии сделать разумное предположение о значении определенных полей. Если мне нужно будет указать значение года, в который вы умрете, возьмите *год рождения, прибавьте 80* и окажетесь в среднем верны, что намного быстрее, чем ожидание окончательного ответа.

- *Замена средним значением* (mean value imputation). Использование среднего значения переменной в качестве заменителя для пропущенных значений обычно целесообразно. Во-первых, добавление большего числа значений со средним значением оставляет среднее значение неизменным, поэтому мы не искажаем нашу статистику такой заменой. Во-вторых, поля со средними значениями придают ванильный вкус большинству моделей, поэтому они оказывают приглушенное влияние на любой прогноз, сделанный с использованием данных.

  Но среднее значение может быть неуместным, если существует систематическая причина пропуска данных. Предположим, мы использовали средний год смерти из Википедии, чтобы подсчитать недостающее значение для всех живых людей. Это может привести к катастрофическим последствиям, поскольку многие люди будут зарегистрированы умершими еще до того, как они действительно родились.

- *Замена случайным значением* (random value imputation). Другой подход заключается в выборе случайного значения в столбце для замены отсутствующего значения. Казалось бы, это готовит нас к потенциально ошибочным догадкам, но в том-то и дело. Регулярный выбор случайных значений позволяет проводить статистическую оценку влияния замены. Если мы запустим модель десять раз с десятью разными значениями замены и получим широко варьирующиеся результаты, то мы, вероятно, не должны сильно доверять этой модели. Такая проверка точности особенно полезна, когда в наборе данных отсутствует значительная часть значений.

- *Замена ближайшим соседом* (imputation by nearest neighbor). Что, если мы найдем полную запись, наиболее точно соответствующую всем присутствующим полям данной, и используем найденного ближайшего соседа для замены отсутствующего значения? Такие прогнозы должны быть точней, чем при замене средним значением, когда есть систематические причины для объяснения расхождений между записями.

  Этот подход требует наличия *функции расстояния* (distance function) для определения наиболее похожих записей. Методы *ближайших соседей* (nearest neighbor) являются важной техникой в науке о данных и будут более подробно обсуждаться в разделе 10.2.

- *Замена интерполяцией* (imputation by interpolation). В более общем смысле, мы можем использовать такой метод, как линейная регрессия (см. раздел 9.1), для прогнозирования значений целевого столбца, учитывая другие поля в записи. Такие модели могут быть обучены на основе полных записей, а затем применены к тем, у которых отсутствуют значения.

Использование линейной регрессии для прогнозирования пропущенных значений работает лучше всего, когда в записи отсутствует только одно поле. Потенциальную опасность здесь создают значительные выбросы из-за плохих прогнозов. Модели регрессии могут легко превратить неполную запись в выброс, заполнив отсутствующие поля необычно высокими или низкими значениями. Это привело бы к тому, что последующий анализ сконцентрирует больше внимания на записях с пропущенными значениями, что в точности противоположно тому, что мы хотим сделать.

Такие проблемы подчеркивают важность обнаружения выбросов — это последний этап процесса очистки, который и будет рассмотрен здесь.

3.3.4. Обнаружение выброса

Ошибки при сборе данных легко могут привести к выбросам, которые способны помешать правильному анализу. Интересный пример относится к самому большому из когда-либо обнаруженных позвонков динозавров. Размером в 1500 миллиметров, он мог бы принадлежать существу длинной 188 футов (57,30 метров). Это удивительно, особенно потому, что *второй по величине* обнаруженный экземпляр давал всего 122 фута (37,19 метра).

Наиболее вероятное объяснение (см. [32]) заключается в том, что этой гигантской окаменелости на самом деле никогда не было: она отсутствовала в Американском музее естественной истории более ста лет. Возможно, первоначальное измерение было выполнено для кости обычного размера, но две цифры в центре случайно поменялись местами, что дает позвонок в 1050 миллиметров.

Выбросы элементов зачастую возникают из-за ошибок при вводе данных, как, очевидно, было и здесь. Они также могут возникать из-за ошибок при очистке, например, из-за неправильного форматирования, приводящего к тому, что номер сноски интерпретируется как числовое значение. То, что нечто написано, не делает это правдой. Как и в примере с динозавром, один выброс может привести к серьезным ошибочным интерпретациям.

Общая проверка работоспособности требует просмотра наибольших и наименьших значений в каждой переменной или столбце, чтобы убедиться, не находятся ли они слишком далеко от линии. Лучше всего для этого построить частотную гистограмму и определить местоположение крайних элементов. Визуальный осмотр также может подтвердить, что распределение выглядит так, как должно, обычно в форме колокола.

В нормально распределенных данных вероятность того, что значение равно k при стандартном отклонении от среднего, экспоненциально уменьшается с увеличением k. Это объясняет, почему нет 10-футовых (3,05 метра) баскетболистов, и обеспечивает надежный порог для выявления выбросов. Распределения

по степенному закону обнаружить труднее: Билл Гейтс действительно "стоит" в 10 000 раз больше, чем средний человек.

Очень просто взять и удалить строки, содержащие поля с выбросами и двигаться дальше. Выбросы зачастую указывают на систематические проблемы, с которыми нужно справляться. Рассмотрим набор данных исторических личностей по продолжительности жизни. Легко замерить библейского Мафусаила (969 лет), принять его за выброс и удалить.

Но лучше выяснить, не свидетельствует ли он о других цифрах, которые мы должны рассмотреть, чтобы удалить. Обратите внимание: у Мафусаила нет точных дат рождения и смерти. Возможно, опубликованный возраст любого человека без дат следует считать достаточно подозрительным, чтобы его можно было сократить. Напротив, человек с самой короткой продолжительностью жизни в Википедии (Иоанн I, король Франции) прожил всего пять дней. Но его рождение (15 ноября) и смерть (20 ноября) в 1316 году убеждают меня, что его продолжительность жизни точна.

3.4. Случай из жизни: игры на фондовом рынке

При каждой встрече мой аспирант Венбин говорил мне, что мы зарабатываем деньги. Но каждый раз он был все менее и менее оптимистичен.

Наша система анализа отношений Lydia получала массу текстов из новостей и социальных сетей, сводя их к ежедневным временным рядам по частоте упоминаний и отношению миллионов различных людей в пределах упомянутых мест и организаций. Когда некто выигрывает спортивный чемпионат, появляется много статей, описывающих событие и восхваляющих этого атлета. Но когда этого спортсмена арестовывают за наркотики, тон статей о нем немедленно меняется. Подсчитывая в текстовом потоке относительную частоту связанных с событием положительных слов ("победоносный") и отрицательных слов ("арестован"), мы можем построить *сигналы настроения* (sentiment signal) по любому объекту, достойному новостей.

Венбин изучал, как можно использовать сигналы настроения для прогнозирования таких будущих событий, как кассовый сбор для некого фильма, в ответ на качество опубликованных обзоров или слухов. Но особенно он хотел использовать эти данные для игры на фондовом рынке. Акции поднимаются и опускаются в зависимости от новостей. Отчет об упущенной прибыли — *плохая* новость для компании, поэтому цена понижается. Утверждение Управлением по санитарному надзору за качеством пищевых продуктов и медикаментов (FDA) нового препарата — *отличная* новость для компании, которой он принадлежит, поэтому его цена растет. Если бы Венбин мог использовать наш

сигнал настроения для прогнозирования будущих цен на акции... проще говоря, мне больше не пришлось бы платить ему как научному сотруднику.

Таким образом, он смоделировал стратегию покупки акций, которые показали самые высокие настроения в новостях этого дня, а также продажи акций с самыми низкими настроениями. Он получил отличные результаты. "Видите, — сказал он, — мы зарабатываем деньги".

Цифры выглядели великолепно, но у меня было одно возражение. Использование результатов сегодняшних новостей для прогнозирования текущих смещений цен было не совсем корректно, поскольку описанное в статье событие могло уже сместить цену, прежде чем мы успели прочесть об этом. Цены на акции способны очень быстро реагировать на важные новости.

Поэтому Венбин создал модель стратегии покупки акций, основанную на настроениях из новостей предыдущего дня, чтобы создать разрыв между наблюдаемыми новостями и изменениями цен. Доходность существенно снизилась, но все еще оставалась положительной. "Видите, — сказал он, — мы все еще зарабатываем деньги".

Но мне было все же немного не по себе. Многие экономисты считают, что финансовые рынки эффективны, а это значит, что все опубликованные новости мгновенно отражаются в изменении цен. Цены, безусловно, изменились в ответ на новости, но вы не сможете получить их достаточно быстро, чтобы использовать информацию. Мы должны были оставаться достаточно скептически настроенными, чтобы не было проблем с данными и временем, которыми мы могли бы объяснить наши результаты.

Поэтому я спросил у Венбина, как именно он выполнял симуляцию. Его стратегия покупки и продажи строилась по цене закрытия на каждый день. Но до открытия следующего дня оставалось шестнадцать часов, и у мира было достаточно времени, чтобы отреагировать на события, которые произошли, пока я спал. Он переключил свою модель покупки на цену открытия. Опять же, уровень доходности существенно снизился, но он все еще оставался положительным. "Видите, — сказал он, — мы все еще зарабатываем деньги".

Но могут ли быть и другие неточности в том, как мы рассчитывали наши данные, получая, по сути, завтрашнюю газету сегодня? Добросовестно ли мы исключили все другие возможности, о которых могли не подумать, например, отражали ли даты опубликованной статьи время их выпуска, а не момент написания. После того как мы стали крайне скептичны, его стратегии, казалось, все равно демонстрировали положительную отдачу от настроений в новостях.

Наша статья об этом анализе [10] была принята хорошо, и Венбин стал успешным специалистом по количественному анализу, использующим настроения наравне с другими сигналами для торговли на финансовых рынках. Но я немного обеспокоен этим результатом. Очистку наших данных для точной

отметки времени каждой новостной статьи было очень трудно выполнить правильно. Изначально наша система была разработана для создания ежедневных временных рядов в пакетном режиме, поэтому трудно быть уверенным в том, что мы сделали все правильно в миллионах статей, загруженных за несколько лет, чтобы теперь выполнять более точный анализ.

Извлеченный урок заключается в том, что, когда речь идет о деньгах, чистота важна. Кроме того, лучше спроектировать чистую среду в начале анализа, чем неистово мыть все в конце.

3.5. Краудсорсинг

Ни у одного человека нет ответов на все вопросы. Даже у меня. Некоторыми из плодов мудрости является то, как мы получаем знания, а также то, как мы собираем мнения из знаний и опыта других.

Краудсорсинг (crowdsourcing) использует знания и труд большого количества людей для достижения общей цели. Он использует *мудрость толпы* (wisdom of crowds), ведь коллективное знание группы людей могло бы быть лучше, чем знание самого умного человека среди них.

Понимание этого началось с быка. Фрэнсис Гальтон (Francis Galton), основатель статистической науки и родственник Чарльза Дарвина, посетил местную ярмарку домашнего скота в 1906 году. На празднике сельским жителям было предложено угадать вес некоего быка, причем человек, чье предположение оказалось наиболее близким к истине, получал приз. На это пошли почти 800 участников. Никто не назвал фактический вес 1178 фунтов (534,33 кг), но Гальтон заметил, что среднее предположение было удивительно близко: 1179 фунтов (534,79 кг)! Эксперимент Гальтона предполагает, что для определенных задач можно получить лучшие результаты, задействовав группу разных людей, а не просто опрашивая экспертов.

Краудсорсинг служит важным источником данных при построении моделей, особенно для задач, связанных с человеческим восприятием. Люди остаются самой современной системой обработки естественного языка и компьютерного зрения, достигая высочайшего уровня производительности. Лучший способ сбора данных об обучении зачастую требует, чтобы люди набрали определенный текст или ввели изображение. Реализация этого в достаточно большом масштабе, чтобы собрать существенные данные для обучения, обычно требует большого количества корреспондентов, даже толпы.

Социальные сети и другие новые технологии облегчают сбор и обобщение мнений в массовом масштабе. Но как мы можем отделить мудрость толпы от криков толпы?

3.5.1. Демонстрация пенсов

Давайте начнем с небольшого эксперимента с мудростью толпы. На рис. 3.4 представлены фотографии банки с монетами, которые я накопил в своем офисе за многие годы. Сколько *пенсов* (pennies)[3] у меня в этой банке? Сделайте свое собственное предположение сейчас, поскольку я собираюсь сказать вам ответ на следующей странице.

Чтобы получить правильный ответ, мне пришлось попросить своего сотрудника Джастина Гардена (Justin Garden) взвесить пенсы на точных лабораторных весах. Деление на вес одного пенса дает их сумму. На рис. 3.4 (справа) видно, что Джастин усердно выполняет свою задачу.

Рис. 3.4. Угадайте, сколько пенсов у меня в этой банке? (слева) Правильный ответ был определен с использованием точных научных методов (справа)

Поэтому я снова спрашиваю: сколько у меня пенсов в этой банке? Я проводил этот эксперимент на студентах в моей группе науки о данных. Каков будет ваш ответ по сравнению с их?

Сначала я попросил одиннадцать своих студентов написать свое мнение на карточках и тихо передать их мне. Таким образом, эти догадки были полностью независимы друг от друга. Результаты были отсортированы для удобства:

537, 556, 600, 636, 1200, 1250, 2350, 3000, 5000, 11 000, 15 000.

Затем я написал эти цифры на доске и вычислил некоторые статистические данные. Медианой этих предположений было 1250, а средним 3739. На самом деле в банке было ровно 1879 пенсов. Среднее значение у моих студентов было ближе к правильному значению, чем любое предположение.

[3] Судя по фото, имелись в виду центы США. — *Примеч. ред.*

Но перед тем, как раскрыть реальную сумму, я попросил угадать его еще дюжину студентов. Единственное отличие заключалось в том, что эта группа видела догадки первого набора студентов, написанные на доске. Их выбор был таков:

750, 750, 1000, 1000, 1000, 1250, 1400, 1770, 1800, 3500, 4000, 5000.

Разглашение группе догадок других людей существенно повлияло на распределение, устраняя все выбросы: минимум во второй группе стал больше, чем четыре предыдущие догадки, а максимум стал меньше или равен трем из предыдущего раунда. В этой группе медиана составляла 1325, а среднее — 1935. Оба они оказались несколько ближе к фактическому ответу, но вполне понятно, что для этого понадобилось групповое мышление.

Привязка (anchoring) — это общеизвестная когнитивная тенденция: суждения людей иррационально фиксируются на первом услышанном числе. Продавцы автомобилей используют его регулярно, первоначально называя завышенную цену на автомобиль, чтобы последующие цены походили на скидку.

Наконец, я сделал последний тест, прежде чем раскрыть ответ. Я позволил моим студентам делать ставки на банку, а это значит, что они должны были быть достаточно уверенными, чтобы рисковать деньгами. Это дало ровно две заявки от храбрых студентов, по 1500 и 2000 пенсов соответственно. Я принял от простаков ставки максимум 1,21 долл., но оба оказались довольно близки. Это не удивительно: люди, желающие поставить свои деньги на событие, по определению уверены в своем выборе.

3.5.2. Когда толпа проявляет мудрость?

Согласно книге *The Wisdom of Crowds* [33] Джеймса Шуровьески (James Surowiecki), толпа проявляет мудрость, когда выполняются четыре условия.

- *Когда мнения независимы.* Наш эксперимент показал, как легко группе перейти в групповое мышление. На людей, естественно, влияют другие. Если вы хотите чье-то истинное мнение, вы должны спросить его лично.
- *Когда толпа состоит из людей с разными знаниями и подходами.* Толпа создает информацию только тогда, когда есть разногласия. Комитет, состоящий из совершенно единодушных экспертов, не даст ничего большего, чем вы могли бы получить у любого из них. В задаче угадывания количества пенсов некоторые люди оценивали объем контейнера, в то время как другие измеряли провисание моей руки, когда я поднимал тяжелую массу. К альтернативным подходам мог бы относиться подсчет количества пенсов, которое я мог бы накопить за двадцать лет случайного опустошения своих карманов, или их собственный опыт накопления.

- *Когда проблема лежит в области, не требующей специальных знаний.* Я доверяю консенсусу толпы в определенных важных решениях, например, какой тип машины купить или кто должен быть президентом моей страны. Но когда дело доходит до решения, является ли мой образец опухоли злокачественным или доброкачественным, я доверяю словам одного доктора, выбранного из 1000 имен, взятых случайным образом из телефонной книги.

  Почему? Потому что этот вопрос требует весьма специализированных знаний и опыта. Есть подлинная причина, почему врач *должен* знать куда больше, чем все остальные. Для простых задач толпа годится, но нужно быть осторожным, чтобы не спрашивать у толпы то, чего она не может знать.

- *Мнения могут быть достаточно едины.* Наименее полезной частью любой формы массового опроса является открытое поле "Скажите нам, что вы думаете"! Проблема здесь в том, что невозможно объединить эти мнения для формирования консенсуса, потому что у разных людей разные задачи и проблемы. Возможно, эти тексты можно было бы классифицировать по сходству, но это трудно сделать эффективно.

  Использование ответов на такие свободные вопросы обычно анекдотическое. Люди выбирают наиболее интересные из них, а затем используют их, чтобы произвести впечатление на босса.

На заметку. Будьте уникальным элементом в повседневном порядке жизни. Разнообразное и независимое мышление вносит самый большой вклад в толпу.

3.5.3. Механизмы объединения

Извлечение смысла из набора ответов требует использования подходящего механизма объединения. Для оценки числовых величин подходят стандартные методы, такие как построение графика плотности распределения и вычисление сводной статистики. Как среднее значение, так и медиана неявно предполагают, что ошибки распределены симметрично. Быстрого взгляда на форму распределения может быть вполне достаточно, чтобы подтвердить или отклонить эту гипотезу.

Медиана обычно является более подходящим выбором, чем среднее значение в решении проблем объединения. Это снижает влияние выбросов, что является особенно серьезной проблемой в случае массовых экспериментов, где определенная часть ваших участников, вероятно, окажется идиотами. По нашим

данным об угадывании количества пенсов, среднее показало страшно завышенную оценку 3739, которая уменьшилась до 2843 после удаления наибольшего и наименьшего предположений, а затем снизилась до 2005 после удаления двух выбросов на каждом конце (напомним, правильный ответ был 1879).

Удаление выбросов — это очень хорошая стратегия, но у нас могут быть и другие основания судить о надежности наших субъектов, например их результаты в других тестах, где мы знаем ответ. Получив *средневзвешенное значение* (weighted average), где мы придаем больший вес тем оценкам, которые считаются более достоверными, можно учесть также степень доверия.

Для задач классификации голосование является основным механизмом объединения. *Теорема Кондорсе о присяжных* (Condorcet jury theorem) оправдывает нашу веру в демократию. В ней говорится, что если вероятность того, что каждый избиратель прав по данному вопросу, составляет $p > 0{,}5$, то вероятность того, что большинство избирателей правы ($P(n)$), больше, чем p. Фактически это точно:

$$P(n) = \sum_{i=(n+1)/2}^{n} \binom{n}{i} p^i (1-p)^{n-i}.$$

Большее количество избирателей обеспечивает статистическую достоверность даже для очень спорных выборов. Предположим, что $p = 0{,}51$, т.е. правые силы имеют большинство. Жюри из 101 члена будет принимать правильное решение в 57% случаев, в то время как $P(1001) = 0{,}73$ и $P(10001) = 0{,}9999$. Вероятность правильного решения приближается к 1 при $n \to \infty$.

Однако у избирательных систем существуют естественные пределы. *Теорема Эрроу* (Arrow's impossibility theorem) гласит, что ни одна избирательная система для суммирования сочетаний предпочтений при голосовании не удовлетворяет четырем естественным условиям честности выборов. Она будет подробней обсуждаться в разделе 4.6 в контексте оценок и рейтингов.

3.5.4. Службы краудсорсинга

Краудсорсинговые службы, такие как Amazon Turk и CrowdFlower, позволяют вам нанять большое количество людей для выполнения небольших сдельных работ. Они помогают вам нанять людей, чтобы создать данные за вас.

Эти краудсорсинговые службы предоставляют большой контингент внештатных работников, выступая в качестве посредника между ними и потенциальными работодателями. Эти работники, обычно называемые *туркáми* (Turkers), получают списки доступных работ и их оплату, как показано на рис. 3.5. Работодатели, как правило, обладают некоторой возможностью контроля местонахождения и квалификации тех, кого они нанимают, и могут от-

казаться от результатов усилий работника без оплаты, если они сочтут их недостаточными. В то же время публикуются статистические данные об уровне отказов работодателей, и хорошие работники вряд ли согласятся работать на плохих нанимателей.

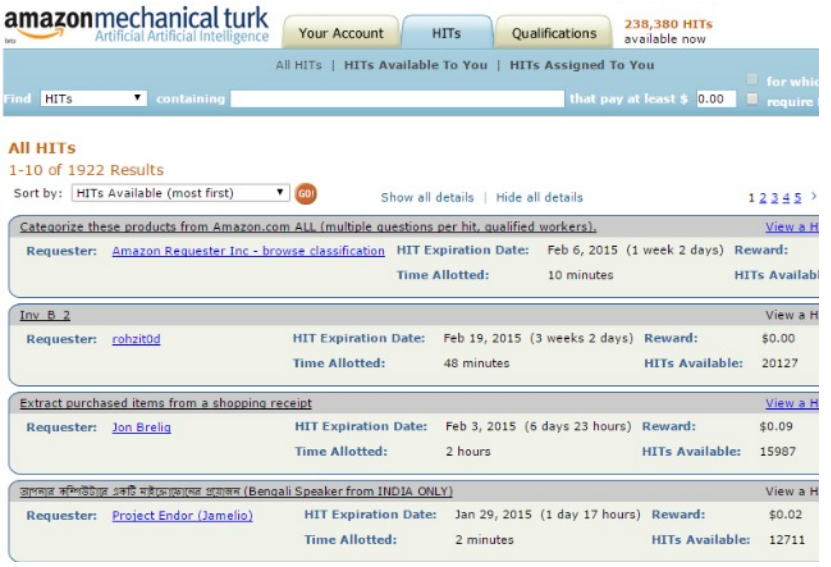


Рис. 3.5. Предлагаемые задачи на Mechanical Turk

Возлагаемые на турок задачи обычно подразумевают простые когнитивные усилия, которые в настоящее время не могут быть хорошо выполнены компьютерами. Полезные случаи применения турок таковы.

- *Измерение аспектов человеческого восприятия*. Краудсорсинговые системы обеспечивают эффективные способы сбора репрезентативных мнений по простым вопросам. Одним из хороших случаев применения было установление взаимосвязи между цветами в красно-зелено-голубой области и именами, по которым люди обычно идентифицируют их на языке. Это важно знать при создании описаний продуктов и изображений. Так где же граница цветового пространства между “синим” и “голубым” или “цветом яйца малиновки” и “цветом морской волны”? Правильные названия являются функцией культуры и условностей, а не физики. Чтобы выяснить это, вы должны опросить людей, а краудсорсинг позволяет легко опрашивать сотни или тысячи разных людей.
- *Получение учебных данных для классификаторов машинного обучения*. Наш основной интерес в краудсорсинге будет заключаться в том, чтобы создавать человеческие комментарии, которые послужат данными для обучения. Многие задачи машинного обучения заключаются в стремлении

выполнить определенную задачу "так же, как люди". Для этого требуется большое количество обучающих примеров, чтобы установить, как поступают люди, когда им предоставляется некая возможность.

Предположим, например, что мы стремимся создать систему анализа настроений, способную прочитать письменный отзыв и решить, является ли его мнение о некоем продукте благоприятным или неблагоприятным. Нам понадобится большое количество обзоров, помеченных комментаторами, чтобы они послужили данными для обучения и проверки. Кроме того, нам нужны одни и те же обзоры, которые многократно комментируются разными комментаторами, чтобы выявить разногласия между ними касательно точного смысла текста.

- *Получение оценочных данных для компьютерных систем. A/B-тест* (*A/B testing*) является стандартным методом оптимизации пользовательских интерфейсов: предоставьте половине жюри версию *A* данной системы, а другой половине версию *B*. Затем по некоему показателю проверьте, какая группа работала лучше. Турки могут дать отзыв о том, насколько интересным является данное приложение или насколько хорошо работает новый классификатор.

  Один из моих аспирантов (Яньцин Чен) использовал службу CrowdFlower для оценки созданной им системы определения наиболее релевантной категории Википедии для конкретной сущности. Какая категория лучше описывает Барака Обаму: *Президенты США* или *Афроамериканские авторы*? За 200 долл. он заставил людей ответить на 10 000 таких вопросов с несколькими вариантами ответов, и этого было достаточно, чтобы он мог правильно оценить свою систему.

- *Ввод людей в машину.* До сих пор существует множество интересных задач, которые люди выполняют намного лучше, чем машины. Продуманный интерфейс может предоставлять пользовательские запросы людям, сидящим внутри компьютера (people sitting inside the computer) и ожидающим тех, кто в них нуждается.

  Предположим, вы хотели создать приложение, которое поможет людям с нарушениями зрения, позволяя пользователю сделать снимок и попросить кого-нибудь о помощи. Возможно, они находятся у себя на кухне, и им нужен кто-то, кто прочтет этикетку на банке. Это приложение может вызвать Turker в качестве подпрограммы, чтобы решить такую задачу, если это необходимо.

  Конечно, эти пары "изображение-комментарий" следует сохранить для будущего анализа. Они могут служить учебными данными для программы машинного обучения, чтобы по возможности вывести людей из этого цикла.

- *Независимые творческие усилия.* Краудсорсинг может быть использован для инициации большого количества творческих работ по требованию. Вы можете заказать сообщения в блоге или статьи по требованию или письменные обзоры продукта (как хорошие, так и плохие). Все, что вы можете себе представить, может быть создано, если вы просто укажете, чего вы хотите.

  Вот два примера глупых проектов, которые меня очень забавляют.

  - The Sheep Market (овечий рынок) (`http://www.thesheepmarket.com`) заказал 10 000 рисунков овец за пенс каждый. Он пытается продать их по высокой цене как концептуальное произведение искусства. Каких творческих усилий вы можете ожидать от людей, работающих на вас по цене 0,25 долл. за рисунок?
  - Emoji Dick (`http://www.emojidick.com`) сделал попытку, используя краудсорсинг, полностью перевести великий американский роман *Моби Дик* в образы эмодзи (смайлики). Его создатели разделили книгу примерно на 10 000 частей и раздали каждую часть на перевод трем разным туркам. Другие турки были наняты для выбора лучшего из вариантов для включения в финальную книгу. В проекте приняли участие более 800 турок, а общая его стоимость составила 3676 долл. (наем проходил на сайте Kickstarter).
- *Экономические и психологические эксперименты.* Краудсорсинг оказался благом для социологов, проводящих эксперименты по поведенческой экономике и психологии. Вместо того чтобы нанимать местных студентов для участия в их исследованиях, эти исследователи теперь могут распространить свой трудовой резерв на весь мир. Они получают возможность использовать большие группы населения, проводить независимую репликацию в разных странах и, таким образом, проверять, существуют ли культурные предубеждения в их гипотезах.

Существует множество увлекательных задач, которые можно весьма недорого выполнить с помощью краудсорсинга. Тем не менее вы обречены на разочарование, если нанимаете турок для выполнения неправильной задачи неправильным образом. К плохим случаям использования краудсорсинга относятся следующие.

- *Любое задание, требующее повышенной квалификации.* Хотя каждый человек обладает уникальными навыками и опытом, краудсорсинговые работники не проходят специальной подготовки. Они предназначены для использования в качестве сменных частей (interchangeable part). Вы не устанавливаете личные отношения с этими работниками, и любое

реальное общение будет слишком коротким, чтобы провести тренинг продолжительностью более нескольких минут.

Задачи, требующие определенных технических навыков, не являются подходящими для краудсорсинга. Тем не менее они могут быть выполнены субподрядчиками в традиционных долгосрочных соглашениях.

- *Любое задание, которое вы не можете сформулировать четко.* У вас нет механизма обратной связи с турками. По правде говоря, они не могут задавать вам вопросы. Таким образом, система работает только в том случае, если вы можете кратко, четко и недвусмысленно поставить свои задачи.

  Это намного сложнее, чем кажется. Поймите, что вы пытаетесь программировать не компьютеры, а людей со всеми присущими им ошибками, связанными с идеей "делай, как я говорю", а не "делай, что я имею в виду". Проверьте свои спецификации на ближайшем окружении, прежде чем открывать свою задачу для широких масс, а затем проведите небольшой тестовый прогон на своей краудсорсинговой платформе, чтобы оценить, как оно пойдет, прежде чем растратить большую часть своего бюджета. Возможно, вас ждут сюрпризы. Вещи, которые кажутся вам очевидными, могут означать нечто совершенно иное для работника на другой стороне земного шара.

- *Любая задача, где вы не можете проверить, хорошо ли она выполнена.* У турок есть одна мотивация, когда они берут на себя сдельную работу: они стараются максимально эффективно конвертировать свое время в деньги. Они ищут рабочие места, предлагающие лучшую цену за свой труд, а самые умные будут стремиться выполнить вашу задачу как можно быстрее и бездумно.

  Краудсорсинговые платформы позволяют работодателям удерживать оплату, если работа по контракту выполнена неприемлемо. Чтобы воспользоваться этим, необходим эффективный способ проверки качества продукта. Возможно, вам следует попросить их выполнить определенные задания, если вы уже знаете правильный ответ. Возможно, вы можете сравнить их ответы с ответами других независимых работников и отказаться от их работы, если она слишком часто не согласуется с консенсусом.

  Очень важно использовать какой-то механизм контроля качества. Некоторая часть доступных работников на любой платформе — это боты, которые ищут задачи с множественным выбором для выполнения случайным образом. Другие могут быть людьми с языковыми навыками,

совершенно не подходящими для данной задачи. Вы должны проверить и отказаться от них, чтобы не быть одураченным.

Тем не менее вы не можете обоснованно жаловаться на результаты плохо определенных задач. Отказ от слишком высокой доли работы ухудшит вашу репутацию среди работников на платформе. Особенно плохая идея — отказываться платить людям, но все равно использовать их рабочий продукт.

- *Любая нелегальная задача или слишком бесчеловечная, чтобы предлагать людям.* Вам не разрешается просить турка сделать что-то незаконное или неэтичное. Классический пример — нанимать кого-то, чтобы писать плохие отзывы о продуктах вашего конкурента. Наем киллера делает тебя таким же виновным в убийстве, как и тот, кто выстрелил. Имейте в виду, что существуют электронные следы, по которым можно сразу найти размещающего подобные заказы.

  Люди в образовательных и исследовательских учреждениях имеют более высокий стандарт, чем закон, поддерживаемый *институциональным наблюдательным советом* (Institutional Review Board — IRB). IRB — это комитет исследователей и административных чиновников, которые должны одобрить любое исследование на людях, прежде чем оно будет предпринято. Добровольные краудсорсинговые исследования, такие как те, которые мы обсуждали, регулярно утверждаются после того, как исследователи прошли короткий онлайн-курс обучения, чтобы убедиться, что они понимают правила.

  Всегда помните, что на другом конце машины находится человек. Не задавайте ему задачи, которые являются оскорбительными, унизительными, нарушающими конфиденциальность или слишком стрессовыми. Вы, вероятно, получите лучшие результаты от своих работников, если будете относиться к ним как к людям.

Чтобы заставить людей выполнять ваши задания, необходимы надлежащие стимулы, а не только четкие инструкции. В жизни вы обычно получаете то, за что платите. Помните о преобладающей в настоящее время минимальной почасовой оплате труда в вашей стране и соответственно оценивайте свои задачи. Это не юридическое требование, но в целом это хороший бизнес.

Зловещее свечение, которое исходит от найма работников по цене 0,50 долл. в час, быстро исчезает, как только вы видите низкое качество работников, которых привлекают ваши задачи. Вы можете легко растратить все свои сбережения, если будете тщательно корректировать свой рабочий продукт, возможно, заплатив нескольким работникам за это. Задачи с более высокой оплатой труда

находят работников намного быстрее, поэтому будьте готовы подождать, если вы не установите достаточный тариф. Боты и их функциональные эквиваленты с куда большей радостью принимают нищенскую заработную плату, чем работники, которых вы действительно хотите нанять.

3.5.5. Игрофикация

Существует альтернатива плате людям за комментирование или расшифровку ваших данных. Вместо этого все можно сделать так весело, чтобы люди работали на вас бесплатно!

Игры со смыслом (Games With A Purpose — GWAP) — это системы, которые маскируют сбор данных под игру, в которую люди хотят играть, или под задачу, которую люди сами захотят решить. При правильном сочетании игры, мотивов и воображения можно делать удивительные вещи. Успешные примеры таковы.

- *CAPTCHA для оптического распознавания символов (OCR).* CAPTCHA — это искаженные текстовые изображения, с которыми вы нередко сталкиваетесь при создании учетной записи в Интернете. Они требуют, чтобы вы вводили содержимое текстовых строк, показанных на изображении, чтобы доказать, что вы человек. Это позволяет запретить доступ ботам и другим программным системам.

  Системы ReCAPTCHA были разработаны для получения полезных данных из более чем 100 миллионов CAPTCHA, отображаемых каждый день. В каждой отображаются две текстовые строки, одну из которых система проверяет, чтобы разрешить запись. Другая представляет трудный случай для системы OCR, которая оцифровывает старые книги и газеты. Ответы возвращаются назад, чтобы улучшить оцифровку архивных документов, расшифровывая более 40 миллионов слов в день.

- *Психологическое/IQ тестирование в играх и приложениях.* Психологи установили пять основных качеств личности как важные и воспроизводимые аспекты личности. Академические психологи используют личностные тесты с множественным выбором, чтобы измерить местоположение людей на шкале личностей для каждой из пяти основных черт: открытость, добросовестность, экстраверсия, доброжелательность и эмоциональность.

  Превратив эти опросы в игровые приложения ("Каковы черты *вашей* личности?"), психологи собрали данные о личностях более 75 000 различных людей, а также другие данные о предпочтениях и поведении. Это создало огромный набор данных для изучения многих интересных вопросов в психологии личности.

- *Игра FoldIt для предсказания белковых структур.* Прогнозирование структур, образованных белковыми молекулами, является одной из важнейших вычислительных задач в науке. Несмотря на многолетнюю работу, то, что заставляет белок складываться в определенную форму, до сих пор не совсем понятно.

  FoldIt (`https://fold.it`) — это игра, в которой людям, не являющимся биологами, предлагается создавать молекулы белка, которые складываются в определенную форму. Игроки оцениваются по тому, насколько их дизайн приближается к заданной цели, при этом игроки с наивысшим результатом представляются в таблице лидеров. На основании победных проектов было опубликовано несколько научных работ.

Ключом к успеху здесь является создание игры, в которую можно играть, чтобы стать популярным. Это намного сложнее, чем может показаться. В магазине приложений миллионы бесплатных приложений, в основном игры. Очень немногие когда-либо пробовало более нескольких сотен человек, что далеко не достаточно, чтобы быть интересным с точки зрения сбора данных. Добавление дополнительного ограничения на то, что игра генерирует интересные научные данные, делает задачу повышения ее популярности еще сложнее.

Для решения этой задачи следует использовать мотивационные методы. Ведение счета является важной частью любой игры, и игра должна быть спроектирована таким образом, чтобы результативность сначала быстро росла, чтобы заинтересовать игрока. Индикаторы прогресса дают стимул для достижения следующего уровня. Награждение значками и предоставление списков лидеров, видимых другими, поощряет бо́льшие усилия. Наполеон учредил множество лент и украшений для своих солдат, отметив, что "Просто удивительно, на что способны люди за полоску ткани".

Основной принцип разработки таких игр, как FoldIt, заключается в том, чтобы абстрагироваться от технической специфики области в пользу функции подсчета очков. Игра настроена таким образом, что игрокам не обязательно нужно было разбираться в вопросах молекулярной динамики, просто некоторые изменения приводят к повышению показателей, а другие — к снижению. Игрок будет вырабатывать собственную интуицию о предметной области во время игры, что приведет к разработке, которая может никогда не прийти в голову специалистам в данной области.

3.6. Дополнительная информация

Цитата Чарльза Бэббиджа в начале этой главы взята из его книги *Passages from the Life of a Philosopher* [34]. Я рекомендую роман в картинках Сидни Падуа

(Sydney Padua) [35] в качестве забавного, но содержательного (хотя и вымышленного) введения в его работу и отношения с Адой Лавлейс (Ada Lovelace).

Практическим вопросам обработки данных на определенных языках программирования посвящено множество книг. Особенно полезны книги издательства O'Reilly по науке о данных на языке Python, в том числе [36, 37].

История нашей системы ставок в игре хай-алай, в том числе роль очистки веб-сайта описана в моей книге *Calculated Bets* [8]. Это краткий и увлекательный обзор того, как строить имитационные модели для прогнозирования, и он будет предметом раздела "Случай из жизни" в разделе 7.8.

Провал космических полетов из-за ошибок в компьютерных вычислениях хорошо описан в популярных СМИ, см. обсуждения космических миссий Ariane 5 и Mars Climate Orbiter в публикациях Глейка [38], а также Стивенсона и др. [39] соответственно.

Гениальная идея использования акселерометров в сотовых телефонах для обнаружения землетрясений возникла у Фолкнера и др. [40]. Репрезентативные исследования больших наборов изображений сайта Flickr осуществлены Кисилевичем и др. [41].

Киттур [42] сообщает об опыте краудсорсинговых исследований с использованием Amazon Turk. Наше использование сайта CrowdFlower для идентификации соответствующих описаний исторических фигур было представлено в публикации [43]. Методы игрофикации и инструкции обсуждаются в публикациях [44, 45]. Системы ReCAPTCHA представлены в публикации Фон Анна и др. [46]. Масштабный сбор данных о психологических чертах с помощью мобильных приложений принадлежит Коссинскому и др. [47].

3.7. Упражнения

Манипуляция данными

3.1. *[3]* Уделите два часа знакомству с одним из следующих языков программирования: Python, R, MatLab, Wolfram Alpha/Language. Затем напишите краткую статью с вашими впечатлениями о его характеристиках:

- выразительность;
- скорость выполнения;
- разнообразие библиотечных функций;
- среда программирования;
- пригодность для алгоритмически интенсивных задач;
- пригодность для общих задач сбора данных.

3.2. *[5]* Выберите два основных языка программирования для обработки данных и напишите программы для решения следующих задач на обоих из них. Какой язык вы нашли наиболее подходящим для каждой задачи?
(a) Hello World!
(b) Чтение чисел из файла и их вывод в отсортированном порядке.
(c) Чтение текстового файла и подсчет общего количества слов.
(d) Чтение текстового файла и подсчет общего количества *отдельных* слов.
(e) Чтение чисел из файла и построение их частотной гистограммы.
(f) Загрузка страницы из веба и ее очистка.

3.3. *[3]* Поиграйте немного с Python, R и Matlab. Что вам понравилось больше всего? Каковы преимущества и недостатки каждого?

3.4. *[5]* Постройте набор данных из n замеров человеческого роста, причем p% из них будут записаны в английских футах, а остальные в метрических единицах. Используйте статистические тесты, чтобы проверить, отличается ли это распределение от того, которое правильно записано в метрах. Каков предел в зависимости от n и p, когда становится ясно, в чем проблема?

Источники данных

3.5. *[3]* Найдите таблицу цен за временное хранение. Проанализируйте эти данные и составьте прогноз стоимости/объема хранения данных через пять лет. Каковы будут цены на диски через 25 или 50 лет?

3.6. *[5]* Найдите подходящие источники данных для следующих конкурсов *The Quant Shop* и оцените их качество:
- *Miss Universe.*
- *Movie gross.*
- *Baby weight.*
- *Art auction price.*
- *Snow on Christmas.*
- *Super Bowl/college champion.*
- *Ghoul pool*?
- *Future gold/oil price*?

Очистка данных

3.7. *[3]* Узнайте, что было странного в сентябре 1752 года. Какие особые действия могли бы предпринять ученые того времени для нормализации годовой статистики в настоящее время?

3.8. *[3]* Выбросы каких типов можно ожидать в следующих наборах данных.
(a) Студенческие оценки.
(b) Данные о зарплате.
(c) Продолжительность жизни в Википедии.

3.9. *[3]* Датчик состояния здоровья выдает поток, состоящий из двадцати различных значений, включая артериальное давление, частоту сердечных сокращений и температуру тела. Опишите два метода или более, которые вы могли бы использовать для проверки правильности потока данных, поступающих с датчика.

Реализация проектов

3.10. *[5]* Реализуйте функцию, которая извлекает набор хештегов из фрейма данных публикаций Twitter. Хештеги начинаются с символа "#" и содержат любую комбинацию прописных и строчных букв, а также цифр. Необходимо учесть, что хештег может заканчивается там, где есть пробел или знак пунктуации, например запятая, точка с запятой или точка.

3.11. *[5]* Законы, регулирующие регистрацию избирателей, различаются в разных штатах США. Выберите один или несколько штатов с наиболее либеральными правилами и посмотрите, что вы должны сделать, чтобы получить данные. Подсказка: Флорида.

Краудсорсинг

3.12. *[5]* Опишите, как могли бы быть наняты краудсорсинговые работники, чтобы помочь собрать данные для конкурсов *The Quant Shop*.

- *Miss Universe*.
- *Movie gross*.
- *Baby weight*.
- *Art auction price*.
- *Snow on Christmas*.
- *Super Bowl/college champion*.
- *Ghoul pool*.
- *Future gold/oil price*?

3.13. *[3]* Предположим, вы платите туркам за то, что они читают и комментируют тексты, основываясь на простом мнении (положительном или отрицательном), которое передает каждый отрывок. Это задача для обсуждения, но как мы можем судить алгоритмически, ответил турок случайным образом или выполнил свою работу серьезно?

Вопросы на интервью

3.14. *[5]* Предположим, вы создали систему для прогнозирования цен на акции. Как бы вы оценили ее?

3.15. *[5]* Как бы вы оценили в целом выбросы, и что делать, если вы их нашли?

3.16. *[3]* Почему очистка данных играет важную роль в анализе?

3.17. *[5]* Как во время анализа вы обрабатываете пропущенные значения?

3.18. *[5]* Объясните пристрастие выбора. Почему это важно? Как процедуры управления данными, такие как обработка отсутствующих данных, могут ухудшить ситуацию?

3.19. *[3]* Как эффективно очищать данные из веба?

Конкурсы Kaggle

3.20. Облачно с прояснениями.
`https://www.kaggle.com/c/crowdflower-weather-twitter`

3.21. Спрогнозируйте возврат акций в конце дня, не будучи обманутым шумом.
`https://www.kaggle.com/c/the-winton-stock-market-challenge`

3.22. Очистка данных и анализ исторических изменений климата.
`https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data`.

Глава 4

Оценки и ранги

> Деньги — это табло со счетом, позволяющее оценить ваше место в игре против других людей.
>
> — Марк Кьюбан (Mark Cuban)

Функции оценки (scoring function) сводят многомерные записи к одному значению, выделяя некоторое специфическое свойство данных. Знакомым примером функции оценки является оценка знаний студентов на таких курсах, как мой. Студенты могут быть ранжированы (отсортированы) в соответствии с этими числовыми оценками, а затем в соответствии с этим порядком им присваиваются буквенные оценки.

Оценки, как правило, рассчитываются по числовым функциям, которые отражают успеваемость студентов, например баллы, начисляемые за каждую домашнюю работу или экзамен. Каждый студент получает единый комбинированный балл в диапазоне от 0 до 100. Эти баллы обычно следуют из линейной комбинации входных переменных, возможно, с 8%-ным коэффициентом для каждого из пяти домашних заданий и 20%-ным коэффициентом для каждого из трех экзаменов.

В отношении рубрик оценки следует обратить внимание на несколько моментов, которые мы будем использовать в качестве модели для более общих функций оценки и ранжирования.

- *Степень произвольности*. Каждый преподаватель/профессор использует разные соотношения между балами домашних заданий и экзаменов при оценке своих учеников. Некоторые ценят итоговый экзамен больше, чем все остальные переменные. Некоторые перед усреднением нормализуют каждое значение до 100, в то время как другие преобразуют каждый балл в *Z-оценку* (Z-score). Все они различаются по принципу, но каждый преподаватель/профессор уверен, что его система оценок лучшая.
- *Отсутствие данных для проверки*. Нет никакого золотого стандарта, предписывающего преподавателям "правильную" оценку, которую их студенты *должны были* получить на курсе. Студенты часто жалуются, что я должен дать им лучшую оценку, но за этими просьбами, похоже,

кроется скорее личная заинтересованность, чем объективность. На самом деле я редко слышу, чтобы студенты рекомендовали мне понизить их оценку. Без объективной обратной связи или стандартов, на которые можно было бы ориентироваться, у меня нет строгого способа оценить свою систему оценок и улучшить ее.

- *Общая корректность*. И все же, несмотря на использование совершенно несопоставимых и абсолютно не подтвержденных подходов, различные системы оценок обычно дают схожие результаты. В каждой школе есть группа круглых отличников (класса A), монополизирующих значительную часть высших оценок по каждому курсу. Этого не могло бы быть, если бы все эти разные системы оценок произвольно определяли успеваемость учащихся. Студенты класса C, как правило, путаются в средних и нижних слоях основной массы своих групп, вместо того, чтобы чередоваться с A и F на пути к своему окончательному среднему. Все системы оценок различны, но почти все оправданны.

В этой главе мы будем использовать функции оценки и ранжирования в качестве первого этапа анализа данных. Не все любят их так сильно, как я. Функции оценки часто кажутся произвольными и специальными, и в неумелых руках могут создавать впечатляюще выглядящие цифры, которые, по сути, совершенно бессмысленны. Поскольку их эффективность, как правило, не может быть подтверждена, эти методы не так научно обоснованы, как статистические методы и методы машинного обучения, которые мы представим в последующих главах.

Но я полагаю, что функции оценки следует знать, поскольку это удобные эвристические способы извлечения смысла из больших наборов данных. Функцию оценки иногда называют *статистической*, что придает ей больше достоинства и уважения. Мы представим несколько методов для получения осмысленных оценок из данных.

4.1. Индекс массы тела (BMI)

Все любят поесть, и наш современный мир изобилия предоставляет многочисленные возможности для этого. В результате значительный процент населения имеет массу тела, превышающую оптимальную. Но как вы можете определить, являетесь ли вы одним из них?

Индекс массы тела (Body Mass Index — BMI) — это оценка, или статистика, предназначенная для контроля вашего веса. Он определяется так.

$$BMI = \frac{Вес}{Рост^2}.$$

Здесь вес измеряется в килограммах, а рост в метрах.

Когда я пишу это, я имею рост 68 дюймов (1,73 метра) и чувствую себя немного упитанным при весе 150 фунтов (68,0 кг). Таким образом, мой BMI составляет 68,0/(1,73) = 22,8. Это не так страшно, поскольку общепринятые диапазоны BMI в США составляют:

Недостаточный вес: ниже 18,5.
Нормальный вес: от 18,5 до 25.
Избыточный вес: от 25 до 30.
Ожирение: более 30.

Таким образом, я считаю, что нахожусь в нормальном диапазоне, с запасом в еще дюжину фунтов, которые я могу набрать, прежде чем официально считать свой вес избыточным. На рис. 4.1 показаны репрезентативные группы американцев, находящихся в пространстве рост-вес согласно этой шкале. Каждая точка на этом графике — это человек, окрашенный в соответствии со своей весовой классификацией по BMI. Области, казалось бы, сплошного цвета, настолько плотны, что точки пересекаются. Внешние точки справа соответствуют самым толстым людям.

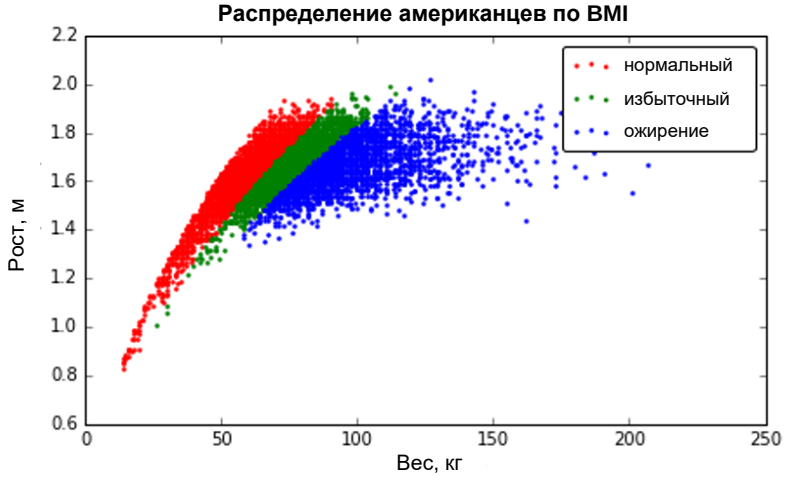


Рис. 4.1. Диаграмма разброса по росту и весу для 1000 репрезентативных групп американцев. Различные оттенки иллюстрируют метки классов в распределении BMI

Индекс BMI является примером очень успешной функции оценки/статистики. Он широко используется и общепринят, хотя некоторые в области общественного здравоохранения утверждают, что доступны более качественные статистические данные.

Логика BMI почти безупречна. Квадрат высоты должен быть пропорционален площади. Но масса должна расти пропорционально *объему*, а не площади, так почему же это не *вес*/*рост*^{3}? Исторически индекс BMI был разработан для корреляции с процентным содержанием жира в организме человека, что гораздо сложнее измерить, чем рост и вес. Эксперименты с несколькими простыми оценочными функциями, включая m/l и m/l^3, показали, что индекс BMI работает лучше всего.

Очень интересно посмотреть на распределения индекса BMI для экстремальных популяций. Рассмотрим профессиональных спортсменов по американскому футболу (NFL) и баскетболу (NBA).

- Баскетболисты, как известно, люди высокие. Они также должны много бегать по полю весь день, чтобы улучшить физическую форму.
- Игроки в американский футбол, общеизвестно, люди крупные. В частности, линейные игроки предназначены только для блокирования или перехвата других линейных игроков, для чего в первую очередь нужен объем.

Давайте посмотрим на некоторые данные. На рис. 4.2 показано распределение индекса BMI для баскетболистов и футболистов. Действительно, почти все баскетболисты имеют нормальный индекс BMI, несмотря на их очень ненормальный рост. А футболисты почти одинаковы, причем большинство из них считаются тучными, несмотря на то, что они также являются хорошо подготовленными спортсменами. Футболисты, как правило, оптимизированы по силе, а не по сердечно-сосудистой системе.

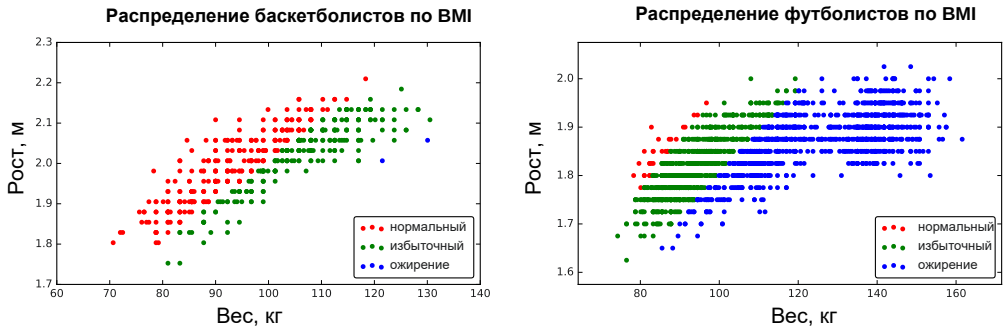


Рис. 4.2. Распределения индексов BMI профессиональных баскетболистов (слева) и футболистов (справа)

В главе 6 мы обсудим методы визуализации, позволяющие выделить представляемые данные, однако давайте начнем развивать нашу эстетику сейчас. Мы используем *точечные диаграммы*, чтобы показать каждого человека как точку в пространстве рост-вес, а метки (весовой класс или позиция игрока) показаны цветом.

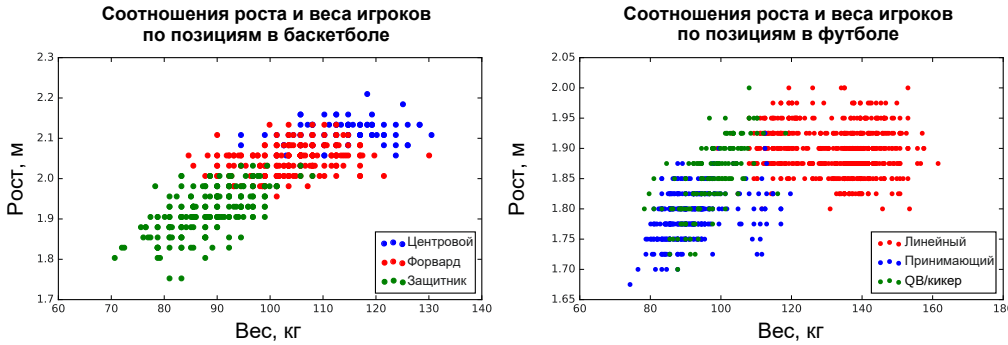


Рис. 4.3. Позиция в баскетболе (слева) и футболе (справа) во многом определяется размерами

Разделение индексов BMI по положению также показательно и приведено на рис. 4.3. В баскетболе защитники быстры и стройны, а играющие в центре — высоки и устрашающи. Таким образом, все эти позиции разделены точно по размеру. В футболе умелые игроки (квотербеки, кикеры и пантеры) оказываются значительно меньше говяжьих туш на линии.

4.2. Разработка систем оценки

Оценки (scores) — это функции, которые отображают характеристики каждого объекта на числовое значение качества. В этом разделе будут рассмотрены основные подходы к созданию эффективных систем оценки.

4.2.1. Золотые стандарты и прокси

Исторически бумажные деньги были обеспечены золотом, это означало, что один бумажный доллар всегда можно было обменять на золото ценой 1 доллар. Вот почему мы знали, что наши деньги стоят больше, чем бумага, на которой они напечатаны.

В науке о данных *золотой стандарт* (gold standard) — это набор меток или ответов, которые, как мы полагаем, верны. В исходной формулировке BMI золотым стандартом были проценты жира в организме, тщательно измеренные для небольшого числа субъектов. Конечно, такие измерения подвержены некоторой ошибке, но, определяя эти значения как золотой стандарт для физической формы, мы принимаем их как правильную меру. Мы верим в золото.

Наличие золотого стандарта обеспечивает строгий способ разработки хорошей системы оценок. Мы можем использовать метод аппроксимации кривой, такой как линейная регрессия (будет обсуждаться в разделе 9.1), для нахождения весовых коэффициентов входных характеристик, чтобы наилучшим образом аппроксимировать “правильные ответы” на экземпляры золотого стандарта.

Но настоящие золотые стандарты может быть трудно найти. *Прокси* (proxy) — это данные, которые легче найти и которые *должны* хорошо коррелировать с желаемой, но недостижимой абсолютной истиной. Индекс BMI был задуман так, чтобы быть показателем процента жира в организме. Его легко вычислить по росту и весу, и он хорошо коррелирует с телесным жиром. Это означает, что редко бывает необходимо проверить плавучесть в резервуарах с водой, мерить штангенциркулем или применять более интрузивные методы, которые непосредственно определяют количество жировых отложений.

Предположим, я хотел бы улучшить систему оценок, которую использую в следующем году для курса науки о данных. У меня есть данные студентов за прошлый год, т.е. их оценки по домашним заданиям и тестам, но у меня нет золотого стандарта на то, какие оценки эти студенты *заслуживают*. У меня есть только та оценка, которую я им дал, что абсолютно бессмысленно, если я пытаюсь улучшить систему.

Для неизвестного "реального" показателя знаний курса мне нужен прокси. Хорошим кандидатом на эту роль может быть совокупный средний балл каждого студента на *других* курсах. По правде говоря, успеваемость студентов должна быть примерно одинакова на всех курсах. Если моя система оценок ухудшает средний балл лучших студентов и повышает у не самых лучших, я, вероятно, делаю что-то не так.

Прокси особенно хороши при оценке систем оценки/ранжирования. В нашей книге *Who's Bigger?* [12] мы использовали Википедию для ранжирования исторических личностей по "значимости". У нас не было данных о значении золотого стандарта, определяющего, *насколько* важны эти люди. Для оценки своей объективности мы использовали несколько прокси.

- Цены, которые коллекционеры готовы платить за автографы знаменитостей, в целом *должны* соответствовать значению знаменитости. Чем выше цена, которую люди готовы заплатить, тем больше звезда.
- Статистика того, насколько хорош игрок в бейсбол, как правило, *должна* соответствовать значимости игрока. Чем лучше спортсмен, тем важнее он будет.
- В опубликованных рейтингах, появляющихся в книгах и журналах, перечислены главные президенты, кинозвезды, певцы, авторы и т.д. Президенты, получившие более высокие оценки историков, должны, как правило, оцениваться нами выше. Такие мнения в совокупности должны в целом соотноситься со значением этих исторических фигур.

Более подробно мы обсудим тему оценки значимости исторических деятелей в разделе 4.7.

4.2.2. Оценки или ранги

Ранжирование (ranking) — это перестановки, упорядочивающие n сущностей по достоинству, как правило, за счет сортировки выходных данных по какой-либо системе оценки. Популярными примерами рейтингов и рейтинговых систем являются следующие.

- *Двадцатка лучших футболистов/баскетболистов.* Пресс-агентства обычно оценивают лучшие спортивные команды колледжей, собирая голоса тренеров или спортивных журналистов. Как правило, каждый избиратель предоставляет свой личный рейтинг из двадцати лучших команд, и каждая команда получает тем больше очков, чем выше они расположены в списке избирателей. Суммирование баллов от каждого избирателя дает общий балл для каждой команды, а сортировка этих баллов определяет рейтинг.
- *Академические рейтинги университетов.* Журнал *U.S News and World Report* публикует ежегодные рейтинги лучших американских колледжей и университетов. Их методология является собственной и меняется каждый год, по-видимому, чтобы мотивировать людей покупать новые системы рейтингов. Но это, как правило, оценка, полученная из таких статистических данных, как соотношение преподавателей и студентов, коэффициент приема, стандартизированные результаты оценок студентов и абитуриентов и, возможно, результаты его футбольных/баскетбольных команд. Опросы академических экспертов также входят в смесь.
- *Google PageRank/результаты поиска.* Каждый запрос к поисковой системе запускает значительное количество вычислений, неявно ранжируя релевантность каждого документа в Интернете по отношению к запросу. Документы оцениваются на основе того, насколько хорошо они соответствуют тексту запроса, в сочетании с оценками качества, присущего каждой странице. Самым известным показателем качества страницы здесь является *PageRank*, централизованный сетевой алгоритм, который будет рассмотрен в разделе 10.4.
- *Рейтинг в классе.* Большинство средних школ ранжирует своих учащихся в соответствии с их оценками, при этом учащиеся с наивысшим рейтингом считаются отличниками (valedictorian). Функцией оценки, лежащей в основе этих рейтингов, обычно является *средний балл успеваемости* (Grade-Point Average — GPA), где вклад каждого курса взвешен по количеству заслуг, а каждая возможная буквенная оценка сопоставляется с числом (обычно $A = 4{,}0$). Но есть и варианты: многие школы предпочитают ценить более сложные курсы выше, чем легкие, такие как гимнастика, чтобы отразить большую сложность получения хороших оценок.

По правде говоря, сортировка результатов системы оценок дает числовое ранжирование. Но с другой стороны, позиция каждого элемента в рейтинге (скажем, 493 место из 2196) также дает числовую оценку этому элементу.

Поскольку оценки и рейтинги взаимозависимы, что обеспечивает более значимое представление данных? Как и при любом сравнении, лучший ответ зависит от ответов на такие вопросы.

- *Будут ли цифры представлены изолированно*? Рейтинги хороши в предоставлении контекста для интерпретации результатов. На момент написания этой книги баскетбольная команда университета в Стоуни-Брук занимает 111 место среди 351 команд колледжей страны благодаря нашему *индексу процентного рейтинга* (Ratings Percentage Index — RPI), равному 39,18. Какое значение дает нам лучшее представление о том, хороша наша команда или плоха: 111 или 39,18?
- *Каково базовое распределение оценок*? По определению объект с самым высоким рейтингом имеет лучшую оценку, чем объект со вторым рейтингом, но это ничего не говорит о величине разницы между ними. Они практически равны, или №1 существенно лучше?

  Различия в рейтингах *кажутся* линейными: разница между 1 и 2 кажется такой же, как и разница между 111 и 112. Но это не всегда верно в системах оценок. Действительно, небольшие абсолютные различия в оценках зачастую могут вести к существенным различиям в рейтинге.
- *Интересуют крайности или среднее*? Хорошо разработанные системы оценок зачастую имеют колоколообразное распределение. При концентрации оценок вокруг среднего значения небольшие различия в оценках могут означать большие различия в рейтинге. При нормальном распределении увеличение вашей оценки от среднего на одно стандартное отклонение (σ) перемещает вас с 50 *процентиля* (percentile) на 84. Но такое же изменение с 1σ до 2σ приведет вас только с 84 процентиля на 92,5.

  Поэтому, когда организация скатывается с первых в десятые, ее руководители должны уходить. Но когда команда Стоуни-Брук скатывается с 111 места на 120, это, вероятно, представляет незначительную разницу в счете и может быть сброшено со счетов. Рейтинги хороши для выявления самых хороших и самых плохих сущностей в группе, но в куда меньшей степени для различий в области медианы.

4.2.3. Выявление хороших функций оценки

Хорошие функции оценки хороши потому, что они легко интерпретируемы и в целом правдоподобны. Здесь мы рассмотрим свойства статистики, указывающие на следующие направления.

- *Простота вычислений.* Хорошая статистика может быть легко описана и представлена. Индекс BMI является отличным примером: он содержит только два параметра и вычисляется с использованием только простой алгебры. Он был найден в результате поиска во всех простых функциональных формах небольшого числа легко получаемых релевантных переменных. Это отличное практическое упражнение для мозгового штурма возможной статистики из заданного набора функций в наборе данных, который вы хорошо знаете.
- *Простота для понимания.* Из описания статистики должно быть ясно, что ранжирование имеет отношение к рассматриваемому вопросу. "Масса с поправкой на рост" вполне объясняет, почему индекс BMI связан с ожирением. Четкое объяснение идей, лежащих в основе вашей статистики, совершенно необходимо, чтобы другие люди доверяли ей в достаточной мере.
- *Монотонная интерпретация переменных.* Вы должны иметь представление о том, как каждая из функций, используемых в вашей функции оценки, соотносится с целью. Масса *должна* положительно коррелировать с индексом BMI, поскольку избыточный вес подразумевает большой вес. Рост *должен* коррелировать отрицательно, поскольку высокие люди, естественно, весят больше, чем невысокие.

  По правде говоря, вы создаете функцию оценки без фактического золотого стандарта для сравнения. Это требует понимания того, что означают ваши переменные, поэтому ваша оценочная функция будет правильно коррелировать с этой задачей.
- *Дает в целом удовлетворительные результаты по выбросам.* В идеале в любой хорошей системе оценки вы знаете достаточно много обо всех отдельных точках, чтобы иметь представление о том, где они находятся. Если я действительно удивлен идентичностью главных объектов, выявленных системой оценки, то это, вероятно, ошибка, а не особенность. Когда я рассчитываю оценки студентов своих курсов, я уже знаю имена нескольких звезд и нескольких идиотов по их вопросам в аудитории. Если вычисленные мной оценки не соответствуют этим впечатлениям, значит, существует потенциальная ошибка, которую необходимо выявить.

Если элементы данных вам действительно совершенно непонятны, то вам, вероятно, следует лучше изучить свою область деятельности. По крайней мере, создайте искусственные примеры ("Суперзвезда" и "Супертормоз") с такими значениями признаков, чтобы они находились в верхней и нижней части рейтинга, а затем посмотрите, как они соответствуют реальным данным.

- *Использует систематически нормализованные переменные*. Переменные, взятые из колоколообразных распределений, ведут себя в функциях оценки разумно. В хвостах с обоих концов будут присутствовать выбросы, которые соответствуют лучшим/худшим элементам, плюс пик в середине, где оценки элементов должны быть относительно одинаковыми.

  Эти нормально распределенные переменные должны быть преобразованы в Z-оценки (см. раздел 4.3) перед их суммированием, чтобы все функции имели сопоставимые значения среднего и дисперсии. Это уменьшает зависимость функции оценки от магических констант для корректировки весов, поэтому ни одна особенность не оказывает слишком сильного влияния на результаты.

  По правде говоря, суммирование Z-оценок с использованием правильных знаков (плюс для положительных коррелирующих переменных и минус для отрицательных корреляций) при одинаковых весовых коэффициентах будет вести себя примерно правильно. Лучшая функция может взвешивать эти переменные по важности, в зависимости от силы корреляции с целью. Но вряд ли это будет иметь большое значение.

- *Разделяет совпадения осмысленным образом*. Функции ранжирования имеют очень ограниченную ценность, когда есть совпадения. Оценивать ловкость людей по тому, сколько у них пальцев, не очень показательно. Будет очень маленькая группа с двенадцатью пальцами, подавляющее большинство с десятью, а затем небольшие группы жертв несчастных случаев со все более тяжелыми травмами, пока мы не дойдем до нуля. В целом оценки должны быть действительными числами в пределах допустимого диапазона, чтобы минимизировать вероятность совпадений. Предоставление вторичных функций для разрыва совпадений является ценным и имеет смысл, если эти функции также коррелируют с интересующим вас свойством.

4.3. Z-оценки и нормализация

Важным принципом науки о данных является то, что мы должны постараться сделать так, чтобы наши модели могли делать правильные вещи. Методы машинного обучения, такие как линейная регрессия, имеют целью найти линию, оптимально подходящую для данного набора данных. Но очень важно нормализовать все различные переменные, чтобы сделать их диапазон/распределение сопоставимыми, прежде чем мы попытаемся использовать их для чего-то подходящего.

Нашим основным методом нормализации будут *Z-оценки* (Z-score). Они вычисляются так:

$$Z_i = (a_i - \mu)/\sigma,$$

где μ — это среднее значение распределения, а σ — соответствующее стандартное отклонение.

Z-оценки приводят произвольные наборы переменных к единому диапазону. Z-оценки роста, измеренные в дюймах, будут точно такими же, как и у роста, измеренного в милях. Среднее значение Z-оценок по всем точкам равно нулю. На рис. 4.4 показан набор целых чисел, приведенных к Z-оценкам. Значения, превышающие среднее значение, становятся положительными, а значения, которые меньше среднего значения, становятся отрицательными. Стандартное отклонение Z-оценок равно 1, поэтому все распределения Z-оценок имеют сходные свойства.

$$\mu(B) = 21.9 \quad \sigma(B) = 1.92$$
$$\mu(Z) = 0 \quad \sigma(Z) = 1$$

B	19	22	24	20	23	19	21	24	24	23
Z	−1.51	0.05	1.09	−0.98	0.57	−1.51	−0.46	1.09	1.09	0.57

Рис. 4.4. Получение Z-оценок из набора значений B нормализует их так, чтобы они имели среднее $\mu = 0$ и $\sigma = 1$

Преобразование значений в Z-оценки позволяет достичь двух целей. Во-первых, они помогают визуализировать закономерности и корреляции, гарантируя, что все поля имеют одинаковое среднее значение (нуль) и работают в одинаковом диапазоне. Мы понимаем, что Z-оценка 3,87 должна отражать рост уровня баскетболиста, а 79,8 — нет, без знакомства с единицей измерения (скажем, в дюймах). Во-вторых, использование Z-оценок упрощает наши алгоритмы машинного обучения, делая все различные функции сопоставимыми.

Теоретически выполнение такого линейного преобразования, как Z-оценка, *на самом деле* не делает ничего, что большинство алгоритмов обучения не могли

бы понять самостоятельно. Обычно эти алгоритмы находят лучший коэффициент для умножения каждой переменной, который может быть близок к σ, если алгоритм действительно этого хочет.

Тем не менее здесь вступают в силу реалии числовых вычислений. Предположим, мы пытались построить линейную модель на основе двух переменных, связанных с городами США, скажем, их площади в квадратных милях и численности населения. Первое имеет среднее значение порядка 5 и максимум порядка 100. Второе имеет среднее значение порядка 25 000 и максимум — 8 000 000. Чтобы обе переменные оказали одинаковое влияние на нашу модель, мы должны разделить вторую переменную на коэффициент 100 000 или около того.

Это вызвано проблемой точности чисел, поскольку очень небольшое изменение значения коэффициента вызывает очень большое изменение того, насколько переменная совокупности доминирует в модели. Гораздо лучше было бы, чтобы переменные были примерно одинакового масштаба и диапазона распределения, поэтому вопрос заключается в том, получит ли один элемент вес, скажем, вдвое больше, чем другой.

Z-оценки лучше всего использовать для нормально распределенных переменных, которые, в конце концов, полностью описываются средним значением μ и стандартным отклонением σ. Но для распределения по степенному закону они работают хуже. Рассмотрим распределение богатства в Соединенных Штатах, которое может составлять в среднем, скажем, 200 000 долл. при σ = = 200 000 долл. Z-оценка Билла Гейтса в 80 млрд долл. была бы тогда 4999, что по-прежнему является невероятным выбросом, с учетом нулевого среднего значения.

Ваши самые большие огрехи анализа данных заключаются в использовании неправильно нормализованных переменных. Что мы можем сделать, чтобы уменьшить показатели Билла Гейтса? Мы можем применить логарифмирование, как обсуждалось в разделе 2.4.

4.4. Передовые методы ранжирования

Большинство практических задач ранжирования решаются вычислением оценок в виде линейных комбинаций признаков, с последующей их сортировкой. В отсутствие какого-либо золотого стандарта эти методы дают статистику, которая зачастую является и показательной, и информативной.

Тем не менее было разработано несколько мощных методов вычисления рангов из определенных типов входных данных: результатов парных сравнений, сетей отношений и даже сборок других рангов. Эти методы мы и рассмотрим здесь.

4.4.1. Рейтинг Эло

Рейтинги зачастую формируются в результате анализа последовательностей двоичных сравнений, которые естественным образом возникают в соревнованиях между сущностями.

- *Результаты спортивных соревнований.* Типичные спортивные соревнования, будь то футбольные или шахматные матчи, стравливают команды A и B между собой. Победит только одна из них. Таким образом, каждый матч по сути является двоичным сравнением достоинств.
- *Голоса и опросы.* Знающих людей нередко просят сравнить варианты и решить, какой выбор они считают лучшим. На выборах эти сравнения называются голосами. Основным компонентом рейтинга некоторых университетов является опрос профессоров: какая школа лучше, A или B?

  В фильме *Социальная сеть* показано, как Марк Цукерберг (Mark Zuckerberg) из Facebook начинает свою работу с FaceMash, веб-сайтом, демонстрирующим пользователям два лица и просящим их выбрать наиболее привлекательное из них. Затем, основываясь на этих парных сравнениях, его сайт ранжировал все лица от наименее до наиболее привлекательных.
- *Неявные сравнения.* С правильной точки зрения данные объектов могут быть обоснованно интерпретированы как парные сравнения. Предположим, что студент был принят обоими университетами A и B, но выбирает университет A. Это можно считать неявным голосованием за то, что университет A лучше, чем B.

Каков правильный способ интерпретации таких голосов, особенно когда есть много кандидатов, и не все пары игроков соревнуются друг с другом? Нет смысла говорить о том, кто выиграл больше всего потому, что (а) одни могли бы участвовать в большем количестве сравнений, чем другие игроки, и (б) они могли бы избежать сильных соперников и победить только слабых.

Система Эло (Elo system) начинает с оценки всех игроков, предположительно в равной степени, а затем постепенно корректирует оценку каждого игрока в соответствии с результатом каждого матча согласно формуле:

$$r'(A) = r(A) + k(S_A - \mu_A),$$

где $r(A)$ и $r'(A)$ представляют предыдущую и обновленную оценки для игрока A. k — это фиксированный параметр, отражающий максимально возможную корректировку оценки по результатам одного матча. Небольшое значение k приводит к довольно статичному ранжированию, в то время как использование слишком большого значения k приводит к резким колебаниям ранга на основе последнего матча.

S_A — это результат, полученный игроком A в рассматриваемом матче. Обычно $S_A = 1$, если A выиграл, и $S_A = -1$, если A проиграл.

μ_A — это ожидаемый результат для A в поединке с B. Если A имеет точно такой же уровень квалификации, что и B, то, вероятно, $\mu_A = 0$. Но предположим, что A — чемпион, а B — новичок или рохля. Мы ожидаем, что A почти наверняка победит в личном поединке, поэтому $\mu_A > 0$ и, вероятно, будет достаточно близко к 1.

Здесь все ясно, кроме определения μ_A. Учитывая оценку вероятности того, что A побьет B ($P_{A>B}$), то

$$\mu_A = 1 \cdot P_{A>B} + (-1) \cdot (1 - P_{A>B}).$$

Эта вероятность выигрыша явно зависит от величины разницы в навыках между игроками A и B, которая и должна быть измерена системой ранжирования. Таким образом, $x = r(A) - r(B)$ представляет эту разницу в квалификации.

Для завершения системы ранжирования Эло нам нужен способ получения действительной переменной x и ее преобразования в значимую вероятность. Это серьезная проблема, с которой мы будем неоднократно сталкиваться в этой книге и которая решается с помощью такого математического подхода, как *логит-функция* (logit function).

Логит-функция

Предположим, что мы хотим получить действительную переменную $-\infty < x < \infty$ и преобразовать ее в вероятность $0 \le p \le 1$. Существует множество способов сделать это, но самым простым преобразованием является $p = f(x)$, где

$$f(x) = \frac{1}{1 + e^{-cx}}$$

Форма логит-функции $f(x)$ показана на рис. 4.5. Обратите внимание на особые случаи в середине и на концах.

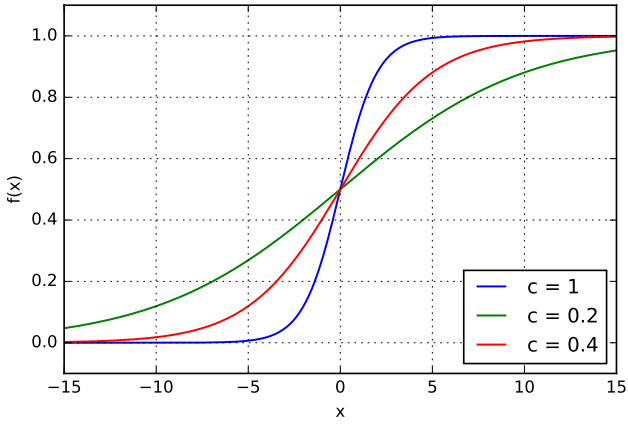


Рис. 4.5. Форма логит-функции для трех разных значений c

- Когда способности двух игроков одинаковы, $x = 0$ и $f(0) = 1/2$, значит, оба игрока имеют равную вероятность выигрыша.
- Когда игрок A имеет огромное преимущество, $x \to \infty$ и $f(\infty) = 1$, значит, игрок A является фаворитом для победы в матче.
- Когда игрок B имеет огромное преимущество, $x \to -\infty$ и $f(-\infty) = 0$, значит, игрок B является фаворитом для победы в матче.

Это именно те значения, которые нам нужны, если x определяет разницу в навыках игроков.

Логит-функция плавно и симметрично интерполируется между этими полюсами. Параметр c в логит-функции определяет, насколько крут переход. Означают ли небольшие различия в навыках большие различия в вероятности выигрыша? Для $c = 0$ ландшафт плоский, как блин: $f(x) = 1/2$ для всех x. Чем больше значение c, тем четче переход, как показано на рис. 4.5. Действительно, когда $c = \infty$, получается ступенчатая функция от 0 до 1.

Параметр $c = 1$ является разумным началом, но правильный выбор зависит от специфики области. Наблюдение за тем, как часто данная величина различий в навыках приводит к неожиданности (выигрыш слабой стороны), помогает определить параметр. Система ранжирования Elo Chess была разработана таким образом, что $r(A) - r(B) = 400$, а значит вероятность выигрыша A в десять раз выше, чем у B.

На рис. 4.6 показаны вычисления Эло в контексте крайне маловероятного турнира с участием трех величайших шахматистов в истории и одного любителя с низким рейтингом. Здесь $k = 40$, что подразумевает максимально возможное отклонение оценки в 80 очков как следствие любого одиночного матча. Стандартная логит-функция давала Каспарову вероятность выиграть у Скиены в первом раунде 0,999886, но благодаря чуду, подобному воскресению Лазаря, матч закончился иначе. Как следствие 80 баллов перешло из рейтинга Каспарова в мой.

На другой стороне поля сражались два настоящих чемпиона по шахматам, причем, к большому сожалению Полгар, она набрала всего 55 очков. Она вытерла мною пол в следующем раунде, достижение, столь ожидаемое, что она получила практически нулевые рейтинговые очки. Метод Эло очень эффективен при обновлении рейтингов в ответ на неожиданность, а не только на победу.

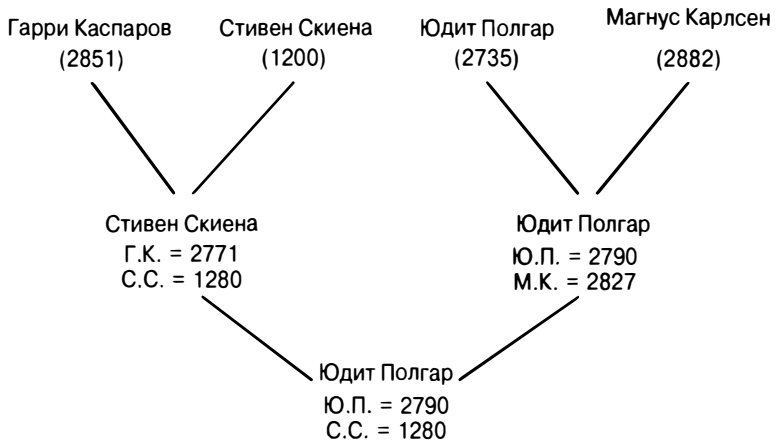


Рис. 4.6. Изменения в результатах ELO в ходе маловероятного шахматного турнира

4.4.2. Слияние рейтингов

Любой отдельный числовой признак f, такой как рост, может инициировать $\binom{n}{2}$ попарных сравнений среди n элементов, проверяя истинность $f(A) > f(B)$ для каждой пары элементов A и B. Мы могли бы передать эти пары в метод Эло, чтобы получить рейтинг, но это был бы довольно глупый способ суждения о вещах. В конце концов, результат любого такого анализа будет просто отражать отсортированный порядок f.

Однако интеграция набора рейтингов по нескольким разным признакам создает более интересную проблему. Здесь мы интерпретируем отсортированный порядок i-го признака как определение перестановок P_i для интересующих нас элементов. Мы ищем согласованную перестановку P, которая каким-то образом лучше всего отражает все перестановки компонентов $P_1, \ldots, P_k$.

Это требует определения *функции расстояния* (distance function) для измерения сходства между двумя перестановками. Аналогичная проблема возникла при определении коэффициента ранговой корреляции Спирмена (см. раздел 2.3.1), где мы сравнили две переменные по мере согласования в относительном порядке элементов.[1]

Метод Борда (Borda's method) создает согласованные ранги из множества других рангов с использованием простой системы начисления баллов. В частности, мы назначаем стоимость или вес каждой из n позиций в перестановке.

[1] Обратите внимание на разницу между показателем сходства и метрикой расстояния. В корреляции оценки увеличиваются, поскольку элементы становятся более похожими, в то время как в функции расстояния разница сводится к нулю. Более подробно метрики расстояния будут обсуждаться в разделе 10.1.1.

Затем для каждого из n элементов мы суммируем веса его позиций по всем k входным рейтингам. Сортировка этих n баллов определяет итоговый согласованный рейтинг.

Теперь все ясно, кроме сопоставления позиций и стоимостей. Простейшая функция стоимости присваивает i пунктов для появления в i-й позиции в каждой перестановке, т.е. мы суммируем ранги элемента по всем перестановкам. Это то, что мы делаем в примере на рис. 4.7. Пункт A получает $3 \cdot 1 + 1 \cdot 2 = 5$ пунктов в силу того, что он появляется первым в трех рейтингах и вторым в одном. Пункт C заканчивает с 12 пунктами, включая 2, 3, 3 и 4. Окончательный согласованный рейтинг $\{A, B, C, D, E\}$ объединяет все голоса из всех входных рейтингов, даже если согласованности нет, то по крайней мере мы можем расстаться со всеми четырьмя первоначальными рейтингами.

1	A	B	A	A		A: 5
2	C	A	B	B		B: 8
3	B	C	C	D	⟶	C: 12
4	D	D	E	C		D: 16
5	E	E	D	E		E: 19

Рис. 4.7. Метод Борда для построения согласованного рейтинга $\{A, B, C, D, E\}$ для набора из четырех входных рейтингов с использованием линейных весов

Однако отнюдь не очевидно, что использование линейных весов является наилучшим выбором, поскольку это предполагает абсолютную уверенность в нашей точности позиционирования элементов на протяжении всех перестановок. Как правило, мы знаем больше всего о достоинствах нашего лучшего выбора, но будем довольно мало осведомлены о том, как именно они соотносятся между собой ближе к середине порядка. Если это так, то лучшим подходом может быть получение большего количества пунктов за различие между 1-м и 2-м, чем между 110-м и 111-м.

Этот тип взвешивания неявно выполняется колоколообразной кривой. Предположим, мы отбираем n элементов через равные интервалы из нормального распределения, как показано на рис. 4.8. Присвоение этих значений x в качестве позиционных весов приводит к большему разбросу среди самых высоких и самых низких рангов, чем в центре. Области по краям действительно такие же широкие, как кажутся, для этих 50 одинаково разнесенных точек: напомним, что 95% вероятностной массы находится в пределах 2σ от центра.

С другой стороны, если мы не уверены в симметричности, мы могли бы сделать выборку из полунормального распределения, поэтому хвост наших

рангов взвешивается на пике нормального распределения. Таким образом, наибольшее разделение существует среди элементов высшего ранга, а минимум различий между элементами в хвосте.

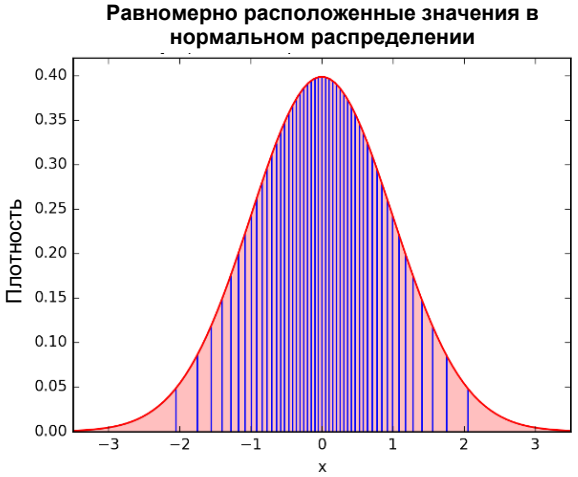


Рис. 4.8. Равномерно расположенные по нормальному распределению значения находятся ближе к середине, чем к концам, что дает соответствующие веса для метода Борда

Здесь ваш выбор весовой функции зависит от области применения, поэтому выберите ту, которая лучше справляется с вашей задачей. Определение *лучшей* функции стоимости является некорректной задачей. И когда мы пытаемся разработать идеальную систему выборов, случаются странные вещи, как будет показано в разделе 4.6.

4.4.3. Рейтинг на основе диграфа

Сети предоставляют альтернативный взгляд на набор результатов голосования в форме "ранг A впереди B". Мы можем построить ориентированный граф/сеть, где есть вершина, соответствующая каждой сущности, и направленное ребро (A, B) для каждого голоса, который повышает ранг A относительно B.

Тогда оптимальным ранжированием будет перестановка P вершин, которая затрагивает наименьшее количество ребер, где ребро (A, B) не подходит, если B предшествует A в конечной перестановке рангов P.

Если бы голоса были совершенно едины, то эта оптимальная перестановка нарушила бы нулевые ребра. Действительно, это тот случай, когда в графе нет направленных циклов. Направленный цикл, подобный (A, C), (C, E), (E, A), представляет собой внутреннее противоречие с любым порядком рангов, поскольку всегда будет несчастливое ребро, независимо от того, какой порядок вы выберете.

Направленный граф без циклов называется *ориентированным ациклическим графом* (Directed Acyclic Graph — DAG). Внимательный читатель с некоторым знанием алгоритмов вспомнит, что нахождение подобного оптимального порядка вершин называется *топологической сортировкой* (topologically sorting) DAG, которая может быть эффективно выполнена за линейное время. Рис. 4.9 (слева) представляет собой граф DAG и имеет ровно два разных порядка, соответствующих направленным ребрам: {*A*, *B*, *C*, *D*, *E*} и {*A*, *C*, *B*, *D*, *E*}.

Однако крайне маловероятно, что реальный набор функций или избирателей окажутся взаимно согласованными. Задача *максимального ациклического подграфа* (maximum acyclic subgraph) заключается в том, чтобы найти наименьшее количество ребер, которые нужно удалить, чтобы оставить граф DAG. Достаточно удалить ребро (*E*, *A*) на рис. 4.9 (справа). К сожалению, проблемой поиска лучших рангов здесь является NP-полная задача, а значит не существует эффективного алгоритма для нахождения оптимального решения.

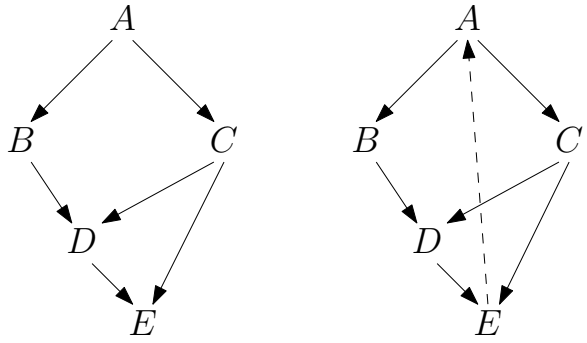


Рис. 4.9. Последовательно упорядочиваемые предпочтения дают ациклический граф, или DAG-граф (слева). Непоследовательные предпочтения приводят к направленным циклам, которые можно разорвать, удалив небольшие наборы тщательно отобранных ребер, они показаны здесь пунктиром (справа)

Но здесь есть естественная эвристика (natural heuristic). Хорошей подсказкой о принадлежности вершины *v* является разность d_v между входящей (in-degree) и исходящей (out-degree) степенями вершины. Когда разность d_v имеет большое отрицательное значение, она, вероятно, относится к начальной части перестановок, поскольку доминирование существует над многими элементами и отсутствует только над несколькими. Можно создать вполне достойную перестановку рангов, сортируя вершины в соответствии с этими различиями. Еще лучше последовательно помещать наиболее отрицательную (или наиболее положительную) вершину *v* в ее логическую позицию, удаляя ребра, попадающие на *v*, а затем корректировать счетчики перед позицией следующей лучшей вершины.

4.4.4. Алгоритм PageRank

Но существует и другой, более известный способ упорядочения вершин в сети по важности: алгоритм PageRank, лежащий в основе поисковой системы Google.

Сеть состоит из веб-страниц, большинство из которых содержат ссылки на другие веб-страницы. Ваша веб-страница, ссылающаяся на мою, является неявным подтверждением того, что вы считаете мою страницу довольно хорошей. Если это воспринимается как голосование "вы считаете, что моя страница лучше вашей", мы можем создать сеть ссылок и рассматривать ее как проблему максимального ациклического подграфа, которая обсуждалась в предыдущем разделе.

Но доминирование — не совсем правильная интерпретация для ссылок в Интернете. Вместо этого алгоритм PageRank вознаграждает вершины, которые имеют больше всего ссылок на нее: если все дороги ведут в Рим, то Рим должен быть довольно важным местом. Кроме того, он взвешивает эти ссылки по мощи источника: ссылка на меня с важной страницы должна весить куда более, чем с одного из спамовых сайтов.

Детали здесь интересны, но я отложу более глубокое обсуждение до раздела 10.4, когда мы обсудим сетевой анализ. Однако я надеюсь, что это краткое введение в алгоритм PageRank поможет вам оценить следующую историю.

4.5. Случай из жизни: месть Клайда

В старших классах средней школы у меня возникла идея написать программу, которая предсказывала бы результаты профессиональных футбольных игр. Я не был серьезно заинтересован в футболе как в спорте, но видел, как несколько моих одноклассников ставили свои деньги на обед на результаты футбольных игр выходного дня. Мне казалось очевидным, что написание программы, которая точно предсказывает исход футбольных игр, может иметь большое значение, а кроме того, это очень крутая вещь.

Оглядываясь назад, можно сказать, что задуманная мной программа кажется сейчас безнадежно грубой. Для прогнозирования количества очков команды x в игре против команды y моя программа усредняла очки, набранные командой x, и очки, пропущенные командой y.

$$P_x = \frac{((\text{очки, набранные командой } x) + (\text{очки, пропущенные командой } y))}{2 \cdot (\text{сыграно игр})},$$

$$P_y = \frac{((\text{очки, набранные командой } y) + (\text{очки, пропущенные командой } x))}{2 \cdot (\text{сыграно игр})}.$$

Затем я скорректировал бы эти числа вверх или вниз в ответ на такие факторы, как преимущество домашнего поля, округлил бы числа соответствующим образом и назвал результат своим прогнозом счета игры.

Эта компьютерная программа, *Clyde* (Клайд), была моей первой попыткой создать функцию оценки для некоторого аспекта реального мира. В этом была определенная доля логики. Хорошие команды набирают больше очков, чем допускается, в то время как плохие команды отдают больше очков, чем набирают. Если команда *x* играет с командой *y*, которая набрала довольно много очков, то команда *x* должна набрать больше очков, когда играет против команды *y*, чем когда играет против других команд лиги. Аналогичным образом, чем больше очков набрала команда *x*, играя против остальной части лиги, тем больше очков она может набрать, играя против *y*.

Конечно, эта грубая модель не могла охватить все аспекты футбольной реальности. Предположим, что команда *x* до сих пор играла во всех матчах в этом сезоне, а команда *y* играла с лучшими командами лиги. Команда *y* может быть гораздо лучшей, чем команда *x*, хотя ее результаты пока плохие. Эта модель игнорирует также любые факторы, от которых страдает команда: жаркая или холодная погода и состояние команды. Она игнорирует все факторы, которые делают спорт по своей природе непредсказуемым.

И все же даже такая простая модель способна предсказать результат футбольных игр. Если вы подсчитаете средние очки, как указано выше, и дадите домашней команде дополнительные три очка в качестве бонуса, вы выберете победителя почти в двух третях из всех футбольных матчей. Сравните это с еще более грубой моделью подбрасывания монеты, которая правильно предсказывает только половину бросков. Это был первый серьезный урок, который Clyde преподал мне:

Даже грубые математические модели могут иметь реальную прогнозирующую силу.

Будучи дерзким 16-летним парнем, я написал в нашу местную газету, *The New Brunswick Home News*, объяснив, что у меня есть компьютерная программа для прогнозирования результатов футбольных игр, и я был готов предложить им эксклюзивную возможность публиковать свои прогнозы каждую неделю. Помните, что это было в 1977 году, задолго до того, как персональные компьютеры утвердились в общественном сознании. В те дни идея старшеклассника, *использующего* компьютер, имела значительную ценность по новизне. Чтобы оценить, как все изменилось, посмотрите статью, опубликованную обо мне и программе Clyde на рис. 4.10.

Student uses computers to predict football winners

By JEFF LEEBAW
Home News staff writer

EAST BRUNSWICK — A 16-year-old East Brunswick High School student has found a way to combine an interest in football with a fascination for computers.

Steven Skiena says he can determine, with a high degree of accuracy, the outcome of professional football games by feeding a computer pertinent information about competing teams.

"The winners will almost always be correct," said the high school junior who lives at 5 Currier Road off Dunhams Corner Road. "I had an 86 per cent accuracy rate when I started predicting at the end of last season."

He does it by feeding the computer a myriad of statistics that include team records, points scored and allowed, average yards gained and allowed during a game, a breakdown of the yards gained and allowed into rushing and passing categories, performances at home and on the road, and more.

The information is gathered from weekly compilations of football statistics and standings. Skiena puts the facts on index cards and then types them into one of the six computer terminals at the high school or a terminal at The Library where he works part-time after school.

"I get a winning team, a decimal score for each team and a point spread," said the teen-ager who completed a computer programming course last year at the high school.

His first attempt at picking winners involved a Monday night game between the Oakland Raiders, the eventual Super Bowl victors, and the Cincinnati Bengals.

It was a difficult game to analyze because Cincinnati was fighting for a playoff berth while Oakland had already clinched a spot in the post season competition.

"Nobody knew whether Oakland would be giving 100 per cent," Skiena said. "But my calculations indicated they would win by 24-20. The final score was 35-20. They went all out."

Skiena said he went on to pick 12 of the 14 winners the following week and accurately predicted Oakland would defeat Minnesota in the Super Bowl.

The National Football League's 1976 Record Book, which breaks down last year's statistics for each of the league's 28 teams, will supply most of Skiena's information for the first few weeks of the 1977 season. He will also use statistics from the final two exhibition games played this year by each of the teams.

Skiena wrote a computer program based on 17 statistical variables that might come into play during a football game.

The computer, in essence, asks him questions and he types the answers.

"It starts out by asking for the names of the teams," he said. "Then it will ask for records, points scored, etc...."

The computer program also attempts to include such intangible variables as injuries.

"The injuries are broken down into offense, defense and quarterback," he explained. "Obviously a quarterback injury is the most serious. It's too difficult to break down injuries for every position. When the computer asks for the number of injuries on defense, I'll type in one, two or whatever the figure is.

Will Skiena use his computer results to enter the variety of football pools and contests that are available during the season?

"No," he said. "I don't like to bet on my own predictions. Last season a friend bet on a game I predicted and it happened to be one of the few that were wrong."

STEVEN SKIENA
...to test his accuracy

Predictions published

Steven Skiena will get a chance to display his skill as a pro football prognosticator each Sunday in The Home News.

The youngster's weekly selections will be an "added ingredient to our football coverage," according to Home News Executive Editor Robert E. Rhodes.

"I think it's interesting enough for us to give it a shot," Rhodes said of the teen-ager's computer method of determining the outcome of games. "He seems like an earnest young man and we'll stand behind him."

Skiena will receive a "modest stipend" for predicting the winners, scores for each team and briefly explaining the reasons for his conclusions.

His column will appear for the first time in Sunday's sports section when the National Football League (NFL) opens its 1977 season with a slate of 13 games. Skiena will also predict the outcome of the league's Monday night games.

The Home News is publishing the column in the sports section to test the youngster's system and to offer football fans an entertaining feature. Its purpose is not to encourage betting.

"We'll be printing his selections close enough to the time of the game to prevent betting," Rhodes said. "There's big interest in pro football and more than anything else we want to test his system. I'll be rooting for him."

Рис. 4.10. Моя первая попытка математического моделирования

Я получил работу. Клайд предсказал исход почти каждой игры 1977 года в Национальной футбольной лиге. Насколько я помню, мы с Клайдом завершили сезон с впечатляющим показателем 135–70. Каждую неделю они сравнивали мои прогнозы с прогнозами спортивных журналистов газеты. Насколько я помню, мы все закончили с разрывом в несколько игр друг от друга, хотя большинство авторов спортивных обзоров закончили с лучшими прогнозами, чем компьютер.

Home News были настолько впечатлены моей работой, что не продлили контракт со мной на следующий сезон. Тем не менее прогнозы Клайда для сезона 1978 года были опубликованы в газете *Philadelphia Inquirer*, гораздо большей газете. Хотя у меня не было личной колонки. Вместо этого *Inquirer* включил меня в десятку любительских и профессиональных предсказателей, или зазывал. Каждую неделю мы должны были прогнозировать результаты четырех игр, указав разницу в счете.

В футболе *ставка на разницу в счете* (point spread) — это способ уравнять сильные команды, когда делают ставки. Ставка на разницу в счете предназначена для того, чтобы сделать каждую игру предположением 50/50, а следовательно, значительно затруднить прогнозирование результатов игр.

Мы с Клайдом не очень хорошо противостояли разнице в счете во время сезона 1978 года Национальной футбольной лиги, как и большинство других опросов *Philadelphia Inquirer*. Мы предсказали правильно только 46% игр по разнице в счете, что стало достаточно хорошим (или плохим) результатом, чтобы стать седьмым из десяти лучших предсказателей. Противостояние разнице в счете преподало мне второй главный жизненный урок:

Грубые математические модели не имеют реальной предсказательной силы, когда на кону реальные деньги.

Таким образом, программе Clyde не суждено было совершить революцию в мире футбольного прогнозирования. Я почти забыл об этом, пока не поставил задачу предсказания результатов Суперкубка на своем курсе по науке о данных. Группа, получившая эту задачу, состояла из студентов из Индии, т.е. они знали гораздо больше о крикете, чем об американском футболе, когда начинали.

Тем не менее они приняли вызов, став поклонниками игры, пока создавали большой набор данных о результатах каждой профессиональной и студенческой игры, в которую играли за последние десять лет. Они выполнили логистический регрессионный анализ по 142 различным признакам, включая броски, проходы и удар ногой, время владения и количество ударов. Затем они с гордостью сообщили мне о точности своей модели: правильные прогнозы в 51,52% игр NFL.

Я закричал: "*Что! Это ужасно!*" "Пятьдесят процентов — это то, что вы получаете, подбрасывая монету. Попробуйте усреднить очки, набранные и полученные двумя командами, и дать три очка хозяевам поля. Как работает эта простая модель?"

По созданным ими данным новая подобная Clyde модель правильно предсказывала 59,02% всех игр, что намного лучше, чем их сложная модель машинного обучения. Они потеряли в тумане слишком много функций, которые не были должным образом нормализованы и построены с использованием статистики, собранной за слишком долгую историю, чтобы представлять текущий состав команды. В конце концов студентам удалось придумать модель на основе PageRank, которая показала себя немного лучше (60,61%), но Clyde почти так же хорошо выступил в качестве базовой модели.

Здесь есть несколько важных уроков. Во-первых, мусор на входе, мусор на выходе. Если вы не подготовите чистый, правильно нормализованный набор данных, самые продвинутые алгоритмы машинного обучения не смогут вас спасти. Во-вторых, простые результаты, основанные на скромном объеме знаний по конкретным областям, могут быть на удивление хороши. Кроме того, они позволяют вам быть честным. Создайте и оцените простые, понятные исходные условия, прежде чем инвестировать в более эффективные подходы. Применение программы Clyde в качестве исходного пункта оставил их модель машинного обучения не у дел.

4.6. Теорема Эрроу о невозможности

Мы видели несколько подходов к созданию функций оценки или ранжирования из данных. Если у нас есть золотой стандарт, обусловливающий "правильный" относительный порядок, по крайней мере для некоторых сущностей, то его можно использовать для обучения или проверки нашей функции оценки, чтобы максимально соответствовать этим рейтингам.

Но без золотого стандарта сложно создать лучшую систему ранжирования. Это является следствием *теоремы Эрроу о невозможности* (Arrow's impossibility theorem), которая доказывает, что ни одна избирательная система для объединения перестановок предпочтений не удовлетворяет следующим желательным и невинно выглядящим свойствам.

- Система должна быть однозначной, так как, когда ее просят выбрать между альтернативами A и B, она должна четко сказать, что (1) A предпочтительнее B, (2) B предпочтительнее A или (3) они равнозначны.
- Результаты должны быть транзитивны, т.е. если A предпочтительнее B, а B предпочтительнее C, то A должно быть предпочтительнее C.

- Если для всех людей A предпочтительней B, то система должна отдать предпочтение A, а не B.
- Система не должна зависеть от предпочтений только одного человека, диктатора.
- Предпочтение A по сравнению с B должно быть независимым от предпочтений любых других альтернатив, таких как C.

На рис. 4.11 показаны некоторые аспекты теоремы Эрроу и нетранзитивный характер упорядочения типа "камень-ножницы-бумага". Здесь три избирателя (x, y и z) ранжируют свои предпочтения по цветам. Чтобы установить предпочтение между двумя цветами a и b, логическая система может сравнить, сколько перестановок ранжируют a перед b, а не b перед a. По этой системе красный цвет предпочтительнее зеленого по x и y, поэтому красный выигрывает. Точно так же зеленый цвет предпочтительнее синего по x и z, поэтому зеленый выигрывает. Согласно транзитивности красный должен быть предпочтительнее синего, что подразумевает эти результаты. И все же y и z предпочитают синий цвет красному, нарушая основное условие, которое мы хотим сохранить в нашей избирательной системе.

Избиратель	Красный	Зеленый	Синий
x	1	2	3
y	2	3	1
z	3	1	2

Рис. 4.11. Рейтинг предпочтений для цветов, подчеркивающий потерю транзитивности. Красный предпочтительнее зеленого, а зеленый предпочтительнее синего, но синий предпочтительнее красного

Теорема Эрроу полна сюрпризов, но означает ли это, что мы должны отказаться от рейтинга в качестве инструмента для анализа данных? Конечно нет, теорема Эрроу означает не более чем то, что мы должны отказаться от демократии. Традиционные системы голосования основаны на идее о том, что *правила большинства* в основном хорошо отражают предпочтения населения, будучи соответствующим образом обобщенны для работы с большим количеством кандидатов. А техники, описанные в этой главе, обычно хорошо справляются с ранжированием предметов интересными и значимыми способами.

На заметку. Мы не ищем правильные рейтинги, поскольку это плохо определенная цель. Вместо этого мы ищем полезные и интересные рейтинги.

4.7. Случай из жизни: кто больше?

Иногда мои студенты говорят мне, что я уже история. Я надеюсь, что это еще не совсем так, но я очень интересуюсь историей, как и мой бывший докторант Чарльз Уорд. Мы с Чарльзом поболтали о наиболее значимых фигурах в истории, и как это можно измерить. Как и большинство людей, мы нашли ответы в Википедии.

Википедия — удивительная вещь, распределенный рабочий продукт, созданный более чем 100 000 авторов, который каким-то образом поддерживает в общем здравый стандарт точности и глубины. Википедия содержит удивительный объем человеческих знаний в открытой и машиночитаемой форме.

Мы начали с использования английской Википедии в качестве источника данных для исторического рейтинга. Наш первый шаг состоял в том, чтобы извлечь функциональные переменные со страницы каждого человека в Википедии, которые должны четко соотноситься с историческим значением. Это включает в себя следующие функции.

- *Длина*. Страницы в Википедии у наиболее значительных исторических фигур должны быть куда длинные, чем у простых смертных. Таким образом, длина статьи в словах обеспечивает естественную характеристику, отражающую историческую влиятельность, по крайней мере до некоторой степени.
- *Посещения*. Страницы в Википедии о наиболее значимых фигурах читают чаще, чем другие, поскольку они представляют больший интерес для большего числа людей. Мою страницу в Википедии открывают в среднем двадцать раз в день, что довольно круто. Но страницу Исаака Ньютона читают в среднем 7700 раз в день, что значительно лучше.
- *PageRank*. Значимые исторические фигуры взаимосвязаны с другими значимыми историческими фигурами, что отражают гиперссылки в статьях Википедии. Это определяет ориентированный граф, где вершинами являются статьи, а гиперссылки — ориентированными ребрами. Обработка системой PageRank этого графа позволяет измерить центральность (centrality) каждой исторической фигуры, которая хорошо коррелирует со значимостью.

В целом мы извлекли шесть функций для каждой исторической фигуры. Затем мы нормализовали эти переменные перед агрегацией, по сути, объединяя базовые рейтинги с нормально распределенными весами, как упоминалось в разделе 4.4.2. Мы использовали метод *анализа статистических факторов* (statistical factor analysis), относящийся к анализу главных компонентов (который обсуждался в разделе 8.5.2), чтобы выделить два фактора, объясняющих

большую часть расхождений в наших данных. Простая линейная комбинация этих переменных дала нам функцию оценки, и мы отсортировали оценки, чтобы определить наше начальное ранжирование, что мы и назвали *славой*.

Двадцать лучших фигур, по нашей оценке славы, показаны на рис. 4.12 (справа). Мы изучили эти рейтинги и решили, что они не отражают то, что мы хотели. В двадцатку по славе вошли такие поп-музыканты, как Мадонна и Майкл Джексон, а также три современных президента США. Было ясно, что современные фигуры стоят намного выше, чем должны, по нашему мнению: наша функция оценки оценила текущую известность намного выше, чем историческое значение.

Наше решение состояло в снижении оценок современных фигур, чтобы компенсировать фактор времени. То, что страница нынешней знаменитости в Википедии имеет много посещений, конечно, впечатляет, однако то, что нас все еще интересует некто, кто умер 300 лет назад, впечатляет гораздо больше. Двадцать лучших фигур после коррекции времени показаны на рис. 4.12 (слева).

Значимость	Имя
1	Иисус
2	Наполеон
3	Уильям Шекспир
4	Мухаммед
5	Авраам Линкольн
6	Джордж Вашингтон
7	Адольф Гитлер
8	Аристотель
9	Александр Великий
10	Томас Джефферсон
11	Генрих VIII
12	Елизавета I
13	Юлий Цезарь
14	Чарльз Дарвин
15	Карл Маркс
16	Мартин Лютер
17	Королева Виктория
18	Иосиф Сталин
19	Теодор Рузвельт
20	Альберт Эйнштейн

Слава	Личность
1	Джордж Буш
2	Барак Обама
3	Иисус
4	Адольф Гитлер
5	Рональд Рейган
6	Билл Клинтон
7	Наполеон
8	Майкл Джексон
9	Уильям Шекспир
10	Элвис Пресли
11	Мухаммед
12	Иосиф Сталин
13	Авраам Линкольн
14	Д. Вашингтон
15	Альберт Эйнштейн
16	Джон Ф. Кеннеди
17	Елизавета II
18	Иоанн Павел II
19	Мадонна
20	Бритни Спирс

Рис. 4.12. 20 величайших исторических личностей, ранжированных по значимости (слева) и современной славе (справа)

Теперь это было *то*, чего мы хотели! Мы проверили рейтинги, используя любые прокси для исторического значения, которое мы могли найти: другие опубликованные рейтинги, цены на автографы, спортивную статистику,

учебники истории и результаты выборов в Зал славы. Наш рейтинг показал сильную корреляцию со всеми этими прокси.

Действительно, я думаю, что эти рейтинги чудесно показательны. Мы написали книгу, описывающую все виды вещей, которые можно извлечь из них [12]. Я с гордостью рекомендую вам прочитать ее, если вы интересуетесь историей и культурой. Чем больше мы изучали эти рейтинги, тем больше меня впечатляла их общая устойчивость.

К сожалению, наши опубликованные рейтинги не встретили всеобщего согласия. Отнюдь нет. О наших рейтингах были опубликованы десятки газетных и журнальных статей, многие довольно враждебные. Почему люди не уважают их, несмотря на наши многочисленные проверки? Оглядываясь назад, можно сказать, что бо́льшая часть спорных моментов возникла по трем следующим причинам.

- *Неявные различия в понятии значимости.* Наши методы были разработаны так, чтобы измерять *силу мемов* (meme-strength), т.е. насколько успешно эти исторические личности распространяли свои имена по истории. Но многие читатели считали, что наши методы должны охватывать понятие исторического *величия*. Кто был самым важным с точки зрения изменения мира? И мы имеем в виду весь мир или только англоязычный мир? Как может не быть в списке китайских или индийских фигур, если они представляют более 30% населения мира?

  Мы должны договориться о том, что мы пытаемся измерить, прежде чем измерять это. Рост является отличной мерой размера, но он не поможет справиться с ожирением. Однако рост очень полезен при отборе игроков для баскетбольной команды.

- *Выбросы.* Пробы на запах важны для оценки результатов анализа. Что касается нашего рейтинга, это означало проверку расположения людей, которых мы знали, чтобы подтвердить, что они попали в соответствующие места.

  Я остался доволен нашим методом ранжирования подавляющего большинства исторических личностей. Но было несколько человек, которых наш метод оценил куда выше, чем любой разумный человек, в частности президент Джордж Буш (36) и подростковая телезвезда Хилари Дафф (Hilary Duff) (1626). Эти выбросы можно заметить и отбросить. Но поймите, что мы задействовали почти 850 000 исторических фигур, примерно население Сан-Франциско. Несколько ошибочно выбранных плохих примеров должны быть приведены в правильном контексте.

- *Ограничения на навешенные ярлыки.* Большинство рецензентов видели рейтинги только 100 наших лучших фигур, они жаловались и на то,

где именно мы разместили людей, и на то, что некто не попал в список. Женское телешоу *The View* жаловалось, что у нас не хватает женщин. Я вспоминаю британские статьи, жалующиеся на то, что у нас Уинстон Черчилль (37) был оценен слишком низко, в статьях из Южной Африки говорилось, что мы отказались от Нельсона Манделы (356), статьи на китайском языке о том, что мы вообще обошли Китай, и даже чилийский журнал пожаловался на отсутствие чилийцев.

Отчасти это отражает культурные различия. У этих критиков было неявное представление о значении, отличное от того, которое отражено в английской Википедии. Но многое из этого отражает тот факт, что в топ-100 есть ровно сотня мест. Многие из фигур, которые они считали отсутствующими, были лишь немного за пределами видимого горизонта. Из-за каждого нового человека, которого мы помещали в первую сотню, нам приходилось выбрасывать кого-то другого. Но читатели почти никогда не предлагали имена, которые должны быть пропущены, только те, которые должны были быть добавлены.

Какая здесь мораль? Постарайтесь предвидеть озабоченность публики по поводу вашего рейтинга. Нам было рекомендовано явно называть нашу меру *силой мемов*, а не *значимостью*. Оглядываясь назад, использование этого менее громкого имени позволило бы нашим читателям лучше оценить то, что мы делали. Вероятно, нам также следовало воздержаться от создания сотни лучших рейтингов, а вместо этого сосредоточиться на относительных порядках в группах интересов: кто были ведущими музыкантами, учеными и художниками? Это могло бы оказаться менее спорным, помогая людям укрепить доверие к тому, что мы делали.

4.8. Дополнительная информация

Лангвилл и Майер [48] дают общее представление о большинстве обсуждаемых здесь методов ранжирования, включая Эло и PageRank.

Одной из важнейших тем, не затронутых в этой главе, является изучение методов *ранжирования*, которые используют данные ранжирования по золотому стандарту для обучения соответствующим функциям оценки. Такие абсолютно правдивые данные обычно недоступны, но иногда можно найти прокси. При оценке поисковых систем свидетельство о том, что пользователь щелкнул (скажем) на четвертом представленном ему элементе, может быть истолковано как его голос о том, что данный элемент должен иметь более высокий рейтинг, чем три остальные. SVMrank [49] представляет метод изучения функций ранжирования по таким данным.

Эвристика, предложенная минимизацией краевых конфликтов в порядке вершин, описана Идсом и др. в публикации [50]. Моя презентация теоремы Эрроу о невозможности основана на заметках Уоткинса [51].

Случаи из жизни этой главы были взяты из моих книг *Calculated Bets* и *Who's Bigger?* Не судитесь со мной за самоплагиат.

4.9. Упражнения

Оценки и ранги

4.1. *[3]* Пусть X представляет случайную величину, взятую из нормального распределения, определенного как $\mu = 2$ и $\sigma = 3$. Предположим, что мы наблюдаем $X = 5{,}08$. Найдите Z-оценку x и определите, сколько стандартных отклонений от среднего составляет x.

4.2. *[3]* Какой процент от стандартного нормального распределения ($\mu = 0$, $\sigma = 1$) находится в каждом регионе?
(a) $Z > 1{,}13$.
(b) $Z < 0{,}18$.
(c) $Z > 8$.
(d) $|Z| < 0{,}5$.

4.3. *[3]* Аманда сдала тест GRE (Graduate Record Examination) и набрала 160 баллов по вербальному мышлению и 157 баллов по количественному. Средний балл за вербальное мышление составил 151 со стандартным отклонением 7 по сравнению со средним $\mu = 153$ и $\sigma = 7{,}67$ для количественного мышления. Предположим, что оба распределения нормальные.
(a) Каковы Z-оценки Аманды по этим экзаменационным дисциплинам? Отметьте эти оценки на стандартной кривой нормального распределения.
(b) В какой из дисциплин она оказалась лучше по сравнению с другими студентами?
(c) Найдите ее процентильные оценки для обоих экзаменов.

4.4. *[3]* Определите три успешные и хорошо используемые функции оценки в областях, которые вас интересуют. Для каждого объясните, что делает эту функцию хорошей оценкой и как ее используют другие.

4.5. *[5]* Найдите набор данных о свойствах одного из следующих классов.
(a) Страны мира.
(b) Кино и кинозвезды.
(c) Звезды спорта.
(d) Университеты.
Создайте подходящую функцию ранжирования, отражающую качество или популярность. Насколько хорошо она коррелирует с некой внешней мерой, нацеленной на аналогичный результат?

4.6. *[5]* Создайте две существенно разные, но подходящие функции оценки для одного и того же набора элементов. Насколько отличаются итоговые рейтинги? Свидетельствует ли тот факт, что обе функции должны разумно ограничивать ранжирование, о том, что они очень похожи?

4.7. *[3]* Системы оценки, используемые профессиональными спортивными лигами для выбора наиболее ценного игрока, обычно подразумевают присвоение позиционных весов перестановкам, указанным избирателями. Какие системы они используют в профессиональном бейсболе, баскетболе и футболе? Они похожи? Как, по-вашему, они корректны?

Реализация проектов

4.8. *[5]* Используйте рейтинги Эло для ранжирования всех команд в таком спорте как бейсбол, футбол или баскетбол, где рейтинг корректируют в зависимости от каждого нового результата игры. Насколько точно эти рейтинги Эло предсказывают результаты будущих игр?

4.9. *[5]* Оцените надежность метода Борда, применив k случайных перестановок к каждой из m различных копий перестановок $p = \{1, 2, \ldots, n\}$. Каков порог, после которого метод Борда не может восстановить p как функцию от n, k и m?

Вопросы на интервью

4.10. *[5]* Что делает набор данных золотым стандартом?

4.11. *[5]* Как можно проверить, работает ли новая модель оценки кредитного риска?

4.12. *[5]* Как бы вы прогнозировали продажи конкретной книги, основываясь на общедоступных данных Amazon?

Конкурсы Kaggle

4.13. Ранжировать шахматистов по игровым позициям.
`https://www.kaggle.com/c/chess`

4.14. Разработка системы оценки финансового кредита.
`https://www.kaggle.com/c/GiveMeSomeCredit`

4.15. Предсказать оплату за работу по ее объявлению.
`https://www.kaggle.com/c/job-salary-prediction`

Глава 5

Статистический анализ

Со статистикой лгать легко, но легче лгать без нее.

— Фредерик Мостеллер (Frederick Mosteller)

Признаюсь, у меня никогда не получалось по-настоящему удовлетворительного разговора со статистиком. И это совсем не из-за того, что я не пытался. Несколько раз за эти годы у меня возникали проблемы, представляющие интерес для статистиков, но я всегда возвращался с ответами типа "Вы не можете сделать это таким образом" или "Но это не является независимым" вместо того, чтобы услышать "Вы можете справиться с этим вот как".

По правде говоря, эти статистики, как правило, тоже не любят общаться со мной. Статистики всерьез думают о данных гораздо дольше, чем программисты, и у них есть множество мощных методов и идей, чтобы доказать это. В этой главе я представлю некоторые из этих важных инструментов, включая определения нескольких фундаментальных распределений и проверки на статистическую значимость. В этой главе также будет представлен Байесовский анализ — способ, позволяющий точно оценить, как новые данные должны влиять на наши предыдущие оценки будущих событий.

Процесс статистического рассуждения приведен на рис. 5.1. Существует множество возможных вещей, которые мы можем наблюдать. Фактически из них выбирается лишь относительно небольшое подмножество, в идеале случайным образом, а значит, мы можем наблюдать свойства лишь выбранных элементов. Теория вероятностей описывает, какими свойствами должна обладать наша выборка, чтобы учитывать свойства базовой совокупности. Но *статистический вывод* (statistical inference) работает иначе, он пытается определить, какова общая совокупность, исходя из анализа выборки.

В идеале мы научимся мыслить как статистик: т.е. оставаться достаточно бдительным и защищенным от чрезмерной интерпретации и ошибок, сохраняя при этом уверенность в том, что с данными можно играть и использовать их в том направлении, в котором они нас ведут.

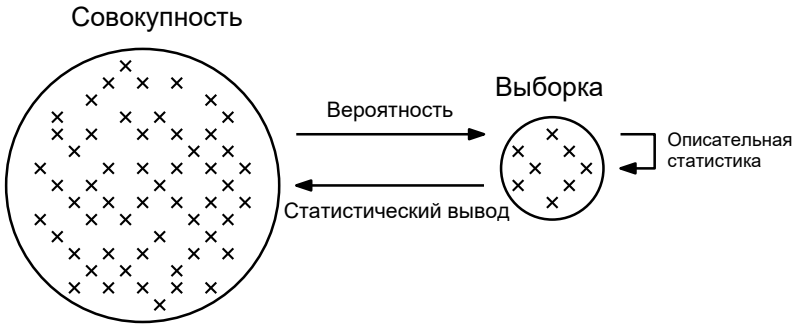


Рис. 5.1. Центральная догма статистики: анализ небольшой случайной выборки позволяет сделать точные выводы о всей совокупности

5.1. Статистические распределения

Каждая переменная, которую мы наблюдаем, определяет конкретную плотность распределения, отражающую частоту возникновения каждого конкретного значения. Уникальные свойства переменных, таких как рост, вес и IQ, фиксируются их распределением. Но формы этих распределений сами по себе не уникальны: огромное разнообразие всех данных в мире подчиняется распределениям лишь нескольких классических форм.

Эти классические распределения имеют два приятных свойства: (1) они описывают формы плотностей распределений, которые часто встречаются на практике, и (2) их нередко можно описать математически с помощью выражений в замкнутой форме при очень небольшом количестве параметров. Будучи абстрагированы от конкретных данных наблюдений, они становятся *распределениями вероятностей* (probability distribution), заслуживающими независимого изучения.

Знание классических распределений вероятностей важно, поскольку они нередко встречаются на практике. Они дают нам словарный запас для разговора о том, как выглядят наши данные. В следующих разделах мы рассмотрим наиболее важные статистические распределения (биномиальное, нормальное, пуассоновское и по степенному закону), подчеркнув свойства, определяющие их характер.

Обратите внимание на то, что ваши наблюдаемые данные не обязательно являются результатом определенного теоретического распределения только потому, что их форма похожа. Для строгого доказательства того, что ваши экспериментально наблюдаемые данные выборки отражают определенное распределение, могут использоваться статистические проверки.

Но я собираюсь избавить вас от необходимости выполнения любой из этих проверок. Я с уверенностью скажу, что ваши реальные данные *не соответствуют* в точности ни одному из известных теоретических распределений.

Почему? Поймите, что мир — это сложное место, что делает его измерение беспорядочным процессом. Ваши наблюдения, вероятно, будут основаны на нескольких выборочных совокупностях, каждая из которых имеет несколько иное базовое распределение. В хвостах любого наблюдаемого распределения обычно происходит нечто странное: внезапный всплеск необычно высоких или низких значений. Связанные с ними измерения будут иметь ошибки, возникающие иногда довольно странными систематическими способами.

5.1.1. Биномиальное распределение

Рассмотрим эксперимент, состоящий из идентичных независимых испытаний, которые имеют два возможных исхода P_1 и P_2 с соответствующими вероятностями p и $q = (1 - p)$. Возможно, ваш эксперимент заключается в подбрасывании монеты, где вероятность орлов ($p = 0{,}5$) точно такая же, как и решек ($q = 0{,}5$). Возможно, это многократное включение и выключение света, когда вероятность внезапного обнаружения необходимости замены лампы ($p = 0{,}001$) намного меньше, чем вероятность увидеть свет ($q = 0{,}999$).

Биномиальное распределение (binomial distribution) демонстрирует вероятности получения именно x событий P_1 в ходе n независимых испытаний в произвольном порядке. Здесь важна независимость: мы предполагаем, что вероятность выхода из строя лампы не имеет отношения к тому, сколько раз она использовалась ранее. Для биномиального распределения pdf определяется так:

$$P(X = x) = \binom{n}{x} p^x (1 - p)^{(n-x)}.$$

В отношении биномиального распределения можно отметить следующее.

- *Оно дискретно.* Оба аргумента биномиального распределения (n и x) должны быть целыми числами. Гладкость графика на рис. 5.2 (слева) — это иллюзия, поскольку значение $n = 200$ довольно велико. Невозможно получить 101,25 орлов при 200 бросках монеты.
- *Вы, вероятно, можете объяснить теорию, лежащую в его основе.* Вы впервые встретились с биномиальным распределением в старших классах школы. Помните треугольник Паскаля? Вероятность получить в итоге точно x орлов при n бросках в определенной последовательности составляет $p^x (1 - p)^{n-x}$ для каждой из $\binom{n}{x}$ отдельных последовательностей бросков.
- *Оно имеет форму в виде своего рода колокола.* Для идеальной монеты ($p = 0{,}5$) биномиальное распределение является совершенно симметричным

со средним значением в середине. В случае с лампочкой это не так: если мы включим лампу только $n = 1000$ раз, количество отказов, вероятней всего, будет равно нулю. Это звон только в половину колокола на рис. 5.2. Тем не менее при $n \to \infty$ мы получим симметричное распределение с пиком в середине.

- *Оно определяется с использованием только двух параметров.* Все, что нам нужно для полного определения данного биномиального распределения, — это значения p и n.

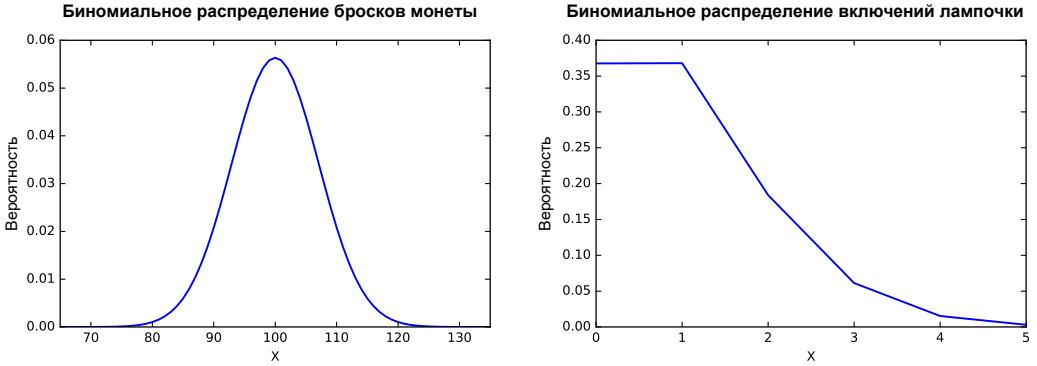


Рис. 5.2. Биномиальное распределение может использоваться для моделирования распределения орлов в 200 бросках монеты при $p = 0,5$ (слева) и количества перегоревших лампочек в 1000 событиях с вероятностью отказа $p = 0,001$ (справа)

Биномиальное распределение позволяет корректно моделировать многие вещи. Напомним, что дисперсия результативности $p = 0,300$ хитера обсуждалась в разделе 2.2.3. Там вероятность отбоя в каждом испытании была $p = 0,3$ при $n = 500$ испытаний за сезон. Таким образом, количество отбоев за сезон берется из биномиального распределения.

Понимание того, что это биномиальное распределение, означало, что нам действительно не нужно было использовать симуляцию для построения распределения. Такие свойства, как ожидаемое количество отбоев $\mu = np = 500 \times 0,3 = 150$ и его стандартное отклонение $\sigma = \sqrt{npq} = \sqrt{500 \cdot 0,3 \cdot 0,7} = 10,25$, просто происходят из формул в замкнутой форме, которые можно при необходимости найти.

5.1.2. Нормальное распределение

Колоколообразные кривые позволяют моделировать многие природные явления. Измеренные характеристики, такие как рост, вес, продолжительность жизни и IQ, соответствуют одной и той же базовой схеме: большая часть значений лежит довольно близко к среднему значению, распределение симметрично,

и никакое значение не является слишком экстремальным. За всю историю мира никогда не было ни 12-футового (3,6576 м) мужчины, ни 140-летней женщины.

Основой всех колоколообразных кривых является *гауссово*, или *нормальное распределение* (normal distribution), которое полностью параметризируется его средним значением и стандартным отклонением:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}.$$

На рис. 5.3 показаны pdf и cdf нормального распределения. Здесь имеет смысл обратить внимание на следующее.

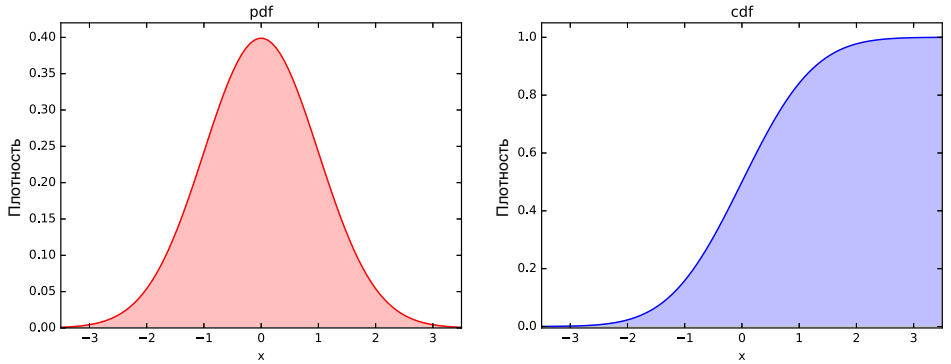


Рис. 5.3. Функция плотности вероятностей (pdf) нормального распределения (слева) с соответствующей кумулятивной функцией плотности (cdf) справа

- *Оно непрерывно.* Аргументы для нормального распределения (среднее значение μ и стандартное отклонение σ) вполне могут быть вещественными числами с единственным ограничением, $\sigma > 0$.
- *Вы, вероятно, не можете объяснить его происхождение.* Нормальное распределение является обобщением биномиального распределения, где $n \to \infty$, а степень концентрации вокруг среднего значения определяется параметром σ. Примените свою интуицию к биномиальному распределению и поверьте, что Гаусс правильно сделал свои вычисления: великий математик разработал нормальное распределение для своей докторской диссертации. Или обратитесь к любой приличной книге по статистике, если вам действительно интересно узнать, откуда оно взялось.
- *Оно действительно имеет форму колокола.* Гауссово распределение является идеальным примером колоколообразной кривой. Поскольку оно работает с непрерывной переменной (например, с ростом), а не с дискретным подсчетом (скажем, количеством событий), его график совершенно гладкий. Поскольку оно продолжается бесконечно в обоих

направлениях, усечения хвостов с обеих сторон не происходит. Нормальное распределение — это теоретическая конструкция, которая помогает объяснить это совершенство.

- *Оно также определяется с использованием только двух параметров.* Однако эти параметры отличны от биномиального распределения! Нормальное распределение полностью определяется его центральной точкой (определяется как среднее значение μ) и его разбросом (определяется стандартным отклонением σ). Это единственные кнопки, которые мы можем использовать для настройки распределения.

Что значит нормальное?

Нормальное распределение моделирует удивительное количество природных явлений. Возможно, наиболее важным является ошибка измерения. Каждый раз, измеряя свой вес на весах в ванной, вы получаете несколько иной ответ, даже если ваш вес не изменился. Иногда показания будут выше, а иногда ниже, в зависимости от температуры в комнате и кривизны пола. Маленькие ошибки более вероятны, чем большие, и довольно высокая вероятность так же вероятна, как и небольшая. Экспериментальные ошибки обычно распределяется как *гауссовский шум* (Gaussian noise).

Такие физические явления, как рост, вес и продолжительность жизни, имеют распределение в форме колокола по аналогичным аргументам. Тем не менее утверждение о том, что такие распределения являются *нормальными*, обычно является слишком поспешным, без точного указания базовой совокупности. Распределен ли рост человека нормально? Конечно нет: у мужчин и женщин разные показатели среднего роста и связанные с ними распределения. Распространен ли нормально рост мужчин? Конечно нет: при включении в группу детей и сгорбленных пожилых людей вы снова получаете смесь несколько разных базовых распределений. Является ли рост взрослых мужчин в Соединенных Штатах нормальным? Определенно нет. Есть нетривиальные популяции с нарушениями роста, такими как карликовость и акромегалия, которые оставляют группы людей значительно ниже и выше, чем можно объяснить нормальным распределением.

Возможно, наиболее известным колоколообразным, но ненормальным распределением является распределение ежедневной доходности (процентное смещение цен) на финансовых рынках. Большое падение рынка определяется большим процентным падением цен: 10 октября 1987 года индекс Доу–Джонса в среднем потерял 22,61% своей стоимости. Большие обвалы фондового рынка происходят с гораздо большей частотой, чем это можно точно смоделировать с помощью нормального распределения. Действительно, каждое существенное падение рынка приводит к увольнению определенного количества

специалистов по количественному анализу, которые полагали положение нормальным и недостаточно застраховались от таких экстремальных явлений. Оказывается, что *логарифм* доходности акций оказывается нормально распределенным, что приводит к распределению с более толстыми хвостами, чем обычно.

Хотя мы должны помнить, что колоколообразные распределения не всегда нормальны, такое предположение является вполне разумным для начала, в отсутствие лучшего понимания.

5.1.3. Значения нормального распределения

Напомним, что любое распределение всегда приблизительно характеризуют среднее значение и стандартное отклонение, как обсуждалось в разделе 2.2.4. Но они отлично справляются с характеристикой нормального распределения, поскольку они и *определяют* нормальное распределение.

Рис 5.4 иллюстрирует известное правило нормального распределения 68%–95%–99%. Шестьдесят восемь процентов вероятностной массы должны находиться в пределах $\pm 1\sigma$ от среднего значения. Далее, 95% вероятности находится в пределах 2σ, а 99,7% в пределах 3σ.

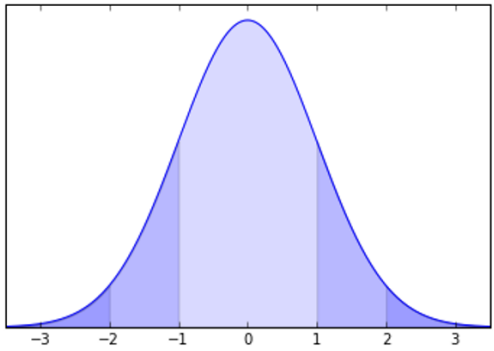


Рис. 5.4. Нормальное распределение подразумевает жесткие границы вероятности нахождения вдалеке от среднего. 68% значений должны лежать в пределах одной сигмы от среднего значения, 95% в пределах 2σ и 99,7% — 3σ

Следовательно, значения, далекие от среднего (в терминах σ), исчезающе редки в любой нормально распределенной переменной. Действительно, термин *шесть сигм* (six sigma) используется для обозначения столь высоких стандартов качества, что дефекты являются невероятно редкими событиями. Мы хотим, чтобы авиакатастрофы были событиями шести сигм. Вероятность события 6σ при нормальном распределении составляет примерно 2 на миллиард.

Интеллект, измеряемый по шкале IQ, обычно распределяется со средним значением 100 и стандартным отклонением $\sigma = 15$. Таким образом, 95% населения находится в пределах 2σ от среднего значения, т.е. от 70 до 130. Это оставляет только 2,5% людей с IQ выше 130, и еще 2,5% с IQ ниже 70. В общей сложности 99,7% общей массы находится в пределах 3σ от среднего значения, т.е. людей с IQ от 55 до 145.

Итак, насколько умен самый умный человек в мире? Если мы предположим, что население составляет 7 миллиардов человек, вероятность того, что случайно выбранный человек окажется самым умным, составляет приблизительно $1{,}43 \cdot 10^{-10}$. Это примерно одинаковая вероятность с тем, что одна выборка будет лежать в более чем $6{,}5\sigma$ от среднего значения. Таким образом, согласно этому расчету самый умный человек в мире должен иметь IQ приблизительно 197,5.

Степень, в которой вы согласны с этим, зависит от того, насколько сильно вы верите, что IQ действительно распределен нормально. Такие модели обычно имеют серьезную вероятность оказаться ошибочными в крайних случаях. Действительно, по этой модели существует вероятность того, что кто-то окажется достаточно глупым, чтобы заработать отрицательный балл по тесту IQ.

5.1.4. Распределение Пуассона

Распределение Пуассона (Poisson distribution) измеряет плотность интервалов между редкими событиями. Предположим, что мы моделируем продолжительность жизни человека по последовательности ежедневных событий, где существует небольшая, но постоянная вероятность $1 - p$ того, что человек перестает дышать сегодня. Продолжительность жизни ровно n дней означает успешное дыхание на протяжении каждого из первых $n - 1$ дней, а затем навсегда нарушает схему в n-й день. Вероятность прожить ровно n дней составляет $Pr(n) = p^{n-1}(1-p)$, давая ожидаемую продолжительность жизни

$$\mu = \sum_{k=o}^{\infty} k\,Pr(k).$$

Распределение Пуассона в основном проистекает из этого анализа, но получает более удобный аргумент, чем p. Вместо этого он основан на μ, среднем значении распределения. Поскольку каждый p определяет конкретное значение μ, эти параметры в некотором смысле эквивалентны, но среднее значение гораздо проще оценить или измерить. Распределение Пуассона дает очень простую замкнутую форму:

$$Pr(x) = \frac{e^{-\mu}\mu^{x}}{x!}.$$

Как только вы начинаете думать правильно, многие распределения начинают выглядеть, как распределение Пуассона, поскольку они представляют интервалы между редкими событиями.

Вспомните модель лампочки с биномиальным распределением из предыдущего раздела. Это позволило легко рассчитать ожидаемое количество изменений на рис. 5.2 (справа), но не распределение продолжительности жизни, которым является распределение Пуассон. На рис. 5.5 приведено соответствующее распределение Пуассона для $\mu = 1/p = 1000$, которое демонстрирует, что мы должны ожидать, что почти все лампочки будут светиться в течение от 900 до 1100 часов перед тем, как свет погаснет.

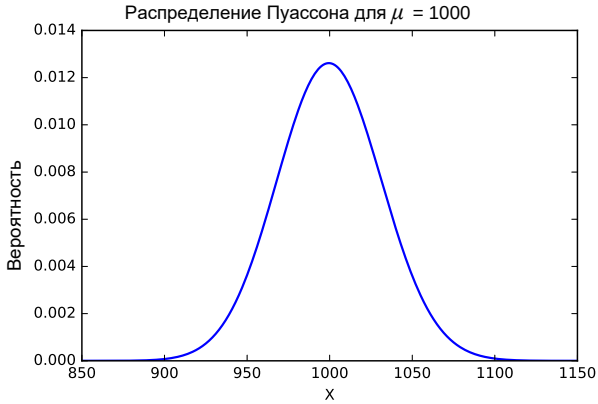


Рис. 5.5. Распределение срока службы лампочек с ожидаемым сроком службы $\mu = 1000$ часов согласно модели распределения Пуассона

В качестве альтернативы предположим, что мы моделируем количество детей в процессе, когда семья продолжает рожать детей до тех пор, пока после одной крупной истерики, распродажи выпечки или стирки белья родители наконец не сломаются: "*Это все! На этом достаточно. Больше никогда!*"

При такой модели размер семьи должен моделироваться как распределение Пуассона, где каждый день существует небольшая, но и ненулевая вероятность срыва, приводящего к закрытию фабрики. Достаточно ли хорошо работает модель "*На этом достаточно*", чтобы предсказать размер семьи? Ломанная линия на рис. 5.6 представляет распределение Пуассона с параметром $\lambda = 2{,}2$, что означает, что семьи имеют в среднем 2,2 ребенка. Точки представляют долю семей с k детьми, взятую из результатов исследования General Social Survey (GSS) в США за 2010 год.

Здесь превосходное совпадение по всем размерам семей, кроме $k = 1$, и, честно говоря, мой личный опыт подсказывает, что один ребенок в семье встречается куда чаще, чем демонстрирует этот набор данных. Тем не менее, зная только среднее значение и формулу распределения Пуассона, мы можем построить разумную оценку распределения реального размера семьи.

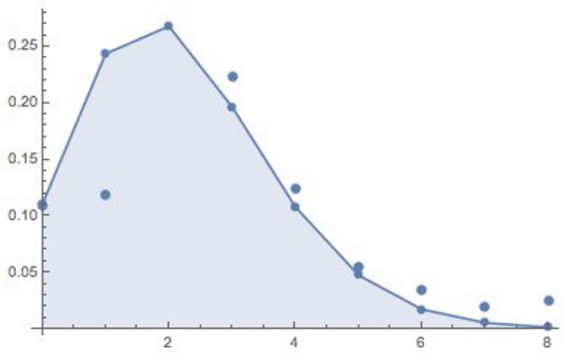


Рис. 5.6. Наблюдаемая доля семей с x детьми (отдельные точки) точно моделируется распределением Пуассона, определяемым средним значением $\mu = 2{,}2$ ребенка на семью (линия)

5.1.5. Распределение по степенному закону

Многие распределения данных имеют гораздо более длинные хвосты, чем это возможно при нормальном распределении или распределении Пуассона. Рассмотрим, например, население городов. По данным Википедии, в 2014 году в США было ровно 297 городов с населением более 100 000 человек. Население k-го по величине города для $1 \leq k \leq 297$ представлено на рис. 5.7 (слева).

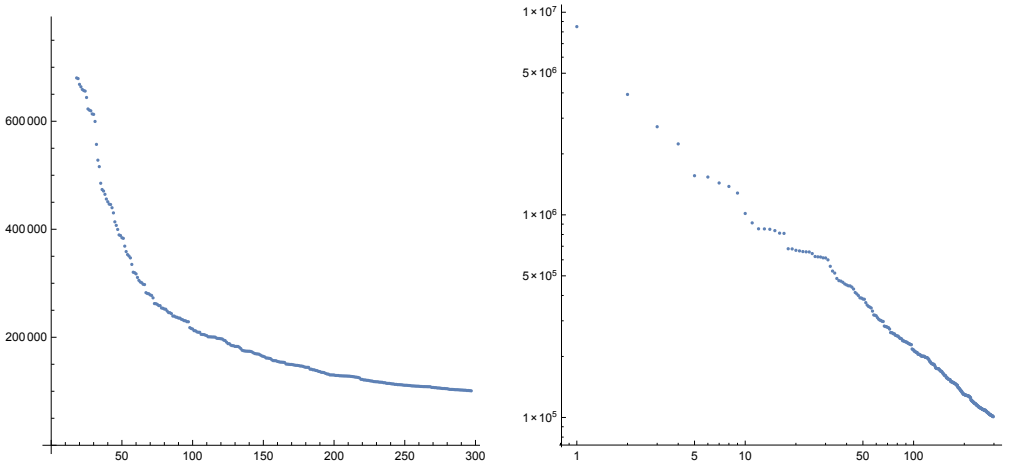


Рис. 5.7. Население городов США по убыванию ранга (слева). Справа — те же данные, но в логарифмическом масштабе. Теперь включены и самые большие города. То, что они выстроились в линию, свидетельствует о распределении по степенному закону

Это демонстрирует, что в сравнительно небольшом количестве городов население существенно доминирует над остальными. В семнадцати крупнейших городах население настолько велико, что они были удалены с этого участка графика, чтобы мы могли увидеть остальные.

В этих городах проживает в среднем 304 689 человек, а ужасающее стандартное отклонение составляет 599 916 человек. Когда стандартное отклонение настолько велико по отношению к среднему значению, что-то не так. При нормальном распределении 99,7% массы находятся в пределах 3σ от среднего значения, что делает маловероятным проживание в каком-либо из этих городов более 2,1 млн человек. Тем не менее население Хьюстона составляет 2,2 млн человек, а Нью-Йорк (8,4 млн человек) более чем на 13σ выше среднего! Население городов явно *не* распределено нормально. На самом деле оно соответствует другому распределению, *распределению по степенному закону* (power law).

Для данной переменной X, определенной степенным законом распределения,

$$P(X = x) = cx^{-\alpha}.$$

Оно параметризуется двумя константами: экспонентой α и константой нормализации c.

Распределение по степенному закону требует больше объяснений для понимания. Общая вероятность, определяемая этим распределением, представляет собой площадь под кривой:

$$A = \int_{x=-\infty}^{\infty} cx^{-\alpha} = c \int_{x=-\infty}^{\infty} x^{-\alpha}.$$

Конкретное значение A определяется параметрами α и c. Константа нормализации c выбирается специально для данного α, чтобы гарантировать, что $A = 1$, как того требуют законы вероятности. Кроме этого, c не имеет для нас особого значения.

Реальное действие происходит с α. Обратите внимание, что, когда мы удваиваем входное значение (с x до $2x$), мы уменьшаем вероятность в порядке $f = 2^{-\alpha}$. Это выглядит плохо, но для любого данного α это просто константа. Итак, что на самом деле гласит степенной закон, так это то, что вероятность события размером $2x$ в 2α раз меньше, чем в случае с событием размером x, для всех x.

Степенным законом хорошо моделируется личное богатство, где $f \approx 0{,}2 = 1/5$. Это означает, что в широком диапазоне, если у Z людей есть x долларов, то у $Z = 5$ человек есть $2x$ долларов. Пятая часть людей имеет скорее 200 000 долл., чем 100 000 долл. Если 625 человек в мире имеет состояние в 5 млрд долл., то примерно 125 мультимиллиардеров должны стоить по 10 млрд долл. каждый. Кроме того, должно быть 25 супермиллиардеров, каждый на сумму 20 млрд долл., пять гипермиллиардеров на уровне 40 млрд долл. и, наконец, один Билл Гейтс с 80 млрд долл.

Степенной закон определяет правило "80/20", описывающее все неравенство в нашем мире: согласно наблюдениям, 20% первых A получают 80% величины B. Степенной закон имеет тенденцию возникать всякий раз, когда богатые становятся богаче, и вероятность того, что вы получите больше, зависит от того, что у вас уже есть. Большие города становятся непропорционально большими потому, что большие города привлекают больше людей. Из-за своего богатства Билл Гейтс получает доступ к гораздо лучшим инвестиционным возможностям, чем я, поэтому его деньги растут быстрее, чем мои.

Многие распределения определяются такими моделями преимущественного роста.

- *Интернет-сайты с x пользователями.* Сайты становятся все более популярными по мере увеличения количества пользователей. У вас больше шансов присоединиться к Instagram или Facebook, поскольку ваши друзья уже присоединились к ним. Предпочтительное присоединение (preferential attachment) приводит к распределению по степенному закону.
- *Слова, используемые с относительной частотой x.* Существует длинный хвост миллионов слов, таких как *algorist* или *defenestrate*[1], которые редко используются в английском языке. С другой стороны, небольшой набор таких слов, как *the*, используется значительно чаще, чем остальные. *Закон Ципфа* (Zipf's law) регулирует распределение использования слов в естественных языках и гласит, что частота использования k-го по популярности (согласно частотному ранжированию) слова составляет $1/k$ частоты использования самого популярного слова.[2] Чтобы оценить, насколько хорошо это работает, рассмотрим распределение слов по частоте их упоминания в английской Википедии.

Ранг	Слово	Количество	Ранг	Слово	Количество	Ранг	Слово	Количество
1	the	25131726	1017	build	41890	10021	glances	2767
110	even	415055	2017	essential	21803	20026	ecclesiastical	881
212	men	177630	3018	sounds	13867	30028	zero-sum	405
312	least	132652	4018	boards	9811	40029	excluded	218
412	police	99926	5018	rage	7385	50030	sympathizes	124
514	quite	79205	6019	occupied	5813	60034	capon	77
614	include	65764	7020	continually	4650	70023	fibs	49
714	knowledge	57974	8020	delay	3835	80039	conventionalized	33
816	set	50862	9021	delayed	3233	90079	grandmom	23
916	doctor	46091	10021	glances	2767	100033	slum-dwellers	17

[1] Defenestrate означает "выгнать кого-то с работы", т.е. уволить в резкой форме.

[2] См. лучше `https://ru.wikipedia.org/wiki/Закон_Ципфа`. — *Примеч. ред.*

Как можно заметить, частота использования быстро падает с ростом ранга: напомним, что *grandmom* (бабушка) — это всего лишь сленговая форма слова *grandma*, а не *the real McCoy*.

Почему это степенной закон? Слово ранга $2x$ имеет частоту $F_{2x} \sim F_1/2x$, по сравнению с $F_x \sim F_1/x$. Таким образом, удвоение ранга удваивает частоту, и это соответствует степенному закону с $\alpha = 1$.

Каков механизм развития языков, приводящий к такому распределению? Наиболее правдоподобное объяснение в том, что люди учат и используют слова так, как они слышат их от других людей. Любой механизм, который благоприятствует уже популярному, ведет к степенному закону.

- *Частота землетрясений магнитудой x.* Шкала Рихтера для измерения силы землетрясений является логарифмической, а значит, землетрясение магнитудой 5,3 в десять раз сильнее, чем событие 4,3 по шкале. Добавление единицы к величине умножает силу в десять раз.

  Такой быстрый рост шкалы свидетельствует, что более крупные события встречаются реже, чем мелкие. Я вызываю землетрясение магнитудой 0,02 каждый раз, когда сливаю воду в туалете. На самом деле такие события случаются миллиардами каждый день, но большие землетрясения становятся все более редкими. Всякий раз, когда количество растет потенциально неограниченным образом, а его вероятность экспоненциально уменьшается, вы получаете степенной закон. Данные показывают, что это так же верно и в отношении энергии, выделяемой землетрясениями, и в случае жертв войн: к счастью, число конфликтов, убивающих x людей, уменьшается согласно степенному закону.

Для распознавания распределений по степенному закону учитесь держать глаза открытыми. Вы найдете их повсюду в нашем несправедливом мире. Они выявляются по следующим свойствам.

- *Степенные законы демонстрируют прямые линии на логарифмических графиках.* Посмотрите на график численности населения городов на рис. 5.7 (справа). Хотя на краях есть некоторые промежутки, где данных становится мало, в общем и целом точки аккуратно лежат на одной линии. Это *главная* характеристика степенного закона. Кстати, наклон этой линии определяется α, постоянной, определяющей форму распределения степенного закона.
- *Среднее значение не имеет смысла.* Один Билл Гейтс добавляет около 250 долл. к состоянию среднего человека в Соединенных Штатах. Это странно. При распределении по степенному закону существует очень малая, но ненулевая вероятность того, что кто-то будет иметь бесконечно богатое состояние, так что же это значит? Медиана намного лучше

справляется с описанием большинства таких распределений, чем наблюдаемое среднее.

- *Стандартное отклонение не имеет смысла.* При распределении по степенному закону стандартное отклонение обычно больше или равно среднему значению. Это значит, что распределение очень плохо характеризуется по μ и σ, в то время как степенной закон дает очень хорошее описание в терминах α и c.
- *Масштаб распределения неизменен.* Предположим, что мы нанесли на карту население 300–600 крупнейших городов США, а не 300 лучших, как на рис. 5.7 (слева). Форма выглядела бы почти так же, население 300 по величине города просто поднялось бы над хвостом. Любая экспоненциальная функция является *масштабно-инвариантной* (scale invariant), потому что она выглядит одинаково при любом разрешении. Это является следствием того, что оно представляет собой прямую линию на логарифмическом графике: любой поддиапазон является сегментом прямой линии, параметры которой в своем окне имеют те же параметры, что и полное распределение.

На заметку. Оставайтесь в поисках распределений по степенному закону. Они отражают мировое неравенство, а значит, они повсюду.

5.2. Выборка из распределений

Выборка точек из данного распределения вероятностей — это обычная практика, с которой стоит уметь справляться. Возможно, вам нужны тестовые данные из распределения по степенному закону, чтобы запустить симуляцию или убедиться, что ваша программа работает в экстремальных условиях. Для проверки того, соответствуют ли ваши данные конкретному распределению, нужно что-то сравнивать, и обычно это должны быть правильно сформированные синтетические данные, взятые из канонического распределения.

Существует общая методика выборки из любого заданного распределения вероятности — *выборка с обратным преобразованием* (inverse transform sampling). Напомним, что, применяя интегрирование и дифференцирование, мы можем перемещаться между функцией плотности вероятностей P и кумулятивной функцией плотности C. Мы можем перемещаться между ними потому, что:

$$P(k = X) = C'(k) = C(X \le k + \delta) - C(X \le k);$$

$$C(X \le k) = \int_{x=-\infty}^{k} P(X = x).$$

Предположим, что я хочу взять точку из этого, возможно, очень сложного распределения. Я могу использовать однородный генератор случайных чисел, чтобы выбрать значение p в интервале [0, …, 1]. Мы можем интерпретировать p как вероятность и использовать ее в качестве индекса кумулятивного распределения C. Точнее, мы сообщаем конкретное значение x, такое как $C(X \leq x) = p$.

Подход приведен на рис. 5.8. Это выборка из нормального распределения. Предположим, что p = 0,729 — это случайное число, созданное нашим однородным генератором. Мы возвращаем значение x таким образом, что y = 0,729, поэтому x = 0,62 согласно этому cdf.

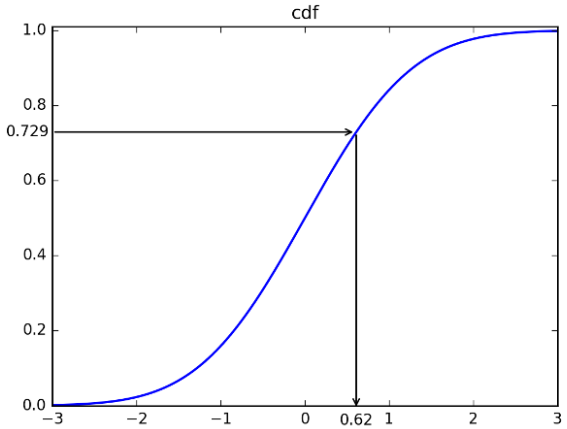


Рис. 5.8. Метод выборки с обратным преобразованием позволяет нам преобразовать однородное случайное число из диапазона [0, 1] (здесь 0,729) в случайную выборку, взятую из любого распределения, с учетом его cdf

Если вы работаете с популярным распределением вероятностей на хорошо поддерживаемом языке, таком как Python, в его библиотеке почти наверняка есть функция для создания случайных выборок. Так что ищите подходящую библиотеку, прежде чем писать свою.

5.2.1. Случайная выборка вне одного измерения

Правильная выборка из заданного распределения становится очень сложной проблемой, когда вы увеличиваете количество измерений. Рассмотрим задачу равномерной выборки точек из круга. Подумайте немного о том, как вы могли бы сделать это, прежде чем мы продолжим.

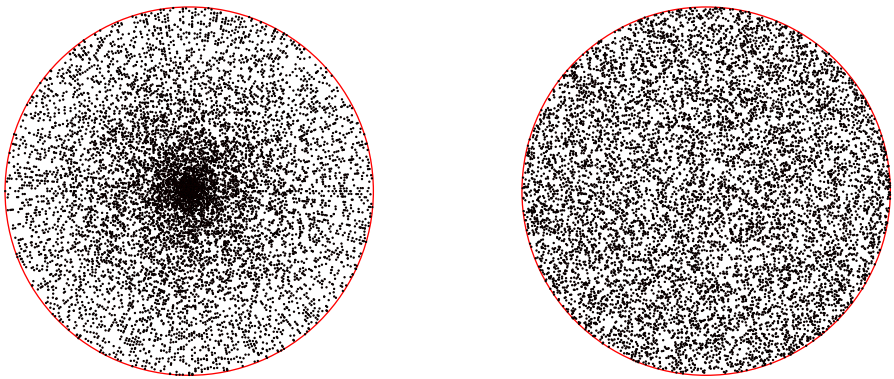


Рис. 5.9. Произвольная генерация 10 000 точек по парам "угол-радиус" явно сосредоточивается вблизи центра круга (слева). Выборка Монте-Карло, напротив, генерирует точки в пределах круга равномерно (справа)

Самый умный из вас может придти к идее выборки угла и расстояния от центра независимо. Угол, который должна иметь любая точка выборки относительно начала координат и положительной оси x, варьируется от 0 до 2π. Расстояние от начала координат должно быть значением от 0 до r. Выбирайте эти координаты случайным образом равномерно, и у вас будет случайная точка в круге.

Этот подход разумный, но неправильный. Конечно, любая точка, созданная таким образом, должна находиться внутри круга. Но точки выбираются не с одинаковой частотой. Этот метод будет генерировать точки, но половина из них будет лежать на расстоянии не более $r/2$ от центра. Однако большая часть площади круга находится дальше от центра! Таким образом, мы сделаем выборку около начала координат в ущерб массе вблизи границ. Это показано на рис. 5.9 (слева), где приведен график из 10 000 точек, полученных с помощью этого метода.

Правильной техникой, оказывается, является *выборка Монте-Карло* (Monte Carlo sampling). Координаты x и y каждой точки в круге варьируются от $-r$ до r, как и у многих точек вне круга. Таким образом, выборка этих значений равномерно случайным образом дает нам точку, которая лежит в ограничивающей рамке круга, но не всегда внутри самой окружности. Это легко проверить: расстояние от (x, y) до начала координат должно быть не больше r, т.е. равно $\sqrt{x^2 + y^2} \leq r$. Если это так, то мы нашли случайную точку в круге. Если нет, мы отбрасываем ее и пробуем снова. На рис. 5.9 (справа) показаны 10 000 точек, созданных с использованием этого метода: посмотрите, насколько равномерно они заполняют круг, без каких-либо явных мест чрезмерной или недостаточной выборки.

Эффективность здесь полностью зависит от отношения объема требуемой области (площадь круга) к объему ограничивающего прямоугольника (площадь квадрата). Поскольку 78,5% этого ограниченного прямоугольника занято кругом, в среднем достаточно менее двух попыток, чтобы найти каждую новую точку круга.

5.3. Статистическая значимость

Статистиков серьезно заботит степень значимости наблюдаемых данных. Компьютерный анализ легко найдет множество шаблонов и корреляций в любом интересном наборе данных. Но отражает ли конкретная корреляция реальные явления, а не просто случайные совпадения? Другими словами, когда наблюдение действительно *значимо* (significant)?

Достаточно сильные корреляции в больших наборах данных могут показаться "вполне очевидно" значимыми, но зачастую бывают едва уловимые проблемы. С одной стороны, *корреляция не подразумевает причинно-следственной связи*. Изображение на рис. 5.10 убедительно демонстрирует, что объем глубоких исследований в области компьютерных наук коррелирует с количеством видеоигр. Мне хотелось бы надеяться, что я привел к алгоритмам больше людей, чем Nintendo, но, может быть, это одно и то же? Графики таких ложных корреляций буквально заполняют книгу [52], причем очень забавную.

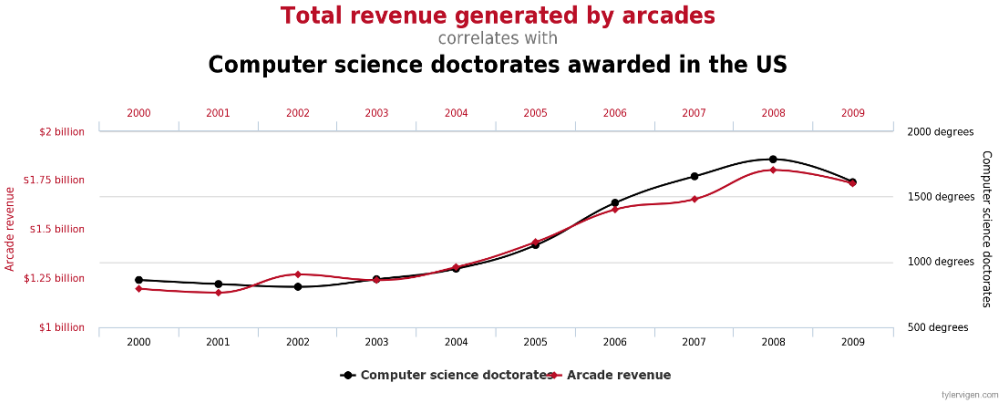


Рис. 5.10. Корреляция и причинно-следственная связь: количество ежегодно присуждаемых в Соединенных Штатах званий докторов компьютерных наук точно коррелирует с доходами от видеоигр и аркад (из [52])

Статистическая дисциплина вступает в свои права, позволяя выявить едва уловимые различия, согласно которым наблюдение является значимым или нет. Классический пример, определение эффективности медикаментозного

лечения, взят из медицинской статистики. Фармацевтическая компания проводит эксперимент, сравнивая два препарата. Препарат *A* вылечил 19 пациентов из 34. Препарат *B* вылечил 14 пациентов из 21. Действительно ли препарат *B* лучше, чем препарат *A*? Утверждение FDA нового лекарства может добавить, а может вычесть миллиарды из состояния фармацевтических компаний. Но можете ли вы быть уверены, что новый препарат обеспечивает реальное улучшение? Как по-вашему?

5.3.1. Значение значимости

Статистическая значимость — это мера нашей уверенности в том, что между двумя распределениями данных существует подлинная разница. Это важно. Но статистическая значимость не измеряет важность или величину этой разницы. Для достаточно больших размеров выборки, чрезвычайно малые различия могут быть очень значимы в статистических тестах.

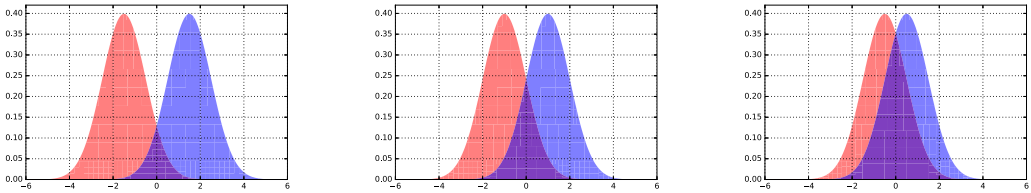


Рис. 5.11. Пары нормальных распределений с одинаковой дисперсией, но уменьшающейся разницей в их средних значениях слева направо. Чем ближе средние значения распределений, тем больше их совпадение

Предположим, например, что меня дурят при ставках на монету, которая выпадает орлом в 51% случаев вместо 50%, как ожидается от идеальной монеты. После 100 бросков идеальной монеты я ожидаю увидеть 51% или более орлов в 46,02% случаев, поэтому у меня нет абсолютно никаких оснований для жалоб, когда я делаю ставку. После 1000 бросков вероятность увидеть как минимум 510 орлов падает до 0,274. К 10 000 броскам вероятность увидеть столько орлов составляет всего 0,0233, и я должен начать подозревать монету. После 100 000 бросков вероятность честности снизится до $1{,}29 \cdot 10^{10}$, т.е. она настолько мала, что мне придется подать официальную жалобу, даже если я считаю своего соперника джентльменом.

Но вот в чем дело. Хотя теперь совершенно ясно, что меня обманули с помощью несимметричной монеты, последствия этого факта невелики. Почти любую проблему в жизни можно рассматривать с разных сторон, я предпочитаю позитивный взгляд, поскольку ставки просто недостаточно высоки. При ставке в 1 долл. за бросок моя ожидаемая потеря даже после 100 000 бросков составила бы только 1000 долл.

Значимость свидетельствует о том, насколько маловероятно нечто связанное с случайностью, но не его важность. Нас действительно заботит *величина эффекта* (effect size), степень различий между двумя группами. Мы неофициально классифицируем величину эффекта *среднего* уровня как заметную невооруженным глазом при внимательном наблюдении. В этом масштабе выявляются большие эффекты, а *малые* эффекты не являются полностью тривиальными [53]. Есть несколько статистических показателей, которыми пытаются измерять величину эффекта.

- *Стандартизированная величина эффекта по Коуэну d* (Cohen's d). Важность различия между двумя средними значениями μ и μ' зависит от абсолютной величины изменения, а также от естественного изменения распределений, измеряемого с помощью σ или σ'. Эту величину эффекта можно измерить как:

  $$d = (|\mu - \mu'|)/\sigma.$$

  Разумный порог для небольшой величины эффекта > 0,2, средней величины эффекта > 0,5 и большой величины эффекта > 0,8.

- *Коэффициент корреляции Пирсона r*. Измеряет степень линейной взаимосвязи между двумя переменными по шкале от –1 до 1. Пороговые значения для величин эффекта сравнимы со сдвигом среднего: небольшие эффекты начинаются с ±0,2, средние — порядка ±0,5, а большие величины эффекта требуют корреляций ±0,8.

- *Коэффициент вариации* r^2. Квадрат коэффициента корреляции отражает долю дисперсии в одной переменной, которая объясняется другой. Пороги следуют из квадрата, приведенного выше. Небольшие эффекты объясняют как минимум 4% дисперсии, средние эффекты — дисперсией ≥25%, а большие величины эффекта — как минимум 64%.

- *Процент перекрытия*. Площадь ниже линии любого распределения вероятностей по определению равна 1. Площадь пересечения между двумя заданными распределениями является хорошей мерой их сходства, как показано на рис. 5.12. У одинаковых распределений перекрытие составляет 100%, в то время как перекрытие непересекающихся интервалов равно 0%. Разумные пороговые значения таковы: для небольших эффектов перекрытие составляет 53%, средних — 67%, а для больших — 85%.

Конечно, любой значительный эффект, который не является статистически значимым, по своей сути является подозрительным. Корреляция количества исследований и видеоигр на рис. 5.10 было настолько высокой (r = 0,985), что размер эффекта был бы огромным, если бы количество точек выборки и методология были достаточно обоснованными, чтобы подтвердить заключение.

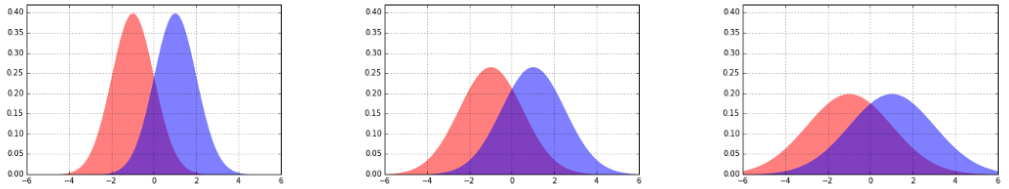


Рис. 5.12. Пары нормальных распределений с одинаковой разницей в среднем значении, но с возрастающей дисперсией слева направо. По мере увеличения дисперсии между распределениями перекрытие становится больше, что усложняет их разделение

> *На заметку.* Статистическая значимость зависит от количества выборок, а величина эффекта — нет.

5.3.2. T-критерий: сравнение средних значений совокупностей

Как мы уже видели, большие смещения среднего значения между двумя совокупностями предполагают большие размеры эффекта. Но сколько измерений нам нужно, прежде чем мы сможем с уверенностью сказать, что это явление реально? Предположим, мы измеряем IQ двадцати мужчин и двадцати женщин. Данные показывают, что одна группа в среднем умнее? Конечно, средние значения выборок будут отличаться, хотя бы немного, но значительна ли эта разница?

T-критерий оценивает, отличаются ли средние значения совокупности для двух выборок. Эта проблема обычно возникает при *AB-тесте* (AB testing), связанном с оценкой того, влияет ли изменение продукта на производительность. Предположим, вы демонстрируете одной группе пользователей версию *A*, а другой группе версию *B*. Предположим также, что вы измеряете значение производительности системы по каждому пользователю, например количество кликов по объявлениям или количество звезд, которые они дают, когда их спрашивают о впечатлении. С помощью t-критерия измеряется, является ли наблюдаемая разница между двумя группами значимой.

- *Разница в среднем значении относительно велика.* Это имеет смысл. Можно сделать вывод, что мужчины в среднем весят больше, чем женщины, поскольку величина эффекта очень велика. По данным Центра по контролю за заболеваниями[3], в 2010 году средний американский мужчина весил 195,5 фунта (88,45 кг), тогда как средняя американская женщина

[3] http://www.cdc.gov/nchs/fastats/obesity-overweight.htm

весила 166,2 фунта (75,39 кг). Довольно много. Чтобы доказать гораздо более тонкое различие, например IQ, реально требуется куда больше доказательств, чтобы быть столь же убедительным.

- *Стандартные отклонения достаточно малы.* Это также имеет смысл. Легко убедить себя в том, что у мужчин и женщин в среднем одинаковое количество пальцев, поскольку наблюдаемые нами подсчеты очень плотно сгруппированы вокруг среднего значения: {10, 10, 10, 10, 9, 10, ...}. Гипотеза о подсчете равных количеств пальцев потребовала бы гораздо больше доказательств, если бы цифры сильно колебались. Я бы с трудом согласился с истинностью среднего значения распределения $m = 10$, если бы наблюдал значения {3, 15, 6, 14, 17, 5}.
- *Количество выборок достаточно велико.* Это снова имеет смысл. Чем больше данных я вижу, тем тверже я убежден в том, что образец выборки точно представляет ее базовое распределение. Например, мужчины, несомненно, в среднем имеют меньше пальцев, чем женщины, что является следствием большего количества приключений с электроинструментами.[4] Но для подтверждения этого относительно редкого явления потребуется очень большое количество выборок наблюдений.

Расчет t-критерия начинается с вычисления *статистики критерия* (test statistic) по двум наборам наблюдений. T-статистика определяется как

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}},$$

где $\bar{x}_i$, σ_i и n_i — это среднее значение, стандартное отклонение и размер популяции в выборке i соответственно.

Давайте тщательно разберем это уравнение. Числитель — это разница между средними значениями, поэтому, чем больше эта разница, тем больше значение t-статистики. Стандартные отклонения находятся в знаменателе, поэтому, чем меньше значение σ_i, тем больше значение t-статистики. Если это не понятно, то вспомните, что происходит, когда вы делите x на число, приближающееся к нулю. Увеличение размера выборки n_i также уменьшает знаменатель, поэтому, чем больше n_i, тем больше значение t-статистики. Во всех случаях факторы, которые делают нас более уверенными в реальной разнице между этими двумя распределениями, увеличивают значение t-статистики.

[4] Одного этого наблюдения может быть вполне достаточно для объяснения взаимозависимости между полом и IQ, без всякой необходимости в дополнительных статистических данных.

Интерпретация смысла определенного значения t-статистики происходит в результате поиска числа в соответствующей таблице. Для желаемого *уровня значимости* (significance level) α и *количества степеней свободы* (degrees of freedom) (по существу, размера выборки) в таблице указывается значение v, которое t-статистика t должна превышать. Если $t > v$, то наблюдение является значимым для уровня α.

Почему это работает?

Статистические тесты, такие как t-критерий, зачастую выглядят для меня как вуду, поскольку мы смотрим на число с какого-то волшебного стола и относимся к нему как к Евангелию. *Оракул сказал: разница значительна!* Конечно, за проверкой значимости стоит настоящая математика, однако вывод включает вычисления и странные функции (такие как *гамма-функция* (gamma function) $\Gamma(n)$, вещественное обобщение факториалов). Именно из-за этих сложных вычислений возникла договоренность искать необходимое в предварительно вычисленной таблице, а не вычислять ее самостоятельно.

Вы можете найти вывод соответствующих формул в любой хорошей книге по статистике, если вы заинтересованы. Эти тесты основаны на таких идеях, как случайная выборка. Мы видели, как среднее значение и стандартное отклонение ограничивают форму любого основного распределения вероятности. Получение среднего значения выборки очень далеко от среднего означает неудачу. Согласно теории случайный выбор значений для нескольких стандартных отклонений от среднего значения для совокупности маловероятен. Это повышает вероятность того, что наблюдение такой большой разницы является результатом извлечения из другого распределения.

Большая часть технических деталей здесь является следствием работы с едва уловимыми феноменами и небольшими наборами данных. Исторически наблюдаемые данные были очень скудным ресурсом, и он остаются таковым во многих ситуациях. Вспомните наше обсуждение проверки эффективности лекарств, когда кто-то должен умереть за каждый набранный нами балл. Большой мир данных, в котором вы, вероятно, будете жить, обычно содержит больше наблюдений (все посещают нашу веб-страницу), более низкие ставки (покупают ли клиенты больше, когда вы показываете им зеленый фон вместо голубого?) и, возможно, меньшие размеры эффекта (насколько велико должно быть улучшение, чтобы оно оправдало переход на новый цвет фона?).

5.3.3. Критерий Колмогорова–Смирнова

С помощью t-критерия сравниваются две выборки из предположительно нормальных распределений в соответствии с расстоянием между их соответ-

ствующими средними значениями. *Критерий Колмогорова–Смирнова* (KS), напротив, сравнивает функции кумулятивных распределений (cumulative distribution function — cdf) двух выборочных распределений и оценивает, насколько они похожи.

Это показано на рис. 5.13. Функции cdf двух разных выборок нанесены на одну и ту же самую диаграмму. Если две выборки взяты из одного и того же распределения, диапазоны значений x должны в значительной степени перекрываться. Кроме того, поскольку оба распределения представлены в виде cdf, ось y представляет совокупную вероятность от 0 до 1. Обе функции монотонно возрастают слева направо, где $C(x)$ — это доля выборки $\leq x$.

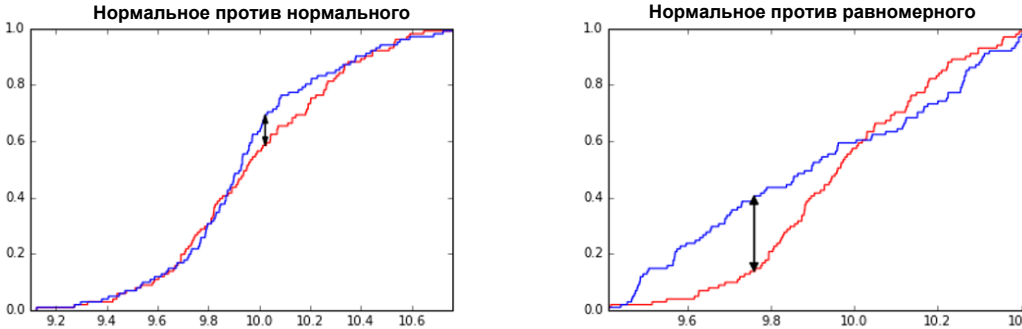


Рис. 5.13. Критерий Колмогорова–Смирнова количественно определяет разницу между двумя вероятностными распределениями по максимальному промежутку y-расстояния между двумя функциями кумулятивного распределения. Слева две выборки с одинаковым нормальным распределением. Справа сравнение выборок из равномерного и нормального распределений, проведенных в одном и том же x-диапазоне

Нам нужно найти такое значение x, для которого соответствующие значения y в двух cdf отличаются максимально. Расстояние $D(C_1, C_2)$ между распределениями C_1 и C_2 является разностью значений y при этом критическом x, формально определяемым как

$$D(C_1, C_2) = \max_{-\infty \leq x \leq \infty} |C_1(x) - C_2(x)|.$$

Чем более существенно различие в двух выборках распределений при некотором значении, тем более вероятно, что они были взяты из разных распределений. На рис. 5.13 (слева) показаны две независимые выборки из одного и того же нормального распределения. Обратите внимание на имеющийся крошечный разрыв между ними. На рис. 5.13 (справа), напротив, сравнивается выборка, взятая из нормального распределения, с выборкой, взятой из равномерного распределения. Критерий Колмогорова–Смирнова не обмануть: посмотрите на большие зазоры возле хвостов, где мы и ожидаем его увидеть.

Критерий Колмогорова–Смирнова сравнивает значение $D(C_1,C_2)$ с конкретной целью, заявляя, что два распределения отличаются на уровне значимости α, когда:

$$D(C_1,C_2) > c(\alpha)\sqrt{\frac{n_1+n_2}{n_1 n_2}},$$

где $c(\alpha)$ является константой для поиска в таблице.

В основе функции размеров выборки лежит некая интуиция. Для простоты предположим, что оба образца имеют одинаковый размер n. Тогда

$$\sqrt{\frac{n_1+n_2}{n_1 n_2}} = \sqrt{\frac{2n}{n^2}} = \sqrt{\frac{2}{n}}.$$

Значение $\sqrt{n}$ возникает естественно в задачах выборки, таких как стандартное отклонение биномиального распределения. Ожидаемая разница между количеством орлов и решек при n бросках монеты составляет порядка $\sqrt{n}$. В контексте критерия Колмогорова–Смирнова это также отражает ожидаемое отклонение, когда два образца следует считать одинаковыми. Критерий Колмогорова–Смирнова отражает то, что происходит в самом сердце распределения, на основе которого можно сделать достоверное утверждение.

Мне нравится критерий Колмогорова–Смирнова. Он предоставляет графики распределений, которые я могу понять, демонстрирующие самое слабое место в предположении об их идентичности. В этом критерии меньше технических предположений и вариантов, чем в t-критерии, что означает меньшую вероятность допустить ошибку при его использовании. Кроме того, критерий Колмогорова–Смирнова может быть применен ко многим задачам, в том числе к проверке выброса точек из нормального распределения.

Проверка на нормальность

При построении графика нормальное распределение дает колоколообразную кривую. Но не каждое распределение в форме колокола является нормальным, и иногда важно знать разницу.

Существуют специализированные статистические тесты для проверки нормальности заданных выборок распределений f_1. Но для этого мы можем использовать и общий критерий Колмогорова–Смирнова при условии, что мы можем определить значимый f_2 для сравнения с f_1.

Именно поэтому я представил методы случайной выборки в разделе 5.2. Используя описанный нами метод кумулятивного распределения, можно получить статистически обоснованные случайные выборки из n точек для любого распределения, в котором вы знаете cdf, для любого n. Для f_2 мы должны выбрать значимое количество точек для сравнения. Мы можем использовать

$n_2 = n_1$ или, возможно, несколько бо́льшую выборку, если n_1 очень мало. Мы хотим быть уверены, что с помощью нашей выборки мы фиксируем форму желаемого распределения.

Таким образом, если мы построим нашу случайную выборку для f_2 из нормального распределения, критерий Колмогорова–Смирнова не сможет отличить f_1 от f_2, если f_1 также исходит из нормального распределения при тех же μ и σ.

Одно предостережение. Достаточно чувствительный статистический тест, вероятно, откажет в нормальности практически любому наблюдаемому распределению. Нормальное распределение — это абстракция, а мир — сложное место. Но просмотр графика критерия Колмогорова–Смирнова покажет вам, где именно происходят отклонения. Хвосты слишком толстые или слишком худые? Распределение перекошено? С этим пониманием вы можете решить, достаточно ли велики различия, чтобы иметь для вас значение.

5.3.4. Поправка Бонферрони

В науке давно принято использовать $\alpha = 0{,}05$ в качестве границы между статистической значимостью и случайностью. Статистическая значимость 0,05 означает, что существует вероятность 1/20, что этот результат был бы получен чисто случайно.

Это не является бессмысленным стандартом при сборе данных для проверки сложной гипотезы. Ставка на лошадь с коэффициентом 20 к 1 на выигрыш заслуживает внимания. Если только вы не делали ставки одновременно на миллионы других лошадей. Тогда похвальба о небольшом количестве ставок 20:1, в которых вы действительно выиграли, будет по меньшей мере вводить в заблуждение.

Таким образом, предварительный сбор фактов, в ходе которого проверяют миллионы гипотез, должен соответствовать более высоким стандартам. Эта ошибка привела к сильной, но ложной корреляции между количеством докторов наук в области информатики и видеоиграми на рис. 5.10. Она была обнаружена в ходе сравнения тысяч графиков друг с другом и сохранения только самых забавных с наиболее высокой оценкой корреляции.

Поправка Бонферрони (Bonferroni correction)[5] обеспечивает важный баланс при оценке того, насколько мы доверяем явно значимому статистическому результату. Следовательно, то, как именно вы нашли корреляцию, может быть так же важно, как и сила самой корреляции. У того, кто купит миллион лотерейных билетов и однажды выиграет, будет гораздо менее впечатляющий успех, чем у того, кто покупает всего один билет и выигрывает.

[5] Я всегда считал, что "*Поправка Бонферрони*" станет великолепным названием для боевика. С Дуэйном Джонсоном в роли Бонферрони.

В поправке Бонферрони говорится, что при одновременной проверке n разных гипотез результирующее значение p должно возрасти до уровня α/n, чтобы считаться значимым на уровне α.

Как и в любом статистическом тесте, правильное применение коррекции таит в себе множество тонкостей. Но главный принцип здесь важно понять. Связанные с компьютерами люди особенно склонны проводить масштабные сравнения всего со всем или охотиться за необычными выбросами и моделями. В конце концов, когда вы написали программу анализа, почему бы не запустить ее для всех ваших данных? Представление только лучших, специально отобранных результатов позволяет легко обмануть других людей. Поправка Бонферрони — это способ не дать вам обмануть себя.

5.3.5. Частота ложных открытий

Поправка Бонферрони защищает нас от поспешного принятия значимости единственной успешной гипотезы во многих испытаниях. Но зачастую при работе с большими многомерными данными мы сталкиваемся с другой проблемой. Вероятно, все m переменных коррелируют (возможно, слабо) с целевой переменной. Если n достаточно велико, многие из этих корреляций будут статистически значимыми. Действительно ли мы сделали так много важных открытий?

Метод Бенджамини–Хохберга (Benjamini–Hochberg procedure) для минимизации *частоты ложных открытий* (False Discovery Rate — FDR) предоставляет очень простой способ провести черту между интересными и неинтересными переменными на основе значимости. Отсортируем переменные по силе их p-значения, чтобы наиболее значимые переменные находились слева, а наименее значимые — справа. Теперь рассмотрим переменную i-го ранга в этом порядке. Мы принимаем значение этой переменной на уровне α, если

$$\forall_{j=1}^{i}(p_j \leq \frac{j}{m}\alpha).$$

Ситуация проиллюстрирована на рис. 5.14. Значения p сортируются в порядке возрастания слева направо, что обозначено нерегулярной синей кривой. Если мы принимаем все p-значения меньше α, мы принимаем слишком много. Вот почему Бонферрони разработал свою поправку. Однако требование, чтобы все p-значения соответствовали стандарту поправки Бонферрони (где кривая пересекает α/m), является слишком строгим.

Метод Бенджамини–Хохберга признает, что если многие значения действительно значимы для определенного стандарта, то определенная их доля должна быть значимой для гораздо более высокого стандарта. Диагональная линия на рис. 5.14 обеспечивает этот уровень контроля качества.

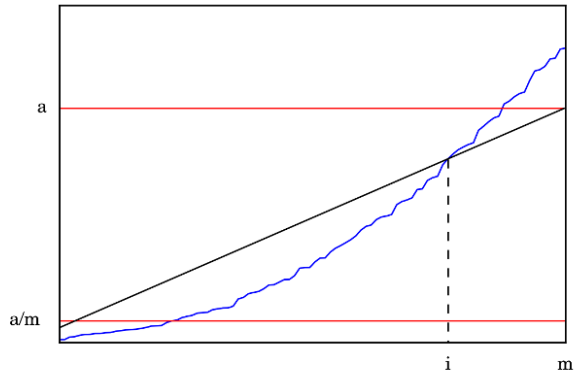


Рис. 5.14. Метод Бенджамини–Хохберга минимизирует частоту ложных открытий, принимая p-значения, только когда $p_i \leq ai/m$. Синяя кривая показывает отсортированные p-значения, а диагональная черта определяет, когда такое p-значение является значительным

5.4. Случай из жизни: поиск фонтана молодости

Это была красивая свадьба. Мы были очень рады за Рэйчел и Дэвида, жениха и невесту. Я ел как король, танцевал с моей любимой женой и наслаждался теплым блеском грудинки, когда мне показалось, что что-то не так. Я осмотрел комнату и сделал странное наблюдение. Каким-то образом, впервые за много лет, я стал моложе, чем большинство людей в толпе.

Это может показаться вам не слишком важным, но это потому, что вы читатель, вероятно, моложе, чем большинство людей во многих ситуациях. Но поверьте мне, придет время, когда вы заметите такие вещи. Я помню, как впервые осознал, что уже ходил в колледж в то время, когда рождалось большинство из моих учеников. Потом они начали рождаться, когда я учился в аспирантуре. Сегодняшние студенты колледжа родились не только после того, как я стал профессором, но и после того, как я получил здесь должность. Так как же я мог быть моложе большинства людей на этой свадьбе?

Было две возможности. Либо случайно в комнату вошло слишком много пожилых людей, либо была причина, объясняющая это явление. Именно поэтому были изобретены статистические тесты значимости и *p*-значения, чтобы помочь отличить что-то от ничего.

Так какова была вероятность того, что тогда, в 54 года, я окажусь моложе, чем большинство из 251 человека на свадьбе Рэйчел? Согласно Wolfram Alpha (точнее, по пятилетним оценкам American Community Survey за 2008–2012 годы), в Соединенных Штатах насчитывалось 309,1 миллиона человек,

из которых 77,1 миллиона были в возрасте 55 лет и старше. Когда я пишу эти строки, почти 25% населения старше меня.

Вероятность того, что большинство из 251 случайно выбранных американцев будет старше 55 лет, определяется следующим образом:

$$p = \sum_{i=126}^{251} \binom{251}{i} (1-0{,}75)^i (0{,}75)^{(251-i)} = 8{,}98 \cdot 10^{-18}.$$

Хотя это намного больше, чем предыдущее значение p, оно все равно невероятно мало: сродни выбрасыванию сразу 27 орлов идеальной монетой. Просто запретить детям вход было недостаточно, чтобы снова сделать меня молодым.

Я вернулся к Рэйчел и заставил ее признаться. Оказывается, у ее матери было необычно большое количество двоюродных братьев, и она поддерживала *со всеми* ними исключительно хорошие связи. Вспомните теорию относительности Эйнштейна, где $E = mc^2$ означает, что каждый являющийся двоюродным братом моей матери, исключается дважды. *Все* эти двоюродные братья были приглашены на свадьбу. С семьей Рэйчел, существенно превосходящей необычно крошечный клан жениха, эта когорта старших кузенов стала доминировать на танцплощадке.

Действительно, мы можем вычислить количество старших двоюродных братьев (c), которых нужно пригласить, чтобы получить шансы 50/50, что я буду моложе, чем средний гость, при условии, что остальные из 251 гостя были выбраны случайным образом. Оказывается, что при $c = 65$ одиноких двоюродных братьев (или 32,5 супружеских пар) будет вполне достаточно, как только дети будут исключены ($f = 0{,}672$).

$$p = \sum_{i=126}^{251} \binom{251}{i} (1-0{,}0672)^i (0{,}672)^{(251-i)} = 9{,}118 \cdot 10^{-9}.$$

Мораль здесь в том, что важно вычислить вероятность любого интересного наблюдения, прежде чем объявлять его чудом. Никогда не останавливайтесь на частичном объяснении, если оно не снижает удивление до правдоподобных уровней. Вероятно, есть подлинное явление, лежащее в основе любого достаточно редкого события, и обнаружение того, с чем оно связано, делает науку о данных захватывающей.

5.5. Критерии перестановки и p-значения

Традиционные критерии статистической значимости оказываются весьма эффективными при решении вопроса о том, действительно ли две выборки взяты из одного и того же распределения. Тем не менее эти критерии должны быть вычислены правильно, чтобы делать свою работу. У многих стандартных

критериев есть свои тонкости, такие как вопросы односторонних и двусторонних критериев, предположения о распределении и многое другое. Правильное вычисление этих критериев требует умения и внимания.

Критерии перестановок (permutation test) позволяют использовать более общий и дуракоустойчивый способ определения значимости. Если ваша гипотеза подтверждается данными, то случайно перемешанные наборы данных должны быть менее вероятными для ее подтверждения. Проведя множество испытаний с использованием случайных данных, мы можем точно установить, насколько необычным является феномен, который вы проверяете.

Рассмотрим рис. 5.15, где мы обозначаем независимую переменную (пол: мужской или женский) и зависимую переменную (скажем, рост). Исходное распределение (слева) выглядит совершенно по-разному для мужчин и женщин, отражая подлинные различия в росте. Но насколько необычна эта разница? Мы можем построить новый набор данных, случайно присваивая пол исходным переменным (в центре). Сортировка внутри каждой группы показывает, что распределение результатов по псевдо-мужчинам/женщинам теперь гораздо более сбалансировано, чем в исходных данных (справа). Это свидетельствует о том, что пол действительно был важным фактором в определении роста, и мы пришли к еще более важному выводу, что это случается снова и снова после 1 000 или 1 000 000 испытаний.

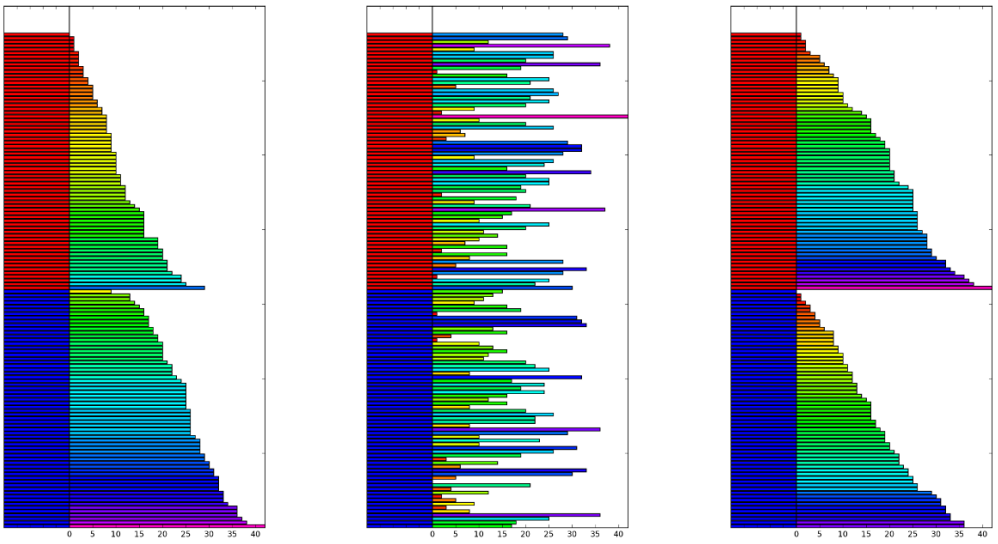


Рис. 5.15. Критерий перестановок показывают значимость корреляции между полом и ростом (слева). Случайное распределение пола по росту (в центре) приводит к существенно разному распределению результатов при сортировке (справа), подтверждая значимость исходных отношений

Ранг критериев статистики на реальных данных среди распределения статистических значений из случайных перестановок определяет уровень значимости или *p-значение* (*p*-value), рис. 5.16 (слева) демонстрирует то, что мы ищем. Реальная стоимость лежит на правом краю распределения, свидетельствуя о значимости. На рисунке справа реальная стоимость лежит в средней части распределения, свидетельствуя об отсутствии эффекта.

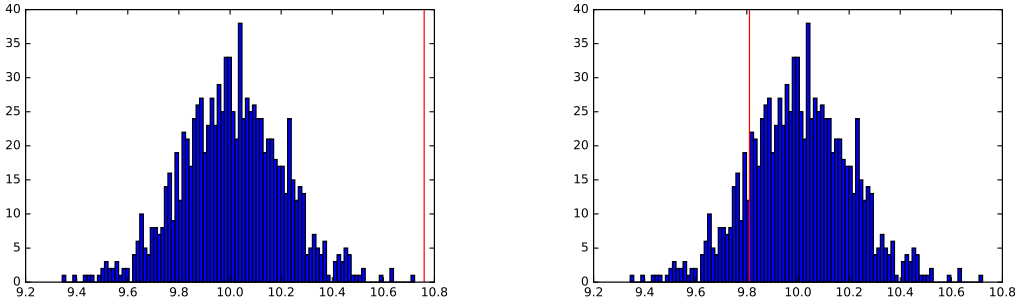


Рис. 5.16. Критерии перестановок оценивают значимость по позиции оценки на фактических данных по сравнению с распределением оценок, произведенных случайными перестановками. Позиция на краю хвоста (слева) очень важна, а позиция в середине распределения (слева) не интересна

Критерии перестановки требуют, чтобы вы разработали статистику, которая отражает вашу гипотезу о данных. Коэффициент корреляции является разумным выбором, если вы хотите установить важные отношения между конкретной парой переменных. В идеале наблюдаемая корреляция в реальных данных будет сильнее, чем в любой случайной перестановке этих данных. При проверке связи между полом и ростом, нашей статистикой, возможно, может быть разница в среднем росте мужчин и женщин. Опять же, мы надеемся, что в реальных данных это окажется больше, чем в большинстве случайных перестановок.

Будьте изобретательны в выборе статистики: сила критериев перестановки заключается в том, что они могут работать практически со всем, что вы можете придумать, чтобы доказать свою правоту. Лучше всего, если ваша статистика минимизирует вероятность связей, так как вы обязаны считать все связи свидетельствующими против вашей гипотезы.

На заметку. Критерии перестановки дают вам вероятность ваших данных с учетом вашей гипотезы, а именно, что статистика будет отличаться от распределения случайной выборки. Это не совсем то же самое, что и доказательство вашей гипотезы с помощью данных, что является традиционной целью проверки статистической значимости. Но это намного лучше, чем ничего.

Оценка значимости или p-значение критерия перестановки зависит от того, сколько случайных попыток выполнено. Всегда старайтесь сделать не менее 1000 случайных испытаний и даже больше, если это возможно. Чем больше перестановок вы опробуете, тем более значительным может быть ваше p-значение, по крайней мере до определенной степени. Если данный ввод на самом деле является лучшим из всех $k!$ перестановок, то самое экстремальное значение p, которое вы можете получить, составляет $1/k!$, независимо от того, сколько случайных перестановок вы опробовали. Избыточная выборка увеличивает ваш знаменатель, не увеличивая при этом вашу истинную уверенность.

> *На заметку.* P-значения вычисляются для повышения *вашей* уверенности в том, что наблюдение является реальным и интересным. Это работает только тогда, когда вы честно вычисляете критерий перестановки, выполняя эксперименты, которые могут обеспечить достаточную степень удивления.

5.5.1. Создание случайных перестановок

Создание случайных перестановок является еще одной важной проблемой выборки, с которой люди нередко сталкиваются. Оба приведенных ниже алгоритма используют последовательности случайных перестановок для перемешивания начальной перестановки $\{1, 2, \ldots, n\}$.

Но гарантировать, что все $n!$ перестановок генерируются случайно и равномерно — это довольно сложное дело. Действительно, только один из этих алгоритмов делает это правильно так:

```
for i = 1 to n do a[i] = i;
for i = 1 to n - 1 do swap[a[i], a[Random[i, n]];
```

или так:

```
for i = 1 to n do a[i] = i;
for i = 1 to n - 1 do swap[a[i], a[Random[1, n]].
```

Рассмотрим это внимательно: разница здесь очень тонкая. Это настолько тонко, что вы можете даже не заметить этого в коде. Критическая разница — это 1 или i в вызове `Random`. Один из этих алгоритмов является правильным, а один — нет. Если вы думаете, что можете сказать, что один работает, а другой нет, убедительно объясните, почему.

Если вы действительно желаете знать, то правильным алгоритмом является первый. Он выбирает случайный элемент от 1 до n для первой позиции, затем оставляет его в покое и возвращается к остальным. Он генерирует перестановки равномерно случайным образом. Второй алгоритм дает определенным

элементам лучший шанс оказаться первым, доказывая, что распределение не является равномерным.

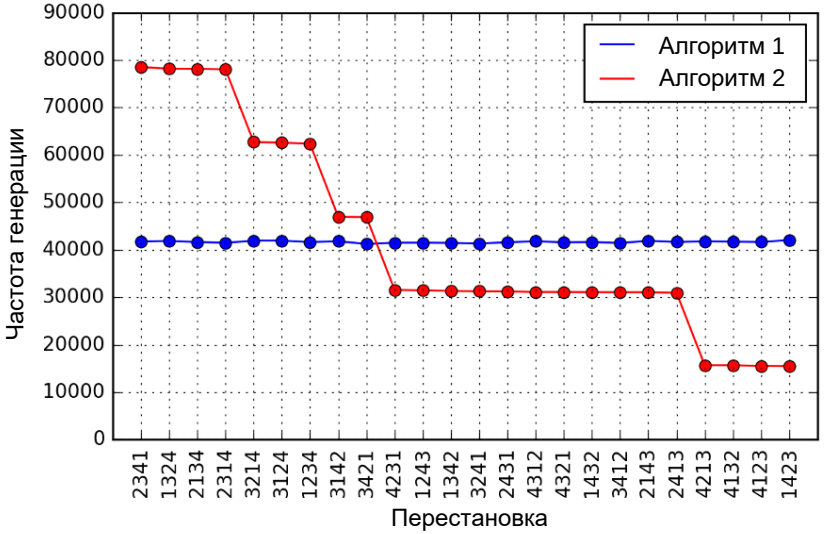


Рис. 5.17. Частота генерации для всех 4! = 24 перестановок с использованием двух разных алгоритмов. Алгоритм 1 генерирует их с одинаковой частотой, в то время как алгоритм 2 существенно меняет ее

Но если вы не можете доказать это теоретически, вы можете использовать идею критерия перестановок. Реализуйте оба алгоритма и выполните 1 000 000 прогонов каждого, построив случайные перестановки, скажем, для $n = 4$ элементов. Посчитайте, как часто каждый алгоритм генерирует каждый из 4! = 24 различных перестановок. Результаты такого эксперимента показаны на рис. 5.17. Алгоритм 1 оказывается невероятно устойчивым со стандартным отклонением 166,1 случая. В отличие от него существует восьмикратное различие между наиболее и наименее частыми перестановками в алгоритме 2, при этом $\sigma = 20\,923{,}9$.

Мораль здесь в том, что случайная генерация может быть очень тонким делом. И что эксперименты типа Монте-Карло, такие как критерии перестановок, могут устранить необходимость в тонких рассуждениях. Сначала проверяй, потом доверяй.

5.5.2. Страйк хитов Ди Маджо

Одним из самых удивительных рекордов в бейсболе является страйк хитов (hitting streak) на 56 игр Джо Ди Маджо (DiMaggio). Задача бэттера в том,

чтобы принимать хиты, и они получают, возможно, четыре шанса в каждой игре, чтобы получить их. Даже очень хорошие хитеры терпят неудачу.

Но в 1941 году Джо Ди Маджо добился успеха в 56 играх подряд, и это поистине удивительное достижение. Ни один игрок за семьдесят пять лет с тех пор не приблизился к этому рекорду, равно как ни один до него.

Но насколько необычным была такая длина страйка в контексте его карьеры? Ди Маджо сыграл 1736 игр, с 2214 хитами при 6821 бэте. Таким образом, он должен получить хиты примерно в $1 - (1 - 2214/6821)^4 = 79{,}2\%$ его игр с четырьмя бэтами. Какова вероятность того, что кто-то с его уровнем квалификации сможет справиться с такой последовательной серией игр в ходе своей карьеры?

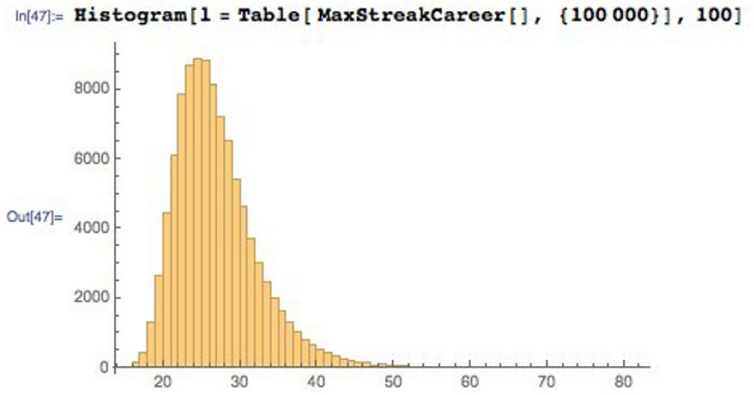


Рис. 5.18. Распределение самых длинных страйков хитов на более чем 100 000 моделях карьер. Фактический страйк хитов в 56 играх Ди Маджо располагается у самого хвоста этого распределения, демонстрируя тем самым сложность этого подвига

Для тех из вас, кто устал от моих аналогий в бейсболе, предлагаю рассмотреть это в другом контексте. Предположим, вы студент, который в среднем набрал по тестам 90 баллов. Вы очень хороший ученик, но не идеальный. Каковы шансы, что у вас может быть полоса везения, когда вы набрали более 90 баллов на десяти тестах подряд? Как насчет двадцати подряд? Не могли бы вы пройти успешно 56 тестов подряд?[6] Если случилась такая длинная полоса, значит ли это, что вы перешли на другой уровень обучения или вам просто повезло?

Итак, когда у Ди Маджо был страйк хитов, это было просто ожидаемым следствием его бесспорных навыков, или ему просто повезло? Он был одним из самых лучших хитеров своего времени или всех времен, звездой каждого сезона своей тринадцатилетней карьеры. Но мы также знаем, что Ди Маджо время от времени везло. В конце концов, он *был* женат на кинозвезде Мэрилин Монро.

[6] Нет, если вы посещаете один из моих курсов, уверяю вас.

Чтобы решить этот вопрос, мы использовали случайные числа для моделирования, когда он получил хиты за свою синтетическую "карьеру" из 1736 игр. В каждой симулированный игре Джо получал четыре шанса на поражение и добился успеха с вероятностью $p = 2214/6821 = 0{,}325$. Затем мы могли бы определить самую длинную серию хитов в течение этой смоделированной карьеры. Моделируя 100 000 карьер Ди Маджо, мы получаем плотность распределения страйков, которые могут продемонстрировать редкость его достижений в контексте, получая в процессе p-значения.

Результаты показаны на рис. 5.18. Только в 44 из 100 000 смоделированных карьер ($p = 0{,}00044$) Ди Маджо провел серию не менее чем из 56 игр. Таким образом, длина совершенно не соответствует тому, что можно было бы ожидать от него. Вторая самая длинная полоса среди всех хитеров в высшей лиге — всего 44 игры, так что она также не соответствует всем остальным. Но он также однажды попал в 61 игру подряд на более низком уровне конкуренции, поэтому он, похоже, обладал исключительной способностью к последовательности.

Ударные полосы можно рассматривать как проходы между играми без ударов, и поэтому их можно смоделировать с помощью распределения Пуассона. Но симуляции Монте-Карло дают ответы без подробной математики. Критерии перестановок дают нам понимание с минимальными знаниями и интеллектуальными усилиями.

5.6. Байесовский вывод

Условная вероятность $P(A|B)$ измеряет вероятность события A с учетом того, что произошло событие B. В этой книге мы будем полагаться на условную вероятность, поскольку она позволяет нам обновлять нашу уверенность в событии в ответ на новые данные, такие как данные наблюдений.

Теорема Байеса (Bayes' theorem) — это важный инструмент для работы с условными вероятностями, поскольку он позволяет нам обращать условные выражения:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}.$$

С помощью теоремы Байеса мы можем преобразовать вопрос о $P(outcome|data)$ в $P(data|outcome)$, который зачастую гораздо проще вычислить. В некотором смысле теорема Байеса является просто следствием алгебры, но она приводит к другому способу мышления о вероятности.

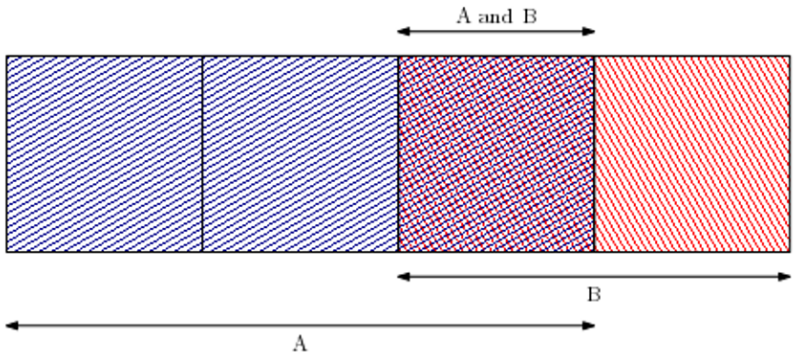


Рис. 5.19. Теорема Байеса в действии

На рис. 5.19 теорема Байеса иллюстрируется в действии. *Пространство событий* (event space) состоит из выбора одного блока из четырех. Комплексные события A и B представляют поддиапазоны блоков, где $P(A) = 3/4$ и $P(B) = 2/4 = 1/2$. Подсчитав блоки на рисунке, мы можем заметить, что $P(A|B) = 1/2$ и $P(B|A) = 1/3$. Это также следует непосредственно из теоремы Байеса:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}=\frac{(1/3)\cdot(3/4)}{(1/2)}=1/2$$

$$P(B|A)=\frac{P(A|B)P(B)}{P(A)}=\frac{(1/2)\cdot(1/2)}{(3/4)}=1/3.$$

Байесовский вывод отражает, как обновляется априорная вероятность $P(A)$, чтобы дать апостериорную вероятность $P(A|B)$ перед новым наблюдением B, в соответствии с отношением вероятности $P(B|A)$ и предельной вероятности $P(B)$. *Предыдущая* вероятность $P(A)$ отражает наше первоначальное предположение о мире, которое должно быть пересмотрено на основе дополнительных доказательств B.

Байесовский вывод — это важный способ взглянуть на мир. Придя на свадьбу Рэйчел и Дэвида, я предполагал, что распределение по возрасту будет отражать распределение по миру в целом. Но моя уверенность слабла с приходом каждого пожилого кузена, пока она наконец не исчезла.

Мы будем использовать байесовский вывод для построения классификаторов в разделе 11.1. Но помните об этой философии при анализе данных. Вы должны подходить к каждому заданию с предварительным представлением о том, какими должны быть ответы, а затем пересмотреть их в соответствии со статистическими данными.

5.7. Дополнительная информация

Каждый аналитик данных должен пройти курс элементарной статистики. К рекомендуемым текстам обычно относятся Фридман [54], а также и Джеймс и др. [21]. Уилан [22] — это общее введение, а Хафф [55] — это классический трактат о том, как лучше опираться на статистику.

Донохо [56] представляет увлекательную историю науки о данных с точки зрения статистики. Он дает убедительный аргумент в пользу того, что большинство основных принципов современной науки о данных изначально были разработаны статистиками, хотя они и не были быстро приняты дисциплиной в целом. Современные статистики начали проводить гораздо более приятные беседы с программистами по этим вопросам, поскольку их интересы взаимно сходились.

Виген [52] представляет забавную коллекцию ложных корреляций, взятых из большого количества интересных временны*х* рядов. Пример приведен на рис. 5.10 и перепечатан с разрешения.

Было продемонстрировано, что размер американских семей достаточно хорошо соответствует распределению Пуассона. Фактически анализ распределения размеров домохозяйств из 104 стран показывает, что модель "с меня хватит" работает во всем мире [57].

5.8. Упражнения

Статистические распределения

5.1. *[5]* Объясните, какое распределение (биномиальное, нормальное, пуассоновское или по степенному закону) кажется наиболее подходящим для следующего явления?

(a) Количество листьев на полностью выросшем дубе.
(b) Возраст, когда у людей седеют волосы.
(c) Количество волос на головах 20-летних людей.
(d) Количество людей, пострадавших от молнии x раз.
(e) Количество миль, пройденных до того, как вашему автомобилю понадобится новая коробка передач.
(f) Количество пробежек бэттера на каждый овер крикета.
(g) Количество пятен леопарда на квадратный фут его шкуры.
(h) Количество людей с ровно x пенни в ящиках буфета.
(i) Количество приложений на сотовых телефонах людей.
(j) Ежедневная посещаемость курса Скиены по науке о данных.

5.2. *[5]* Объясните, какое распределение (биномиальное, нормальное, пуассоновское или по степенному закону) кажется наиболее подходящим для следующего явления *The Quant Shop*?
(a) Красота конкурсантов на конкурсе Мисс Вселенная.
(b) Кассовый сбор фильмов, созданных голливудскими студиями.
(c) Вес детей при рождении.
(d) Цена произведений искусства на аукционе.
(e) Количество снега, выпадающего на Рождество в Нью-Йорке.
(f) Количество команд, которые выиграют x игр в данном футбольном сезоне.
(g) Продолжительность жизни известных людей.
(h) Дневная цена на золото в течение данного года.

5.3. *[5]* С учетом, что соответствующее распределение является нормальным, оцените вероятность следующих событий.
(a) Что в следующей сотне бросков идеальной монеты будет 70 или более орлов?
(b) Что случайно выбранный человек будет весить более 300 фунтов (136,078 кг)?

5.4. *[3]* Средний балл на экзамене по истории составил 85 баллов из 100 при стандартном отклонении 15. Было ли распределение баллов по этому экзамену симметричным? Если нет, какую форму вы ожидаете для этого распределения? Объясните свои рассуждения.

5.5. *[5]* Данные Facebook свидетельствуют, что 50% его пользователей имеют сотню или более друзей. Кроме того, среднее количество друзей пользователя составляет 190. Что эти результаты говорят о форме распределения количества друзей пользователей Facebook?

Проверка значимости

5.6. *[3]* Какие из следующих событий, скорее всего, независимы, а какие нет?
(a) Броски монеты.
(b) Броски в баскетболе.
(c) Успех партий на президентских выборах.

5.7. *[5]* По оценкам American Community Survey за 2010 год, 47,1% женщин в возрасте 15 и старше лет состоят в браке.
(a) Случайно выберите трех женщин в этом возрасте. Какова вероятность того, что третья выбранная женщина окажется единственной замужней?
(b) Какова вероятность того, что все три женщины состоят в браке?
(c) Сколько женщин в среднем вы ожидаете отобрать до выбора замужней женщины? Каково стандартное отклонение?

(d) Если бы доля замужних женщин на самом деле составляла 30%, сколько женщин вы бы ожидали отобрать до выбора замужней женщины? Каково стандартное отклонение?

(e) На основании ваших ответов на вопросы (c) и (d), как уменьшение вероятности события влияет на среднее значение и стандартное отклонение времени ожидания до достижения успеха?

Критерии перестановки и p-значения

5.8. *[5]* Докажите, что алгоритм генерации перестановок на стр. 197 правильно создает перестановки, что означает случайное равномерное распределение.

5.9. *[5]* Получите данные о росте мужчин m и женщин w.

(a) Используйте t-критерий, чтобы определить, являются ли мужчины в среднем выше женщин.

(b) Вычислите критерий перестановок, чтобы установить то же самое: мужчины в среднем выше женщин.

Реализация проектов

5.10. *[5]* На спортивных соревнованиях хорошие команды, как правило, побеждают. Но потому ли, что они знают, как победить, или просто потому, что после достаточно долгой игры лучшая команда все равно победит? Поэкспериментируйте с моделью случайного подбрасывания монеты, где лучшая команда имеет вероятность $p > 0{,}5$ переиграть другую за один период. Для n периодов игры, как часто лучшая команда выигрывает, и как часто она проигрывает при заданной вероятности p? Как это соотносится со статистикой из реальных видов спорта?

5.11. *[8]* 2 февраля — это День сурка в Соединенных Штатах, когда говорят, что, если сурок видит свою тень, последует еще шесть недель зимы. Принимая во внимание, солнечно ли 2 февраля в качестве показателя для сурка, есть ли какая-то предсказательная сила для этой традиции? Проведите исследование, основанное на записях о погоде, и сообщите о точности прогнозов зверя вместе с его статистической значимостью.

Вопросы на интервью

5.12. *[3]* Что такое условная вероятность?

5.13. *[3]* Что такое теорема Байеса? И почему она полезна на практике?

5.14. *[8]* Как бы вы улучшили алгоритм обнаружения спама, который использует наивный байесовский классификатор?

5.15. *[5]* Монету подбрасывают десять раз, и в результате получается две решки и восемь орлов. Как вы можете определить, идеальна ли монета? Каково p-значение для этого результата?

5.16. *[8]* Теперь предположим, что десять монет подбрасывают по десять раз, всего 100 бросков. Как бы вы проверили, идеальны ли монеты?

5.17. *[8]* Муравей находится на бесконечно длинной ветке. Муравей может двигаться на один шаг назад или на один шаг вперед с одинаковой вероятностью в течение дискретных временны*х* промежутков. Какова вероятность того, что муравей вернется к своей начальной точке после $2n$ шагов?

5.18. *[5]* Вы собираетесь сесть на самолет в Сиэтл. Вам стоит захватить зонтик? Вы звоните трем своим друзьям, которые там живут, и спрашиваете каждого независимо, идет ли дождь. Каждый из друзей имеет 2/3 шанса сказать вам правду и 1/3 шанса солгать. Все трое друзей говорят, что в Сиэтле идет дождь. Какова вероятность того, что на самом деле идет дождь?

Конкурсы Kaggle

5.19. Решите, является ли купленный на аукционе автомобиль плохой покупкой. `https://www.kaggle.com/c/DontGetKicked`

5.20. Прогноз спроса на товар на протяжении данной недели. `https://www.kaggle.com/c/grupo-bimbo-inventory-demand`

5.21. Сколько осадков выпадет за следующий час? `https://www.kaggle.com/c/how-much-did-it-rain`.

Глава 6

Визуализация данных

> Графика — это в лучшем случае инструмент для рассуждений.
>
> — Эдвард Тафти (Edward Tufte)

Эффективная визуализация данных является важным аспектом науки о данных, по крайней мере по трем причинам.

- *Исследовательский анализ данных.* Как на самом деле выглядят ваши данные? Получение информации о том, с чем вы имеете дело, является первым шагом любого серьезного анализа. Графики и визуализация — это лучший способ сделать это.
- *Обнаружение ошибок.* Вы сделали что-то глупое в своем анализе? Подача невизуализируемых данных в любой алгоритм машинного обучения вызывает проблемы. Проблемы с точками выброса, недостаточной очисткой и ошибочными предположениями сразу же обнаруживаются при правильной визуализации ваших данных. Слишком часто сводная статистика (с точностью 77,8%!) скрывает то, что на самом деле делает ваша модель. Тщательный анализ того, что вы делаете правильно, а что нет — это первый шаг к лучшему результату.
- *Коммуникация.* Можете ли вы эффективно представить то, что вы узнали, другим? Значимые результаты становятся действенными только после их публикации. Ваш успех в качестве аналитика данных зависит от способности убедить других людей в том, что вы знаете, о чем говорите. Картина стоит 1000 слов, особенно когда вы делаете презентацию скептически настроенной аудитории.

Вы, вероятно, делали графики и диаграммы с начальной школы. Вездесущее программное обеспечение позволяет легко создавать профессионально выглядящие изображения. Так что же такого сложного в визуализации данных?

Для ответа предлагаю притчу. Страшный инцидент из моей молодости о нападении на чемпионку по фигурному катанию. Бандит ударил ее по колену палкой, надеясь выбить ее из предстоящих Олимпийских игр. К счастью, он не попал по колену, и фигуристка выиграла серебряную медаль.

Однако, узнав своего клиента, адвокат бандита выступил с интересной защитой. Это преступление, по его словам, было явно слишком сложным, чтобы его клиент смог задумать его самостоятельно. Это произвело на меня впечатление, так как это означало, что я недооценил познавательную способность, необходимую для того, чтобы кто-то кого-то ударил по ноге палкой.

Моя мораль здесь в том, что многие вещи сложнее, чем кажутся. В частности, я говорю о проблеме нанесения данных на график, чтобы уловить то, что он говорит. Удивительно высокая доля диаграмм, которые я видел в презентациях, *ужасна*, они либо не говорят ничего, либо неправильно передают то, что на самом деле показывают данные. Плохие графики могут иметь негативное значение, что уведет вас в неправильном направлении.

В этом разделе мы познакомимся с принципами, которые заставляют работать стандартные графики, и покажем, как они могут вводить в заблуждение, если не используются должным образом. Исходя из этого опыта, мы попытаемся развить ваше понимание того, когда графики могут лгать, и как вы можете их улучшить.

6.1. Исследовательский анализ данных

Появление массивных наборов данных меняет способ, которым делается наука. Традиционный научный метод *основан на гипотезах* (hypothesis driven). Исследователь формулирует теорию о том, как устроен мир, а затем пытается поддержать или отвергнуть эту гипотезу на основе данных. В отличие от этого, наука *на основании данных* (data-driven), начинает со сбора существенного набора данных, а затем ищет шаблоны, которые в идеале будут играть роль гипотез для будущего анализа.

Исследовательский анализ данных (exploratory data analysis) — это поиск закономерностей и тенденций в заданном наборе данных. Техники визуализации играют важную роль в этом процессе. Внимательное изучение ваших данных важно по нескольким причинам, включая выявление ошибок при сборе и обработке, выявление нарушений статистических допущений и выдвижение интересных гипотез.

В этом разделе мы обсудим, как выполнить анализ исследовательских данных и что приносит визуализация как часть процесса.

6.1.1. Противостояние новому набору данных

Что вам делать при обнаружении нового набора данных? В первую очередь это зависит от того, почему вы заинтересованы в нем, но начальные этапы исследования оказываются практически независимыми от области применения.

Я рекомендую следующие этапы, чтобы ознакомиться с любым новым набором данных, которые я иллюстрирую при изучении набора данных измерений тела NHANES по адресу `https://www.statcrunch.com/app/index.php?dataid=1406047`. Это табличные данные, но общие принципы здесь применимы к более широкому классу ресурсов.

- *Ответьте на основные вопросы.* Есть несколько вещей, которые вы должны знать о своем наборе данных, прежде чем открыть файл. Задайте следующие вопросы.
  - *Кто создал этот набор данных, когда и почему?* Понимание того, как были получены ваши данные, дает представление о том, насколько они важны и должны ли мы доверять им. Это также указывает нам на правильных людей, если нам нужно больше узнать о происхождении данных. Немного покопавшись, я обнаружил, что они были получены в ходе Национальной программы обследования состояния здоровья и питания (National Health and Nutrition Examination Survey — NHANES) за 2009–2010 гг., а также узнал, кто именно отвечал за его публикацию.
  - *Насколько он велик?* Насколько богаты данные с точки зрения количества полей или столбцов? Насколько набор данных велик по количеству записей или строк? Если он слишком большой, чтобы его можно было легко исследовать с помощью интерактивных инструментов, извлеките небольшую выборку и проведите ее первоначальные исследования. Этот набор данных содержит 4978 записей (2452 для мужчины и 2526 для женщин), каждая из которых имеет семь полей данных плюс пол.
  - *Что означают поля?* Просмотрите все столбцы в вашем наборе данных и убедитесь, что вы понимаете, что это такое. Какие поля являются числовыми, а какие категориальными? В каких единицах измерялись величины? Какие поля являются идентификаторами или описаниями, а не данными для вычисления? Быстрый обзор показывает, что рост и вес здесь были измерены с использованием метрической системы в сантиметрах и килограммах соответственно.
- *Ищите знакомые или понятные записи.* Я считаю чрезвычайно ценным знакомство с несколькими записями еще до того, как я узнаю их названия. Записи, как правило, связаны с человеком, местом или предметом, о котором вы уже знаете, поэтому вы можете поместить их в контекст и оценить достоверность данных. Но если нет, найдите несколько записей, представляющих особый интерес, чтобы узнать, возможно, те, которые имеют максимальные или минимальные значения наиболее важных полей.

Если знакомых записей нет, иногда их стоит создать. Грамотный разработчик базы данных медицинских карт сказал мне, что он использовал 5000 лучших имен из *Who's Bigger*, чтобы они служили именами пациентов при разработке продукта. Это была гораздо более вдохновляющая идея, чем создание искусственных названий, таких как "Пациент F1253". Они были достаточно забавными, чтобы стимулировать игру с системой, и достаточно запоминающимися, чтобы их можно было помечать и сообщать о случаях выброса, например: "С Францем Кафкой что-то серьезно не так".

- *Сделайте статистические сводки.* Посмотрите основные статистические данные каждого столбца. *Пятичисловая сводка* (five number summary) Тьюки является отличным началом для числовых значений, состоящих из экстремальных значений (max и min), а также медианных и квартильных элементов.

  Применительно к компонентам нашего набора данных роста и веса мы получаем:

	Минимум	**25%**	**Медиана**	**75%**	**Максимум**
Возраст	241	418	584	748	959
Вес	32,4	67,2	78,8	92,6	218,2
Рост	140	160	167	175	204
Длина ног	23,7	35,7	38,4	41	55,5
Длина рук	29,5	35,5	37,4	39,4	47,7
Окружность рук	19,5	29,7	32,8	36,1	141,1
Талия	59,1	87,5	97,95	108,3	172

  Это очень информативно. Для начала, что за средний возраст 584 года? Возвращаясь к данным, мы узнаем, что возраст измеряется месяцами, т.е. медиана составляет 48,67 года. Длина руки и ноги примерно одинакова, но длина ноги незначительно больше. *Я не знал этого.* Но внезапно я понимаю, что, описывая людей, чаще отмечают длинные/короткие ноги, чем длинные/короткие руки, возможно, именно поэтому.

  Для категориальных полей, таких как профессия, аналогичной сводкой будет отчет о том, сколько разных типов меток появляется в столбце и какие три наиболее популярные категории имеют соответствующие частоты.

- *Парные корреляции.* Матрица коэффициентов корреляции между всеми парами столбцов (или, по крайней мере, столбцов с зависимыми переменными, представляющими интерес) дает представление о том, насколько легко будет построить успешную модель. В идеале у нас будет несколько особенностей, которые сильно коррелируют с результатом, но не сильно коррелируют друг с другом. Только один столбец из набора идеально

коррелированных объектов имеет какое-либо значение, поскольку все остальные функции полностью определены из любого отдельного столбца.

	Возраст	Вес	Рост	Длина ног	Длина рук	Окружность рук	Талия
Возраст	1,000						
Вес	0,017	1,000					
Рост	–0,105	0,443	1,000				
Длина ног	–0,268	0,238	0,745	1,000			
Длина рук	0,053	0,583	0,801	0,614	1,000		
Окружность рук	0,007	0,890	0,226	0,088	0,444	1,000	
Талия	0,227	0,892	0,181	–0,029	0,402	0,820	1,000

Эти парные корреляции довольно интересны. Почему рост *отрицательно* коррелирует с возрастом? Все люди здесь взрослые (241 месяц = 20,1 года), поэтому все они выросли полностью. Но предыдущее поколение было ниже современных людей. Кроме того, с возрастом люди становятся ниже, поэтому совместно это, вероятно, объясняет данное явление. Сильная корреляция между весом и размером талии (0,89) отражает неудобную правду о природе.

- *Разделение на классы.* Существуют ли интересные способы разделения вещей по основным категориальным переменным, таким как пол или местоположение? С помощью сводной статистики вы можете определить, есть ли разница между распределениями, если они обусловлены категорией. Особенно ищите там, где, по вашему мнению, должны быть различия, основанные на вашем понимании данных и области применения.

  Корреляции были в целом схожи по полу, но были некоторые интересные различия. Например, корреляция между ростом и весом сильнее у мужчин (0,443), чем у женщин (0,297).

- *Графики распределений.* Эта глава посвящена методам визуализации данных. Используйте типы диаграмм, которые мы обсудим в разделе 6.3, чтобы просмотреть распределения в поисках шаблонов и выбросов. Какова общая форма каждого распределения? Должны ли данные быть очищены или преобразованы, чтобы сделать их более колоколообразными?

  На рис. 6.1 показана мощь сетки точечных графиков различных переменных. С первого взгляда мы видим, что нет огромных выбросов, пары которых коррелируют, а также характер всех линий тенденций. Вооружившись этим единственным графиком, мы теперь готовы применить этот набор данных к любой задаче.

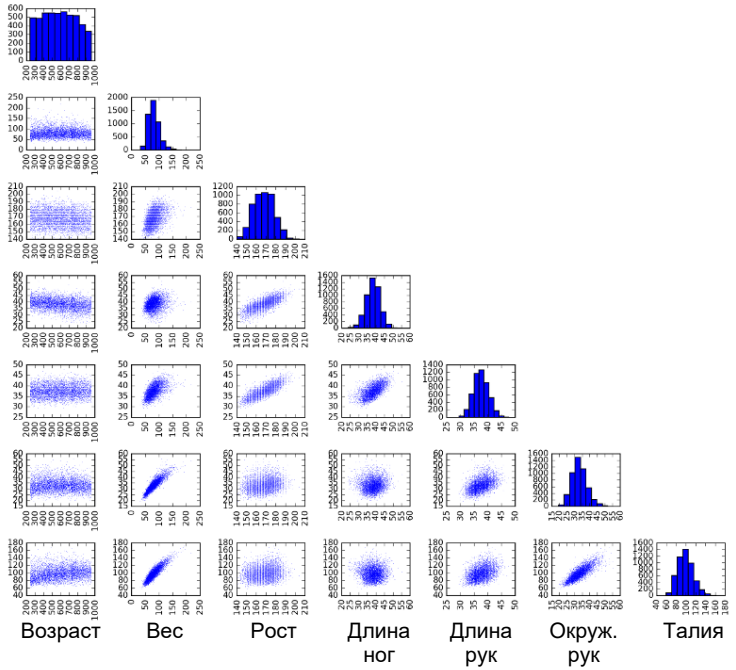


Рис. 6.1. Массив точечных графиков пар переменных позволяет быстро понять фору распределений значений данных и их корреляции

6.1.2. Сводная статистика и квартет Энскомба

Существуют пределы того, насколько хорошо вы можете понимать данные без методов визуализации. Лучше всего это изображает *квартет Энскомба* (Anscombe's quartet): четыре двумерных набора данных, каждый из которых имеет одиннадцать точек (рис. 6.2). Все четыре набора данных имеют одинаковые средние значения для значений x и y, одинаковые отклонения для значений x и y и точно такую же корреляцию между значениями x и y.

Эти наборы данных должны быть очень похожи, верно? Изучите немного цифры, чтобы вы могли понять, как они выглядят.

Поняли? Теперь посмотрите на точечные графики этих наборов данных на рис. 6.3. Все они выглядят по-разному и рассказывают существенно разные истории. Одна тенденция линейная, а вторая выглядит почти параболической. Две других — почти идеально линейные с выбросами по модулю, но с сильно отличающимися наклонами.

	I		II		III		IV	
	x	y	x	y	x	y	x	y
	10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
	8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
	13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
	9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
	11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
	14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
	6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
	4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
	12.0	10.84	12.0	9.31	12.0	8.15	8.0	5.56
	7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
	5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89
Среднее	9.0	7.5	9.0	7.5	9.0	7.5	9.0	7.5
Отклонение	10.0	3.75	10.0	3.75	10.0	3.75	10.0	3.75
Корреляция	0.816		0.816		0.816		0.816	

Рис. 6.2. Четыре набора данных с одинаковыми статистическими свойствами. На что они похожи?

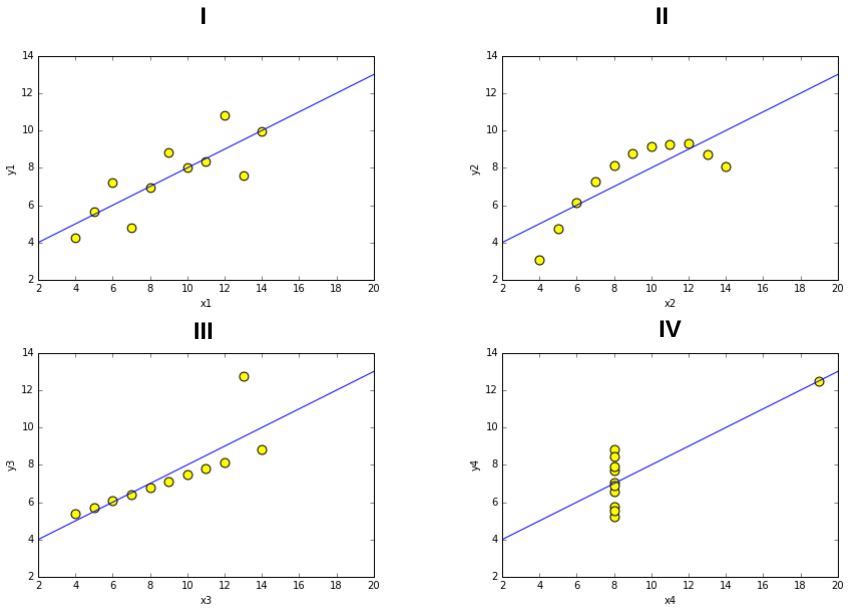


Рис. 6.3. Графики квартета Энскомба. Все эти наборы данных существенно отличаются, хотя и имеют одинаковую сводную статистику

Дело здесь в том, что вы можете мгновенно оценить эти различия, взглянув на график. Даже простые визуализации являются мощными инструментами для понимания того, что происходит в наборе данных. Любой разумный аналитик данных стремится в полной мере использовать методы визуализации.

6.1.3. Инструменты визуализации

Для визуализации доступна обширная коллекция программных средств. По правде говоря, задачи визуализации делятся на три категории, и правильный выбор инструментов зависит от того, какова ваша задача.

- *Исследовательский анализ данных.* Здесь мы собираемся выполнять быстрое, интерактивное исследование имеющегося набора данных. Программы для работы с электронными таблицами, такие как Excel, и среды программирования для ноутбуков, такие как iPython, R и Mathematica, вполне эффективны при построении графиков стандартных типов. Ключевым моментом здесь является сокрытие сложности, поэтому процедуры построения графиков стандартно делают то, что нужно, но при необходимости их можно и настроить.
- *Графики качества, подходящего для публикаций и презентаций.* Тот факт, что программа Excel очень популярна, вовсе не означает, что она создает лучшие графики. Хорошая визуализация — это результат взаимодействия между ученым и программным обеспечением, с использованием всех преимуществ гибкости инструмента для максимизации информационного содержания графики.

  Графические библиотеки, такие как MatPlotLib или Gnuplot, поддерживают множество опций, позволяющих придать вашему графику точно такой вид, как вы хотите. Статистический язык R имеет весьма обширную библиотеку визуализаций данных. Просмотрите каталоги типов графиков, поддерживаемых вашей любимой библиотекой, позволяющих найти лучшее представление для ваших данных.
- *Интерактивная визуализация для внешних приложений.* Создание инструментальных панелей, облегчающих взаимодействие пользователя с собственными наборами данных, является типичной задачей для инженеров, ориентированных на науку о данных. Типичной задачей здесь является создание инструментов поддержки исследовательского анализа данных для персонала, менее технически квалифицированного, но более ориентированного на область применения.

  С использованием стандартных графических библиотек такие системы могут быть легко встроены в языки программирования, например Python. Существует также класс сторонних систем для построения информационных панелей, таких как Tableau. Эти системы создаются на более высоком уровне, чем другие инструменты, они поддерживают определенные парадигмы взаимодействия и связанные с ними средства для разных представлений данных.

6.2. Выработка эстетики визуализации

Грамотная оценка произведения искусства или вина требует наличия определенного вкуса или эстетики. Дело не в том, нравится ли вам что-то, а в том, чтобы понять, почему вам это нравится. Эксперты-искусствоведы говорят о диапазоне палитры художника, об использовании света или напряженности композиции. Знатоки вина говорят об аромате, крепости, кислотности и прозрачности своего любимого напитка, а также о том, сколько в нем содержится дубильных веществ или танина. Они всегда могут сказать что-то лучше, чем просто "это вкусно".

Чтобы отличать хорошие визуализации от плохих, требуется выработать эстетику дизайна и словарный запас, чтобы говорить о представлениях данных. На рис. 6.4 представлены два знаменитых шедевра западной живописи. Какой из них лучше? Этот вопрос не имеет смысла без чувства эстетики и словарного запаса для его описания.

Рис. 6.4. Какая картина вам нравится больше? Формирование интеллектуальных предпочтений в искусстве или визуализации зависит от наличия отличительной визуальной эстетики

Моя визуальная эстетика и словарный запас во многом получены из книг Эдварда Тафти (Edward Tufte) [58, 59, 60]. Он художник: однажды мне довелось встретиться с ним в его бывшей художественной галерее напротив Челси Пирс (Chelsea Piers) на Манхэттене. Он долго думал о том, что делает диаграмму или график информативным и красивым. Опираясь на эстетику дизайна, он пришел к следующим выводам.

- *Максимизация соотношения данных и чернил.* Ваша визуализация должна показать ваши данные. Так почему же на графиках вы видите так много фоновых сеток, теней и меток?
- *Минимизация фактора лжи.* Как и ученый, ваши данные должны раскрывать истину, в идеале — ту, которую вы хотите увидеть раскрытой. Но честны ли вы со своей аудиторией или используете графические средства, которые вводят их в заблуждение, чтобы показать то, чего на самом деле нет?

- *Минимизация неинформативных элементов.* Современное программное обеспечение для визуализации часто добавляет интересные визуальные эффекты, которые имеют мало общего с вашим набором данных. Ваш график интересен из-за ваших данных или несмотря на них?
- *Правильные масштабы и ясные маркеры.* Точная интерпретация данных зависит от элементов, не связанных с данными, таких как масштаб и маркировка. Оптимизированы ли ваши описательные материалы по ясности и точности?
- *Эффективное использование цвета.* Человеческий глаз способен различать небольшие градации оттенков и насыщенности цвета. Используете ли вы цвет для выделения важных свойств ваших данных или просто для художественного заявления?
- *Сила повторения.* Массивы схожей графики с разными, но связанными элементами данных обеспечивают краткий и мощный способ визуального сравнения. Ваши однотипные диаграммы облегчают сравнения или просто избыточны?

Каждый из этих принципов будет подробно описан далее.

6.2.1. Максимизация соотношения данных и чернил

На любом графике часть чернил используется для представления фактических базовых данных, а остальные — для графических эффектов. По правде говоря, визуализация должна фокусироваться на демонстрации самих данных. Мы определяем *соотношение данных и чернил* (data-ink ratio) так.

$$\text{Соотношение данных и чернил} = \frac{\text{Чернила для данных}}{\text{Общее количество чернил, используемых в графике}}.$$

На рис. 6.5 представлена средняя заработная плата в разбивке по полу (Бюро статистики труда, 2015 г.), что поможет прояснить это понятие. Какое представление данных вы предпочитаете? Изображение слева говорит: "Круто, как вы создали тени и такой эффект трехмерной перспективы?" Изображение справа говорит: "Вау, женщинам действительно недоплачивают во всех точках спектра доходов. Но почему для консультантов разрыв самый маленький?"

Максимизация соотношения данных и чернил позволяет данным говорить, что в первую очередь является главной целью упражнения по визуализации. Изображение справа позволяет точнее сравнивать высоту столбцов, поэтому мужчины не выглядят как женщины с высокими прическами. Цвета тоже хорошо делают свою работу, позволяя нам сравнивать яблоки с яблоками.

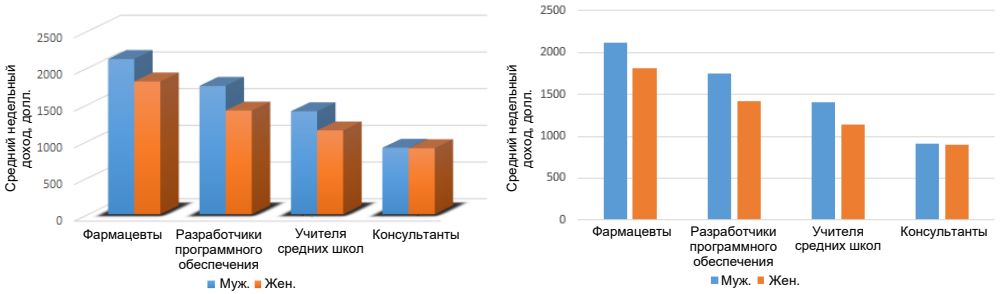


Рис. 6.5. Объемные столбики, отбрасывающие тени (слева), могут выглядеть впечатляюще. Но на самом деле это просто неинформативные элементы, которые служат для снижения четкости и соотношения данных и чернил по сравнению с простым отображением данных (справа)

Существуют более экстремальные способы увеличения соотношения данных и чернил. Зачем нам вообще столбцы? Та же самая информация может быть передана точками на соответствующей высоте, и, несомненно, это будет лучше, если мы нанесем гораздо больше, чем восемь точек, показанных здесь. Имейте в виду, что при визуализации данных меньше может быть лучше.

6.2.2. Минимизация фактора лжи

Визуализация стремится рассказать правдивую историю о том, что говорят данные. Самая лживая форма лжи — это обман ваших данных, но вполне возможно, что ваши данные точны, но некто намеренно вводит аудиторию в заблуждение относительно того, что они говорят. Тафти определяет *фактор лжи* (lie factor) диаграммы как:

$$\text{Фактор лжи} = \frac{\text{Размер эффекта на графике}}{\text{Размер эффекта в данных}}.$$

Целостность графика требует минимизации этого факта лжи и избегания методов, имеющих тенденцию вводить в заблуждение. К плохим практикам относятся следующие.

- *Представление среднего без вариаций.* Значения данных {100, 100, 100, 100, 100} и {200, 0, 100, 200, 0} рассказывают разные истории, даже если средние значения обоих равны 100. Если вы не можете построить фактические точки со средним, то по крайней мере покажите дисперсию, чтобы прояснить степень, в которой среднее отражает распределение.
- *Представление интерполяции без фактических данных.* Линии регрессии и подогнанные кривые эффективны для передачи трендов и упрощения больших наборов данных. Но без показа точек данных, на которых оно основано, невозможно определить качество соответствия.

- *Искажения масштаба.* Соотношение сторон фигуры может оказать огромное влияние на то, как мы интерпретируем то, что видим. На рис 6.6 представлены три визуализации некого финансового временного графика, вполне идентичные, за исключением соотношения сторон диаграммы.

  В нижнем варианте серия выглядит обычно: здесь не о чем беспокоиться. Справа прибыль упала прямо с обрыва: небо рухнуло! График в левом углу демонстрирует серьезное снижение, но с признаками сезонного скачка.

  Какой сюжет правильный? Люди обычно привыкли видеть графики, представленные в соответствии с золотым сечением, подразумевая, что ширина должна быть примерно в 1,6 раза больше высоты. Придайте им эту форму, если у вас нет серьезных причин, почему это неуместно. Психологи сообщают, что 45-градусные линии являются наиболее легко интерпретируемыми, поэтому избегайте форм, которые существенно уводят линии от этого ориентира.

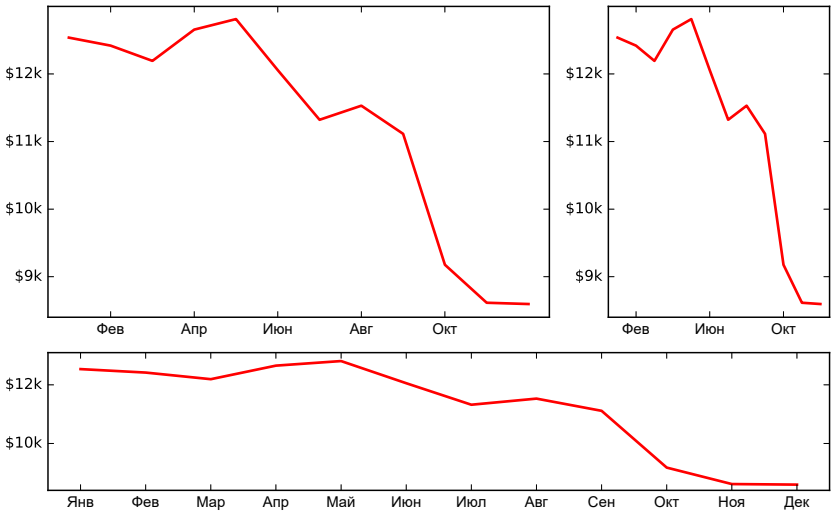


Рис. 6.6. Три визуализации одного и того же финансового временного ряда. Какая наиболее точно отражает ситуацию?

- *Удаление меток с числовых осей.* Даже самые плохие искажения масштаба могут быть полностью скрыты, если не выводить числовые метки на осях. Только с помощью числовой разметки можно восстановить действительные значения данных на графике.
- *Удаление с графика исходной точки.* В большинстве графиков неявно предполагается, что диапазон значений по оси y отображаются от нуля до $y$max. Мы теряем способность визуально сравнивать величины, если диапазон y вместо этого распространяется от $y_{min} - \varepsilon$ до y_{max}. Наибольшее значение внезапно начинает выглядеть во много раз больше, чем наименьшее значение, а не масштабируется в нужной пропорции.

Если бы рис. 6.5 (справа) был нарисован с узким [900, 2500] диапазоном значений *y*, то создалось бы впечатление о том, что консультанты голодали, а не зарабатывали зарплаты почти наравне с учителями, разработчиками программного обеспечения и фармацевтами. Такой обман может быть обнаружен, если на оси отмечены значения, но поймать на нем трудно.

Несмотря на формулу Тафти, фактор лжи не может быть вычислен механически, поскольку он требует понимания текущей повестки дня, которая стоит за этим искажением. При чтении любого графика важно знать, кто его создал и почему. Понимание текущей повестки дня должно привлечь ваше внимание к потенциально вводящим в заблуждение сообщениям, закодированным в графике.

6.2.3. Минимизация неинформативных элементов

Посторонние визуальные элементы отвлекают от сообщения, которое пытаются передать данные. На захватывающем графике данные рассказывают историю, а не демонстрируют неинформативные элементы.

На рис. 6.7 представлены помесячные временны*е* ряды продаж в компании, которая начинает переживать плохие времена. Рассматриваемый график представляет собой *гистограмму* (bar plot) — совершенно надежный способ представления данных временного ряда, и она нарисована с использованием обычных, возможно, стандартных параметров, с применением вполне обычного пакета для построения графиков.

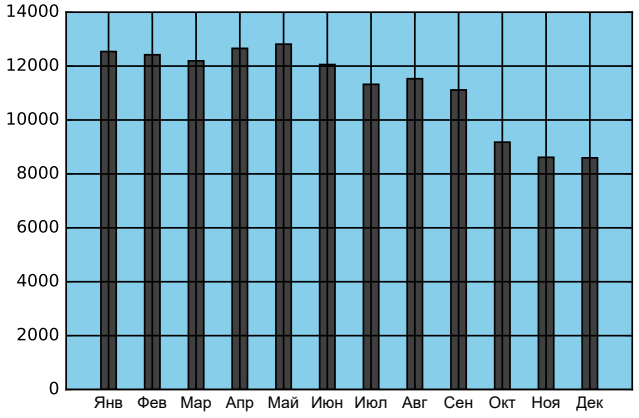


Рис. 6.7. Временной ряд месячных объемов продаж. Как мы можем улучшить/упростить эту гистограмму временного ряда?

Но можем ли мы упростить этот график, удалив элементы так, чтобы лучше выделить данные? Подумайте об этом минуту, прежде чем взглянуть на рис. 6.8, где представлена серия из четырех последовательных упрощений этого графика. Критически важные операции таковы.

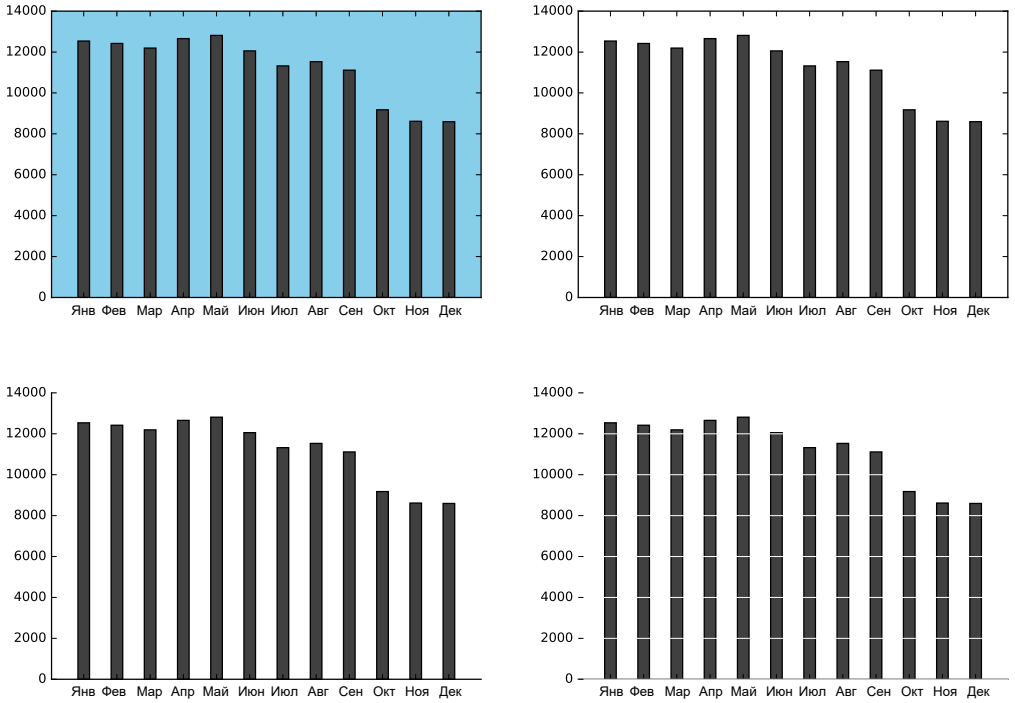


Рис. 6.8. Четыре последовательных упрощения рис. 6.7 за счет удаления посторонних неинформативных элементов

- *Освободите данные из тюрьмы (вверху слева).* Тяжелые решетки заточают ваши данные, визуально доминируя над содержимым. Часто графики можно улучшить, удалив сетку или хотя бы осветлив ее.

  Потенциальная ценность сетки данных заключается в том, что она способствует более точной интерпретации числовых величин. Таким образом, сетки, как правило, наиболее полезны на графиках с большим количеством значений, которые, возможно, должны быть указаны точно. Светлые сетки могут вполне адекватно справляться с такими задачами.

- *Прекратите отбрасывать тень (вверху справа).* Цветной фон здесь не способствует интерпретации графики. Удаление его увеличивает соотношение данных и чернил, а также делает его менее навязчивым.

- *Откажитесь от рамок (внизу слева).* Ограничительная рамка на самом деле не несет никакой информации, особенно верхняя и правая границы, которые не определяют оси. Уберите их и дайте больше воздуха вашему графику.

- *Заставьте отсутствие чернил работать на вас (внизу справа).* Эффект координатной сетки можно восстановить, удалив из столбцов линии вме-

сто добавления элементов. Это облегчает сравнение величин самых больших чисел, сосредоточивая внимание на больших изменениях в относительно небольшой верхней части, вместо небольших изменений в длинном столбике.

Архитектор Людвиг Мис ван дер Роэ (Mies van der Rohe) сказал, что "меньше значит больше". Удаление элементов из графиков зачастую улучшает их гораздо больше, чем их добавление. Сделайте это частью вашей философии графического дизайна.

6.2.4. Правильные масштабы и ясные маркеры

Недостатки в масштабировании и маркировке являются основным источником преднамеренной или случайной дезинформации на графиках. Метки должны указывать правильную величину чисел, а масштаб должен показывать эти цифры с правильным разрешением, чтобы облегчить сравнение. По правде говоря, данные должны быть отмасштабированы так, чтобы полностью заполнить пространство, выделенное для них на графике.

Разумные люди вполне способны различать, нужно ли масштабировать оси по всему теоретическому диапазону переменной или сократить его так, чтобы отразить только наблюдаемые значения. Однако некоторые решения явно необоснованны.

График на рис. 6.9 (слева) был составлен моим студентом, на нем показана корреляция между двумя переменными для почти ста языков. Поскольку корреляция колеблется между [–1, 1], он построил график в этом интервале. Огромное море белого на этом графике отражает только то, что мы могли бы добиться большего, приблизив корреляцию к 1,0. Но в остальном график не читается.

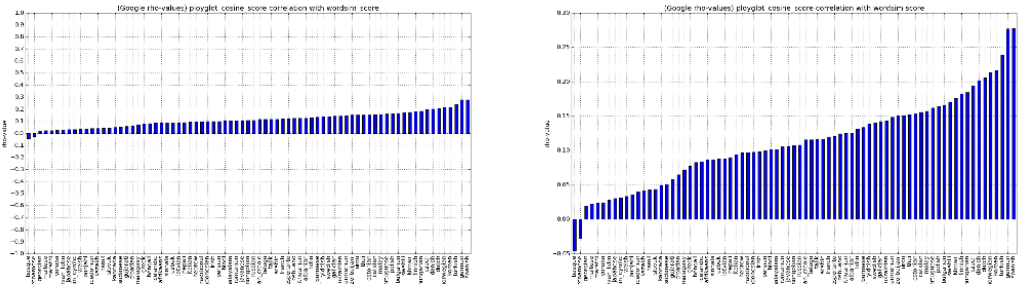


Рис. 6.9. Масштабирование по максимально возможному диапазону (слева) выглядит глупо, когда все, что он показывает, это пустое пространство. Лучшее масштабирование позволяет проводить более значимые сравнения (справа)

На рис. 6.9 (справа) представлены точно такие же данные, но в усеченном масштабе. *Теперь* мы можем видеть, где происходит увеличение результативности по мере перемещения слева направо, а также считывать результаты для любого языка. Раньше столбцы были так далеко от меток, что было трудно вообще сопоставлять их с названиями.

Самый большой недостаток усеченных масштабов проявляется тогда, когда вы показываете не весь столбец, поэтому его длина не отражает относительное значение переменной. Здесь мы показываем линию $y = 0$, помогая читателю понять, что каждый столбец должен быть целым. Освобождение данных из-за тюремной решетки также помогло бы.

6.2.5. Эффективное использование цвета

Цвета все чаще воспринимаются как часть любой графической коммуникации. Они играют две основные роли в диаграммах — отмечают различия классов и кодируют числовые значения. Представление точек разных типов, кластеров или классов разными цветами создает еще один слой информации на обычном точечном графике. Это отличная идея, когда мы пытаемся установить степень различий в распределении данных по классам. Самое главное, чтобы классы легко отличались друг от друга по выделению основным цветом.

Лучше всего, когда цвета выбраны так, чтобы иметь мнемонические значения для естественной связи с текущим классом. Потери должны быть напечатаны красными чернилами, экологические вопросы — зелеными, нации — цветами с их флага, а спортивные команды — цветами с их формы. Для представления мужчин точки окрашивают синим цветом, а женщин — красным, эта подсказка помогает зрителю интерпретировать график рассеяния, как показано на рис. 9.17.

Выбор цветов для представления числового масштаба является более сложной проблемой. Карты цветов радуги воспринимаются нелинейно, а значит, никому не очевидно, лежит ли фиолетовый цвет до или после зеленого. Таким образом, при отображении чисел в цветах радуги группировка одинаковых чисел в одинаковые цвета делает незаметными относительные величины без явной ссылки на цветовую шкалу. На рис. 6.10 представлены для сравнения несколько цветовых шкал из библиотеки MatPlotLib языка Python.

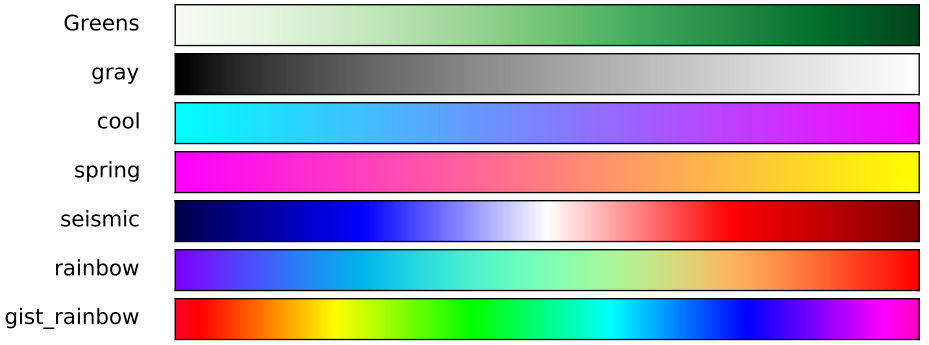


Рис. 6.10. Цветовые шкалы из Python MatPlotLib различаются оттенками, насыщенностью и яркостью. Радужные цветовые карты воспринимаются нелинейно, что затрудняет распознавание различий

Гораздо лучшие цветовые шкалы основаны на различной яркости или насыщенности. *Яркость* цвета модулируется за счет смешивания оттенка с оттенком серого, где-то между белым и черным. *Насыщенность* контролируется смешиванием с частью серого, где 0 дает чистый оттенок, а 1 удаляет цвет полностью.

Другая популярная цветовая шкала имеет различные положительные и отрицательные цвета (скажем, синий и красный, как в цветовой шкале seismic на рис. 6.10), отображаемые вокруг белого или серого центра в нуле. Таким образом, оттенок говорит зрителю о полярности числа, а яркость/насыщенность отражает их величину. Определенные цветовые гаммы намного лучше для дальтоников, особенно тех, кто избегает использования красного и зеленого.

Как правило, большие площади на графиках должны быть показаны ненасыщенными цветами. Обратное верно для небольших площадей, которые лучше выделять насыщенными цветами. Системы цветопередачи — это удивительно технический и сложный вопрос, а это значит, что вы должны всегда использовать хорошо проработанные цветовые шкалы, а не изобретать свои собственные.

6.2.6. Сила повторения

Множественные небольшие графики и таблицы являются отличным способом представления многомерных данных. Вспомните мощь сетки, демонстрирующей все двумерные распределения на рис. 6.1.

Есть много приложений создания нескольких небольших графиков. Мы можем использовать их, чтобы разделить распределение по классам, возможно, построив отдельные, но сопоставимые диаграммы по регионам, полу или периоду времени. Массивы графиков облегчают сравнение: что изменилось между различными распределениями.

Графики временных рядов позволяют сравнивать одни и те же величины в разных точках календаря. Еще лучше сравнить несколько временны*х* рядов в виде линий на одном графике или нескольких графиков в логическом массиве, отражающем их взаимосвязь.

6.3. Типы диаграмм

В этом разделе мы рассмотрим обоснование основных типов визуализаций данных. По каждой диаграмме я представляю лучшие практики их использования и обрисовываю степень свободы, которую вы должны обеспечить, чтобы ваша презентация была максимально эффективной.

Ничто не говорит "Вот график некоторых данных", как бездумный рисунок, созданный с использованием стандартных настроек какого-либо программного инструмента. Мои студенты слишком часто представляют мне такие сырые продукты данных, и этот раздел является своего рода личной реакцией на это.

> *На заметку.* У вас есть сила и ответственность создавать зн*а*чимые и понятные презентации. Эффективная визуализация включает в себя итеративный процесс просмотра данных, принятия решения о том, какую историю они пытаются рассказать, а затем улучшения отображения, чтобы рассказать историю лучше.

На рис. 6.11 представлено ориентировочное дерево решений от Абела [61], помогающее выбрать правильное представление данных. В этом разделе рассматриваются наиболее важные диаграммы, однако используйте это дерево, чтобы лучше понять, *почему* определенные визуализации более подходят в определенных контекстах. Нам нужно составить правильный график для имеющегося набора данных, а не просто первое, что приходит на ум.

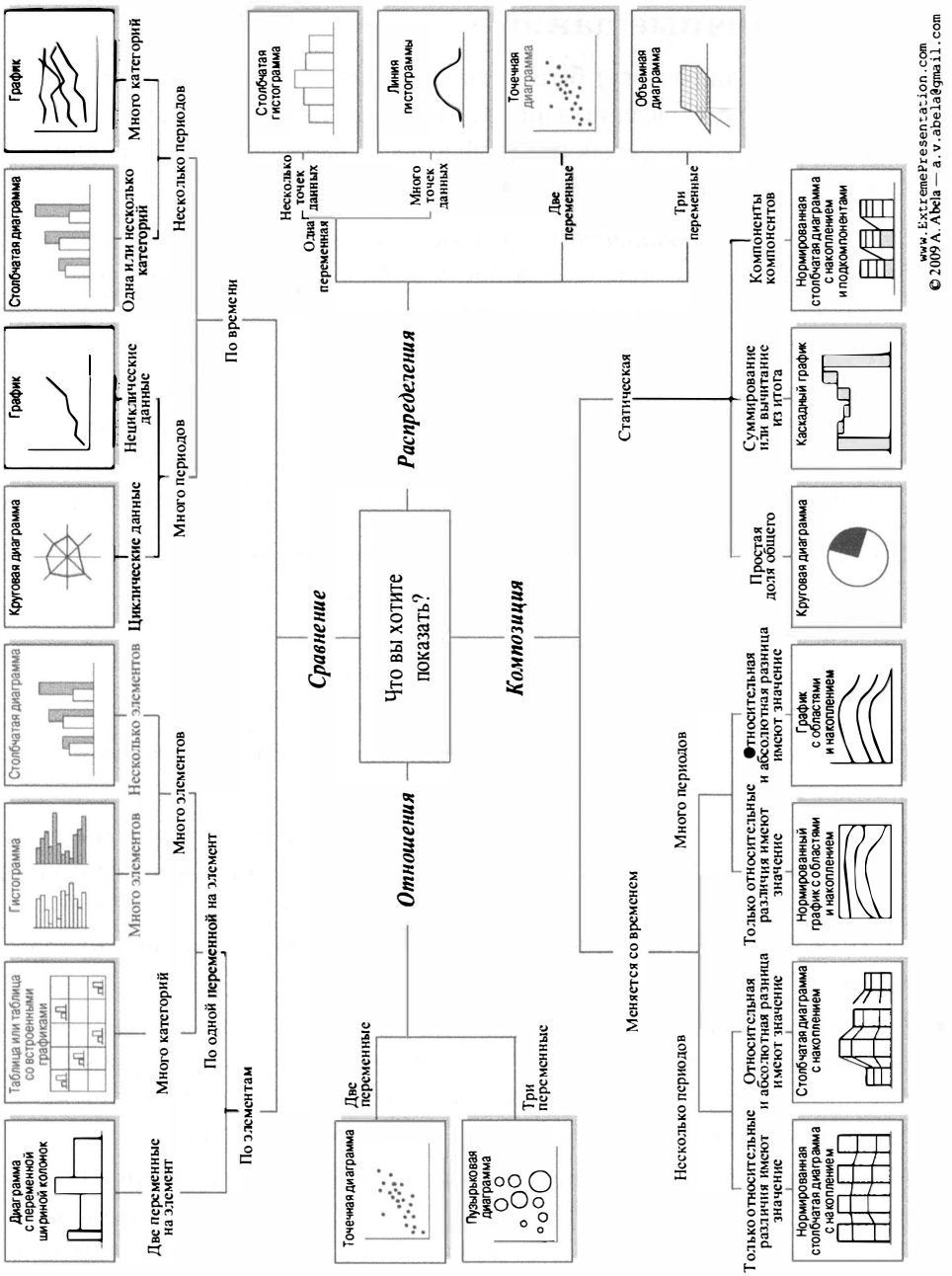


Рис. 6.11. Интеллектуальное дерево решений, помогающее определить лучшее визуальное представление данных. Перепечатано с разрешения Абелы [61]

6.3.1. Табличные данные

Числовые таблицы могут быть весьма красивы и являются очень эффективным способом представления данных. Хотя может *показаться*, что им не хватает визуальной привлекательности графических представлений, таблицы имеют ряд преимуществ перед другими представлениями.

- *Представление точности*. Разрядность чисел говорит вам о том, как оно было получено: средняя зарплата в 79 815 долл. говорит о чем-то отличном от 80 000 долл. Такие тонкости, как правило, теряются на графиках, но наглядно видны в числовых таблицах.
- *Представление масштаба*. Длины чисел в таблице можно сравнить с гистограммой в логарифмическом масштабе. Выравнивание по правому краю наилучшим образом говорит о различиях в порядке величин, поскольку (в куда меньшей степени) воспроизводит первые цифры чисел в столбце.

По левому краю	По центру	По правому краю
1	1	1
10	10	10
100	100	100
1000	1000	1000

  Выравнивание по левому краю предотвращает такие сравнения, поэтому всегда избегайте их.
- *Многомерная визуализация*. Геометрию сложно понять, когда мы выходим за пределы двух измерений. Но таблицы могут оставаться управляемыми даже для большого количества переменных. Вспомните бейсбольную статистику Бэйбла Рута на рис. 1.1, таблицу из двадцати восьми столбцов, которую легко может понять любой знающий фанат.
- *Гетерогенные данные*. Таблицы, как правило, являются лучшим способом представления сочетания числовых и категориальных атрибутов, таких как текст и метки. Для представления значений определенных полей могут даже использоваться такие символы, как эмодзи (смайлики).
- *Компактность*. Таблицы особенно полезны для представления небольшого количества точек. Две точки в двух измерениях можно нарисовать как линию, но зачем? Маленькая таблица, как правило, лучше, чем большая картинка.

Представление табличных данных кажется простым ("просто положите их в таблицу"), как стукнуть палкой по ноге. Но все тонкости заключаются в создании наиболее информативных таблиц. К лучшим практикам относятся следующие.

- *Упорядочивайте строки так, чтобы облегчить сравнение.* Вы можете упорядочивать строки в таблице любым удобным для вас способом, так что воспользуйтесь этим. Сортировка строк по значениям важного столбца — это, как правило, хорошая идея. Группировка строк также полезна для облегчения сравнения.

  Во многих контекстах сортировка по размеру или дате может быть более показательной, чем название. Использование канонического порядка строк (скажем, лексикографического по имени) может быть полезна для поиска элементов по имени, но это, как правило, не имеет значения, если в таблице много строк.

- *Упорядочивайте столбцы так, чтобы подчеркнуть важность или взаимоотношения.* Глаза, бегающие по всей странице слева направо, неспособны к эффективному визуальному сравнению, но соседние поля легко сравнить. По правде говоря, столбцы должны быть организованы так, чтобы группировать похожие поля, скрывая наименее важные поля справа.

- *Числа с одинаковой точностью выравнивайте по правому краю.* Визуально сравнивать 3,1415 с 39,2 в таблице — безнадежная задача: большее число должно выглядеть больше. Лучше всего выровнять их по правому краю и установить одинаковую точность для всех: 3,14 против 39,20.

- *Чтобы выделить важные записи, используйте* **выделение**, шрифт *или* цвет. Маркируйте экстремальные значения в каждом столбце так, чтобы они выделялись, это позволит сразу увидеть важную информацию. Однако. легко, **будьте** осмотрительны.

- *Избегайте слишком длинных заголовков столбцов.* Белые ленты в таблицах отвлекают и, как правило, появляются из-за того, что метки столбцов длиннее значений, которые они представляют. Используйте аббревиатуры или разделение строк, чтобы минимизировать проблему и прояснить любую двусмысленность в подписи к таблице.

Чтобы проиллюстрировать возможные огрехи, здесь приведена таблица, в которой записаны шесть свойств пятнадцати разных стран, причем порядок строк и столбцов задан случайным образом. Видите ли вы какие-либо возможные пути ее улучшения?

Страна	Площадь	Плотность	Рождаемость	Население	Смертность	ВВП
Россия	17075200	8,37	99,6	142893540	15,39	8900,0
Мексика	1972550	54,47	92,2	107449525	20,91	9000,0
Япония	377835	337,35	99,0	127463611	3,26	28200,0

Окончание таблицы

Страна	Площадь	Плотность	Рождаемость	Население	Смертность	ВВП
Британия	244820	247,57	99,0	60609153	5,16	27700,0
Новая Зеландия	268680	15,17	99,0	4076140	5,85	21600,0
Афганистан	647500	47,96	36,0	31056997	163,07	700,0
Израиль	20770	305,83	95,4	6352117	7,03	19800,0
Соединенные Штаты	9631420	30,99	97,0	298444215	6,50	37800,0
Китай	9596960	136,92	90,9	1313973713	24,18	5000,0
Таджикистан	143100	51,16	99,4	7320815	110,76	1000,0
Бирма	678500	69,83	85,3	47382633	67,24	1800,0
Танзания	945087	39,62	78,2	37445392	98,54	600,0
Тонга	748	153,33	98,5	114689	12,62	2200,0
Германия	357021	230,86	99,0	82422299	4,16	27600,0
Австралия	7686850	2,64	100,0	20264082	4,69	29000,0

Существует множество возможных порядков расположения строк (стран). Сортировка по любому столбцу является улучшением по сравнению со случайным порядком, хотя мы также можем сгруппировать их по регионам и континентам. Порядок столбцов можно сделать более понятным, поставив подобное рядом с подобным. Наконец, такие хитрости, как выравнивание чисел по правому краю, удаление неинформативных цифр, добавление запятых и выделение наибольшего значения в каждом столбце, облегчают чтение данных.

Страна	Население	Площадь	Плотность	Смертность	ВВП	Рождаемость
Афганистан	31 056 997	647 500	47,96	**163,07**	700	36,0
Австралия	20 264 082	7 686 850	2,64	4,69	29 000	**100,0**
Бирма	47 382 633	678 500	69,83	67,24	1800	85,3
Китай	**1 313 973 713**	9 596 960	136,92	24,18	5000	90,9
Германия	82 422 299	357 021	230,86	4,16	27 600	99,0
Израиль	6 352 117	20 770	305,83	7,03	19 800	95,4
Япония	127 463 611	377 835	**337,35**	3,26	28 200	99,0
Мексика	107 449 525	1 972 550	54,47	20,91	9000	92,2
Новая Зеландия	4 076 140	268 680	15,17	5,85	21 600	99,0
Россия	142 893 540	**17 075 200**	8,37	15,39	8900	99,6

Окончание таблицы

Страна	Население	Площадь	Плотность	Смертность	ВВП	Рождаемость
Таджикистан	7 320 815	143 100	51,16	110,76	1000	99,4
Танзания	37 445 392	945 087	39,62	98,54	600	78,2
Тонга	114 689	748	153,33	12,62	2200	98,5
Британия	60 609 153	244 820	247,57	5,16	27 700	99,0
Соединенные Штаты	298 444 215	9 631 420	30,99	6,50	**37 800**	97,0

6.3.2. Точечные и линейные графики

Точечные и линейные графики являются наиболее распространенными формами графического представления данных, обеспечивая визуальное представление функции $y = f(x)$, определенной набором точек (x, y). Точечные графики просто показывают точки данных, в то время как линейные графики соединяют их или интерполируют для определения непрерывной функции $f(x)$. На рис. 6.12 показано несколько различных стилей линейных графиков, различающихся по степени акцентирования представляемых точек, по сравнению с интерполированной кривой. К преимуществам линейных графиков относятся:

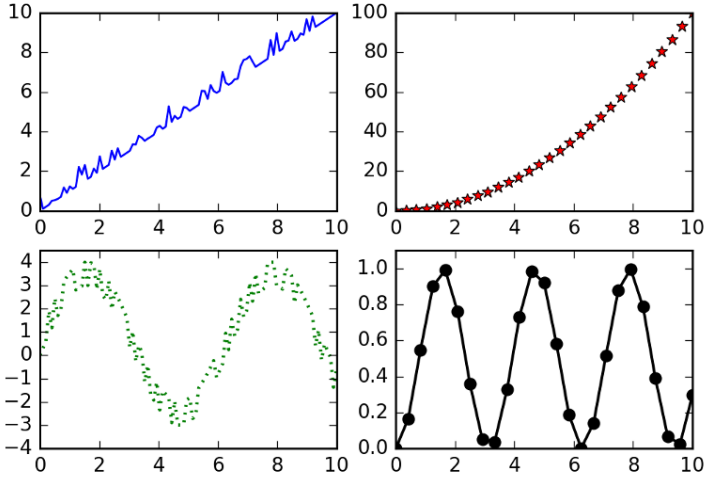


Рис. 6.12. Многие из линейных графиков, которые мы видели, поддерживаются пакетом MatPlotLib языка Python

- *Интерполяция и подгонка.* Полученная из точек интерполяционная кривая обеспечивает прогноз для $f(x)$ во всем диапазоне возможных значений x. Это позволяет нам проверять правильность или ссылаться на другие значения и четко отображать тенденции, показанные в данных.

Наложение подогнанной или сглаженной кривой на тот же график, что и исходные данные, является очень мощной комбинацией. Подход обеспечивает модель, объясняющую, что говорят данные, в то время как фактические точки позволяют нам сделать обоснованное суждение о том, насколько хорошо мы доверяем модели.

- *Точечные графики.* Отличительной чертой линейных графиков является то, что вы не обязаны показывать линию, что приводит к созданию *точечного графика* (dot plot). Соединение точек отрезками (ломаной линией) во многих ситуациях вводит в заблуждение. Если функция определена только в целочисленных точках или значения x представляют различные условия, тогда нет смысла в интерполяции между ними.

  Кроме того, ломаные линии резко отклоняются от своего пути, чтобы захватить выбросы, визуально побуждая нас сосредоточиться именно на тех точках, которые мы должны в основном игнорировать. Частые движения вверх-вниз отвлекают нас от наблюдения более общей тенденции, для чего обычно и смотрят на график.

Вот наилучшие практики применения линейных диаграмм.

- *Показать данные точек, а не просто подгонку.* Как правило, важно показывать фактические данные, а не только подогнанные или интерполированные линии.

  Главное, удостовериться, что одно не мешает другому. Чтобы ненавязчиво представить большое количество точек, мы можем (а) уменьшить размер точек, возможно, до булавочных уколов, и/или (б) осветлить оттенок точек, чтобы они могли находиться на заднем плане. Помните, что существует пятьдесят оттенков серого, и эта тонкость является ключевой.

- *Если возможно, показывайте полный диапазон переменных.* Стандартно большинство графических программ строит графики от $x_{\min}$ до $x_{\max}$ и от $y_{\min}$ до $y_{\max}$, где минимальные и максимальные значения определяются значениями входных данных. Но логические минимум и максимум зависят от контекста, и демонстрация всего диапазона может снизить фактор лжи. Логически счет должен начинаться с нуля, а не с $y_{\min}$.

  Но иногда демонстрация всего диапазона не информативна, поскольку полностью нивелирует эффект, который вы пытаетесь проиллюстрировать. Так и произошло на рис. 6.9. Одним из возможных решений является использование логарифмической шкалы для оси, чтобы совместить более широкий диапазон чисел с эффективным использованием пространства. Но если вам необходимо обрезать диапазон, поясните, что вы делаете, используя метки осей с отметки, чтобы устранить любую двусмысленность.

- *Учитывайте неопределенность при отображении средних значений.* Точки, появляющиеся на линейном или точечном графике, зачастую получают в результате усреднения нескольких наблюдений. Полученное среднее лучше отражает распределение, чем любое отдельное наблюдение. Но среднее значение имеют разные интерпретации, основанные на дисперсии. Средние значения наборов {8,5; 11,0; 13,5; 7,0; 10,0} и {9,9; 9,6; 10,3; 10,1; 10,1} составляют 10,0, но степень, в которой мы доверяем им, чтобы быть точными, существенно отличается.

  Есть несколько способов учета уровня неопределенности измерений на наших графиках. Я предпочитаю просто отображать все значения базовых данных на том же графике, что и среднее значение, используя то же значение x, что и соответствующее среднее. Эти точки будут визуально незаметны по отношению к более толстой линии тенденции, при условии, что они нарисованы в виде маленьких точек и слегка затенены, но теперь они доступны для проверки и анализа.

  Второй подход — показать стандартное отклонение σ вокруг y как ус, демонстрируя интервал $[y - \sigma, y + \sigma]$. Это представление интервала является честным, обозначая диапазон с 68% значений при нормальном распределении. Более длинные усы означают, что точность средств измерения более подозрительна, в то время как короткие усы подразумевают бо́льшую точность.

  График в виде *ящика с усами* (box plot) довольно кратко отображает диапазон и распределение значений в точке с прямоугольником. Этот прямоугольник отображает диапазон значений квартилей (25% и 75%), и он разрезан по медиане (50-й %). Усы (whiskers) обычно добавляют для того, чтобы показать диапазон самых высоких и самых низких значений. На рис. 6.13 показан график зависимости веса от роста с учетом зависимости от выборки совокупности. Медианный вес увеличивается с ростом, но не максимумом, поскольку меньшее количество точек в самом высоком участке уменьшает вероятность получения максимального значения.

  Похоже, что настоящим ученым нравятся сюжеты типа "ящик с усами", но лично я нахожу их излишними. Если вы действительно не можете представить фактические точки данных, просто покажите контурные линии, окружающие ваше среднее значение/медиану на 25-м и 75-м процентах. Это передает в точности ту же информацию, что и прямоугольник на графике ящика, но с меньшим количеством неинформативных элементов.

- *Никогда не соединяйте точки для категориальных данных.* Предположим, вы измеряете некоторую переменную (возможно, средний доход) для нескольких разных классов (скажем, для пятидесяти штатов, от Алабамы до Вайоминга). Возможно, имеет смысл отобразить это в виде точечного

графика с $1 \leq x \leq 50$, но соединение точек было бы глупо и вводило бы в заблуждение. Почему? Потому что между штатом i и штатом $i + 1$ нет значимой смежности. Такие графики лучше строить как гистограммы, обсуждаемые в разделе 6.3.4.

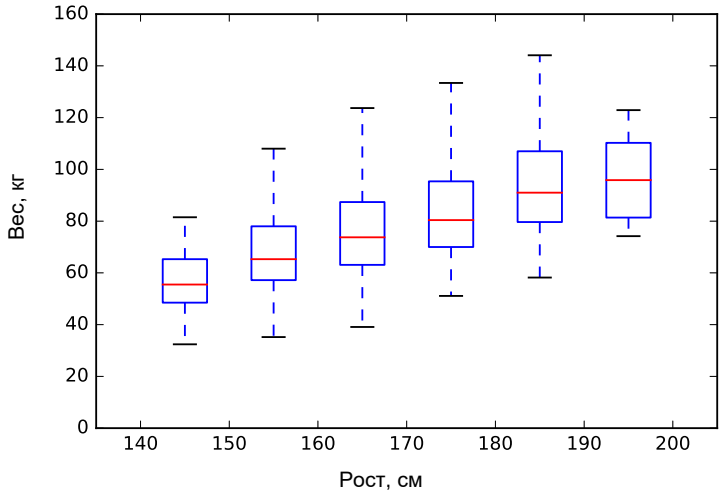


Рис. 6.13. Графики в виде ящика с усами кратко демонстрирует диапазон и квартили (т.е. медиану и дисперсию) распределения

Соединение точек ломанными линиями очень часто являются неинформативным элементом. Линии тенденций или подгонки зачастую более показательны и информативны. Постарайтесь показать сами исходные данные, желательно не очень навязчиво.

- *Используйте цвет и штриховку, чтобы различать линии/классы.* Мы нередко сталкиваемся с представлением одной и той же функции $f(x)$ для двух или более классов, возможно, доход как функция от обучения отдельно для мужчин и женщин.

  Это лучше всего изображать, назначая разные цвета линиям или точкам каждого класса. Штриховые линии (точечные, пунктирные, сплошные и полужирные) также могут быть использованы, но их обычно труднее различить, чем цвета, если только носитель не является черно-белым. На практике на одном графике можно различить от двух до четырех таких линий до того, как визуальный эффект превратится в хаос. Чтобы визуализировать большое количество групп, разбейте их на логические кластеры и используйте графики с несколькими линиями, в каждом из которых достаточно линий, чтобы их можно было различать.

6.3.3. Диаграммы рассеяния

Массивные наборы данных представляют собой реальную проблему для эффективного представления, поскольку большое количество точек легко выходит за рамки возможностей графического представления, в результате чего появляется изображение черного шара смерти. Но при правильном построении диаграммы рассеяния способны показывать тысячи двумерных точек в ясной и понятной форме.

Диаграммы рассеяния показывают значения (x, y) каждой точки в данном наборе данных. Мы использовали диаграммы рассеяния в разделе 4.1 для представления состояния массы тела людей в виде точек в пространстве рост-вес. Цвет каждой точки отражал их классификацию как нормальную, избыточный вес тела или ожирение. К лучшим практикам, связанным с диаграммами рассеяния, относятся следующие.

- *Правильный размер точек рассеяния*. В фильме *О, Боже!* Джордж Бернс как Создатель вспоминает авокадо как о свою самую большую ошибку. Почему? Потому, что он сделал косточку слишком большой. Самая большая ошибка большинства людей с диаграммами рассеяния заключается в том, что они делают точки слишком большими.

  Диаграммы рассеяния на рис. 6.14 показывают распределение BMI для более чем 1000 американцев с двумя разными размерами точек. Обратите внимание, что более мелкие точки показывают более тонкую структуру, поскольку они с меньшей вероятностью будут перекрывать и заслонять другие точки данных.

  Теперь мы видим тонкую структуру плотного ядра, все еще будучи в состоянии обнаружить легкий ореол выбросов. Стандартный размер точек у большинства программ печати составляет около пятидесяти поинтов. Но для больших наборов данных используйте меньшие точки.

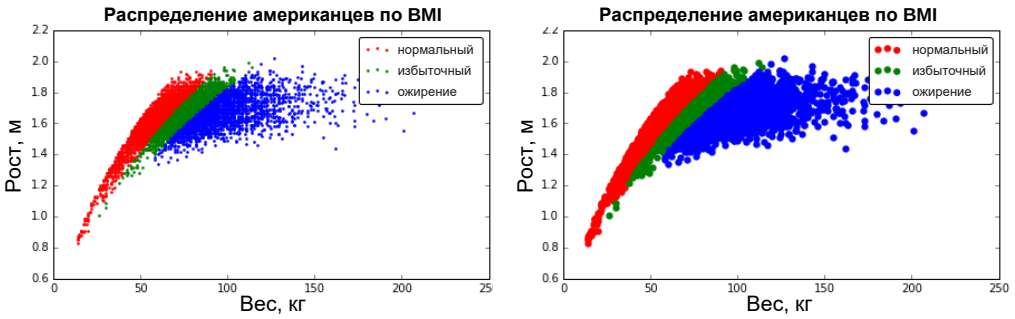


Рис. 6.14. Меньшие точки на диаграммах рассеяния (слева) показывают больше деталей, чем точки стандартного размера (справа)

- *Подкрасьте или затемните целочисленные точки, прежде чем наносить их на график рассеяния.* Диаграммы рассеяния демонстрируют шаблоны сетки, когда значения *x* и *y* имеют целочисленные значения, поскольку между ними нет плавных градаций. Эти диаграммы рассеяния выглядят неестественно, но, что еще хуже, как правило, скрывают данные, поскольку зачастую несколько точек имеют одинаковые координаты.

  Есть два резонных решения. Первое подразумевает раскрашивание каждой точки в зависимости от частоты ее появления. Такие графики называются *тепловыми картами* (heatmap) и при условии использования разумной цветовой шкалы центры концентрации становятся легко заметными. На рис. 6.15 показана тепловая карта данных о росте-весе, которая гораздо лучше выявляет места концентрации точек, чем соответствующий простой точечный график.

  Подобная идея заключается в уменьшении *непрозрачности* (что эквивалентно повышению *прозрачности*) точек, которые мы рассеиваем на графике. Стандартно точки обычно отображаются непрозрачными, что дает сплошную массу при наличии перекрывающихся точек. Но предположим, что мы позволяем этим точкам быть слегка затемненными и прозрачными. Теперь перекрывающиеся точки выглядят темнее, чем одиночные, создавая тепловую карту в нескольких оттенках серого.

  Второй подход заключается в добавлении небольшого случайного шума в каждую точку, чтобы перемешать их в пределах круга радиусом в субъединицу вокруг ее исходного положения. Теперь мы увидим полное множество точек и нарушим отвлекающую регулярность сетки.

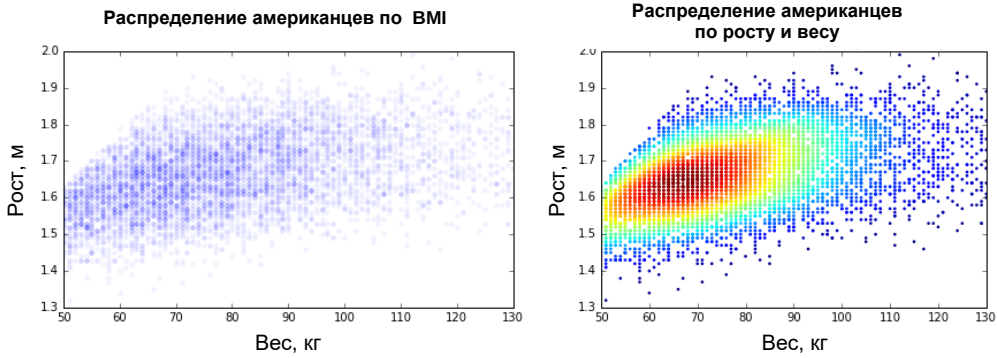


Рис. 6.15. Перекрывающиеся точки могут скрывать смысл графиков рассеяния, особенно у больших наборов данных. Уменьшение непрозрачности точек (слева) показывает некоторую тонкую структуру данных (слева). Но цветная тепловая карта куда более четко демонстрирует распределение точек (справа)

- *Спроецируйте многомерные данные в два измерения или используйте массивы попарных графиков.* Существам из нашей вселенной трудно визуализировать наборы данных в четырех и более измерениях. Наборы многомерных данных обычно можно спроецировать в два измерения, прежде чем отображать их на точечных графиках, используя такие методы, как анализ главных компонентов и самоорганизующиеся карты. Прекрасный пример приведен несколько глав спустя на рис. 11.16, где мы проецируем сто измерений в два, создавая весьма согласованное представление этого многомерного набора данных.

  Одним из преимуществ таких графиков является то, что они могут обеспечить эффективное представление вещей, которых мы иначе не могли бы увидеть. Плохо то, что эти два измерения больше ничего не значат. Конкретней, новые измерения не имеют имен переменных, способных передавать значение, поскольку каждое из двух "новых" измерений кодирует свойства всех исходных измерений.

  Альтернативным представлением является построение сетки или решетки всех попарных проекций, каждая из которых показывает только два исходных измерения. Это прекрасный способ получить представление о том, какие пары измерений коррелируют друг с другом, как мы показали на рис. 6.1.

- *Трехмерные диаграммы рассеяния помогают, только когда есть реальная структура для отображения.* В телевизионных передачах о науке о данных всегда показывают, как некий исследователь берет трехмерное облако точек и вращает его в пространстве, стремясь к некоторому важному научному пониманию. Они никогда не находят его, поскольку облако с любого выбранного направления выглядит почти так же, как и с любого другого направления. Как правило, нет точки зрения, при взгляде с которой внезапно становится ясно, как взаимодействуют размерности.

  Исключение составляют случаи, когда данные фактически были получены из структурированных трехмерных объектов, таких как результаты лазерного сканирования некой сцены. Большая часть данных, с которыми мы сталкиваемся в науке о данных, не соответствует этому описанию, поэтому не стоит ожидать интерактивной визуализации. Используйте сетку всех техник двумерных проекций, которые, по существу, похожи на облако со всех ортогональных направлений.

- *Пузырьковые графики различают по цвету и размеру, чтобы представить дополнительные размерности.* Изменение цвета, формы, размера и оттенка точек позволяет точечным графикам отображать дополнительные

измерения на пузырьковых диаграммах. Обычно это работает лучше, чем построение точек в трех измерениях.

На рис. 6.16 четко показаны четыре измерения (ВВП, ожидаемая продолжительность жизни, население и географический регион) с использованием координат x, y, размера и цвета соответственно. В такой пузырьковой диаграмме есть много чего почитать: она четко показывает корреляцию между ВВП и здоровьем (с помощью прямой линии, хотя обратите внимание на то, что значения x не являются разнесенными линейно), новый мир, как правило, богаче старого мира, и самые большие страны (Китай и Индия), как правило, находятся в середине. Некоторые богатые, но отсталые страны ведут себя ужасно из-за своего народа (например, Южная Африка), в то время как такие страны, как Куба и Вьетнам, похоже, выдвинулись выше своего веса.

6.3.4. Гистограммы и круговые диаграммы

Гистограммы и круговые диаграммы являются инструментами для представления относительных пропорций категориальных переменных. Обе работают, разделяя геометрическое целое, будь то полоса или круг, на области, пропорциональные частоте каждой группы. Оба элемента эффективны в случаях, когда необходимо сделать возможным сравнение. Действительно, разделение каждой полосы на части позволяет получить *гистограмму с накоплением* (stacked bar chart).

На рис. 6.17 представлены данные об избирателях за три года президентских выборов в США, представленные в виде круговой диаграммы и гистограммы. Синий представляет голоса за демократов, красные — голоса за республиканцев. Круги куда наглядней показывают, какая сторона победила на каждых выборах, но столбики демонстрируют, что итоги голосования республиканцев оставались довольно стабильными, пока голоса за демократов в целом росли. Обратите внимание, что эти столбцы можно легко сравнить, поскольку они выровнены по левому краю.

У некоторых критиков круговые диаграммы вызывают дикое отвращение, потому что они занимают больше места, чем необходимо, и, как правило, их сложнее читать и сравнивать. Но круговые диаграммы, возможно, лучше подходят для отображения процентов от совокупности. Многим людям они нравятся, поэтому их можно использовать в небольших количествах. Лучшие практики для гистограмм и круговых диаграмм таковы.

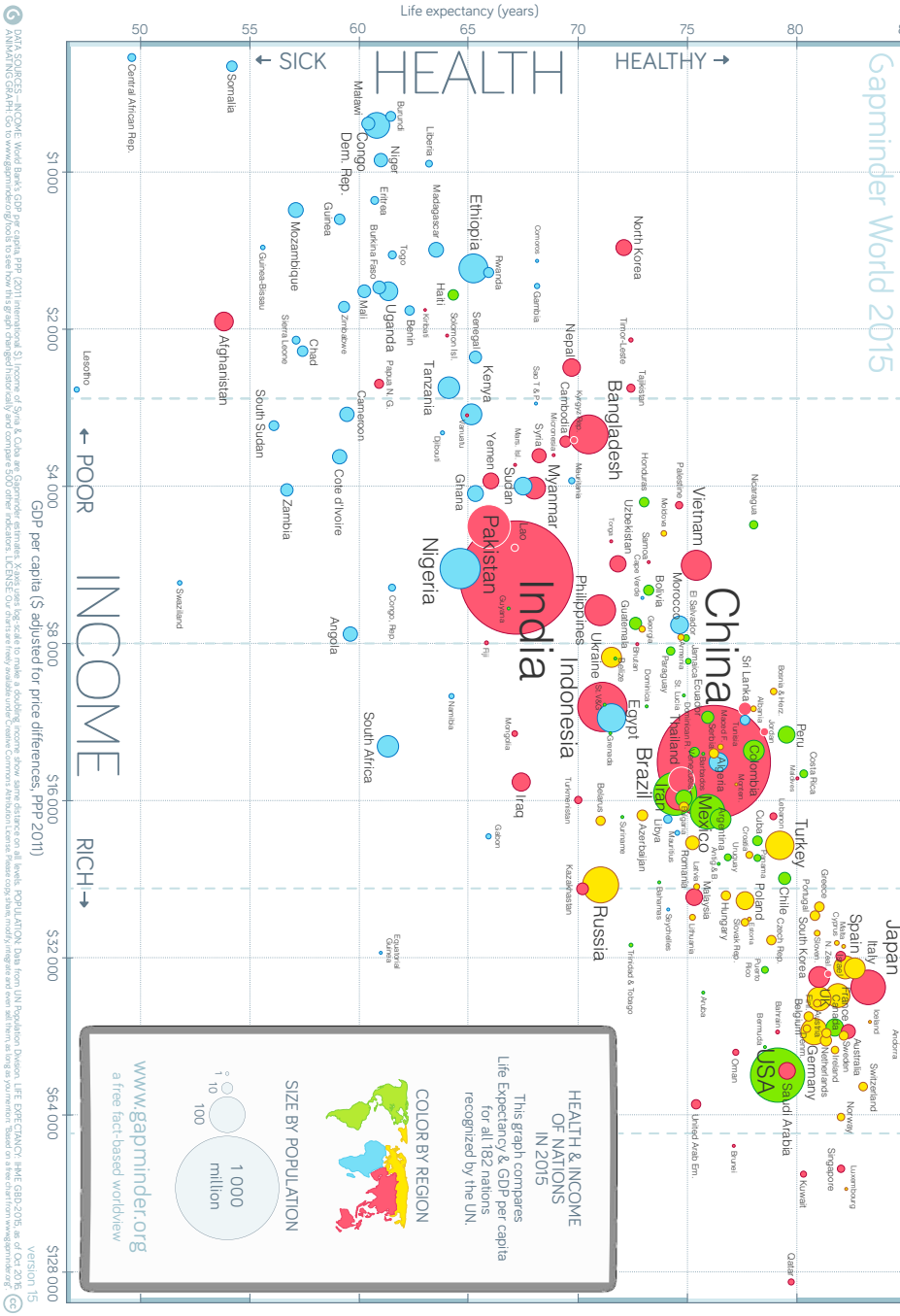


Рис. 6.16. Добавление к диаграмме рассеяния размера и цвета приводит к натуральным четырехмерным визуализациям, иллюстрирующим в данном случае свойства народов мира. На основе общедоступного графика от `http://www.gapminder.org`

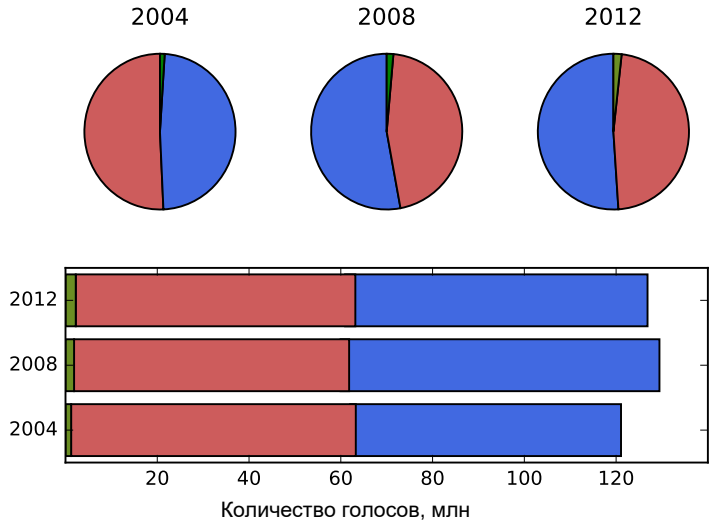


Рис. 6.17. Данные о голосах избирателей на трех президентских выборах в США. Гистограммы и круговые диаграммы отображают пропорции частот категориальных переменных. Относительные величины во временном ряду можно отобразить, модулируя разделительные линии полосы или круга

- *Непосредственная маркировка масштаба на секторах круга.* Круговые диаграммы обычно сопровождаются легендой, обозначающей то, чему соответствует каждый цвет. Это очень отвлекает, поскольку ваши глаза должны двигаться вперед и назад между легендой и кругом, чтобы интерпретировать его.

  Гораздо лучше пометить каждый сектор непосредственно, внутри сектора или сразу за его ободом. Это имеет еще одно преимущество, заключающееся в том, что не рекомендуется использовать слишком много секторов, поскольку узкие осколки обычно становятся не интерпретируемыми. Это помогает сгруппировать осколки в один сектор, обычно называемый *другое*, а затем, возможно, представить вторую круговую диаграмму, разложенную на основные компоненты *другого*.

- *Соответствующий масштаб, в зависимости от необходимости выделить абсолютную величину или пропорцию.* Круговые диаграммы предназначены для представления долей целого. При создании серии круговых диаграмм или гистограмм самым важным решением является то, хотите вы показать размер целого или только доли каждой подгруппы.

  На рис. 6.19 показаны две гистограммы с накоплением, представляющие статистику выживания на обреченном корабле "Титаник" сообразно клас-

су билетов. Гистограмма (слева) точно воспроизводит размеры каждого класса и полученные результаты. Диаграмма с одинаковой длиной столбцов (справа) лучше отражает рост смертности среди низших классов.

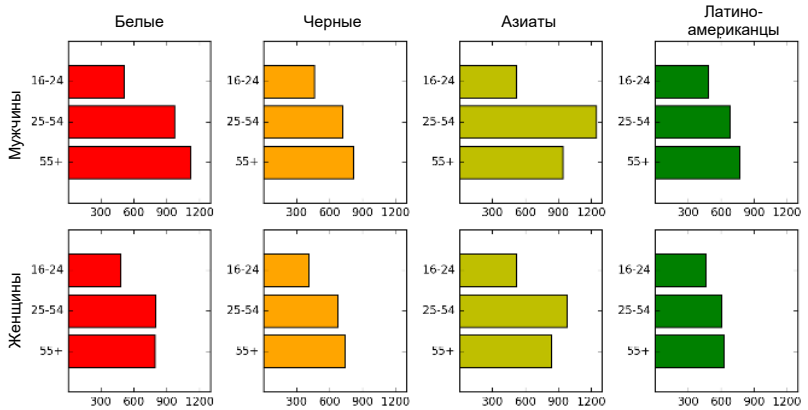


Рис. 6.18. Небольшие диаграммы/таблицы с несколькими столбцами являются отличным способом представления многомерных данных для сравнения

Круговые диаграммы также можно использовать для отображения изменений в величине за счет изменения площади круга, определяющего круговую диаграмму. Но глазу сложнее рассчитать площадь, чем длину, что затрудняет сравнение. Модуляция радиуса для отражения величины вместо площади еще более обманчива, поскольку удвоение радиуса круга умножает его площадь на четыре.

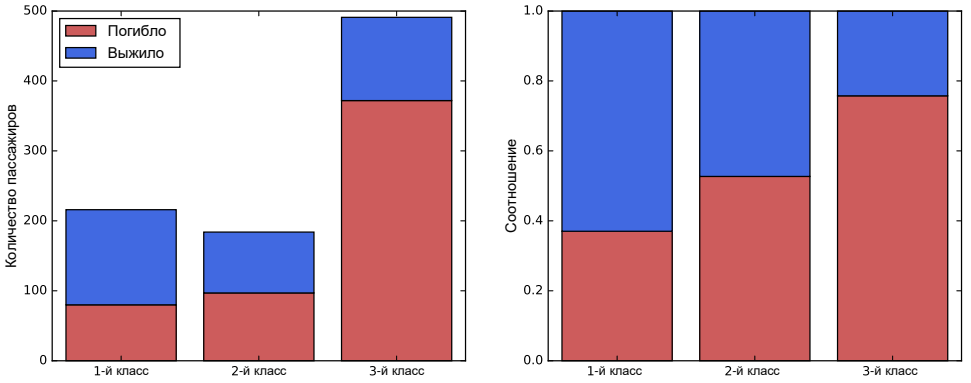


Рис. 6.19. Гистограммы с накоплением, иллюстрирующие выживаемость на обреченном корабле "Титаник" по классу билетов. Гистограмма (слева) информирует нас о размере каждого класса, но масштабированные столбцы (справа) лучше отражают пропорции. Главный вывод: вам лучше не ездить "на руле" (третий класс)

Плохая круговая диаграмма

На рис. 6.20 показаны две круговые диаграммы, демонстрирующие распределение делегатов на съезде республиканцев 2016 года по кандидатам. Левый круг является двухмерным, в то время как диаграмма справа имеет аккуратные толстые срезы, демонстрирующие глубину. Какой из них лучше передает распределение голосов?

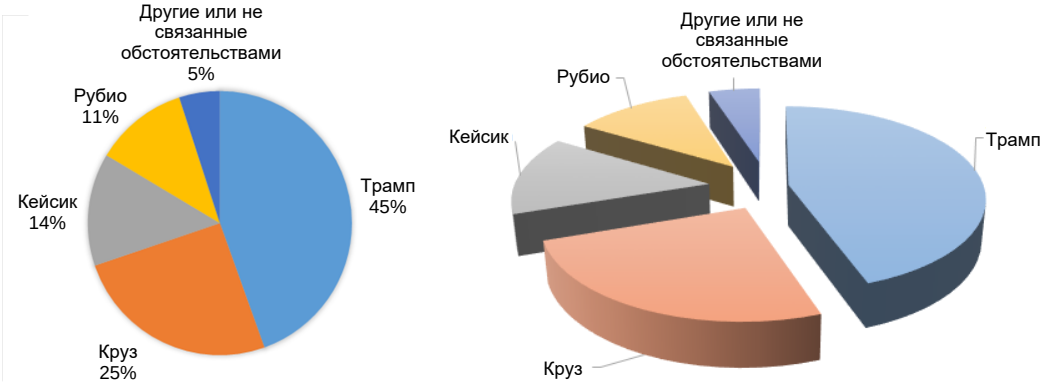


Рис. 6.20. Круговые диаграммы делегатов съезда республиканцев 2016 года по кандидатам. Какой из них лучше и почему?

Понятное дело, трехмерные эффекты и разделение — это чистой воды неинформативные элементы, который только скрывают связь между размером секторов. Фактические значения данных также исчезли, возможно, из-за того, что им не хватило места среди всех этих теней. Но зачем нам вообще круговая диаграмма? Небольшая таблица меток/цветов с дополнительным столбцом, содержащим проценты. Так будет короче и информативней.

6.3.5. Гистограммы

Интересные свойства переменных или признаков определяются плотностью их распределения. Где находится пик распределения, и находится ли мода вблизи среднего значения? Распределение симметрично или искажено? Где хвосты? Может ли оно быть бимодальным, если распределение составлено из двух или более групп населения?

Нередко мы сталкиваемся с большим количеством наблюдений за конкретной переменной и пытаемся построить для них представление. *Гистограммы* представляют собой графики наблюдаемых плотностей распределений. Когда переменная определена в широком диапазоне возможных значений относительно n наблюдений, маловероятно, что мы когда-либо увидим какое-либо точное дублирование. Однако, разделяя диапазон значений на соответствующее

количество столбцов равной ширины, мы можем накапливать различные значения и аппроксимировать лежащую в основе плотность вероятностей.

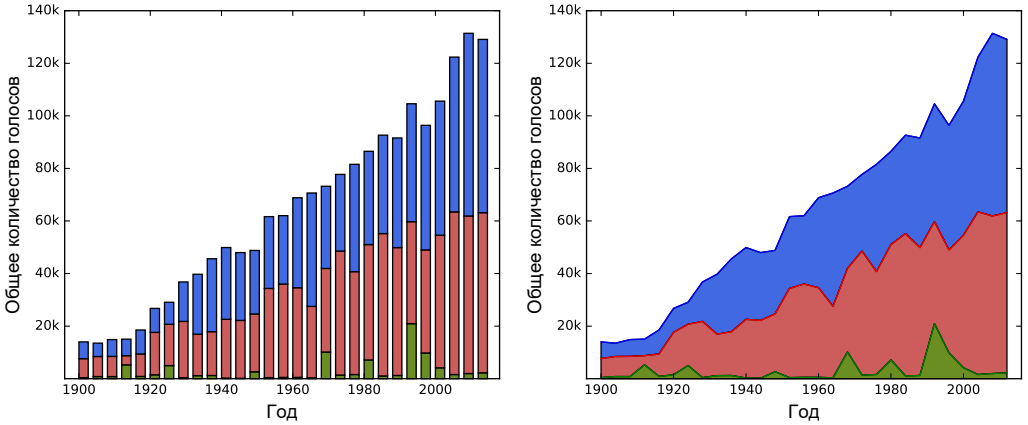


Рис. 6.21. Временные ряды итогов голосования на президентских выборах в США по партиям позволяют нам увидеть изменения в величине и распределении. Демократы показаны синим цветом, а республиканцы красным. Трудно визуализировать изменения, особенно на средних уровнях стека

Самой большой проблемой в построении гистограммы является выбор правильного количества столбцов. Слишком много столбцов, и даже в самом популярном столбце будет всего несколько пунктов. Мы обратились к столбцам, чтобы в первую очередь решить именно эту проблему. Но, используя слишком мало столбцов, вы не увидите достаточно деталей, чтобы понять форму распределения.

На рис. 6.22 иллюстрируется влияние размера столбца на внешний вид гистограммы. Графики в верхнем ряду соответствуют 100 000 пунктам из нормального распределения в десять, двадцать и пятьдесят столбцов соответственно. Когда есть достаточно пунктов, чтобы заполнить пятьдесят столбцов, распределение справа выглядит красиво. Графики в нижнем ряду имеют только 100 точек, поэтому график с тридцатью столбцами справа слишком разрежен и нечеток. Здесь семь столбцов (показанных слева), кажется, создают наиболее представительный вид.

Невозможно дать четкие и однозначные правила для выбора лучшего количества столбцов b при демонстрации ваших данных. Поймите, что на глаз вы никогда не сможете различить более ста столбцов, что обеспечивает логичную верхнюю границу. В общем, мне нравится видеть в среднем от 10 до 50 пунктов на столбец, чтобы все было гладко, поэтому $b = \lceil n/25 \rceil$ дает первое разумное предположение. Однако имеет смысл поэкспериментировать с различными

значениями *b*, поскольку правильное количество столбцов будет работать намного лучше, чем неправильное. Вы узнаете это, когда увидите.

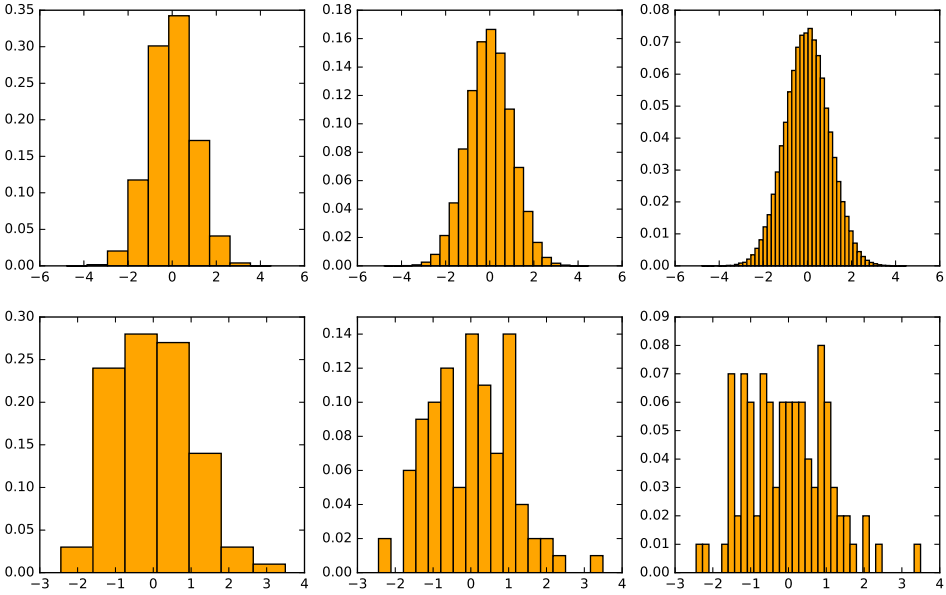


Рис. 6.22. Гистограммы данного распределения могут сильно отличаться в зависимости от количества столбцов. Для большого набора данных выгодно наличие многих столбцов (вверху). Но структура меньшего набора данных выглядит лучше, когда в каждом столбце нетривиальное количество элементов

Лучшие практики для гистограмм таковы.

- *Переведите вашу гистограмму в pdf.* Как правило, мы интерпретируем наши данные как наблюдения, которые аппроксимируют функцию плотности вероятности (pdf) лежащей в основе случайной переменной. Если это так, то лучше маркировать ось *y* долями элементов в каждом сегменте, вместо общего количества. Это особенно верно для больших наборов данных, где столбцы достаточно высоки, чтобы мы не беспокоились о поддержке точного уровня.

  На рис. 6.23 показаны те же данные, изображенные справа в виде pdf, в отличие от гистограммы. Форма одинакова на обоих графиках: все, что изменилось — это метка на оси *y*. И все же результат интерпретировать легче, поскольку он основан на вероятностях, а не на подсчетах.

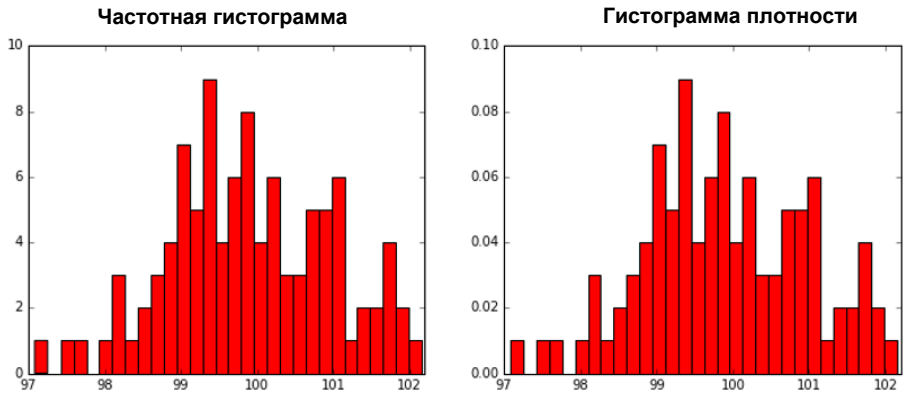


Рис. 6.23. Разделение значений на общее количество приводит к графику плотности вероятности, который в целом можно интерпретировать лучше, даже если формы идентичны

- *Учитывайте cdf.* Кумулятивная функция плотности (cdf) является интегралом pdf, и эти две функции содержат абсолютно одинаковую информацию. Поэтому рассмотрите возможность использования cdf вместо гистограммы для представления своего распределения, как показано на рис. 6.24.

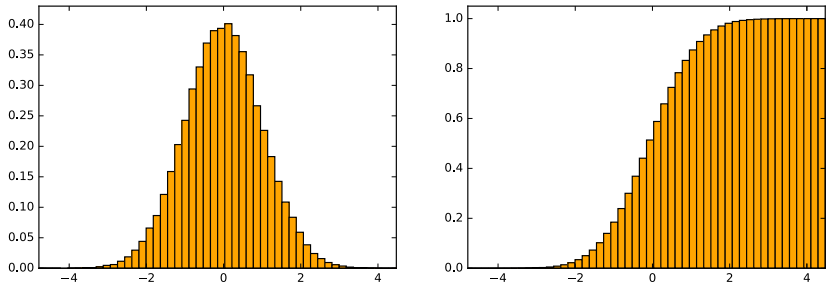


Рис. 6.24. По правде говоря, гистограммы лучше отображают пики в распределении, но cdf лучше демонстрируют хвосты

Преимуществом построения графика в формате cdf является то, что он не полагается на параметр количества столбцов, поэтому он обеспечивает истинное, безошибочное представление ваших данных. Вспомните, как здорово выглядели файлы cdf в критерии Колмогорова–Смирнова на рис. 5.13. Мы рисуем cdf в виде линейного графика с $n + 2$ точками для n наблюдений. Первая и последняя точки $(x_{\min} - \varepsilon, 0)$ и $(x_{\max} + \varepsilon, 1)$. Затем мы сортируем наблюдения, чтобы получить $S = \{s_1, \ldots, s_n\}$ и график $(s_i, i/n)$ для всех i.

Кумулятивные распределения немного сложней для чтения, чем гистограммы. Функция cdf монотонно возрастает, поэтому пиков в распределении нет. Вместо этого мода отмечена самым длинным вертикальным

отрезком. Но cdf гораздо лучше выделяют хвосты распределения. Причина ясна: малые значения в хвостах скрыты осью на гистограмме, но накапливаются в видимый материал в графике cdf.

6.3.6. Карты данных

Карты используют пространственное расположение регионов для представления мест, концепций или предметов. Мы все освоили навигацию по миру с помощью карт — навыки, которые помогают понять соответствующие визуализации.

Традиционные карты данных используют цвет или оттенок, чтобы выделить свойства областей на карте. На рис. 6.25 (слева) каждый штат окрашен в соответствии с тем, проголосовали ли он за Барака Обаму (синий) или за его оппонента Митта Ромни (красный) на президентских выборах 2012 года в США. Карта ясно показывает политическое разделение страны. Северо-восточное и западное побережья сплошь синие, а средний запад и юг красные.

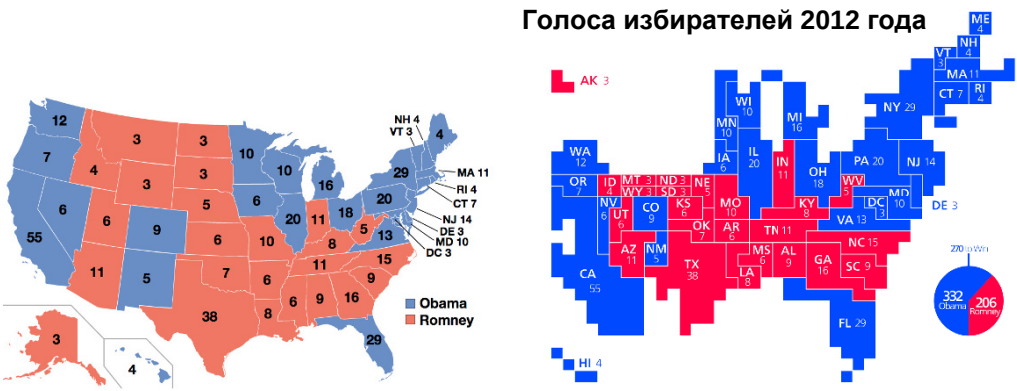


Рис. 6.25. Карты с результатов президентских выборов в США в 2012 году. Получатель голосов избирателей каждого штата представлен на карте данных (слева), в то время как картограмма, где площадь каждого штата пропорциональна количеству голосов (справа), лучше отражает величину победы Обамы. Источник: изображения из Википедии

Карты не ограничиваются географическими регионами. Самая мощная карта в истории научной визуализации — это периодическая таблица элементов. Соответствующие области показывают, где находятся металлы и инертные газы, а также места, где должны были находиться неоткрытые элементы. Периодическая таблица представляет собой карту с достаточным количеством деталей, на которую могут ориентироваться практикующие химики, ее также легко понять и школьникам.

Что дает периодической таблице такую визуализационную мощь?

- *Карте есть что рассказать.* Карты ценны, когда они предоставляют информацию, на которую стоит ссылаться или использовать. Периодическая таблица представляет собой *правильную* визуализацию элементов из-за структуры электронных оболочек и их важности для химических свойств и связей. Карта подвергается повторному изучению, поскольку она способна сказать нам важные вещи.

  Карты данных выглядят захватывающе, поскольку разбивка переменных по регионам так часто приводит к интересным историям. Области на картах обычно отражают культурную, историческую, экономическую и языковую преемственность, поэтому явления, происходящие от любого из этих факторов, как правило, четко проявляются на картах данных.

- *Регионы являются смежными, и смежность что-то значит.* Смежность областей на рис. 6.26 отражена его цветовой схемой, объединяющей элементы, имеющие общие свойства, такие как щелочные металлы и инертные газы. Два элемента, стоящие рядом друг с другом, обычно означают, что они имеют что-то общее.

Рис. 6.26. Периодическая таблица логически группирует элементы, отражая их химические свойства. Источник: `http://sciencenotes.org`

- *Квадраты достаточно большие, чтобы увидеть содержимое.* Важным решением при канонизации периодической таблицы было размещение редкоземельных (элементы 57–71) и актиноидных (элементы 89–103) металлов.

Они условно представлены в виде двух нижних строк, но логически располагаются в теле таблицы в немаркированных зеленых квадратах.

Однако обычный рендеринг позволяет избежать двух проблем. Чтобы сделать все "правильно", эти элементы будут либо сжаты в безнадежно тонкие полоски, либо таблица должна будет стать вдвое шире, чтобы вместить их.

- *Она не слишком достоверна.* Улучшение реальности ради лучших карт — это давняя и почтенная традиция. Напомним, что проекция Меркатора искажает размеры объектов возле полюсов (да,

  Гренландия, я имею в виду именно вас), чтобы сохранить их форму.
- *Картограммы* (cartogram) — это карты, искаженные так, что регионы отражают некоторую базовую переменную, такую как население. На рис. 6.25 (справа) показаны результаты выборов 2012 года на карте, где площадь каждого штата пропорциональна его населению/голосам избирателей. Только теперь масштабы победы Обамы становятся ясными: эти гигантские красные штаты Среднего Запада уменьшаются до соответствующего размера, в результате чего карта становится более синей, чем красной.

6.4. Примеры правильной визуализации

Выработка собственной эстетики визуализации дает вам язык, на котором можно говорить о том, что вам нравится, а что нет. Теперь я призываю вас применить свое суждение для оценки достоинств и недостатков определенных диаграмм и графиков.

В этом разделе мы рассмотрим несколько классических визуализаций, которые я считаю великолепными. Есть так много ужасной графики, которой я бы хотел их противопоставить, но меня учили, что нехорошо смеяться над людьми.

Это особенно верно, когда существуют ограничения авторского права на использование таких изображений в подобной книге. Тем не менее я настоятельно рекомендую вам посетить сайт `http://wtfviz.net/`, чтобы увидеть коллекцию потрясающих диаграмм и графиков. Многие из них довольно забавны, по причинам, которые вы теперь сможете сформулировать, используя идеи этой главы.

6.4.1. Расписание поездов Маре

Тафти указывает график железнодорожного расписания Этьена-Жюля Маре (E.J. Marey) как образец графического дизайна. Он показан на рис. 6.27. Часы

дня представлены на оси x. На самом деле этот прямоугольный участок является цилиндром, резаннымм по метке в 6 утра и развернутым. Ось y представляет все станции на линии Париж — Лион. Каждая линия представляет маршрут конкретного поезда, сообщая, где он должен находиться в каждый момент времени.

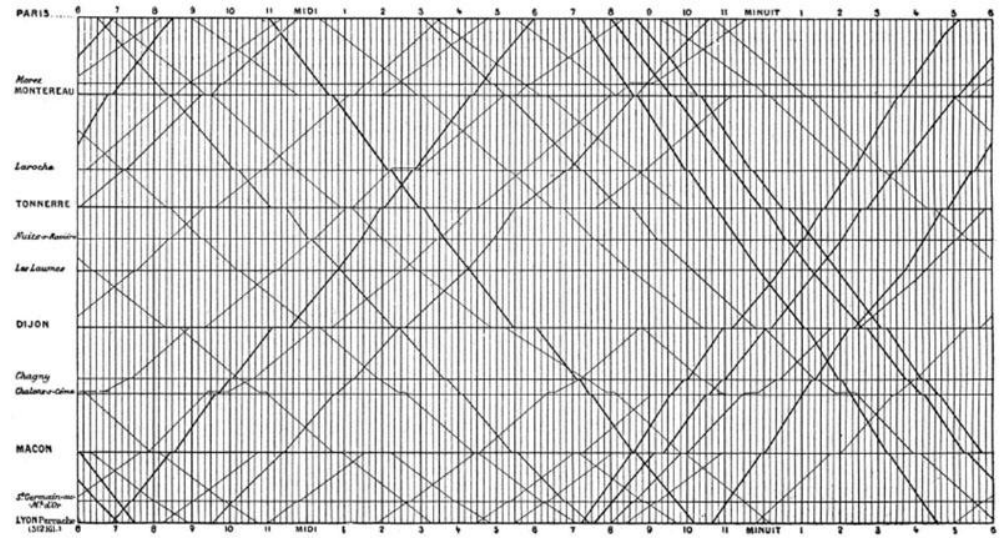


Рис. 6.27. Расписание поездов Маре демонстрирует положение каждого поезда как функцию от времени

Обычные расписания поездов — это таблицы, в которых есть столбец для каждого поезда, строка для каждой станции и запись (i, j), сообщающая время прибытия поезда j на станцию i. Такие таблицы полезны, они способны сказать нам, во сколько прийти, чтобы сесть на наш поезд. Но дизайн Маре дает гораздо больше информации. Что еще вы можете увидеть здесь, чего не можете найти в обычном расписании?

- *Насколько быстро движется поезд?* Скорость определяет угол наклона линии. Чем быстрее поезд, тем больше абсолютный наклон. Более медленные поезда отмечены более пологими линиями, поскольку им требуется больше времени, чтобы покрыть данную дистанцию.

  Особый случай здесь — это определение периодов, когда поезд стоит на станции. В это время линия горизонтальна, что указывает на отсутствие движения по линии вниз.

- *Когда поезда проходят мимо друг друга?* Направление поезда определяется наклоном соответствующей линии. Поезда в северном направлении имеют положительный уклон, а в южном направлении — отрицательный.

Два поезда проходят друг против друга в точках пересечения двух линий, сообщая пассажирам, когда смотреть в окно и махать.

- *Когда наступает час пик?* Около 7 вечера поезда отправляются из Парижа и Лиона, что говорит о том, что это самое популярное время для путешествий. Поездка, как правило, занимала около одиннадцати часов, так что это должен быть поезд спального типа, поскольку путешественники должны были прибыть в пункт назначения рано утром на следующий день.

Время отправления поездов на вокзале, конечно же, тоже есть. Каждая станция отмечена горизонтальной линией, поэтому ищите время, когда поезда пересекают вашу станцию в правильном направлении.

Единственное, что я могу сказать, так это то, что было бы еще лучше с более редкой сеткой данных. Никогда не заключайте ваши данные за решетку!

6.4.2. Карта распространения холеры Сноу

Эта известная карта данных изменила ход истории медицины. Холера была ужасной болезнью, которая убила множество людей в городах 19-го века. Чума приходила внезапно и поражала людей, оставаясь тайной для науки того времени.

Джон Сноу (John Snow) нанес на карту улиц Лондона случаи холеры, вызванные эпидемией 1854 года, в надежде увидеть закономерность. Каждая точка на рис. 6.28 изображает домовладения, пораженные этой болезнью. Что вы видите?

Сноу заметил группу случаев, сосредоточенных вокруг Брод-стрит. В центре скопления был крест, обозначающий колодец, где жители брали питьевую воду. Источник эпидемии был прослежен до ручки единственного водяного насоса. Они изменили ручку, и люди вдруг перестали болеть. Это доказало, что холера была инфекционным заболеванием, вызванным загрязненной водой, и указало путь к ее предотвращению.

6.4.3. Карта погоды в Нью-Йорке

Стоит терпеть нью-йоркскую зиму, чтобы увидеть график, который появляется в *Нью-Йорк таймс* каждый январь и подводит итог погоды предыдущего года. На рис. 6.29 представлена независимая передача тех же данных, в которой показано, почему эта диаграмма интересна. Для каждого дня года мы видим на графике высокую и низкую температуры вместе с историческими данными, чтобы представить ее в контексте: среднесуточный максимум и минимум, а также самые высокие/низкие температуры, когда-либо зарегистрированные в эту дату.

Рис. 6.28. Наибольшая смертность от холеры зарегистрирована вокруг насоса на Брод-стрит, указывая на источник эпидемии

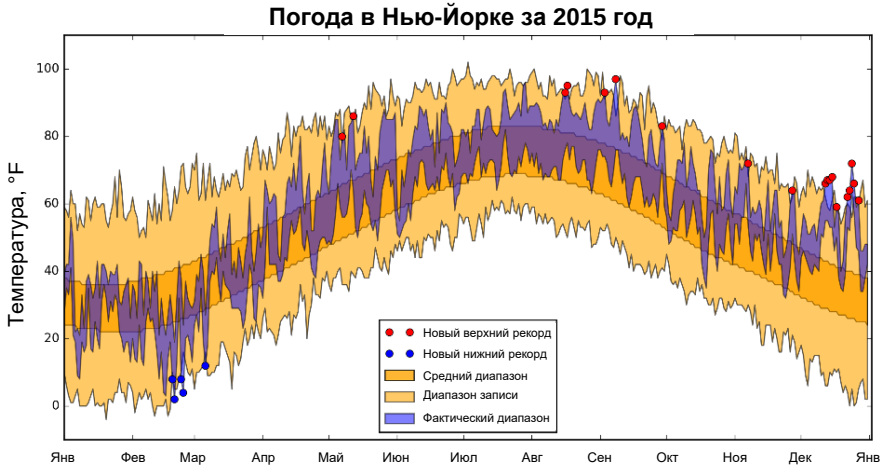


Рис. 6.29. Этот обзор погоды за год показывает четкую историю в более чем 2000 чисел

Так что же в этом такого великолепного? В первую очередь, он показывает $6 \cdot 365 = 2190$ чисел в согласованном виде, что облегчает сравнение по синусоидальной кривой времен года.

- Мы можем сказать, когда были жаркие и холодные периоды и как долго они длились.
- Мы можем сказать, в какие дни были большие перепады температуры, а когда столбик термометра почти не двигался.
- Мы можем сказать, была ли погода необычной в этом году. Был ли это необычно жаркий или холодный год или и то, и другое? Когда были установлены рекордные максимумы/минимумы, и когда они приближались к эти условиям.

Этот единый график богат, понятен и информативен. Пусть он вас вдохновит.

6.5. Чтение графиков

То, что ты видишь, не всегда то, что получаешь. Я видел много графиков, принесенных мне моими студентами за многие годы. Некоторые были удивительны, а большинство достаточно хороши, чтобы выполнить свою работу.

Но я также неоднократно вижу сюжеты с одинаковыми основными проблемами. В этом разделе (для некоторых из моих любимых мозолей) я представляю оригинальный сюжет вместе со способом решения проблемы. Имея опыт, вы сможете определить такие проблемы, просто взглянув на исходный сюжет.

6.5.1. Сокрытие распределения

На рис. 6.30 показана частота для 10 000 английских слов, отсортированных по частоте. Это не выглядит очень увлекательно: все, что вы можете увидеть, — это единственная точка в (1, 2,5). Что случилось, и как вы можете это исправить?

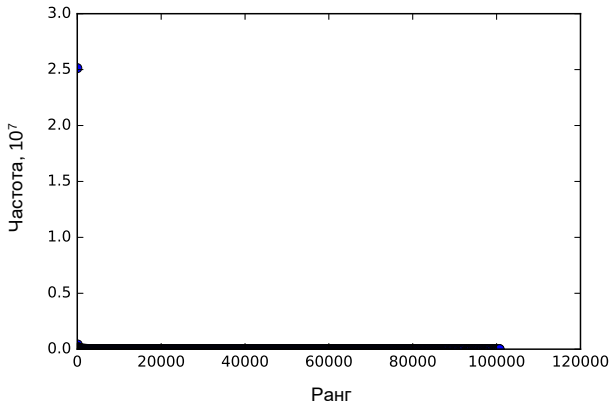


Рис. 6.30. Точечный график частоты слов по рангу. Что вы видите?

Если вы посмотрите на рисунок достаточно долго, то заметите, что на самом деле точек гораздо больше. Но все они находятся на линии $y = 0$ и перекрывают друг друга до такой степени, что образуют неразличимую массу.

Внимательный читатель поймет, что эта единственная точка на самом деле является выбросом, величина которого настолько велика, что сокращает все остальные значения до нуля. Естественная реакция — удалить самую большую точку, но, что любопытно, остальные точки будут выглядеть примерно так же.

Проблема в том, что это распределение по степенному закону, а график степенного закона в линейном масштабе ничего не показывает. Ключевым моментом здесь является построение графика в логарифмическом масштабе, как показано на рис. 6.31 (слева). Теперь вы можете увидеть точки и отображение рангов по частоте. Еще лучше построить его в масштабе логарифм-логарифм, как показано на рис. 6.31 (справа). Прямая линия здесь подтверждает то, что мы имеем дело со степенным законом.

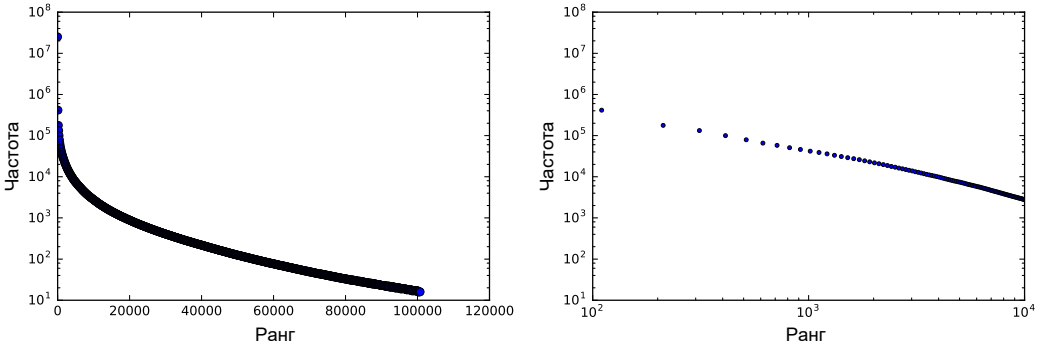


Рис. 6.31. Частота 10 000 английских слов. Отображение частот слов в логарифмическом масштабе (слева) или, что еще лучше, в двойном логарифмическом масштабе показывает, что это степенной закон (справа)

6.5.2. Переинтерпретация дисперсии

В биоинформатике человек пытается узнать, как устроена жизнь, глядя на данные. На рис. 6.32 (слева) представлен график энергии фолдинга генов как функции их длины. Посмотрите на этот график и скажите, можете ли вы сделать открытие.

Вполне понятно, что там что-то происходит. Для генов длиной более 1500 график начинает прыгать, создавая некоторые очень низкие значения. Обнаружили ли мы только что, что энергия изменяется обратно пропорционально длине гена?

Рис. 6.32. Энергия фолдинга генов как функция их длины. Ошибочная дисперсия для сигнала: крайние значения на левом рисунке являются артефактами от усреднения небольшого числа выборок

Нет. Мы просто переоценили дисперсию. Первая подсказка заключается в том, что то, что выглядело очень устойчивым, начинает становиться бесполезным с увеличением длины. Большинство генов очень коротко по длине. Таким образом, точки на правой стороне левого графика основаны на очень небольшом количестве данных. Среднее значение по лишь нескольким точкам совсем не так надежно, как среднее по многим точкам. Действительно, частотный график количества генов по длине (справа) показывает, что их счет уменьшается до нуля там, где график начинает прыгать.

Как мы можем это исправить? Правильнее всего было бы ограничить график, показывая только значения с достаточной поддержкой данных, возможно, длиной 500 или чуть больше. Кроме того, мы можем отсортировать их по длине и взять среднее значение, чтобы доказать, что эффект прыжков исчезает.

6.6. Интерактивная визуализация

Диаграммы и графики, которые мы обсуждали до сих пор, представляли собой статические изображения, предназначенные для изучения зрителем, но не манипулирования ими. Методы интерактивной визуализации становятся все более важными для исследовательского анализа данных.

Мобильные приложения, блокноты и веб-страницы с интерактивными виджетами визуализации могут быть особенно эффективными при представлении данных и их распространении среди широкой аудитории для изучения. Предоставление зрителям возможности играть с фактическими данными позволяет гарантировать, что представленная графикой история является правдивой и полной.

Если вы собираетесь просматривать свои данные в Интернете, имеет смысл сделать это с помощью интерактивных виджетов. Как правило, это расширения основных графиков, которые мы описали в этой главе, с такими функциями, как предоставление всплывающих окон с дополнительной информацией, когда пользователь щелкает на точках, или возможностью изменять диапазоны шкалы с помощью ползунков.

У интерактивной визуализации есть несколько потенциальных недостатков. В первую очередь, с ее помощью труднее передать другим то, что вы видите, по сравнению со статическими изображениями. Другие могут не видеть то же самое, что и вы. Снимки экрана интерактивных систем обычно не могут сравниться по качеству с графикой публикации, оптимизированной на традиционных системах. Проблема с тем, что вы видите, заключается в том, что у систем "*что видишь, то и получаешь*" (what you see is what you get — WYSIWYG) вы, как правило, получаете только то, что видите. Интерактивные системы лучше всего подходят для исследования, а не для презентации.

В интерактивной визуализации нередко возникают излишества. Кнопки и функции добавляются, потому что они могут добавляться, но визуальные эффекты, которые они добавляют, могут отвлекать, а не подчеркивать сообщение. Вращающиеся трехмерные облака точек всегда выглядят круто, но мне трудно их интерпретировать, и они очень редко оказываются познавательными.

Чтобы данные рассказали историю, требуется некоторое понимание того, как рассказывать истории. Фильмы и телевидение представляют современное состояние в повествовательной презентации. Как и они, лучшие интерактивные презентации демонстрируют повествование, например, движение во времени или перебор альтернативных гипотез. Для вдохновения я призываю вас посмотреть выступление покойного Ханса Рослинга (Han Rosling) на TED[1] с использованием анимированных пузырьковых диаграмм, чтобы представить историю социального и экономического развития всех стран мира.

Последние тенденции облачной визуализации позволяют загружать свои данные на сайт, такой как Google Charts `https://developers.google.com/chart/`, чтобы вы могли воспользоваться интерактивными инструментами визуализации и предоставляемыми ими виджетами. Эти инструменты создают очень хорошие интерактивные сюжеты и просты в использовании.

Возможным камнем преткновения является безопасность, поскольку вы передаете свои данные третьим лицам. Я надеюсь и верю, что аналитики ЦРУ имеют доступ к своим собственным внутренним решениям, которые сохраняют секретность их данных. Но не стесняйтесь экспериментировать с этими инструментами в менее секретных ситуациях.

---

[1] `https://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen`

6.7. Случай из жизни: текстовая карта мира

Мой самый большой опыт в визуализации данных стал результатом нашей крупномасштабной системы анализа новостей/настроений. Сайт TextMap предоставляет панель анализа новостей для любой сущности, чье имя фигурировало в новостных статьях того времени. Страница Барака Обамы показана на рис. 6.33. Наша информационная панель была составлена из множества подкомпонентов.

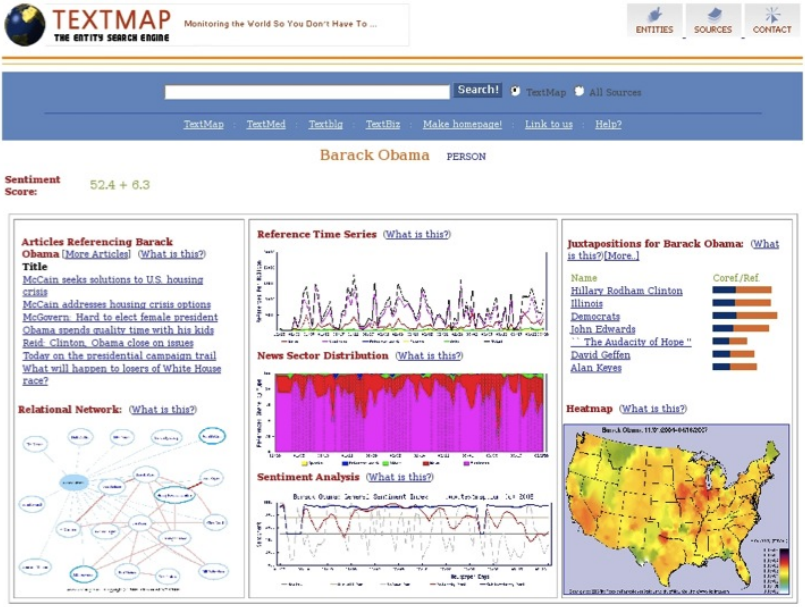


Рис. 6.33. Панель TextMap для Барака Обамы в 2008 году

- *Справочные временные ряды*. Здесь мы сообщаем, как часто данная сущность появлялась в новостях в зависимости от времени. Пики соответствовали более важным новостным событиям. Кроме того, мы разделили эти подсчеты в соответствии с появлениями в каждом разделе газеты: новости, спорт, развлечения или бизнес.
- *Распространение по секторам новостей*. Этот график содержит *в точности* те же данные, что и в эталонном временном ряду, но представлен в виде графика с областями и накоплением, чтобы показать долю ссылок на сущность в каждом разделе статьи. Обама четко представляется в качестве новостной фигуры. Но и другие люди демонстрировали интересные переходы, такие как Арнольд Шварценеггер, когда он перешел от кинематографа к политике.

- *Анализ настроений.* Здесь мы представляем временные ряды, измеряющие отношение к сущности, за счет представления нормализованной разности числа позитивных упоминаний в новостях и общего количества ссылок. Таким образом, нуль представляет нейтральную репутацию, и мы предоставили центральную контрольную линию, чтобы показать текущее положение и перспективу. Здесь отношение к Обаме снижается и изменяется согласно событиям, но в целом остается с правой стороны линии.

  Было приятно наблюдать, как плохие вещи случаются с плохими людьми, глядя на падение отношения к ним в новостях. Насколько я помню, самое низкое из когда-либо достигнутых настроений в новостях было в отношении мамы, которая добилась дурной славы в результате киберзапугивания одного из социальных конкурентов ее дочери, совершившей самоубийство.

- *Тепловая карта.* Здесь мы представили карту данных, демонстрирующую относительную частоту упоминаний сущности. Большое красное пятно Обамы вокруг Иллинойса связано с тем, что он был там сенатором в то время, когда впервые баллотировался на пост президента. Многие классовые организации демонстрировали сильные региональные предубеждения, например спортивные деятели и политики, но в меньшей степени такие деятели из области развлечений, как кинозвезды и певцы.

- *Сопоставления и реляционные сети.* Наша система построила сеть на новостных сущностях, связывая две сущности всякий раз, когда они упоминались в одной и той же статье. Сила этого отношения может быть измерена количеством статей, связывающих их вместе. Наиболее сильные из этих ассоциаций представлены как сопоставления, их частота и сила показаны гистограммой в логарифмическом масштабе.

  На самом деле мы пока не говорили о сетевой визуализации, но основная идея заключается в том, чтобы расположить вершины так, чтобы соседи находились рядом друг с другом, т.е. чтобы края были короткими. Более сильные дружеские отношения показаны более толстыми линиями.

- *Статьи по теме.* Мы предоставили ссылки на репрезентативные новостные статьи с упоминанием данного лица.

Одним из недостатков нашей панели было то, что она не была интерактивной. Фактически она была антиинтерактивна: единственной анимацией было GIF-изображение земного шара, вращающегося в левом верхнем углу. Многие из представленных нами графиков требовали большого объема доступа к данным из нашей неуклюжей базы данных, в частности тепловой карты. Поскольку

она не может быть отображена в интерактивном режиме, мы предварительно вычислили эти карты в автономном режиме и показывали их по требованию.

После того как Михаил (Mikhail) обновил нашу инфраструктуру (см. случай из жизни в разделе 12.2), у нас появилась возможность поддерживать некоторые интерактивные сюжеты, в частности временны*е* ряды. Мы разработали новый пользовательский интерфейс под названием TextMap Access, который позволяет пользователям играть с нашими данными.

Но когда General Sentiment приобрела лицензию на эту технологию, она первым делом избавилась от нашего пользовательского интерфейса. Это не имело смысла для бизнес-аналитиков, которые были нашими клиентами. Это было слишком сложно. Есть существенная разница между привлекательной поверхностью и реальной эффективностью интерфейса для пользователей. Посмотрите внимательно на нашу панель инструментов TextMap: в нескольких местах были кнопки (Что это?). Это было признаком слабости: если бы наша графика была достаточно интуитивно понятной, нам бы они не понадобилась.

Хоть я ворчал по поводу нового интерфейса, я, несомненно, был неправ. В General Sentiment работала группа аналитиков, которые использовали нашу систему в течение всего дня и были готовы поговорить с нашими разработчиками. Интерфейс эволюционировал, чтобы лучше служить им. Лучшие интерфейсы создаются в ходе диалога между разработчиками и пользователями.

Какие уроки можно извлечь из этого опыта? Я все еще нахожу много интересного в этой панели: презентация была достаточно насыщенной, чтобы по-настоящему раскрыть интересные вещи о том, как устроен мир. Но не все с этим согласились. Разные клиенты предпочитали разные интерфейсы, поскольку они делали в нем разные вещи.

Одним из уроков здесь является возможность создания альтернативных представлений одних и тех же данных. Эталонные временны*е* ряды и распределение по новостному сектору были точно такими же данными, но представили совершенно разные идеи, когда были предоставлены по-иному. Все песни сделаны из одних и тех же нот, но то, как вы их аранжируете, делает музыку разной.

6.8. Дополнительная информация

Я настоятельно рекомендую книги Эдварда Тафти [58, 59, 60] всем, кто интересуется искусством научной визуализации. Вам даже не нужно их читать. Достаточно взглянуть на картинки, чтобы увидеть контрасты между хорошей и плохой графикой, а также графики, которые действительно передают истории в данных.

Хорошим источником информации по визуализации данных является также книга Фью [62]. Интересные блоги о визуализации данных находятся по адресам `http://flowingdata.com` и `http://viz.wtf`. Первый сосредоточивается на великолепных визуализациях, второй — на поисках недостатков. История холерной карты Сноу приводится в книге Джонсона [63].

Пример удаления неинформативных элементов из раздела 6.2.3 был навеян примером Тима Брея `http://www.tbray.org`. Квартет Анскомба впервые был представлен в [64]. Базовая архитектура нашей системы анализа новостей Lydia/TextMap описана у Ллойда и др. [65] наравне с дополнительными статьями, описывающими тепловые карты [66] и пользовательский интерфейс Access [67].

6.9. Упражнения

Исследовательский анализ данных

6.1. *[5]* Дайте ответы на вопросы, связанные со следующими наборами данных, которые доступны по адресу `http://www.data-manual.com/data`.

(a) Проанализируйте набор данных *movie* (фильмы). Каков диапазон кассовых сборов фильмов в Соединенных Штатах? Какой тип фильмов наиболее успешен на рынке? Комедии? PG-13? Драмы?

(b) Проанализируйте набор данных *Manhattan rolling sales* (о продаже недвижимости на Манхэттене). Где в Манхэттене находится самая дорогая и самая дешевая недвижимость? Какова связь между продажной ценой и квадратным футом?

(c) Проанализируйте набор данных *2012 Olympic* (олимпиада 2012 года). Что вы можете сказать о связи между населением страны и количеством завоеванных ею медалей? Что вы можете сказать о соотношении между количеством женщин и мужчин, а также ВВП этой страны?

(d) Проанализируйте набор данных *GDP per capita* (о ВВП на душу населения). Каково соотношение темпов роста ВВП стран Европы, Азии и Африки? Когда в стране происходит существенное изменение в ВВП, какие исторические события, вероятно, несут наибольшую ответственность за это?

6.2. *[3]* Ответьте на следующие простые вопросы для одного или нескольких наборов данных, доступных по адресу `http://www.data-manual.com/data`.

(a) Кто его создал, когда и почему?

(b) Насколько он велик?

(c) Что означают поля?

(d) Укажите несколько знакомых или понятных записей.

(e) Предоставьте пятичисловую сводку Тьюки для каждого столбца.
(f) Постройте матрицу попарной корреляции для каждой пары столбцов.
(g) Постройте график попарного распределения для каждой интересующей пары столбцов.

Интерпретация визуализаций

6.3. *[5]* Просмотрите свои любимые новостные сайты и найдете десять интересных графиков или диаграмм, половина из которых хороша, а половина плоха. Для каждого, пожалуйста, оцените следующие аспекты, используя словарь, который мы разработали в этой главе.
(a) График хорошо или плохо представляет данные?
(b) Есть ли в презентации предвзятость, преднамеренная или случайная?
(c) Есть ли на рисунке неинформативные элементы?
(d) Оси помечены ясно и информативно?
(e) Эффективно ли используется цвет?
(f) Как мы можем улучшить графику?

6.4. *[3]* Посетите сайт `http://www.wtfviz.net`. Найдите пять смехотворно плохих визуализаций и объясните, почему они так плохи и забавны.

Создание визуализаций

6.5. *[5]* Создайте наглядную визуализацию некоторых аспектов вашего любимого набора данных, используя:
(a) хорошо спроектированную таблицу;
(b) точечный и/или линейный график;
(c) график рассеяния;
(d) тепловую карту;
(e) гистограмму или круговую диаграмму;
(f) гистограмму;
(g) карту данных.

6.6. *[5]* Создайте десять разных версий линейных графиков для определенного набора точек (x, y). Какие из них лучше, а какие хуже? Объясните почему.

6.7. *[3]* Постройте графики рассеяния для наборов из 10, 100, 1000 и 10 000 точек. Поэкспериментируйте с размером точки, чтобы найти наиболее показательные значения для каждого набора данных.

6.8. *[5]* Поэкспериментируйте с различными цветовыми шкалами, чтобы построить диаграммы рассеяния для некого набора точек (x, y, z), где для представления измерения z используется цвет. Какие цветовые схемы работают лучше всего? Какие хуже? Объясните, почему.

Реализация проектов

6.9. *[5]* Создайте интерактивный виджет для исследования вашего любимого набора данных, используя соответствующие библиотеки и инструменты. Начните с простого, но проявите творческий подход, насколько хотите.

6.10. *[5]* Создайте фильм иди видео, записав или сняв интерактивное исследование данных. Это не должно быть долго, но насколько интересно/показательно вы можете сделать это?

Вопросы на интервью

6.11. *[3]* Опишите некоторые из хороших практик визуализации данных?

6.12. *[5]* Объясните концепцию Тафти о неинформативных элементах.

6.13. *[8]* Как бы вы определили, является ли статистика, опубликованная в статье, неправильной или представленной для поддержки предвзятого представления?

Конкурсы Kaggle

6.14. Проанализируйте данные о совместном использовании велосипедов в Сан-Франциско.
`https://www.kaggle.com/benhamner/sf-bay-area-bike-share`

6.15. Спрогнозируйте, присутствует ли вирус Западного Нила в данное время в определенном месте.
`https://www.kaggle.com/c/predict-west-nile-virus`

6.16. Какой тип преступления наиболее вероятен в данное время в определенном месте?
`https://www.kaggle.com/c/sf-crime`

Глава 7

Математические модели

> Все модели ошибочны, но некоторые из них полезны.
>
> — Джордж Бокс (George Box)

До сих пор в этой книге рассматривалось множество инструментов для манипулирования данными и их интерпретации. Но мы на самом деле не занимались *моделированием*, которое представляет собой процесс инкапсуляции информации в инструмент, способный делать прогнозы.

Прогнозирующие модели построены вокруг некоторого представления о том, что вызывает будущие события. Экстраполяция последних тенденций и наблюдений предполагает мировоззрение, согласно которому будущее будет таким же, как прошлое. Более сложные модели, такие как законы физики, дают принципиальные понятия причинности; фундаментальные объяснения того, почему вещи происходят.

Эта глава будет сосредоточена на разработке и проверке моделей. Эффективная формулировка моделей требует детального понимания пространства возможных выборов.

Точная оценка производительности модели может быть на удивление трудной, но это важно для понимания того, как интерпретировать полученные прогнозы. Лучшая система прогнозирования не обязательно самая точная, это модель с лучшим пониманием ее пределов и ограничений.

7.1. Философия моделирования

Инженеры и ученые часто с подозрением относятся к слову "философия". Однако стоит задуматься неким фундаментальным образом о том, что мы пытаемся сделать и почему. Напомним, что люди обращаются к ученым за мудростью, а не за программой.

В этом разделе мы обратимся к различным способам мышления о моделях, чтобы помочь сформировать способ их построения.

7.1.1. Бритва Оккама

Бритва Оккама (Occam's razor) — это философский принцип, согласно которому "самое простое объяснение — это лучшее объяснение". Согласно Уильяму из Оккама (William of Occam), теологу 13-го века, при рассмотрении двух моделей или теорий, которые одинаково точно дают предсказания, мы должны выбрать более простую и надежную. Скорее всего, она принимает правильное решение по правильным причинам.

Представление Оккама о простом в целом относится к сокращению числа допущений, используемых при разработке модели. Что касается статистического моделирования, бритва Оккама говорит о необходимости минимизировать количество параметров модели. *Переобучение* (overfitting) происходит, когда модель старается достичь слишком точных результатов при обучении. Это происходит, когда существует так много параметров, что модель может запоминать свой обучающий набор, вместо того, чтобы соответствующим образом обобщить его для минимизации последствий ошибок и выбросов.

Переобученные модели, как правило, очень хорошо работают с обучающими данными, но они гораздо менее точны с данными независимых тестов. Принцип бритвы Оккама требует, чтобы у нас был осмысленный способ оценить, насколько точно работают наши модели.

Простота не является абсолютным достоинством, когда она ведет к снижению производительности. *Глубокое обучение* (deep learning) — это мощный метод построения моделей с миллионами параметров, которые мы обсудим в разделе 11.6. Несмотря на опасность переобучения, эти модели отлично справляются с различными сложными задачами. Оккам бы с подозрением относился к таким моделям, но согласился бы с теми, которые обладают значительно большей предсказательной силой, чем альтернативы.

Цените компромисс между точностью и простотой. Почти всегда можно "улучшить" производительность любой модели, добавив дополнительные параметры и правила для управления исключениями. Сложность имеет свою стоимость, что явно отражено в методах машинного обучения, таких как LASSO и гребневая регрессия. Эти методы используют *штрафные функции* (penalty function) для минимизации функций, используемых в модели.

> *На заметку.* Точность — это не лучший показатель для оценки качества модели. Более простые модели, как правило, более надежны и понятны, чем сложные альтернативы. Повышение производительности в определенных тестах часто объясняется дисперсией или переобучением, а не пониманием.

7.1.2. Дилемма смещения-дисперсии

Это противоречие между сложностью модели и производительностью проявляется в статистическом понятии *дилеммы смещения-дисперсии* (bias-variance trade-off).

- *Смещение* (bias) — это ошибка из-за неправильных предположений, встроенных в модель, таких как ограничение интерполяционной функции линейной, а не кривой более высокого порядка.
- *Дисперсия* (variance) — это ошибка чувствительности к колебаниям в обучающем наборе. Если наш обучающий набор содержит ошибку выборки или измерения, этот шум вносит дисперсию в полученную модель.

Ошибки смещения приводят к *недообучению* (underfit) моделей. Они не соответствуют учебным данным настолько плотно, насколько это возможно, если бы им была предоставлена свобода делать это. В популярном обсуждении я ассоциирую слово *смещение* (bias) с *предубеждением* (prejudice), и это вполне уместно: априорное предположение, что одна группа уступает другой, приведет к менее точным предсказаниям, чем непредвзятая модель. Модели, обрабатывавшие плохие данные как при обучении, так и при проверке, оказываются недообучены.

Ошибки приводят к *переобучению* (overfit) моделей: их стремление к точности заставляет их принимать шум за сигнал, и они так хорошо приспосабливаются к обучающим данным, что шум вводит их в заблуждение. Модели, которые намного лучше справляются с тестовыми данными, чем с учебными, являются переобученными.[1]

> *На заметку.* Модели, основанные на первых принципах или предположениях, скорее всего, будут страдать от смещения, в то время как модели, основанные на данных, подвергаются большей опасности переобучения.

7.1.3. Что бы сделал Нейт Силвер?

Нейт Силвер (Nate Silver), пожалуй, самое известное публичное лицо в науке о данных сегодня. Выдающийся член научного общества, который оставил консалтинговую деятельность по вопросам управления ради разработки методов прогнозирования бейсбола. Он также прославился на своем веб-сайте по прогнозам выборов `http://www.fivethirtyeight.com`. Здесь он

[1] Чтобы завершить эту классификацию, модели, которые лучше работают с тестовыми данными, чем с обучающими, считаются *обманчивыми* (cheating).

использовал количественные методы для анализа результатов опроса, чтобы предсказать результаты президентских выборов в США. На выборах 2008 года он точно назвал победителя в 49 штатах из 50, а в 2012 году его результат улучшился до 50 из 50. Результаты выборов 2016 года стали шоком практически для всех, но только Нейт Силвер среди всех общественных комментаторов определил существенный шанс победы Трампа в коллегии выборщиков при одновременном проигрыше в голосах избирателей. Это действительно оказалось так.

Силвер написал отличную книгу *The Signal and the Noise: Why so many predictions fail — but some don't* [3]. Там он пишет о современных прогнозах в некоторых областях, включая спорт, погоду и землетрясения, а также о финансовом моделировании. Он описывает принципы эффективного моделирования, в том числе такие.

- *Думайте вероятностно.* Прогнозы, в которых делаются конкретные заявления, являются куда *менее* значимыми, чем те, которые по своей природе вероятностны. Прогноз, в котором у Трампа есть только 28,3% шансов на победу, является более значимым, чем тот, который категорически утверждает, что он проиграет.

  Реальный мир — это неопределенное место, и успешные модели признают эту неопределенность. Всегда есть набор возможных результатов, которые в реальности вполне могут произойти с небольшими вариациями, и это должно быть отражено в вашей модели. Прогнозы числовых величин не должны быть отдельными числами, а должны сообщать распределения вероятностей. Для описания такого распределения достаточно указать стандартное отклонение σ вместе со средним значением прогноза μ, особенно если оно считается нормальным.

  Некоторые из техник машинного обучения, которые мы будем изучать, естественно, дают вероятностные ответы. Логистическая регрессия обеспечивает уверенность, наряду с каждой классификацией, которую она делает. Методы, которые помечают голоса k за ближайших соседей, определяют естественную меру достоверности, основанную на согласованности меток в окрестности. Собрать десять из одиннадцати голосов за синих означает нечто более сильное, чем семь из одиннадцати.

- *Изменяйте свой прогноз в ответ на новую информацию.* Живые модели гораздо интереснее мертвых. Модель *жива*, если она постоянно обновляет прогнозы в ответ на новую информацию. Построение инфраструктуры, поддерживающей живую модель, куда сложнее одноразовых вычислений, но она гораздо ценнее.

  Живые модели более интеллектуально честны, чем мертвые. Свежая информация *должна* изменять результат любого прогноза. Ученые должны

быть открыты для изменения мнений в ответ на новые данные: действительно, это то, что отличает ученых от хакеров и троллей.

Динамически изменяющиеся прогнозы предоставляют отличные возможности для оценки вашей модели. Они в конечном счете приходят к правильному ответу? Уменьшается ли неопределенность по мере приближения события? Любая живая модель должна отслеживать и отображать свои прогнозы с течением времени, чтобы зритель мог оценить, точно ли изменения отражают влияние новой информации.

- *Ищите консенсус*. Хороший прогноз исходит из нескольких разных источников данных. Данные должны быть получены из максимально возможного количества различных источников. В идеале должно быть построено несколько моделей, каждая из которых пытается по-разному предсказать одно и то же. Вы должны иметь мнение о том, какая модель является лучшей, но будьте осторожны, когда она далеко отбивается от стада.

  Нередко третьи стороны создают конкурирующие прогнозы, которые вы можете отслеживать и сравнивать. Быть другим не значит быть неправым, но это обеспечивает проверку реальности. Кто был лучше в последнее время? Чем объясняются различия в прогнозе? Может ли ваша модель быть улучшена?

  Модель прогнозирования гриппы Flu Trends от Google предсказывала вспышки заболевания гриппом, отслеживая ключевые слова в поиске: всплеск количества людей, ищущих по ключевым словам *aspirin* (аспирин) или *fever* (повышение температуры), может свидетельствовать о распространении заболевания. Модель прогнозирования Google вполне соответствовала статистике Центра по контролю за заболеваниями (CDC) по фактическим случаям заболевания на протяжении нескольких лет, пока они существенно не сбились с пути.

  Мир меняется. Среди изменений было то, что поисковый интерфейс Google начал предлагать поисковые запросы в ответ на историю пользователя. Получив предложение, многие люди начали искать *аспирин* после *повышение температуры*. И старая модель внезапно перестала быть точной. Ошибка Google заключается в том, что они не следили за эффективностью системы и не корректировали ее со временем.

  Определенные методы машинного обучения явно стремятся к консенсусу. *Алгоритмы бустинга* (boosting algorithm) объединяют большое количество слабых классификаторов, чтобы получить сильный. Методы сборки дерева решений создают много независимых классификаторов и голосуют за них, чтобы принять лучшее решение. Такие методы

способны иметь надежность, которая недоступна моделям, использующим один путь.

- *Используйте байесовский вывод.* Теорема Байеса имеет несколько толкований, но, пожалуй, наиболее убедителен расчет изменения вероятности в ответ на новые доказательства. Будучи сформулирован как

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)},$$

он позволяет рассчитать, как изменяется вероятность события A в ответ на новые доказательства B.

Применение теоремы Байеса требует *предварительной* вероятности $P(A)$, вероятности события A *прежде*, чем узнавать состояние определенного события B. Это может быть результатом запуска классификатора для прогнозирования состояния A из других функций или базовых знаний о частотах событий в популяции. Без хорошей оценки этого априора (prior) очень трудно понять, насколько серьезно это относится к классификатору.

Предположим, что A — это событие, когда человек x на самом деле является террористом, а B — это результат классификатора на основе признаков, который решает, выглядит ли x как террорист. При обучении/оценке набора данных для 1000 человек, половина из которых является террористами, классификатор достигал завидной точности, скажем, 90%. Теперь классификатор говорит, что Скиена похож на террориста. Какова вероятность того, что он *действительно* террорист?

Ключевым моментом здесь является то, что предыдущая вероятность "x — это террорист" действительно очень мала. Если в США действуют сотни террористов, то $P(A) = 100/300\ 000\ 000 = 3{,}33 \cdot 10^{-7}$. Вероятность того, что детектор террористов скажет "да", составляет $P(B) = 0{,}5$, а вероятность того, что детектор будет прав, когда он скажет "да", составляет $P(B|A) = 0{,}9$. Умножение этих значений дает очень малую вероятность того, что я плохой парень,

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} = \frac{0{,}9 \cdot 3{,}33 \cdot 10^{-7}}{0{,}5} = 6 \cdot 10^{-7},$$

хотя по общему признанию я теперь больше, чем просто случайный гражданин.

Фактор априорных вероятностей важен для получения правильной интерпретации этого классификатора. Байесовский вывод начинается с предыдущего распределения, а затем взвешивает дальнейшие доказательства того, насколько сильно оно должно влиять на вероятность события.

7.2. Классификация моделей

Модели наступают, ну очень много разных моделей. Частью выработки философии моделирования является понимание доступных вам степеней свободы в дизайне и реализации. В этом разделе мы рассмотрим типы моделей в нескольких различных измерениях, рассмотрев основные технические проблемы, которыми отличается каждый класс.

7.2.1. Линейные модели против нелинейных

Линейные модели управляются уравнениями, которые взвешивают каждую переменную функции с помощью коэффициента, отражающего ее важность, и суммируют эти значения для получения оценки. Мощные методы машинного обучения, такие как линейная регрессия, могут быть использованы для определения наилучших возможных коэффициентов, соответствующих учебным данным, что дает очень эффективные модели.

Но, по правде говоря, мир не линеен. Более богатые математические описания включают полиномы высшего порядка, логарифмы и экспоненты. Они допускают модели, которые соответствуют обучающим данным намного более точно, чем линейные функции. Обычно гораздо сложнее найти наилучшие возможные коэффициенты для нелинейных моделей. Но нам *не нужно* находить наилучшее возможное соответствие: методы глубокого обучения, основанные на нейронных сетях, предлагают отличную производительность, несмотря на трудности, присущие оптимизации.

Ковбои от моделирования зачастую презирают простоту линейных моделей. Но линейные модели предлагают существенные преимущества. Они легко понятны, в целом оправданны, просты в сборке и позволяют избежать наложения на наборы данных небольшого размера. Бритва Оккама говорит нам, что "самое простое объяснение — это лучшее объяснение". Я, как правило, куда довольней надежной *линейной* моделью, дающей точность x%, чем сложным нелинейным зверем, только на несколько десятых процента лучше ее, при ограниченных проверочных данных.

7.2.2. Черные ящики против описательных моделей

Черные ящики (black box) — это устройства, выполняющие свою работу, каким-то неизвестным образом. Что-то входит и что-то выходит, но происходящее внутри совершенно непроницаемо для посторонних.

Мы, напротив, предпочитаем *описательные* модели, которые дают некоторое представление о том, почему они принимают свои решения. Модели, управляемые теорией, как правило, описательны, поскольку они являются явными реализациями конкретной хорошо развитой теории. Если вы верите теории, у вас есть основания доверять базовой модели и любым последующим прогнозам.

Некоторые модели машинного обучения оказываются менее непрозрачными, чем другие. Модели линейной регрессии носят описательный характер, поскольку можно точно определить, какие переменные получают больший вес, и измерить, насколько они способствуют полученному прогнозу. Модели дерева решений позволяют вам следовать точному пути решения, используемому для классификации. "Наша модель отказала вам в ипотеке, потому что ваш доход составляет менее 10 000 долл. в год, у вас более 50 000 долл. задолженности по кредитной карте, и вы были безработным в течение прошлого года".

Но досадная правда в том, что методы моделирования черного ящика, такие как глубокое обучение, могут быть чрезвычайно эффективными. Модели нейронных сетей, как правило, совершенно непрозрачны в отношении того, *почему* они делают то, что делают. Это поясняет рис. 7.1. Он демонстрирует изображения, которые были тщательно разработаны так, чтобы обмануть современные нейронные сети. Они преуспели блестяще. У рассматриваемых сетей было 99,6% уверенности в том, что они нашли правильную метку для каждого изображения на рис. 7.1.

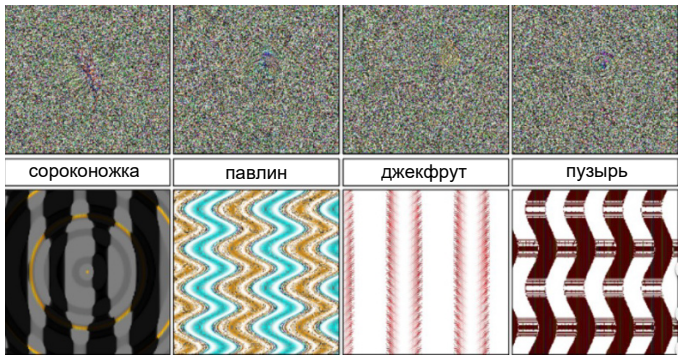


Рис. 7.1. Синтетические изображения, которые ошибочно распознаются как объекты современными нейронными сетями глубокого обучения, причем достоверность каждого из них превышает 99,6%. Источник: [68]

Скандал здесь заключается *не в том*, что сеть неправильно указала метки на этих специальных изображениях, поскольку эти системы распознавания являются весьма впечатляющими. На самом деле они были гораздо точнее, чем

можно было мечтать всего год или два назад. Проблема в том, что создатели этих классификаторов не знали, почему их программы допускают такие ужасные ошибки или как они могут их предотвратить в будущем.

Аналогичная история рассказана о системе, созданной для военных, чтобы отличить изображения легковых автомобилей от грузовиков. Она хорошо показала себя на тренировках, но в реальности — катастрофически. Только впоследствии стало понятно, что учебные изображения автомобилей снимались в солнечный день, а грузовиков — в облачный день, поэтому система научилась связывать небо на заднем плане с классом транспортного средства.

Подобные истории подчеркивают, почему визуализация обучающих данных и использование описательных моделей могут быть такими важными. Вы должны быть уверены, что ваша модель обладает информацией, необходимой для принятия решений, о которых вы спрашиваете, особенно в ситуациях, когда ставки высоки.

7.2.3. Модели первого принципа против моделей управляемых данными

Модели первого принципа (first-principle model) основаны на убеждении в том, как реально работает исследуемая система. Это может быть теоретическое объяснение, такое как законы движения Ньютона. Такие модели могут использовать весь вес классической математики: арифметику, алгебру, геометрию и многое другое. Модель может быть дискретной имитацией событий, как будет обсуждаться в разделе 7.7. Это может быть рассуждение, основанное на понимании предметной области: избиратели недовольны плохой экономикой, поэтому переменные, которые измеряют состояние экономики, должны помочь нам предсказать, кто победит на выборах.

Модели, управляемые данными (data-driven model), напротив, основаны на наблюдаемых корреляциях между входными и выходными переменными. Та же базовая модель может быть использована для прогнозирования погоды на завтра или цены на конкретную акцию, отличаясь только данными, на которых она была обучена. Методы машинного обучения позволяют построить эффективную модель в области, о которой никто ничего не знает, при условии, что у нас достаточно хороший обучающий набор.

Поскольку это книга о науке о данных, вы можете сделать вывод, что мое сердце больше принадлежит моделям, управляемым данными. Но это не совсем так. Наука о данных — это тоже *наука*, и события происходят по понятным причинам. Модели, которые игнорируют это, обречены на неудачу в определенных обстоятельствах.

Однако существует и альтернативный способ сформулировать эту дискуссию. *Специальные модели* (ad hoc model) создаются с использованием знаний, специфичных для предметной области, для определения их структуры и дизайна. Они обычно неустойчивы к изменениям условий, и их трудно применять к новым задачам. Модели машинного обучения для классификации и регрессии, напротив, носят *общий* характер, поскольку в них не используются идеи для решения конкретных проблем, а задействуются только конкретные данные. Перенесите модели на свежие данные, и они адаптируются к меняющимся условиям. Обучите их на другом наборе данных, и они смогут делать что-то совершенно другое. С этой точки зрения общие модели выглядят намного лучше, чем специальные.

Правда в том, что лучшие модели представляют собой смесь теории и данных. Важно понимать вашу предметную область как можно глубже, используя при этом лучшие данные, которые вы можете, чтобы подобрать и оценить ваши модели.

7.2.4. Стохастические модели против детерминированных

Требование одного детерминистического "предсказания" от модели может быть глупым. Мир — это сложное место, состоящее из многих реальностей, с событиями, которые, как правило, не будут разворачиваться точно так же, если время вернется вспять. Хорошие модели прогнозирования включают такое мышление и создают распределение вероятностей по всем возможным событиям.

Стохастический (stochastic) — это причудливое слово, означающее "детерминированный случайно". Методы, которые явно встраивают некоторое понятие вероятности в модель, включают логистическую регрессию и моделирование по методу Монте-Карло. Важно, чтобы ваша модель соблюдала основные свойства вероятностей.

- *Каждая вероятность — это значение от 0 до 1.* Оценки, которые не ограничены этим диапазоном, не оценивают вероятности непосредственно. Решение нередко заключается в том, чтобы пропустить значения через логит-функцию (см. раздел 4.4.1), превратив их в вероятности принципиальным образом.
- *В сумме они должны быть равны 1.* Независимо генерируемые значения от 0 до 1 не означают, что в сумме они составляют единицу вероятности по всему пространству событий. Решение здесь заключается в том, чтобы масштабировать эти значения в ходе деления каждого на функцию

распределения (см. раздел 9.7.4). С другой стороны, переосмыслите свою модель так, чтобы понять, почему она не работает.

- *Редкие события не имеют нулевой вероятности.* Любое возможное событие должно иметь вероятность больше нуля. *Дисконтирование* (discounting) является способом оценки вероятности невидимых, но возможных событий и будет обсуждаться в разделе 11.1.2.

Вероятности — это мера смирения в отношении точности нашей модели и неопределенности сложного мира. Модели должны быть честными и в том, что они делают, и том, чего они не знают.

Однако у детерминированных моделей есть определенные преимущества. Модели первого принципа зачастую дают только один возможный ответ. Законы движения Ньютона *точно* укажут вам, сколько времени потребуется массе, чтобы упасть на заданное расстояние.

То, что детерминированные модели всегда возвращают один и тот же ответ, очень помогает в отладке их реализации. Это говорит о необходимости оптимизировать *повторяемость* (repeatability) при разработке модели. Зафиксируйте начальное число, если вы используете генератор случайных чисел, чтобы вы могли запустить его снова и получить тот же ответ. Создайте набор регрессионных тестов для своей модели, чтобы вы могли подтвердить, что после модификации программы ответы остаются идентичными для заданных входных значений.

7.2.5. Плоские модели против иерархических

Интересные задачи нередко существуют на нескольких разных уровнях, каждый из которых может требовать независимых подмоделей. Прогнозирование будущей цены для конкретной акции действительно должно включать подмодели для анализа таких отдельных вопросов, как (а) общее состояние экономики, (б) балансовая отчетность компании и (в) результаты деятельности других компаний в ее промышленном секторе.

Наложение на модель иерархической структуры позволяет создавать и оценивать ее логичным и прозрачным способом, а не как черный ящик. Некоторые подзадачи пригодны для моделей на основании теории и на основании первого принципа, которые затем могут использоваться в качестве элементов в общей модели, управляемой данными. Явные иерархические модели являются описательными: можно проследить окончательное решение до соответствующей подзадачи верхнего уровня и сообщить, насколько сильно это способствовало достижению наблюдаемого результата.

Первым шагом к построению иерархической модели является явное разложение нашей проблемы на подзадачи. Обычно они представляют собой

механизмы, управляющие моделируемым базовым процессом. От чего *должна* зависеть модель? Если существуют данные и ресурсы для создания принципиальной подмодели для каждой части, отлично! Если нет, то можно оставить ее как пустую модель или базовую линию и явно описать упущение при документировании результатов.

Модели глубокого обучения можно считать одновременно и иерархическими, и неиерархическими. Они, как правило, обучаются на больших наборах необработанных данных, поэтому не существует явного определения подзадач для руководства подпроцессом. В целом сеть делает только одно. Но поскольку они построены из нескольких вложенных слоев (*углубленных* в глубокое обучение), эти модели предполагают, что существуют сложные функции, которые можно изучить из входных данных более низкого уровня.

Я всегда с неохотой признаю, что модели машинного обучения оказываются лучше, чем я, выводя основные организационные принципы в той области, которую я понимаю. Даже при использовании глубокого обучения стоит набросать грубую иерархическую структуру, которая, вероятно, существует для вашей сети. Например, любая сеть обработки изображений должна обобщаться от участков пикселей до краев, а затем от границ до подобъектов и к анализу сцены при переходе к более высоким уровням. Это влияет на архитектуру вашей сети и помогает проверить ее. Видите ли вы доказательства того, что ваша сеть принимает правильные решения по правильным причинам?

7.3. Базовые модели

Один мудрец заметил, что сломанные часы два раза в день показывают правильное время. Как разработчики моделей, мы стремимся быть лучше, чем эти часы, но для доказательства требуется определенный уровень строгости оценки.

Первый этап оценки сложности вашей задачи включается в построении *базовых моделей* (baseline model) — простейших подходящих моделей, дающих ответы, пригодные для сравнения. Более сложные модели *должны* работать лучше, чем базовые, но при этом еще и проверять, действительно ли они работают, и, если да, то насколько их производительность соответствует текущему контексту.

Некоторые задачи прогнозирования по своей природе сложнее, чем другие. Простая базовая линия ("да") оказывается очень точной в прогнозировании восхода солнца завтра. Напротив, вы можете разбогатеть, предсказывая, будет ли фондовый рынок расти или падать в 51% случаев. Только после того, как вы решительно превзойдете свои базовые показатели, ваши модели действительно будут считаться эффективными.

7.3.1. Базовые модели для классификации

Есть две общие задачи моделирования в науке о данных: *классификация* (classification) и *прогнозирование значения* (value prediction). В задачах классификации нам дается небольшой набор возможных ярлыков для любого заданного элемента, например спам или не спам, мужчина или женщина, велосипед, автомобиль или грузовик. Мы ищем систему, которая будет генерировать метку, точно описывающую конкретный экземпляр электронного письма, человека или транспортного средства.

Типичные базовые модели для классификации включают в себя следующее.

- *Равномерный или случайный выбор ярлыков.* Если у вас нет абсолютно никакого предыдущего распределения по объектам, вы можете также сделать случайный выбор, используя метод неработающих часов. Сравнение вашей модели прогнозирования фондового рынка со случайными бросками монеты будет иметь большое значение для демонстрации того, насколько сложна проблема.

  Я называю такой слепой классификатор *обезьяной* (the monkey), поскольку это все равно, что просить своего домашнего питомца принять решение за вас. В задаче предсказания с двадцатью возможными ярлыками или классами достижение значительного превышения 5% является первым доказательством того, что у вас есть некоторое представление о проблеме. Сначала вы должны показать мне, что можете победить обезьяну, прежде чем я начну вам доверять.

- *Наиболее распространенная метка в учебных данных.* Большой учебный набор обычно дает некоторое представление о предварительном распределении по классам. Выбор наиболее часто встречающихся меток лучше, чем их выбор равномерно или случайным образом. Это теория, лежащая в основе базовой модели *солнце-взойдет-завтра* (sun-will-rise-tomorrow).

- *Самая точная однофункциональная модель.* Мощные модели стремятся использовать все полезные функции, присутствующие в данном наборе данных. Но важно знать, на что способна лучшая отдельная функция. Создать лучший классификатор из одной числовой функции x легко: мы объявляем, что элемент находится в классе 1, если $x \geq t$, а иначе в классе 2. Чтобы найти наилучший порог t, мы можем проверить все n возможных порогов в форме $t_i = x_i + \varepsilon$, где x_i — это значение признака в i-м из n обучающих экземпляров. Затем выберите пороговое значение, которое дает наиболее точный классификатор для ваших обучающих данных.

  Бритва Оккама считает простейшую модель лучшей. Только когда ваша сложная модель превосходит все однофакторные модели, она становится интересной.

- *Чужая модель*. Иногда мы не первые, кто пытается выполнить определенную задачу. У вашей компании может быть устаревшая модель, которую вам поручено обновить или пересмотреть. Возможно, подобный вариант проблемы обсуждался в академической статье, а возможно, некто даже опубликовал свой код в Интернете, чтобы вы могли поэкспериментировать с ним.

  Когда вы сравниваете свою модель с чужой работой, может произойти одно из двух: либо вы превосходите их, либо нет. Если вы превосходите, у вас есть нечто, чем стоит похвастаться. Если это не так, то это шанс научиться и усовершенствоваться. Почему ты не выиграл? Тот факт, что вы проиграли, дает вам уверенность в том, что ваша модель может быть улучшена по крайней мере до уровня чужой модели.

- *Ясновидение*. Существуют обстоятельства, когда даже самая лучшая модель теоретически не может достичь 100% точности. Предположим, что две записи данных абсолютно одинаковы в пространстве признаков, но с противоречивыми метками. Не существует детерминированного классификатора, который когда-либо мог бы решить обе эти проблемы правильно, поэтому мы обречены на неудовлетворительную производительность. Но более жесткая верхняя граница от оптимально ясновидящего предиктора может убедить вас, что ваша базовая модель лучше, чем вы думали.

  Нередко возникает необходимость улучшения верхних границ, когда ваши учебные данные являются результатом процесса аннотирования человеком, и несколько аннотаторов оценивают одни и те же экземпляры. Мы получаем противоречия, когда два аннотатора не согласны друг с другом. Я работал над проблемами, где 86,6% правильных ответов были наивысшим результатом. Это ухудшает ожидания. Хороший жизненный совет — помните, от своих собратьев мало чего можно ожидать, и вам придется обходиться гораздо меньшим, чем это.

7.3.2. Базовые модели для прогнозирования значения

В задачах прогнозирования значений нам дан набор пар признаков и значений (f_i, v_i), которые можно использовать для обучения функции F, такой как $F(v_i) = v_i$. Базовые модели для задач прогнозирования значения вытекают из методов, аналогичных тем, которые были предложены для классификации.

- *Среднее или медиана*. Просто игнорируйте функции, чтобы вы всегда могли вывести согласованное целевое значение. Это оказывается достаточно информативной базой, поскольку, если вы не можете существенно

преуспеть, всегда угадывая среднее значение, у вас либо неправильные функции, либо вы работаете над безнадежной задачей.

- *Линейная регрессия.* Мы подробно рассмотрим линейную регрессию в разделе 9.1. Но пока достаточно понять, что эта мощная, но простая в использовании методика строит наилучшую из возможных линейных функций для задач прогнозирования значений. Эта базовая линия позволяет вам лучше оценить производительность нелинейных моделей. Если они не работают существенно лучше, чем линейный классификатор, они, вероятно, не стоят усилий.
- *Значение предыдущего момента времени.* Прогнозирование временных рядов — это обычная задача, в которой нам поручено прогнозировать значение $f(t_n, x)$ в момент t_n заданного набора характеристик x и наблюдаемые значения $f'(t_i)$ для $1 \leq i < n$. Но сегодняшняя погода — это хорошее предположение о том, будет ли дождь завтра. Аналогично значение предыдущего наблюдаемого значения $f'(t_{n-1})$ является разумным прогнозом для времени $f(t_n)$. Зачастую на практике удивительно трудно превзойти этот базовый уровень.

Базовые модели должны быть справедливыми: они должны быть простыми, но не глупыми. Вы хотите представить цель, которую вы надеетесь или ожидаете подстрелить, но не сидячую утку. Вы должны чувствовать облегчение, когда выходите за пределы базовой линии, но не хвастаться и не ухмыляться.

7.4. Оценка моделей

Поздравляем! Вы создали прогностическую модель для классификации или прогнозирования значения. Теперь насколько она хороша?

На этот невинный вопрос нет простого ответа. В следующих разделах мы подробно рассмотрим ключевые технические вопросы. Но неформальная *проба на запах* является, пожалуй, самым важным критерием оценки модели. Вы *действительно* верите, что модель хорошо справляется с вашими учебными и проверочными экземплярами?

Формальные оценки, которые будут подробно описаны ниже, снижают производительность модели до нескольких сводных статистических данных, агрегированных во многих случаях. Но многие огрехи модели могут быть скрыты, когда вы взаимодействуете только с этими совокупными показателями. У вас нет возможности узнать, есть ли ошибки в вашей реализации или нормализации данных, что приводит к большему снижению производительности, чем это должно быть. Возможно, вы смешали учебные и проверочные данные, получив гораздо лучшие результаты на проверочном стенде, чем заслуживаете.

Чтобы по-настоящему узнать, что происходит, вам нужно сделать пробу на запах. Моя личная проба на запах включает в себя тщательный анализ нескольких примеров, чтобы модель поняла это правильно, и нескольких случаев, когда она поняла это неправильно. Задача заключается в том, чтобы убедиться, что я понимаю, почему модель получила те результаты, которые она получила. В идеале это должны быть записи, чьи "имена" вы понимаете, в тех случаях, когда у вас есть некоторые предположения о том, какими должны быть правильные ответы в результате исследовательского анализа данных или знакомства с предметной областью.

> *На заметку.* Слишком многие аналитики данных заботятся только об оценке статистики своих моделей. Но у хороших аналитиков есть понимание того, являются ли ошибки, которые они делают, оправданными, серьезными или несущественными.

Другая проблема — ваша степень удивления оценкой точности модели. Она работает лучше или хуже, чем вы ожидали? Как вы думаете, насколько точно *вы* выполните задание, если будете использовать человеческое мнение?

С этим связан вопрос о том, насколько было бы ценней, если бы модель работала чуть лучше. Задача NLP, которая правильно классифицирует слова с точностью 95%, допускает ошибку примерно раз на два-три предложения. Это достаточно хорошо? Чем лучше его текущая производительность, тем сложнее будет делать дальнейшие улучшения.

Но лучший способ оценки моделей — это *вневыборочные* (out-of-sample) прогнозы, основывающиеся при построении модели на данных, которых вы никогда не видели (или, что еще лучше, которые ранее не существовали). Хорошая работа с данными, на которых вы обучали модели, весьма сомнительна, поскольку модели легко могут быть переобучены. Прогнозы на основе выборок являются ключом к честности, если у вас достаточно данных и времени для их проверки. Вот почему я попросил своих студентов из *The Quant Shop* построить модели для предсказания будущих событий, а затем заставил их посмотреть, были они правы или нет.

7.4.1. Оценка классификаторов

Оценка классификатора означает измерение того, насколько точно наши прогнозируемые метки соответствуют меткам золотого стандарта в оценочном наборе. Для общего случая двух разных меток или классов (двоичная классификация) мы обычно называем меньший и более интересный из двух клас-

сов *положительным* (positive), а больший (другой) класс — *отрицательным* (negative). В проблеме классификации спама спам, как правило, будет положительным, а не спам — отрицательным. Эта маркировка свидетельствует о том, что определить положительные моменты как минимум так же сложно, как и отрицательные, хотя зачастую тестовые экземпляры выбираются таким образом, чтобы классы имели одинаковую мощь.

Существует четыре возможных результата того, что может сделать модель классификации в любом конкретном случае, что определяет *матрица несоответствий* (confusion matrix) или *таблица сопряженности* (contingency table), показанная на рис. 7.2.

		Прогнозируемый класс	
		Да	Нет
Настоящий класс	Да	Истинно положительные (TP)	Ложно отрицательные (FN)
	Нет	Ложно положительные (FP)	Истинно отрицательные (TN)

Рис. 7.2. Матрица несоответствий для двоичных классификаторов, определяющая различные классы правильных и ошибочных предсказаний

- *Истинно положительный* (True Positive — TP). Здесь наш классификатор помечает положительный элемент как положительный, что приводит к победе классификатора.
- *Истинно отрицательный* (True Negative — TN). Здесь классификатор правильно определяет, что член отрицательного класса заслуживает отрицательной метки. Еще одна победа.
- *Ложно положительные* (False Positive — FP). Классификатор по ошибке называет отрицательный элемент положительным, что приводит к ошибке классификации "типа I".
- *Ложно отрицательные* (False Negative — FN). Классификатор ошибочно объявляет положительный элемент отрицательным, что приводит к ошибке классификации "типа II".

На рис. 7.3 показано, где эти результирующие классы располагаются в разделении двух распределений (мужчин и женщин), где решающей переменной является рост, измеряемый в сантиметрах. Оцениваемый классификатор помечает любого ростом выше 168 сантиметров как мужчину. Темные области представляют пересечение как мужчин, так и женщин. Эти хвосты представляют неправильно классифицированные элементы.

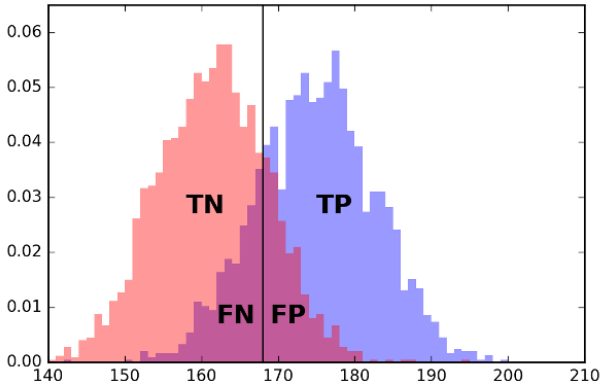


Рис. 7.3. Что произойдет, если мы классифицируем всех ростом выше 168 сантиметров как мужчин? Четыре возможных результата в матрице неточностей отражают, какие экземпляры были классифицированы правильно (TP и TN), а какие нет (FN и FP)

Корректность, точность, отзыв и F-оценка

Существует несколько разных статистических показателей, которые могут быть рассчитаны на основе истинно/ложных положительных/отрицательных значений, указанных выше. Причина, по которой нам нужно так много статистических данных, заключается в том, что мы должны защищать наш классификатор от двух главных оппонентов, жулика и обезьяны.

Жулик (sharp) — это оппонент, который знает, какую систему оценки мы используем, и выбирает базовую модель, которая будет лучше всего соответствовать ей. Жулик попытается заставить оценку статистики выглядеть плохо, достигнув высокой оценки с бесполезным классификатором. Это может означать объявление всех элементов положительными или, возможно, всех отрицательными.

Обезьяна (monkey), напротив, случайно угадывает каждый случай. Чтобы интерпретировать производительность нашей модели, важно установить, насколько она превосходит как жулика, так и обезьяну.

Первый статистический показатель — *корректность* (accuracy) классификатора — это отношение количества правильных прогнозов к общему количеству прогнозов. Таким образом:

$$Корректность = \frac{TP + TN}{TP + TN + FN + FP}.$$

Умножив эту дробь на 100, мы можем получить процент корректности.

Корректность — это значение, которое относительно легко объяснить, поскольку его представляют в любой среде оценки. Насколько корректна обезьяна, когда половина экземпляров положительна, а половина отрицательна? Ожидается, что обезьяна достигнет корректности в 50% за счет случайного угадывания. Одинаковая корректность в 50% будет достигнута жуликом, всегда угадывающим положительное или (что эквивалентно) отрицательное. Он получит половину правильных экземпляров в каждом случае.

Тем не менее только корректность имеет ограничения в качестве метрики оценки, особенно когда положительный класс намного меньше, чем отрицательный. Рассмотрим разработку классификатора для диагностики того, имеет ли пациент рак, когда положительный класс означает заболевание (т.е. тест положительный), а отрицательный — отсутствие заболевания. Априорное распределение заключается в том, что подавляющее большинство людей здоровы, поэтому

$$p = \frac{|положительный|}{|положительный| + |отрицательный|} \ll \frac{1}{2}.$$

Ожидаемая корректность обезьяны с монетой все равно будет 0,5: она должна получить в среднем половину положительных и половину отрицательных значений. Но жулик объявил бы всех здоровыми, достигнув точности $1 - p$. Предположим, что только 5% тестируемых действительно больны. Жулик может похвастаться своей корректностью 95%, одновременно обрекая всех членов класса больных на раннюю смерть.

Таким образом, нам нужны метрики оценки, которые более чувствительны к правильному получению класса положительных меток. *Точность* (precision) измеряет, насколько часто этот классификатор оказывается правым, когда он осмеливается заявить о положительности:

$$Точность = \frac{TP}{TP + FP}$$

Достижение высокой точности невозможно ни для жуликов, ни для обезьян, поскольку доля положительных меток ($p = 0{,}05$) очень мала. Если классификатор выдает слишком много положительных меток, он обречен на низкую точность, поскольку многие пули пролетают мимо, что приводит к множеству ложно положительных результатов. Но если классификатор скуп с положительными метками, очень немногие из них, вероятно, связаны с редкими положительными экземплярами, поэтому классификатор достигает низких истинно положительных результатов. Эти базовые классификаторы достигают точности, пропорциональной вероятности положительного класса $p = 0{,}05$, поскольку они летят вслепую.

	Обезьяна прогнозируемый класс	
	да	нет
да	$pn \cdot q$	$pn(1-p)$
нет	$(1-p)n \cdot q$	$(1-p)n \cdot pn(1-q)$

	Сбалансированный классификатор прогнозируемый класс	
	да	нет
да	$pn \cdot q$	$pn(1-p)$
нет	$(1-p)n \cdot pn(1-q)$	$(1-p)n \cdot q$

Рис. 7.4. Ожидаемая производительность классификатора обезьян в n случаях, где pn положительны, а (1 – p)n отрицательны. Обезьяна угадывает положительные с вероятностью q (слева). Рядом ожидаемая производительность сбалансированного классификатора, который каким-то образом правильно классифицирует членов каждого класса с вероятностью q (справа)

В случае диагностики рака мы можем быть более готовы терпеть ложную положительность (ошибки, когда мы пугаем здорового человека неправильным диагнозом), чем ложную отрицательность (ошибки, когда мы убиваем больного пациента, неправильно диагностируя отсутствие у него болезни). *Отзыв* (recall) измеряет, как часто вы оказались правы во всех положительных случаях:

$$Отзыв = \frac{TP}{TP + FN}.$$

Высокий отзыв подразумевает, что классификатор имеет мало ложных отрицательных меток. Самый простой способ добиться этого — объявить, что есть рак у *всех*, а жулик всегда ответит "да". Этот классификатор имеет высокую степень отзыва, но низкую точность: 95% тестируемых испытают ненужную панику. При построении классификаторов существует неотъемлемый компромисс между точностью и отзывом: чем храбрее ваши прогнозы, тем меньше вероятность того, что они окажутся правильными.

Но люди очень хотят, чтобы одно из измерений описывало производительность их системы. *F-оценка* (F-score) (или иногда F1-оценка) — это такая комбинация, возвращающая среднее гармоническое точности и отзыва:

$$F = 2 \cdot \frac{Точность \cdot Отзыв}{Точность + Отзыв}.$$

F-оценка — это очень жесткая мера. Среднее гармоническое всегда меньше или равно среднему арифметическому, а нижнее число имеет непропорционально большой эффект. Достижение высокой F-оценки требует как высокого отзыва, так и высокой точности. Ни один из наших базовых классификаторов не достигнет приличной F-оценки, несмотря на высокие значения точности и отзыва, поскольку их точность слишком низкая.

F-оценка и соответствующие метрики были разработаны для оценки значимых классификаторов, а не обезьян или жуликов. Чтобы понять, как их интерпретировать, давайте рассмотрим класс магически *сбалансированных* (balanced)

классификаторов, которые так или иначе показывают одинаковую точность как в положительных, так и в отрицательных случаях. Обычно это не так, но классификаторы, выбранные для достижения высоких F-оценок, должны сбалансировать статистику точности и отзыва, а это означает, что они должны показывать приличную производительность как в положительных, так и в отрицательных случаях.

	Обезьяна		**Жулик**		**Сбалансированный классификатор**				
q	0,05	0,5	0,0	1,0	0,5	0,75	0,9	0,99	1,0
корректность	0,905	0,5	0,95	0,05	0,5	0,75	0,9	0,99	1,0
точность	0,05	0,05	–	0,05	0,05	0,136	0,321	0,839	1,0
отзыв	0,05	0,5	0,0	1,0	0,5	0,75	0,9	0,99	1,0
F-оценка	0,05	0,091	–	0,095	0,091	0,231	0,474	0,908	1,0

Рис. 7.5. Производительность нескольких классификаторов при разных показателях производительности

На рис. 7.5 обобщены показатели как базовых, так и сбалансированных классификаторов по нашей проблеме обнаружения рака, сопоставленные по всем четырем нашим оценочным показателям. Отсюда можно извлечь следующие уроки.

- *Корректность вводит в заблуждение, когда размеры классов существенно различаются.* Базовый классификатор, бездумно отвечавший "нет" в каждом случае, достигал точности 95% по проблеме рака, это даже лучше, чем сбалансированный классификатор, который обеспечивал 94% правильности по каждому классу.
- *Отзыв эквивалентен корректности, если и только если классификаторы сбалансированы.* Хорошие вещи случаются, когда корректность распознавания обоих классов одинакова. Этого не происходит автоматически во время обучения, когда размеры классов разные. Это одна из причин, по которой, как правило, в вашем учебном наборе одинаковое количество положительных и отрицательных примеров.
- *Высокой точности очень трудно достичь при несбалансированных размерах классов.* Даже сбалансированный классификатор, который дает корректность 99% как для положительных, так и для отрицательных случаев, не может достичь точности выше 84% для проблемы рака. Это потому, что отрицательных случаев в двадцать раз больше, чем положительных. Ложно положительные результаты неправильной классификации более крупного класса при частоте 1% остаются существенными на фоне 5% истинно положительных результатов.

- *F-оценка лучше справляется с работой для любой отдельной статистики, но чтобы описать производительность классификатора, все четыре работают вместе.* Точность вашего классификатора больше, чем его отзыв? Тогда он помечает слишком мало экземпляров как положительные, и, возможно, вы сможете настроить его лучше. Отзыв выше, чем точность? Может быть, мы можем улучшить F-оценку, будучи менее агрессивными в выборе положительного. Корректность далека от отзыва? Тогда наш классификатор не очень сбалансирован. Так что проверьте, какая сторона хуже, и как мы могли бы это исправить.

Полезный прием для повышения точности модели за счет отзыва — это позволить ей сказать "я не знаю". Классификаторы, как правило, лучше справляются с простыми делами, чем с трудными, когда степень трудности определяется тем, насколько далеко от примера назначена альтернативная метка.

Определение уверенности в *правильности* предложенной вами классификации является ключом к тому, когда вам следует задавать вопрос. Только рискните предположить, когда ваша уверенность выше определенного порога. Пациенты, у которых результаты тестов находятся близко к границе, обычно предпочитают диагноз "пограничный результат", а не "у вас рак", особенно если классификатор не уверен в своем решении.

Наша статистика точности и отзыва должна быть пересмотрена, чтобы должным образом учесть новый неопределенный класс. Нет необходимости менять формулу точности: мы оцениваем только те случаи, которые мы называем положительными. Но знаменатель для отзыва должен явно учитывать все элементы, которые мы отказались пометить. Если предположить, что мы точны в наших мерах доверия, точность увеличится за счет отзыва.

7.4.2. Кривые рабочей характеристики приемника (ROC)

Многие классификаторы поставляются с изначальными средствами управления, которые вы можете настроить, чтобы изменить компромисс между точностью и отзывом. Рассмотрим, например, системы, которые вычисляют числовую оценку, выражаемую "в классах", возможно, полученную в результате оценки того, насколько данный тестовый образец выглядит как рак. Определенные образцы оцениваются более положительно, чем другие. Но где мы проводим границу между положительными и отрицательными?

Если наша оценка "в классах" является точной, то для положительных случаев она обычно должна быть выше, чем для отрицательных. Положительные примеры будут определять распределение оценок, отличное от отрицательных,

как показано на рис. 7.6 (слева). Было бы здорово, если бы эти распределения были полностью непересекающимися, поскольку тогда был бы порог оценки t, чтобы все экземпляры с оценками $\geq t$ были положительными, а все $< t$ — отрицательными. Это определило бы идеальный классификатор.

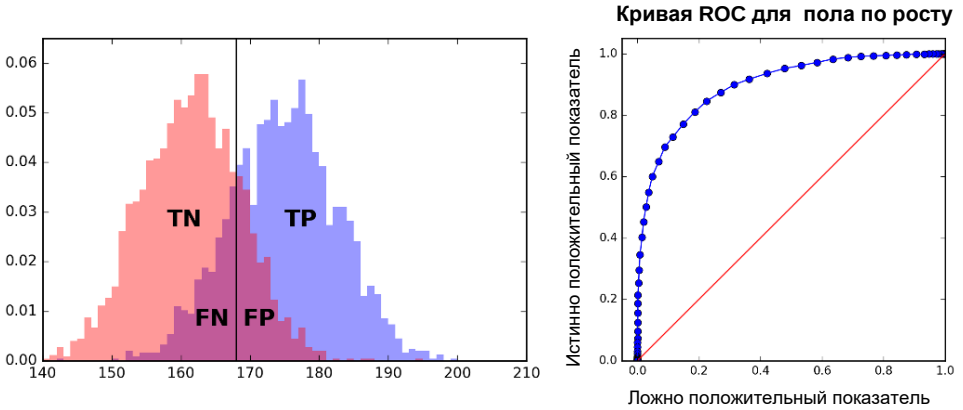


Рис. 7.6. Кривая ROC помогает выбрать наилучший порог для использования в классификаторе, отображая компромисс между истинно положительными значениями и ложно положительными значениями при всех возможных настройках. Главная диагональ здесь — это ROC обезьяны

Но куда вероятней то, что эти два распределения будут перекрываться, по крайней мере до некоторой степени, превращая проблему определения лучшего порога в суждение, основанное на нашем относительном отвращении к ложно положительным и ложно отрицательным значениям.

Кривая *рабочей характеристики приемника* (Receiver Operating Characteristic — ROC) обеспечивает визуальное представление всего нашего пространства возможностей при составлении классификатора. Каждая точка на этой кривой представляет определенный порог классификатора, определяемый его ложно положительными и ложно отрицательными показателями[2]. Эти показатели, в свою очередь, определяются количеством ошибок, деленным на общее количество положительных результатов в оценочных данных и, возможно, умноженным на сотню, чтобы превратить его в проценты.

Давайте рассмотрим, что происходит, когда мы смещаем порог слева направо по этим распределениям. Каждый раз, когда мы переходим к другому примеру, мы либо увеличиваем количество истинно положительных результатов (если этот пример был положительным), либо ложно положительных результатов (если этот пример был фактически отрицательным). С левого края мы

[2] Странное имя этого зверя является наследием его первоначального применения в настройке производительности радиолокационных систем.

достигаем истинно/ложно положительного уровня в 0%, поскольку на этом отрезке классификатор не помечал ничего положительного. При движении далее вправо все примеры будут помечены положительно, а следовательно, оба показателя достигнут 100%. Каждое пороговое значение между ними определяет возможный классификатор, а развертка определяет *ступенчатой кривой* (staircase curve) в пространстве истинно/ложно положительных значений, проводя нас от (0%, 0%) до (100%, 100%).

Предположим, что функция оценки была определена обезьяной, т.е. произвольным случайным значением для каждого экземпляра. Затем, когда мы сместим наш порог вправо, метка следующего примера должна быть положительной или отрицательной с равной вероятностью. Таким образом, мы в равной степени можем увеличить как наш истинно положительный показатель, так и ложно, и кривая ROC должна двигаться вдоль главной диагонали.

Работа лучше обезьяньей подразумевает кривую ROC, которая лежит выше диагонали. Наилучшая из возможных кривых ROC сразу увеличивается с (0%, 0%) до (0%, 100%), что означает, что она предпочитает все положительные случаи любыми отрицательным. Затем она смещается вправо с каждым отрицательным примером, пока, наконец, не достигнет верхнего правого угла.

Область под кривой ROC (area under the ROC curve — AUC) зачастую используется в качестве статистики, измеряющей качество функции оценки, определяющей классификатор. Наилучшая кривая ROC имеет площадь $100\% \cdot 100\% \rightarrow 1$, а обезьяний треугольник имеет площадь 1/2. Чем ближе площадь к 1, тем лучше наша классификационная функция.

7.4.3. Оценка мультиклассовых систем

Многие проблемы классификации не являются бинарными, а значит они должны выбирать из более чем двух классов. В новостях Google есть отдельные разделы для США и мировых новостей, а также для бизнеса, развлечений, спорта, здравоохранения, науки и технологий. Таким образом, классификатор статей, который управляет поведением этого сайта, должен присваивать каждой статье метку из восьми разных классов.

Чем больше у вас возможных меток классов, тем сложнее получить правильную классификацию. Ожидаемая точность классификации обезьяны с d метками составляет $1/d$, поэтому корректность быстро падает с увеличением сложности класса.

Это затрудняет правильную оценку мультиклассовых классификаторов, поскольку низкие показатели успеха приводят в уныние. Лучшей статистикой является *показатель успешности top-k*, который обобщает корректность

для некоторого конкретного значения $k \geq 1$. Как часто оказывается правильной метка среди k лучших вариантов?

Этот критерий хорош, поскольку он дает нам частичную оценку того, насколько мы приблизились к правильному ответу. Насколько близко и достаточно ли хорошо, определяется параметром k. Для $k = 1$ это снижает корректность. Для $k = d$ достаточно *любой* возможной метки, и вероятность успеха составит 100% по определению. Типичные значения — 3, 5 или 10: достаточно высокие, чтобы хороший классификатор достигал точности более 50% и был заметно лучше, чем обезьяна. Но не намного лучше, поскольку эффективная оценка должна дать нам достаточно возможностей для улучшения. На самом деле хорошей практикой является вычисление максимального значения k для всех k от 1 до d или по крайней мере достаточно высокого, чтобы задача стала легкой.

Еще более мощным инструментом оценки является *матрица несоответствий* (confusion matrix) C, т.е. матрица $d \times d$, где $C[x, y]$ сообщает о количестве (или долях) экземпляров класса x, которые помечены как класс y.

На рис. 7.7 показано, как мы читаем матрицу неточностей. Она взята из среды оценки, которую мы создали для проверки классификатора датировки документов, который анализирует тексты, чтобы предсказать период авторства. Такие датировки документов будут постоянным примером оценки в оставшейся части этой главы.

Наиболее важной особенностью является главная диагональ $C[i, i]$, которая подсчитывает, сколько (или какая часть) предметов из класса i были правильно помечены как принадлежащие классу i. Мы полагаемся на большую главную диагональ в нашей матрице. Перед нами трудная задача, и на рис. 7.7 показана сильная, но не идеальная главная диагональ. Здесь есть несколько мест, где документы чаще классифицируются соседним периодом, чем правильным.

Однако наиболее интересными особенностями матрицы неточностей являются большие числа $C[i, j]$, которые *не лежат* вдоль главной диагонали. Обычно они представляют собой запутанные классы. В нашем примере матрица показывает крайне большое количество документов (6%) от 1900 года, классифицированных как 2000, тогда как ни один не классифицирован 1800 годом. Такая асимметрия предлагает рекомендацию по улучшению классификатора.

Есть два возможных объяснения беспорядку в классах. Первая ошибка в классификаторе означает, что нам нужно работать усерднее, чтобы отличить i от j. Но вторая подразумевает смирение от осознания того, что классы i и j могут перекрываться до такой степени, что совершенно неясно, каким должен быть правильный ответ. Возможно, стили письма не так уж и сильно изменились за двадцать лет?

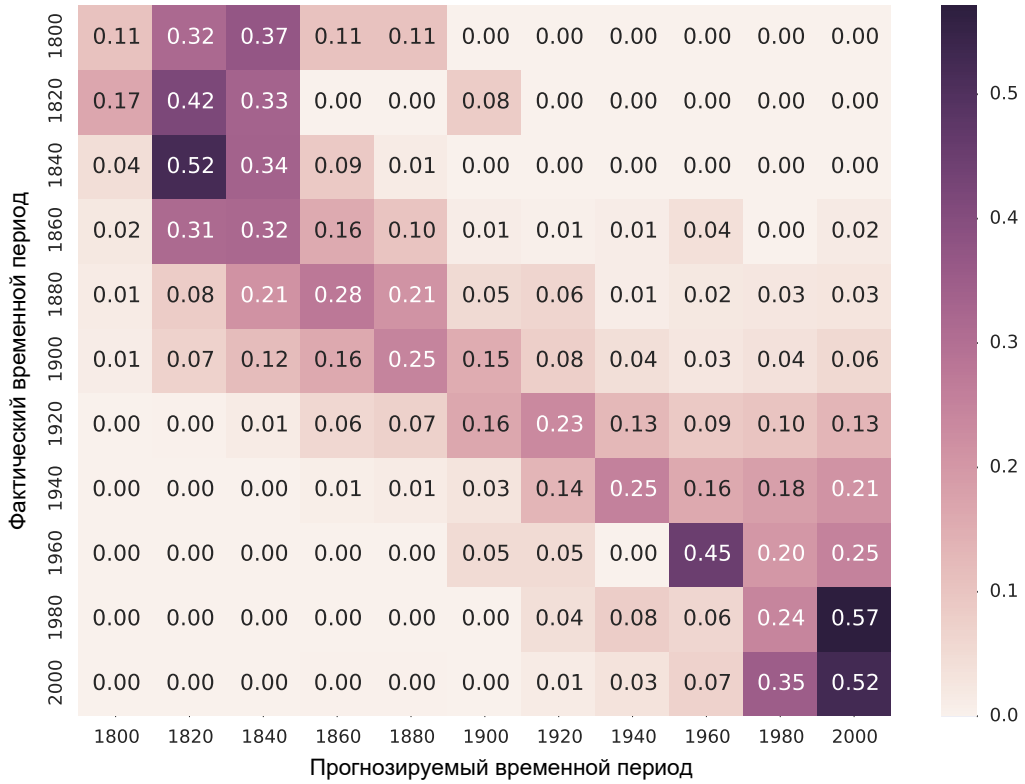


Рис. 7.7. Матрица неточностей для системы датирования документов: главная диагональ отражает точную классификацию

В примере с новостями Google грань между категориями наука и техника очень размыта. Где должна быть статья о коммерческих космических полетах? Google говорит: “Наука”, а я говорю: “Техника”. Таким образом, можно предложить объединение двух категорий, поскольку они представляют собой разницу без различия.

Разреженные строки в матрице неточностей указывают классы, плохо представленные в обучающих данных, в то время как разреженные столбцы указывают метки, которые классификатор не хочет назначать. Любой признак является аргументом, от которого, возможно, нам следует отказаться и объединить две похожие категории.

Строки и столбцы матрицы неточностей предоставляют статистику производительности, аналогичную статистике из раздела 7.4.1 для нескольких классов, параметризованных по классам. *Точность*$_i$ — это доля всех элементов, объявленных классом i, которые на самом деле принадлежат классу i:

$$Точность_i = C[i,i] \Big/ \sum_{j=1}^{d} C[j,i].$$

Отзыв$_i$ — это доля всех членов класса i, которые были правильно определены как таковые:

$$Отзыв_i = C[i,i] \Big/ \sum_{j=1}^{d} C[i,j].$$

7.4.4. Оценка моделей прогнозирования значений

Задачи прогнозирования значений можно рассматривать как задачи классификации, но они охватывают бесконечное количество классов. Однако существуют и более прямые способы оценки регрессионных систем, основанные на расстоянии между прогнозируемыми и фактическими значениями.

Статистика ошибок

Для числовых значений *ошибка* (error) является функцией разницы между прогнозом $y' = f(x)$ и фактическим результатом y. Измерение эффективности системы прогнозирования значений включает два решения: (1) исправление конкретной функции отдельной ошибки и (2) выбор статистики для наилучшего представления полного распределения ошибок. Основные варианты для выбора функции отдельной ошибки таковы.

- *Абсолютная ошибка.* Значение $\Delta = y' - y$ обладает простотой и симметричностью, поэтому знак может указывать случай, когда $y' > y$ или $y > y'$. Проблема заключается в агрегировании этих значений в сводную статистику. Означают ли ошибки смещения, такие как −1 и 1, что система идеальна? Чтобы устранить знак, обычно берется абсолютное значение ошибки.
- *Относительная ошибка.* Абсолютная величина ошибки не имеет смысла без понимания задействованных единиц. Абсолютная погрешность в 1,2 в прогнозируемом росте человека хороша, если она измеряется в миллиметрах, но ужасна, если измеряется в милях.

  Нормализация ошибки по величине наблюдения приводит к получению единицы измерения, которая может быть разумно интерпретирована как дробь или (умножена на 100%) как процент: $\varepsilon = (y - y')/y$. Абсолютная ошибка рассматривает случаи с бо́льшими значениями y как более важные, чем с меньшими, смещение корректируется при вычислении относительных ошибок.
- *Квадратичная ошибка.* Значение $\Delta^2 = (y' - y)^2$ всегда положительно, а следовательно, эти значения могут быть осмысленно суммированы.

Значения больших ошибок вносят несоразмерный вклад в общую сумму при возведении в квадрат: Δ^2 для $\Delta = 2$ в четыре раза больше, чем Δ^2 для $\Delta = 1$. Таким образом, в большом ансамбле выбросы могут легко стать доминирующими в статистике ошибок.

Очень хорошей идеей является построение гистограммы распределения абсолютных ошибок для любого предиктора значений, так как из этого можно многому научиться. Распределение *должно* быть симметричным и сосредоточено вокруг нуля. Оно *должно* быть колоколообразным, т.е. небольшие ошибки встречаются чаще, чем большие. И экстремальные выбросы *должны* быть редкими. Если какое-либо из условий не выполняется, вероятно, существует простой способ улучшить процедуру прогнозирования. Например, если нет центрирования вокруг нуля, добавление постоянного смещения ко всем прогнозам улучшит согласованные результаты.

На рис. 7.8 представлены распределения абсолютных ошибок из двух моделей для прогнозирования года авторства документов по распределению использованных в них слов. Слева мы видим распределение ошибок для обезьяны, случайно угадывавшей годы с 1800 по 2005. Что мы видим? Распределение ошибок является широким и плохим, как мы могли ожидать, но также и асимметричным. Гораздо больше документов вызывало положительные ошибки, чем отрицательные. Почему? Тестовый корпус, по-видимому, содержал больше современных документов, чем старых, поэтому результат (*year–monkey year*) чаще положительный, чем отрицательный. Даже обезьяна может чему-то научиться, увидев распределение.

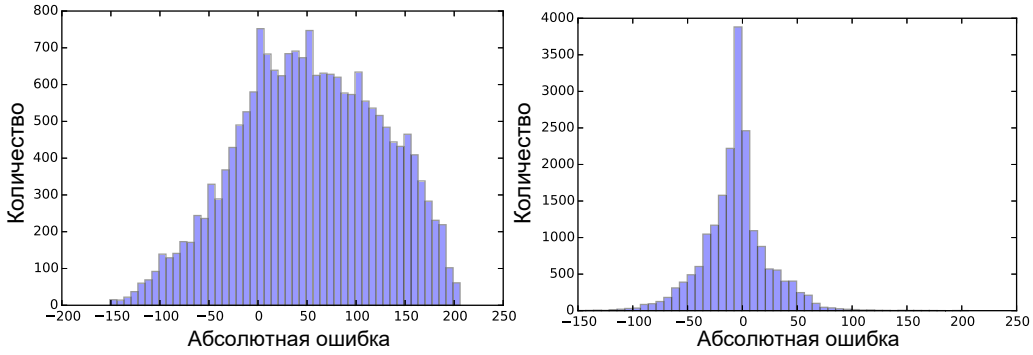


Рис. 7.8. Гистограммы распределения ошибок для случайных (слева) и наивных байесовских классификаторов, предсказывающих год авторства документов (справа)

В отличие от нее, на рис. 7.8 (справа) представлено распределение ошибок для нашего наивного байесовского классификатора по датировке документов. Это выглядит намного лучше: острый пик около нуля и гораздо более узкие хвосты. Но длинный хвост теперь располагается слева от нуля, что говорит нам о том, что мы все еще имеем огорчительно мало старых документов по сравнению с современными. Нам нужно изучить некоторые из этих случаев, чтобы выяснить, почему это так.

Нам нужна сводная статистика, сводящая такое распределение ошибок к одному числу, чтобы сравнить производительность различных моделей прогнозирования значений. Обычно используется такая статистика, как *средний квадрат ошибки* (Mean Squared Error — MSE), которая вычисляется как

$$MSE(Y,Y') = \frac{1}{n}\sum_{i=1}^{n}(y_i' - y_i)^2.$$

Поскольку каждый член взвешивается квадратично, выбросы имеют непропорциональный эффект. Таким образом, среднеквадратическая ошибка может быть более информативной статистикой для шумных случаев.

Среднеквадратическая ошибка (Root mean squared RMSD) — это просто квадратный корень из среднего квадрата ошибки:

$$RMSD(\theta) = \sqrt{MSE(Y,Y')}.$$

Преимущество RMSD в том, что его величина интерпретируется в том же масштабе, что и исходные значения, так же как стандартное отклонение является более интерпретируемой величиной, чем дисперсия. Но это не устраняет проблему, заключающуюся в том, что выбросы могут существенно искажать общее количество.

7.5. Оценка среды

Существенная часть любого проекта науки о данных вращается вокруг создания разумной среды оценки. В частности, вам нужна *программа с одной командой* (single-command program) для запуска вашей модели на оценочных данных и создания графиков/отчетов о ее эффективности, как показано на рис. 7.9.

Почему одна команда? Если вам не будет легко ее запускать, вы не будете делать это достаточно часто. Если результаты сложно читать и интерпретировать, вы не сможете собрать достаточно информации, чтобы она того стоила.

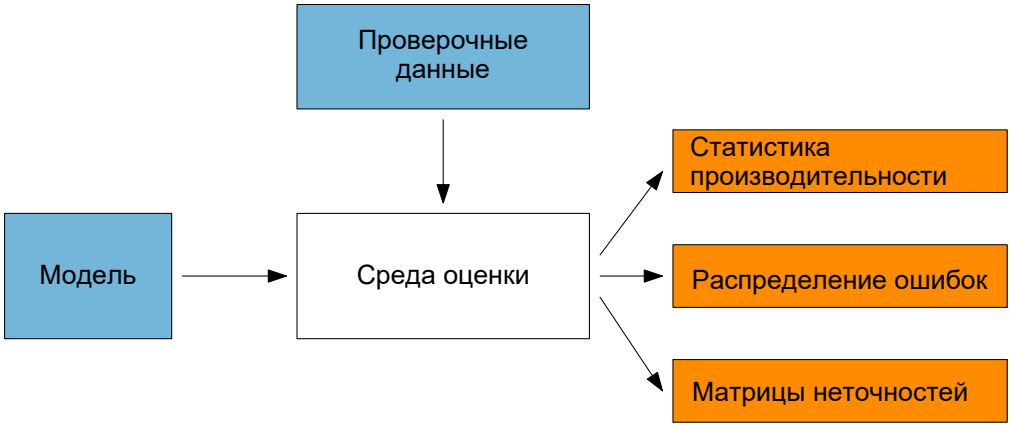


Рис. 7.9. Блок-схема базовой модели оценки среды

Входные данные для среды оценки представляют собой набор экземпляров с соответствующими выходными результатами/метками, а также проверяемой моделью. Система запускает модель для каждого экземпляра, сравнивает каждый результат с его золотым стандартом и выводит сводную статистику и графики распределения, показывающие производительность, достигнутую на этом тестовом наборе.

Хорошая система оценки обладает следующими свойствами.

- В дополнение к двоичным результатам она дает распределение ошибок: насколько близок был ваш прогноз, а не только был ли он правильным или неправильным. Вспомните рис. 7.8 для вдохновения.
- Она автоматически создает отчет с несколькими графиками о нескольких разных входных распределениях, чтобы внимательно прочитать его на досуге.
- Она выводит соответствующую сводную статистику о производительности, чтобы вы могли быстро оценить качество. Вы сделали лучше или хуже, чем в прошлый раз?

В качестве примера на рис. 7.10 представлен результат нашей среды оценки для двух моделей датирования документов, представленных в предыдущем разделе. Напомним, что задача заключается в том, чтобы предсказать год авторства данного документа по использованию слов. Что стоит отметить?

- *Тестовые наборы с разбивкой по типу.* Обратите внимание, что среда оценки разбила входные данные на девять отдельных подмножеств, включая новости (News) и беллетристику (Fiction) длиной от 100 до 2000 слов. Таким образом, мы сразу видим, насколько хорошо мы справляемся с каждым.

	Dataset	Method	MAE	MedAE	Acc
0	NYTimes	Random	73.335463	65.0	0.004895
1	COHA_Fiction_100	Random	79.865017	72.0	0.005287
2	COHA_Fiction_500	Random	80.505849	74.0	0.003825
3	COHA_Fiction_1000	Random	80.604837	72.0	0.003825
4	COHA_Fiction_2000	Random	79.845332	72.0	0.005737
5	COHA_News_100	Random	66.539239	59.0	0.005461
6	COHA_News_500	Random	66.267091	59.0	0.005461
7	COHA_News_1000	Random	66.077670	57.5	0.004956
8	COHA_News_2000	Random	66.225526	58.0	0.005057

	Dataset	Method	MAE	MedAE	Acc
0	NYTimes	NB	21.306301	14	0.029728
1	COHA_Fiction_100	NB	32.302025	22	0.041732
2	COHA_Fiction_500	NB	25.428234	14	0.050056
3	COHA_Fiction_1000	NB	23.493926	13	0.053656
4	COHA_Fiction_2000	NB	22.493363	12	0.054781
5	COHA_News_100	NB	19.384001	14	0.030845
6	COHA_News_500	NB	16.657565	12	0.034891
7	COHA_News_1000	NB	16.282261	12	0.035093
8	COHA_News_2000	NB	16.220065	12	0.035599

Рис. 7.10. Результаты оценки среды для прогнозирования года авторства документов, сравнивающей обезьяну (слева) с наивным байесовским классификатором (справа)

- *Логические прогрессии сложности.* Очевидно, что определить возраст коротких документов куда труднее, чем более длинных. Разделяя более сложные и более простые случаи, мы лучше понимаем наш источник ошибок. Мы видим значительное улучшение в наивном байесовском классификаторе, когда переходим от 100 до 500 слов, но эти успехи заканчиваются после 2000 слов.
- *Проблема соответствующей статистики.* Мы не выводим каждую возможную метрику ошибок, только среднюю и среднюю абсолютную ошибку, а также корректность (как часто мы получали год правильно?). Этого достаточно для того, чтобы мы увидели, что новости легче, чем беллетристика, что наша модель намного лучше, чем обезьяна, и что наши шансы правильно определить фактический год (измеряется корректностью) все еще слишком малы, чтобы о них беспокоиться.

Эта оценка дает информацию, которая нужна, чтобы узнать, как идут дела, не перегружая нас цифрами, которые мы никогда не увидим.

7.5.1. Гигиена данных для оценки

Оценка имеет смысл только тогда, когда вы не обманываете себя. Ужасные вещи случаются, когда люди оценивают свои модели небрежно, не различая учебные, проверочные и оценочные данные.

После определения позиции набора данных для построения прогнозирующей модели ваша первая задача должна заключаться в разделении входных данных на три части.

- *Учебные данные.* Это то, с чем вы можете играть совершенно свободно. Используйте их для изучения предметной области и установки параметров

вашей модели. Обычно около 60% полного набора данных должно быть посвящено обучению.

- *Проверочные данные*. Включают около 20% полного набора данных, это то, что вы используете для оценки, насколько хороша ваша модель. Обычно люди экспериментируют с несколькими подходами машинного обучения или базовыми настройками параметров, поэтому проверка позволяет установить относительную производительность всех этих разных моделей для одной и той же задачи.

  Проверка модели, как правило, показывает, что она работает не так, как хотелось бы, и, таким образом, начинается еще один цикл проектирования и доработки. Низкая производительность в тестовых данных по сравнению с учебными данными наводит на мысль о переобучении модели.

- *Оценочные данные*. Последние 20% данных должны быть отложены на дождливый день: для подтверждения работоспособности окончательной модели непосредственно перед ее введением в эксплуатацию. Это сработает только в том случае, если вы никогда не открывали оценочные данные, пока они действительно не стали необходимы.

Причина такого разделения должна быть очевидна. На экзаменах студентам было бы намного лучше, если бы им заранее был предоставлен доступ к ответу, поскольку они точно знали бы, что учить. Но это не будет отражать того, насколько высоки их знания на самом деле. Хранение проверочных данных отдельно от учебных гарантирует, что тесты измеряют что-то важное в том, что понимает модель. А хранение окончательных оценочных данных для использования только после того, как модель станет стабильной, гарантирует, что специфика проверочного набора не проникла в модель через повторяющиеся итерации проверок. Оценочный набор служит вневыборочными (out-of-sample) данными для проверки окончательной модели.

Выполняя первоначальное разбиение, вы должны быть осторожны, чтобы не создавать нежелательные артефакты и не уничтожать желаемые. Простое разбиение файла в том порядке, в котором он был задан, опасно, поскольку любое структурное различие между совокупностями корпуса обучения и проверки означает, что модель будет работать не так, как должна.

Но предположим, что вы строили модель для прогнозирования будущих цен на акции. Было бы довольно опасно случайным образом выбирать 60% примеров по всей истории в качестве обучающих данных вместо всех примеров за первые 60% времени. Почему? Предположим, что ваша модель "изучает", что будет временем роста и спада на рынке по учебным данным, а затем использует эту информацию для создания виртуальных прогнозов для других

акций в эти же дни. Такая модель будет работать намного лучше при проверке, чем на практике. Правильные методы отбора проб — дело довольно тонкое, оно обсуждается в разделе 5.2.

Важно как можно дольше хранить завесу тайны над вашими оценочными данными, поскольку вы испортите их, как только используете. Шутки никогда не бывают смешными, когда вы их слышите во второй раз, после того, как вы уже знаете суть. Если вы нарушаете целостность своих наборов для проверки и оценки, лучшее решение — начать со свежих данных, вневыборочных, но это не всегда доступно. В противном случае случайным образом разделите полный набор данных на новые образцы для обучения, проверки и оценки и повторно обучите все свои модели с нуля, чтобы перезапустить процесс. Но это должно быть признано неудачным результатом.

7.5.2. Усиление малых оценочных наборов

Идея жесткого разделения входных данных на обучающие, тестовые и оценочные наборы имеет смысл только для достаточно больших наборов данных. Предположим, у вас есть 100 000 записей. Качественной разницы между обучением на 60 000 записях вместо 100 000 не будет, поэтому лучше проводить тщательную оценку.

Но что, если у вас есть только несколько десятков примеров? На момент написания этой книги было всего 45 президентов США, поэтому любой анализ, который вы можете провести по ним, представляет очень небольшую выборочную статистику. Новые данные поступают очень медленно, только раз в четыре года или около того. Подобные проблемы возникают в медицинских исследованиях, которые очень дороги в проведении, потенциально давая данные о порядка ста пациентах. Любое приложение, в котором нам придется платить за аннотации людей, означает, что у нас будет меньше данных для обучения, чем нам хотелось бы.

Что вы можете сделать, если не можете позволить себе отдать часть своих данных для тестирования? *Перекрестная проверка* (cross-validation) разделяет данные на k частей равного размера, а затем обучает k различных моделей. Модель i обучается на объединении всех $x \neq i$ блоков, суммируя $(k-1)/k$ данных, и проверяется на выделенном i-м блоке. Средняя производительность этих k классификаторов является предполагаемой корректностью для всей модели.

Реальное преимущество перекрестной проверки заключается в том, что она дает стандартное отклонение производительности, а не только среднее значение. Каждый классификатор, обученный на определенном подмножестве данных, будет немного отличаться от своих аналогов. Кроме того, тестовые данные для каждого классификатора будут отличаться, что приведет к разным показателям

производительности. Сочетание среднего значения со стандартным отклонением и допущением нормальности дает вам распределение производительности и лучшее представление о том, насколько хорошо можно доверять результатам. Это делает перекрестную проверку очень полезной и для *больших* наборов данных, поскольку вы можете позволить себе сделать несколько разделов и переобучиться, увеличивая таким образом уверенность в том, что ваша модель хороша.

Что из k моделей, полученных в результате перекрестной проверки, следует выбрать в качестве конечного продукта? Возможно, вы могли бы использовать тот, который показал наилучшие результаты по квоте тестирования. Но лучшей альтернативой является перетренировка *всех* данных и уверенность в том, что они будут по крайней мере такими же хорошими, как и менее обученные модели. Это не идеально, но если вы не можете получить достаточно данных, вы должны делать все возможное с тем, что у вас есть.

Вот несколько других идей, которые могут помочь усилить небольшие наборы данных для обучения и оценки.

- *Создайте отрицательные примеры из предыдущего распределения.* Предположим, некто хотел создать классификатор, чтобы определить, кто будет выбран кандидатом в президенты. Реальных примеров кандидатов в президенты (положительные примеры) очень мало, но, по-видимому, элитный резерв настолько мал, что случайный человек почти наверняка не будет выбран. Когда положительные примеры редки, все остальные, скорее всего, отрицательны и могут быть помечены так, чтобы при необходимости предоставлять данные для обучения.
- *Исказите реальные примеры, чтобы создать похожие, но искусственные.* Полезный прием, позволяющий избежать переобучения, подразумевает создание новых учебных экземпляров за счет добавления случайного шума, искажающего помеченные примеры. Затем мы сохраняем исходную метку результата с новым экземпляром.

  Предположим, например, что мы пытаемся обучить систему оптического распознавания символов (OCR) распознавать буквы некого алфавита на отсканированных страницах. Первоначально оплачиваемому сотруднику было дано задание пометить несколько сотен изображений содержащимися в них символами. Мы можем увеличить это количество до нескольких миллионов изображений, добавляя случайный шум и поворачивая, переводя и расширяя интересующую область. Классификатор, обученный на этих синтетических данных, должен быть гораздо более надежным, чем классификатор, ограниченный исходными аннотированными данными.

- *Выдайте частичный кредит, если можете.* Если у вас меньше примеров для обучения/тестирования, чем вы хотите, вы должны выжать как можно больше информации из каждого из них.

  Предположим, что наш классификатор выводит значение, измеряющее его уверенность в своем решении, в дополнение к предложенной метке. Этот уровень достоверности дает нам дополнительное разрешение для оценки классификатора, помимо того, правильно ли он получил метку. Это более серьезный удар по классификатору, когда он делает неверный прогноз, чем в случае, когда он полагал, что ответ был ошибочным. Что касается задачи с президентами, я бы поверил классификатору, который получил 30 правильных и 15 неправильных ответов с точными значениями достоверности намного больше, чем таковому с 32 правильными и 13 неправильными, но с достоверными значениями по всей карте.

7.6. Случай из жизни: 100% корректности

Два бизнесмена выглядели немного неловко в университете в неуместных темно-синих костюмах среди наших шорт и кроссовок. Их звали Пабло и Хуан. Но они нуждались в нас, чтобы сделать свое в*и*дение реальностью.

"Деловой мир все еще работает на бумаге", — объяснил Пабло. Он был один в темном костюме. "У нас есть контракт на оцифровку всех финансовых документов Уолл-стрит, которые все еще печатаются на бумаге. Они заплатят нам целое состояние за то, чтобы заставить компьютер осуществлять сканирование. Прямо сейчас они нанимают людей, чтобы печатать каждый документ в трех экземплярах, просто чтобы гарантировать, что они поняли его абсолютно правильно".

Это звучало захватывающе, и у них были ресурсы, чтобы это осуществить. Но был один нюанс. "Наша система не может делать никаких ошибок. Иногда можно сказать "я не знаю". Но всякий раз, когда она называет букву, она должна быть на 100% корректной".

"Нет проблем". Сказал я им. "Просто позвольте ей сказать "я не знаю" в 100% случаев, но я могу разработать систему, соответствующую вашим требованиям".

Пабло нахмурился. "Но это будет стоить нам целое состояние. В системе, которую мы хотим построить, изображения случаев *я не знаю* будут переданы операторам-людям для чтения. Но мы не можем позволить им платить столько, чтобы они прочитали все".

Мои коллеги и я согласились взяться за эту работу, и со временем мы разработали разумную систему распознавания текста с нуля. Но одна вещь беспокоила меня.

"Эти парни с Уолл-стрит, которые дали вам деньги, они умны, не так ли?" — однажды спросил я у Пабло.

"Чертовски умны", — ответил он.

"Тогда как они могли поверить, когда вы сказали, что можете построить систему OCR, которая была бы корректной на 100%?"

"Поскольку они думали, что я уже делал это раньше", — сказал он со смехом.

Похоже, что предыдущая компания Пабло создала систему, которая оцифровывала данные о ценах с телевизионных мониторов того времени. В этом случае буквы представляли собой точные шаблоны битов, все они были написаны абсолютно одинаковым шрифтом и одинакового размера. Телевизионный сигнал также был цифровым, без каких-либо ошибок в изображении, случайно образовавшихся пятен чернил от принтеров, темных пятен или складок на бумаге. Было вполне тривиально проверить точное соответствие между идеальным шаблоном битов (изображением) и другим идеальным шаблоном битов (символ в шрифте устройства), поскольку источника неопределенности не было. Но эта проблема не имела ничего общего с распознаванием текста, даже если оба были связаны только с чтением букв.

Наша разумная система оптического распознавания символов сделала все возможное с предоставленными им деловыми документами, но, конечно, мы не смогли достичь 100%. В конце концов, ребята с Уолл-стрит перевели свой бизнес на Филиппины, где заплатили трем людям за то, что они напечатали каждый документ и сравнивали два из трех, если возникли разночтения.

Мы сменили направление и занялись чтением рукописных опросных листов, представленных потребителями, соблазненными обещаниями купонов для продуктовых магазинов. Эта задача была сложнее, но ставки не так высоки: они платили нам каких-то 0,22 долл. за форму и не ожидали совершенства. Нашей проблемой было дело, которое использовало каторжный труд для набора данных. Нас привлекли в момент, когда один из этих заключенных отправил письмо с угрозами на адрес, который они нашел в форме опроса, и этот бизнес перебросили нам. Но даже наша обширная автоматизация не смогла прочитать эти формы менее чем по 0,40 долл. за штуку, поэтому контракт вернулся в тюрьму после того, как мы свернули себе мозги.

Основной урок здесь в том, что ни одна система распознавания образов для любой разумной проблемы не будет точна в 100% случаев. Единственный способ никогда не ошибаться — никогда не делать прогнозов. Чтобы узнать, насколько хорошо работает ваша система и где она допускает ошибки, необходима тщательная оценка, чтобы сделать систему лучше.

7.7. Имитационные модели

Существует важный класс моделей первого принципа, которые основаны в первую очередь не на данных, но оказываются очень ценными для понимания самых разнообразных явлений. *Моделирование* (simulation) — это создание моделей, которые пытаются воспроизвести реальные системы и процессы, чтобы мы могли наблюдать и анализировать их поведение.

Модели важны для демонстрации правильности нашего понимания системы. Простая модель, которая отражает бо́льшую часть поведенческой сложности системы, должна объяснить, как она работает. Известный физик Ричард Фейнман (Richard Feynman) сказал: "То, чего я не могу создать, я не понимаю". То, чего вы не можете смоделировать и получить некоторый уровень точности в наблюдаемых результатах, вы не понимаете.

Модель *Монте-Карло* использует случайные числа для синтеза альтернативных реалий. Воспроизведение события миллионы раз в слегка измененных условиях позволяет нам генерировать распределение вероятностей на множестве результатов. Это была идея критерия перестановок для статистической значимости. Мы также увидели (в разделе 5.5.2), что случайные броски монет могут заменять попадания или хиты бэттера, поэтому мы могли смоделировать произвольное количество карьер и наблюдать, что происходило в течение них.

Ключом к построению эффективной модели Монте-Карло является разработка подходящей модели дискретного события. Моделью репликации каждого решения или результата события используется новое случайное число. В транспортной модели, возможно, вам придется решать, идти ли налево или направо, поэтому подбросьте монету. Модель в области здравоохранения или страховки, возможно, должна решать, будет ли у конкретного пациента сердечный приступ сегодня, поэтому подбросьте соответственно взвешенную монету. Цена акции в финансовой модели может увеличиваться или уменьшаться любую секунду, что опять же может быть броском монеты. Баскетболист попадет или промажет, вероятность зависит от его навыков и способностей защитника.

Точность такой модели зависит от вероятностей, которые вы назначаете для орла и решки. Это определяет, как часто встречается каждый результат. Вполне очевидно, что вы не ограничены использованием идеальной монеты, т.е. только 50/50. Вместо этого вероятности должны отражать предположения о вероятности события с учетом состояния модели. Эти параметры часто устанавливаются с помощью статистического анализа, наблюдая за распределением событий в том виде, в котором они встречаются в данных. Частично ценностью модели Монте-Карло является то, что она позволяет нам играть с альтернативными реалиями, изменяя определенные параметры и наблюдая за тем, что происходит.

Критическим аспектом эффективного моделирования является оценка. Ошибки в программировании и недостатки в моделировании достаточно распространены, поэтому модель не может быть принята на веру. Главное — удерживаться от непосредственного включения в вашу модель одного или нескольких классов наблюдений системы. Это обеспечивает поведение вне выборки, поэтому мы можем сравнить распределение результатов моделирования с этими наблюдениями. Если они не вяжутся, ваша модель — просто пустая болтовня. Не допускайте такого.

7.8. Случай из жизни: вычисление ставок

Там, где есть азартные игры, есть деньги, а там, где есть деньги, будут модели. В детстве во время наших семейных поездок во Флориду у меня появилась страсть к такому спорту, как хай-алай. И, когда я научился строить математические модели как взрослый, я стал одержим разработкой прибыльной системы ставок для спорта.

Хай-алай — это спорт баскского происхождения, когда противоборствующие игроки или команды попеременно бросают мяч в стену и ловят его, пока один из них, наконец, не промахивается и не потеряет очко. Бросание и ловля осуществляются с помощью удлиненной корзины, или *цеста* (cesta), мяч, или *пелота* (pelota), сделаны из козьей кожи и твердой резины, а стена — из гранита или бетона; ингредиенты, которые приводят к быстрому и захватывающему действию, изображенному на рис. 7.11. В Соединенных Штатах хай-алай больше всего ассоциируется с штатом Флорида, где разрешены ставки на результаты матчей.

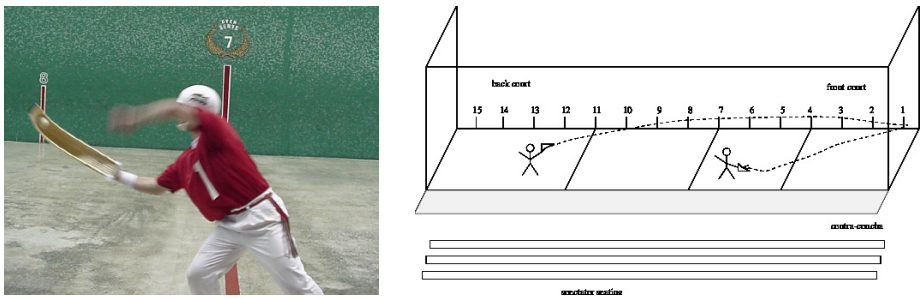


Рис. 7.11. Хай-алай — быстрая, захватывающая игра с мячом, похожая на гандбол, и вы можете на нее ставить

Что делает хай-алай особенно интересным, так это его уникальная и *очень* своеобразная система начисления очков. В каждом матче хай-алая участвуют восемь игроков, с номерами от 1 до 8, чтобы отразить их порядок в игре.

Каждый матч начинают игроки 1 и 2, а остальные терпеливо ждут своей очереди. Все игроки начинают игру с нулевыми очками у каждого. Каждое очко в матче подразумевает двух игроков — того, кто победит, и того, кто проиграет. Проигравший доходит до конца линии, а победитель прибавляет к сумме очков и ждет следующего очка, пока не наберет достаточно очков, чтобы претендовать на матч.

Для меня даже в детстве было очевидно, что эта система начисления очков не будет одинаково справедливой для всех игроков. Находящиеся в начале очереди имеют больше шансов сыграть, и даже *клудж* (kludge), добавленный ими для удвоения значения очков, позже в матче не сможет этого исправить. Но понимание силы этих неравноправий может дать мне преимущество в ставках.

Мое стремление построить систему ставок для хай-алая началось с моделирования этой очень необычной системы подсчета очков. Матч хай-алая состоит из последовательности отдельных событий, описываемых следующей структурой.

Инициализация текущих игроков как 1 и 2.
Инициализация очереди игроков как {3, 4, 5, 6, 7, 8}.
Инициализация общего количества очков каждого игрока нулевым значением.
Пока текущий победитель имеет менее 7 очков:
– выберите случайное число, чтобы решить, кто победит в следующей точке;
– добавьте единицу (или две, если за седьмым очком) к сумме очков моделируемого победителя;
– поместите модель проигравшего в конец очереди;
– получите следующего игрока с головы очереди.
И *так далее*.
Определите победителя текущего очка как победителя матча.

Единственный требующий дополнительной проработки этап — это моделирование пункта между двумя игроками. Если цель нашего моделирования заключается в том, чтобы увидеть, как предвзятости в системе начисления очков влияют на исход матча, то имеет смысл рассмотреть случай, когда все игроки одинаково умелые. Чтобы дать каждому игроку шанс 50/50 выиграть каждое очко, которое они разыгрывают, мы можем подбросить смоделированную монету, чтобы определить, кто победит, а кто проиграет.

Я реализовал симуляцию хай-алай на своем любимом языке программирования и запустил ее на 1 000 000 игр хай-алай. В результате симуляции была получена статистическая таблица, показывающая, как часто окупаются все результаты ставок, при условии, что все игроки одинаково искусны. На рис. 7.12

показано количество смоделированных побед для каждой из восьми стартовых позиций. Какие идеи мы можем извлечь из этой таблицы?

Позиция	Модель		Наблюдение	
		Процент побед	Победы	Процент побед
1	162 675	16,27	1750	14,1
2	162 963	16,30	1813	14,6
3	139 128	13,91	1592	12,8
4	124 455	12,45	1425	11,5
5	101 992	10,20	1487	12,0
6	102 703	10,27	1541	12,4
7	88 559	8,86	1370	11,1
8	117 525	11,75	1405	11,3
			12 383	100,0

Рис. 7.12. Наблюдаемые пристрастия по выигрышам в моделируемых матчах хай-алай хорошо совпадают с результатами, полученными в реальных матчах

- Позиции 1 и 2 имеют существенное преимущество перед остальной частью поля. Любой из первых игроков почти в два раза чаще окажется первым, вторым или третьим, чем бедняга в позиции 7.
- Позиции 1 и 2 выигрывают с одинаковой частотой. Так и должно быть, поскольку оба игрока начинают игру на корте, а не в очереди. Тот факт, что игроки 1 и 2 имеют очень схожую статистику, увеличивает нашу уверенность в правильности симуляции.
- Позиции 1 и 2 не имеют *одинаковой* статистики, поскольку мы смоделировали "только" один миллион игр. Если вы подбросите монету миллион раз, она почти наверняка не даст ровно половины орлов и половины решек. Однако *соотношение* орлов к решкам должно приближаться к 50/50, если мы будем подбрасывать больше монет.

  Моделируемый разрыв между игроками 1 и 2 говорит нам нечто об *ограничениях* на корректность нашего моделирования. Мы не должны доверять никаким выводам, которые зависят от таких небольших различий в наблюдаемых значениях.

Чтобы проверить корректность модели, мы сравнили наши результаты со статистикой по фактическим результатам более чем 12 000 матчей хай-алай, также см. на рис. 7.12. Результаты в основном согласуются с моделированием при учете ограничений малого размера выборки. Позиции 1 и 2 выигрывали чаще всего в реальных матчах, а позиция 7 реже.

Теперь мы знали вероятность того, что каждая возможная ставка в хай-алае окупилась. Были ли мы готовы начать зарабатывать деньги? К сожалению, нет. Несмотря на то что мы установили, что позиция в очереди является основным фактором, определяющим исход матчей хай-алай, возможно, доминирующим, нам еще пришлось преодолеть несколько препятствий, прежде чем мы могли сделать осмысленную ставку.

- *Влияние навыков игрока.* Очевидно, что хороший игрок выиграет с большей вероятностью, чем плохой, независимо от его позиции в очереди. Понятно, что лучшая модель для прогнозирования исхода матчей хай-алай будет учитывать относительные навыки в модели очереди.
- *Догадливость общественности.* Многие люди заметили влияние позиции раньше, чем я. Анализ данных показал, что люди, делающие ставки в хай-алай, в значительной степени учли влияние позиции на шансы. К счастью для нас, однако, во многом не означало полностью.
- *Срез дома* (house cut). Фронтоны (frontons) содержат порядка 20% всего пула ставок от дома (house), и поэтому нам пришлось намного легче, чем обычным игрокам, чтобы просто достичь безубыточности.

Моя модель предоставила информацию о том, какие результаты были наиболее вероятными. Само по себе это не определяло, какие из ставок лучше. Хорошая ставка зависит как от вероятности наступления события, так и от выплаты, если оно произойдет. Выплаты определяются остальной совокупностью ставок. Чтобы найти лучшие ставки, нам пришлось работать намного усерднее.

- Мы должны были проанализировать данные прошлых матчей, чтобы определить, кто был лучше. Как только мы бы узнали, кто был лучше, мы смогли бы сместить броски монет в модели в их пользу, чтобы сделать нашу модель корректней для каждого отдельного матча.
- Мы должны были проанализировать данные о выплате, чтобы построить модель предпочтений *другого* делающего ставки. В хай-алай вы делаете ставки против публики, поэтому вам нужно иметь возможность смоделировать их мышление, чтобы предсказать выплаты по конкретной ставке.
- Мы должны были смоделировать влияние сечения дома на пул ставок. Определенные ставки, которые в противном случае могли бы быть прибыльными, попадают в красную зону, когда вы учитываете эти затраты.

Суть в том, что мы сделали это с 544%-ной прибылью на нашу первоначальную ставку. Полная история нашей игровой системы описана в моей книге *Calculated Bets* [8]. Прочитайте ее: держу пари, вам понравится. Забавно читать об успешных моделях, но еще интереснее создавать их.

7.9. Дополнительная информация

Силвер [3] является отличным введением в сложности моделирования и прогнозирования в различных областях. К хорошим учебникам по вопросам математического моделирования относятся Бендер [69] и Джордано [70].

Проект Google Flu Trends является отличным примером как в отношении преимуществ, так и недостатков анализа больших данных. Имеет смысл отметить Гинсберга и др. [71] за оригинальность описания и Лазера и др. [72] за увлекательный рассказ о том, как все пошло не так.

Технические аспекты системы OCR, представленные в разделе 7.6, описаны у Сазаклис и др. [73]. Работа над определением года авторства (и пример соответствующей среды оценки) принадлежит моим ученикам Вивеку Кулкарни, Парту Дандивале (Parth Dandiwala) и Йингтао Тьян [74].

7.10. Упражнения

Свойства моделей

7.1. *[3]* Квантовая физика намного сложнее, чем ньютоновская. Какая модель проходит проверку бритвой Оккама и почему?

7.2. *[5]* Определите набор представляющих интерес моделей. Для каждого из них определите, какие свойства имеют эти модели.
(a) Они дискретны или непрерывны?
(b) Они линейные или нелинейные?
(c) Это черный ящик или они описательны?
(d) Они являются общими или специальными?
(e) Они основаны на данных или на первом принципе?

7.3. *[3]* Приведите примеры моделей, основанных на данных и на первом принципе, которые используются на практике.

7.4. *[5]* Обсудите для одной или нескольких из следующих задач *The Quant Shop*, какие из моделей являются более перспективным подходом на базе принципа или управляемых данных.
- *Miss Universe.*
- *Movie gross.*
- *Baby weight.*
- *Art auction price.*
- *Snow on Christmas.*
- *Super Bowl/college champion.*
- *Ghoul pool.*
- *Future gold/oil price.*

7.5. *[5]* Разделите одну или несколько из следующих задач *The Quant Shop* на подзадачи, которые можно смоделировать независимо.
- *Miss Universe.*
- *Movie gross.*
- *Baby weight.*
- *Art auction price.*
- *Snow on Christmas.*
- *Super Bowl/college champion.*
- *Ghoul pool.*
- *Future gold/oil price.*

Среды оценки

7.6. *[3]* Предположим, вы создаете классификатор, который отвечает *да* на каждый возможный ввод. Какую точность и отзыв даст этот классификатор?

7.7. *[3]* Объясните, что такое точность и отзыв. Как они связаны с кривой ROC?

7.8. *[5]* Лучше иметь слишком много ложно положительных или слишком много ложно отрицательных? Объясните.

7.9. *[5]* Объясните, что такое переобучение и как вы будете его контролировать.

7.10. *[5]* Предположим, что $f \leq 1/2$ — доля положительных элементов в классификации. Какова вероятность p, что обезьяна сможет угадать положительные значения как функцию от f, чтобы максимизировать конкретный показатель оценки, приведенный ниже? Укажите p и ожидаемый балл оценки, достигнутый обезьяной.
(a) Корректность.
(b) Точность.
(c) Отзыв.
(d) F-оценка.

7.11. *[5]* Что такое перекрестная проверка? Как мы можем выбрать правильное значение k для перекрестной проверки в k-кратном размере?

7.12. *[8]* Как мы можем узнать, собрали ли мы достаточно данных для обучения модели?

7.13. *[5]* Объясните, почему у нас есть наборы данных для обучения, проверки и оценки и как их использовать эффективно?

7.14. *[5]* Предположим, мы хотим обучить бинарный классификатор, где один класс очень редок. Приведите пример такой задачи. Как мы должны обучать эту модель? Какие показатели мы должны использовать для измерения производительности?

7.15. *[5]* Предложите базовые модели для одной или нескольких из следующих задач *The Quant Shop*.

- *Miss Universe.*
- *Movie gross.*
- *Baby weight.*
- *Art auction price.*
- *Snow on Christmas.*
- *Super Bowl/college champion.*
- *Ghoul pool.*
- *Future gold/oil price.*

Реализация проектов

7.16. *[5]* Создайте модель для прогнозирования результатов одного из следующих типов событий и тщательно проанализируйте ее с помощью обратной проверки.

(a) Такие виды спорта, как футбол, баскетбол и скачки.

(b) Объединенные ставки, включающие несколько событий, таких как футбольные чемпионаты или баскетбольный турнир NCAA.

(c) Азартные игры, такие как лотереи, фэнтези-спорт и покер.

(d) Прогнозы местных выборов и выборов в конгресс.

(e) Прогноз цен на акции или товары.

Тщательное тестирование, вероятно, подтвердит, что ваши модели недостаточно сильны для прибыльных ставок, и это на 100% нормально. Будьте честны: убедитесь, что вы используете достаточно актуальные цены и шансы, чтобы отразить возможности ставок, которые все еще будут доступны на момент размещения моделированной ставки. Чтобы убедить меня, что ваша модель на самом деле действительно прибыльна, пришлите мне немного денег, и тогда я поверю вам.

7.17. *[5]* Создайте общую систему оценки моделей на своем любимом языке программирования и настройте ее с правильными данными для оценки моделей по конкретной проблеме. Ваша среда должна сообщать статистику производительности, распределение ошибок и/или матрицы неточностей в зависимости от ситуации.

Вопросы на интервью

7.18. *[3]* Оцените априорные вероятности для следующих событий.

(a) Солнце взойдет завтра.

(b) Большая война с участием вашей страны начнется в следующем году.

(c) Новорожденный ребенок будет жить до 100 лет.

(d) Сегодня ты встретишь человека, за которого выйдешь замуж.

(e) Chicago Cubs выиграют World Series в этом году.

7.19. *[5]* Что мы имеем в виду, когда говорим о дилемме смещения — дисперсии?

7.20. *[5]* Тест дает истинно положительную оценку 100% и ложно положительную 5%. В этой популяции 1 из 1000 человек имеет условие, которое определяет тест. Учитывая положительный тест, какова вероятность того, что этот человек на самом деле имеет это состояние?

7.21. *[5]* Что лучше: иметь хорошие данные или хорошие модели? А как вы определяете хорошее?

7.22. *[3]* Что вы думаете об идее введения шума в ваш набор данных для проверки чувствительности ваших моделей?

7.23. *[5]* Как бы вы определили и измерили прогностическую силу метрики?

Конкурсы Kaggle

7.24. Будет ли финансироваться конкретная заявка на грант?
`https://www.kaggle.com/c/unimelb`

7.25. Кто победит в баскетбольном турнире NCAA?
`https://www.kaggle.com/c/march-machine-learning-mania-2016`

7.26. Прогноз ежегодных продаж в данном ресторане.
`https://www.kaggle.com/c/restaurant-revenue-prediction`.

Глава 8

Линейная алгебра

> Мы нередко слышим, что математика состоит в основном из "доказуемых теорем". Является ли работа писателя в основном "написанием предложений"?
>
> — Джан-Карло Рота (Gian-Carlo Rota)

Элемент данных вашего проекта в науке о данных включает в себя часть всей релевантной информации, которую вы можете найти, в одной или нескольких матрицах данных, в идеале как можно большего размера. Строки каждой матрицы представляют элементы или примеры, в то время как столбцы представляют различные элементы или атрибуты.

Линейная алгебра — это математика матриц: свойства расположений чисел и операции, которые можно с ними выполнять. Это делает ее языком науки о данных. Многие алгоритмы машинного обучения лучше всего понять через линейную алгебру. Алгоритмы для таких задач, как линейная регрессия, могут быть сведены к одной формуле за счет умножения правильной цепочки матричных произведений, чтобы получить желаемые результаты. Такие алгоритмы могут быть одновременно и простыми, и пугающе сложными, и тривиальными в реализации, и неординарными с точки зрения эффективности и надежности.

Возможно, вы когда-то проходили курс по линейной алгебре, но, вероятно, многое забыли. Здесь я рассмотрю бо́льшую часть того, что вам нужно знать: основные операции над матрицами, чем они полезны, и как выработать интуицию о том, что они делают.

8.1. Сила линейной алгебры

Почему линейная алгебра столь сильна? Она регулирует работу матриц, а матрицы везде. Матричные представления важных объектов включают следующее.

- *Данные*. Наиболее полезное представление числовых наборов данных в виде матриц размером $n \times m$. Строки количеством n представляют объекты, элементы или экземпляры, в то время как каждый из m столбцов представляет отдельные элементы или измерения.

- *Наборы геометрических точек*. Матрица $n \times m$ может представлять собой облако точек в пространстве. Каждая из n строк представляет геометрическую точку, а столбцы m определяют размеры. Некоторые матричные операции имеют различные геометрические интерпретации, что позволяет нам обобщать двумерную геометрию, которую мы можем фактически визуализировать в многомерные пространства.
- *Системы уравнений*. линейное уравнение определяется суммой переменных, взвешенных по постоянным коэффициентам, например:

$$y = c_0 + c_1 x_1 + c_2 x_2 + \ldots c_{m-1} x_{m-1}.$$

  Система из n линейных уравнений может быть представлена в виде матрицы $n \times m$, где каждая строка представляет уравнение, и каждый из m столбцов связан с коэффициентами конкретной переменной (или постоянной "переменной" 1 в случае c_0). Зачастую для каждого уравнения необходимо представлять значение y. Обычно это делается с использованием отдельного массива $n \times 1$ или вектора значений решений.
- *Графики и сети*. Графы состоят из вершин и ребер, где ребра определены как упорядоченные пары вершин, например (i, j). Граф с n вершинами и m ребрами может быть представлен в виде матрицы M размером $n \times n$, где $M[i, j]$ обозначает количество (или вес) ребер от вершины i до вершины j. Существуют удивительные связи между комбинаторными свойствами и линейной алгеброй, такие как связь между путями в графах и матричным умножением, а также то, как кластеры вершин соотносятся с собственными значениями/векторами соответствующих матриц.
- *Перестановка операций*. Матрицы могут *делать* нечто. Тщательно разработанные матрицы могут выполнять такие геометрические операции над наборами точек, как перемещение, вращение и масштабирование. Умножение матрицы данных на соответствующую *матрицу перестановок* (permutation matrix) приведет к переупорядочению ее строк и столбцов. Движения могут быть определены *векторами* (vector), матрицами $n \times 1$, достаточно мощными для кодирования таких операций, как перевод и перестановка.

Повсеместное распространение матриц означает, что для манипулирования ими была разработана обширная инфраструктура инструментов. В частности, наличие высокопроизводительных библиотек линейной алгебры для вашего любимого языка программирования означает, что вам никогда не придется реализовать какой-либо базовый алгоритм самостоятельно. Лучшие реализации библиотек оптимизируют также такие вещи, как точность вычислений, ошибки кеширования и использование нескольких ядер, вплоть до уровня ассемблера. Наша цель — сформулировать задачу с использованием линейной алгебры, а алгоритмы оставить этим библиотекам.

8.1.1. Интерпретация линейных алгебраических формул

Краткие формулы, написанные как произведения матриц, могут позволять делать удивительные вещи, включая линейную регрессию, сжатие матриц и геометрические преобразования. Алгебраическая замена в сочетании с богатым набором тождеств дает элегантные механические способы манипулирования такими формулами.

Однако мне очень трудно интерпретировать такие последовательности операций способами, которые я действительно понимаю. Например, возьмем "алгоритм", лежащий в основе линейной регрессии наименьших квадратов:

$$c = (A^T A)^{-1} A^T b,$$

где система $n \times m$ — это $Ax = b$, а w — это вектор коэффициентов наилучшей аппроксимирующей линии.

Одна из причин, по которой я нахожу линейную алгебру сложной, — это *номенклатура* (nomenclature). Есть много разных терминов и концепций, которые нужно понять, чтобы действительно понять то, что происходит. Но еще большая проблема заключается в том, что большинство доказательств по вполне веской причине являются алгебраическими. На мой взгляд, алгебраические доказательства обычно не переносят интуиции о том, почему все работает так, как оно работает. Алгебраические доказательства легче проверить пошагово механическим способом, чем за счет понимания идей, лежащих в основе аргумента.

В этом тексте я представлю только одно формальное доказательство. По замыслу и теорема, и доказательство неверны.

Теорема 1. $2 = 1$.
Доказательство.

$$\begin{aligned} a &= b, \\ a^2 &= ab, \\ a^2 - b^2 &= ab - b^2, \\ (a+b)(a-b) &= b(a-b), \\ a + b &= b, \\ 2b &= b, \\ 2 &= 1. \end{aligned}$$

Если вы никогда раньше не видели такого доказательства, вы можете найти его убедительным, хотя, я надеюсь, вы понимаете на концептуальном уровне, что $2 \neq 1$. Каждая строка следует из предыдущей в ходе прямой алгебраической замены. Проблема, как выясняется, возникает при сокращении $(a - b)$, поскольку на самом деле мы делим на нуль.

Каковы уроки этого доказательства? Доказательства — это идеи, а не просто алгебраические манипуляции. Отсутствие идеи означает отсутствие доказательств. Чтобы понять линейную алгебру, ваша цель должна состоять в том, чтобы сначала проверить простейший интересный случай (обычно два измерения), чтобы выработать интуицию, а затем попытаться представить, как она может обобщаться для более высоких измерений. Всегда есть особые случаи, за которыми нужно следить, например деление на нуль. В линейной алгебре эти случаи включают несоответствия размеров и сингулярные (т.е. необратимые) матрицы. Теория линейной алгебры работает, за исключением случаев, когда она не работает, и лучше думать с точки зрения общих случаев, а не патологических.

8.1.2. Геометрия и векторы

Существует удобная интерпретация "векторов", означающих матрицы $1 \times d$, как *векторов* (vectors) в геометрическом смысле, представляющие собой направленные лучи от начала координат через данную точку в d измерениях.

Нормализация каждого такого вектора v на единицу длины (за счет деления каждой координаты на расстояние от v до начала координат) помещает его в d-мерную сферу, как показано на рис. 8.1: круг для точек на плоскости, реальная сфера для $d = 3$ и некоторая невизуализируемая гиперсфера для $d \geq 4$.

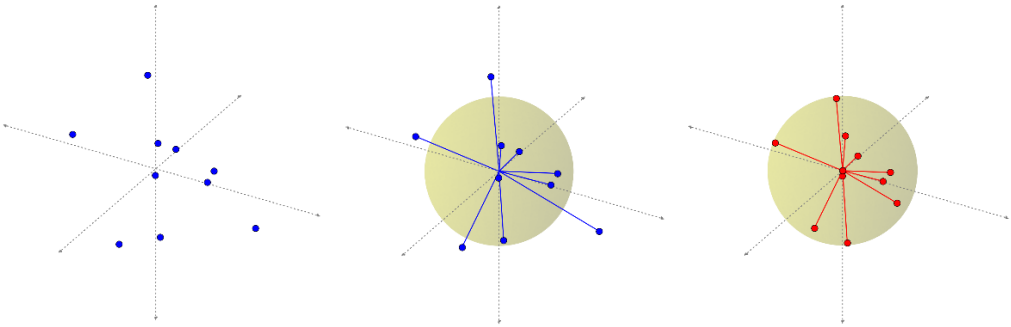


Рис. 8.1. Точки могут быть сведены до векторов на единичной сфере плюс их величины

Эта нормализация оказывается полезной вещью. Расстояния между точками становятся углами между векторами в целях сравнения. Две близлежащие точки будут иметь небольшой угол между ними относительно начала координат: малые расстояния означают небольшие углы. Игнорирование величин является формой масштабирования, делающей все точки непосредственно сопоставимыми.

Скалярное произведение (dot product) является весьма полезной операцией, сводящей векторы к скалярным величинам. Скалярное произведение двух векторов A и B длиной n определяется следующим образом:

$$A \cdot B = \sum_{i=1}^{n} A_i B_i$$

Мы можем использовать операцию скалярного произведения для вычисления угла $\theta = \angle AOB$ между векторами A и B, где O — это начало координат:

$$\cos(\Theta) = \frac{A \cdot B}{\|A\| \, \|B\|}$$

Попробуем разобрать эту формулу. Символ $\|V\|$ означает "длина вектора V". Для единичных векторов она по определению равна 1. В общем, это величина, на которую мы должны разделить V, чтобы сделать его единичным вектором.

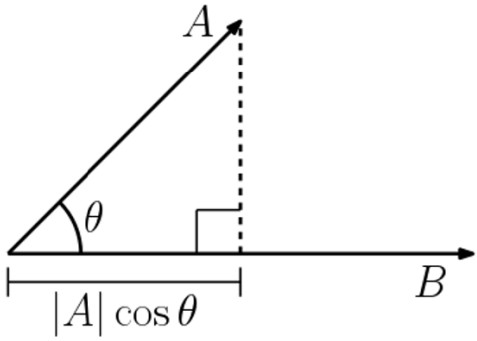


Рис. 8.2. Скалярное произведение двух векторов определяет косинус угла между ними

Но какова связь между скалярным произведением и углом? Рассмотрим простейший случай угла, определенного между двумя лучами: A, направленного вдоль нулевого градуса, и $B = (x, y)$. Таким образом, единичный луч равен $A = (1, 0)$. В этом случае скалярное произведение равно $1 \cdot x + 0 \cdot y = x$, что и должно быть $\cos(\theta)$, если B — это единичный вектор. Мы можем принять на веру, что это обобщение для B подходит и для более высоких измерений.

Таким образом, меньший угол означает более близкие точки на сфере. Но есть другая связь между вещами, которые мы знаем. Напомним особые случаи функции косинуса, приведенные здесь в радианах:

$$\cos(0) = 1, \ \cos(\pi/2) = 0, \ \cos(\pi) = -1 .$$

Значения функции косинуса находятся в диапазоне [–1, 1], точно в том же диапазоне, что и коэффициент корреляции. Кроме того, интерпретация

одинакова: два одинаковых вектора идеально коррелируют, а антиподальные точки коррелируют отрицательно. Ортогональные точки/векторы (случай $\Theta = \pi/2$) имеют меньше всего общего друг с другом.

Функция косинуса — это в точности корреляция двух переменных с нулевым средним. Для единичных векторов $||A|| = ||B|| = 1$, поэтому угол между A и B полностью определяется скалярным произведением.

> *На заметку*. Скалярное произведение двух векторов измеряет сходство точно так же, как коэффициент корреляции Пирсона.

8.2. Визуализация матричных операций

Я предполагаю, что вы уже имели опыт работы с основными матричными операциями, такими как транспонирование, умножение и инверсия. Этот раздел предназначен для напоминания, а не начального обучения.

Рис. 8.3. Пример матричного изображения: Линкольн (слева) и его мемориал (справа). Центральное изображение представляет собой линейную комбинацию того, что слева, и того, что справа, для $\alpha = 0{,}5$

Но чтобы выработать лучшую интуицию, я буду представлять матрицы в виде изображений, а не чисел, чтобы мы могли видеть происходящее, когда оперируем ими. На рис. 8.3 показаны наши основные матричные изображения: президент Авраам Линкольн (слева) и здание его мемориала (справа). Первое представляет собой человеческое лицо, а второе содержит ярко выраженные ряды и колонки.

Имейте в виду, что мы будем постепенно масштабировать матрицу между каждыми операциями, поэтому абсолютный цвет не имеет значения. Интересные шаблоны кроются в различиях между светлым и темным, т.е. самыми маленькими и самыми большими числами в текущей матрице. Обратите также внимание, что исходный элемент матрицы M [1, 1] представляет левый верхний угол изображения.

8.2.1. Сложение матриц

Сложение матриц (matrix addition) является простой операцией: для матриц A и B размером $n \times m$ каждая означает, что для $C = A + B$:

$$C_{ij} = A_{ij} + B_{ij}, \text{ для всех } 1 \le i \le n \text{ и } 1 \le j \le m.$$

Скалярное умножение (scalar multiplication) позволяет изменить вес каждого элемента в матрице одновременно, возможно, чтобы нормализовать их. Для любой матрицы A и числа c, $A' = c \cdot A$ означает, что

$$A'_{ij} = cA_{ij}, \text{ для всех } 1 \le i \le n \text{ и } 1 \le j \le m.$$

Сочетание сложения матриц со скалярным умножением позволяет выполнять *линейные комбинации* (linear combination) матриц. Формула $\alpha \cdot A + (1 - \alpha) \cdot B$ имеет плавное затухание между A (для $\alpha = 1$) и B (для $\alpha = 0$), как показано на рис. 8.3. Это и обеспечивает способ преобразования изображений из A в B.

Транспонирование (transpose) матрицы M меняет местами строки со столбцами, превращая матрицу $a \times b$ в матрицу $b \times a$, где M^T — это

$$M^T_{ij} = M_{ji}, \text{ для всех } 1 \le i \le n \text{ и } 1 \le j \le m.$$

В результате транспонирования квадратной матрицы получается квадратная матрица, поэтому M и M^T можно безопасно складывать и умножать. В более общем смысле: транспонирование — это операция, которая используется для изменения ориентации матрицы, чтобы ее *можно* было складывать или умножать на другую.

Транспонирование матрицы как бы "поворачивает" ее на 180 градусов[1], поэтому $(A^T)^T = A$. В случае квадратных матриц добавление матрицы к ее транспозиции (т.е. к ее транспонированной версии) симметрично, как показано на рис. 8.4 (справа). Причина ясна: $C = A + A^T$ подразумевает, что

$$C_{ij} = A_{ij} + A_{ji} = C_{ji}.$$

Рис. 8.4. Линкольн (слева) и его транспозиция (в центре). Сумма матрицы и ее транспозиции симметрична вдоль главной диагонали (справа)

[1] В данном случае автор говорит образно. На самом деле в результате транспонирования матрицы она зеркально поворачивается на 90 градусов против часовой стрелки, поскольку строки становятся столбцами, а столбцы — строками с теми же номерами.

8.2.2. Умножение матриц

Матричное умножение представляет собой совокупную версию *скалярного произведения* (dot product), или *внутреннего произведения* (inner product), векторов. Напомним, что для двух n-элементных векторов, X и Y, скалярное произведение $X \cdot Y$ определено как:

$$X \cdot Y = \sum_{i=1}^{n} X_i Y_i.$$

Скалярные произведения измеряют, насколько "синхронизированы" эти два вектора. Мы уже видели скалярное произведение при вычислении косинусного расстояния и коэффициента корреляции. Это операция, которая сводит пару векторов к одному числу.

Матричное произведение XY^T этих двух векторов создает матрицу 1×1, содержащую скалярное произведение $X \cdot Y$. Для матриц общего вида произведение $C = AB$ определяется как:

$$C_{ij} = \sum_{i=1}^{k} A_{ik} B_{kj}.$$

Чтобы это работало, A и B должны иметь одинаковые внутренние размеры, подразумевая, что если A равно $n \times k$, то B должно иметь размеры $k \times m$. Каждый элемент произведения матрицы C размером $n \times m$ является скалярным произведением i-й строки A с j-м столбцом B.

Наиболее важные свойства умножения матриц таковы.

- *Они не коммутативны*. *Коммутативность* (commutativity) — это напоминание о том, что порядок не имеет значения $xy = yx$. Хотя мы принимаем коммутативность как должное при умножении целых чисел, порядок *имеет* значение при умножении матриц. Для любой пары неквадратных матриц A и B, как минимум один из AB или BA имеет совместимые размеры. Но умножение даже квадратной матрицы не коммутативно, как показывают скаляры ниже:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \neq \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}$$

  и ковариационные матрицы на рис. 8.5.

- *Матричное умножение ассоциативно*. *Ассоциативность* (associativity) дает нам право применять скобки по своему усмотрению, выполняя операции в относительном порядке, который мы выбираем. При вычислении произведения ABC у нас есть выбор из двух вариантов: $(AB)C$ или $A(BC)$.

Более длинные цепочки матриц дают еще большую свободу, причем количество возможных скобок растет экспоненциально при увеличении длины цепочки. Все они будут возвращать один и тот же ответ, как показано здесь:

$$\left(\begin{bmatrix}1 & 2\\ 4 & 4\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix}\right)\begin{bmatrix}3 & 2\\ 1 & 0\end{bmatrix} = \begin{bmatrix}1 & 4\\ 3 & 8\end{bmatrix}\begin{bmatrix}3 & 2\\ 1 & 0\end{bmatrix} = \begin{bmatrix}7 & 2\\ 17 & 6\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\left(\begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix}\begin{bmatrix}3 & 2\\ 1 & 0\end{bmatrix}\right) = \begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\begin{bmatrix}3 & 2\\ 2 & 0\end{bmatrix} = \begin{bmatrix}7 & 2\\ 17 & 6\end{bmatrix}$$

Есть две основные причины, почему ассоциативность имеет для нас значение. В алгебраическом смысле это позволяет нам идентифицировать соседние пары матриц в цепочке и заменять их в соответствии с тождеством, если оно у нас есть. Но еще одной проблемой являются вычисления. Размер промежуточных матричных произведений легко может резко возрасти в середине. Предположим, мы стремимся вычислить $ABCD$, где A равно $1 \times n$, B и C равно $n \times n$, а D равно $n \times 1$. Произведение $(AB)(CD)$ состоит всего из $2n^2 + n$ операций, при условии обычного алгоритма умножения матриц со вложенными циклами. Напротив, $(A(BC))D$ потребует уже $n^3 + n^2 + n$ операций.

Рис. 8.5. Мемориал Линкольна M (слева) и его ковариационные матрицы. Большой блок в середине MM^T (в центре) является результатом сходства всех рядов от средней полосы M. Рисунок плотной сетки M^TM (справа) отражает регулярный шаблон колонн на здании мемориала

Алгоритм умножения матриц с вложенными циклами, которому вас учили в старших классах школы, тривиально прост в программировании, и он действительно рассматривается далее. Но не спешите программировать его. Гораздо более быстрые и численно устойчивые алгоритмы есть в высоко оптимизированных библиотеках линейной алгебры вашего любимого языка программирования. Формулировка алгоритмов в виде матричных произведений на основе больших массивов чисел вместо использования специальной логики

является нелогичным для большинства кибернетиков. Но эта стратегия может дать очень большие выигрыши на практике.

8.2.3. Применение матричного умножения

На первый взгляд, умножение матриц — довольно неуклюжая операция. Когда я впервые столкнулся с линейной алгеброй, я не мог понять, почему мы не могли просто умножить числа попарно, как сложение матрицы, и покончить с этим.

Причина, по которой мы занимаемся умножением матриц, заключается в том, что с ним можно многое сделать. Здесь мы рассмотрим эти случаи.

Ковариационные матрицы

Умножение матрицы A на ее транспозицию A^T является очень распространенной операцией. Почему? С одной стороны, мы *можем* их умножить: если A — это матрица $n \times d$, то A^T — это матрица $d \times n$. Таким образом, они всегда совместимы для умножения AA^T. Они одинаково совместимы для умножения и другим способом, т.е. A^TA.

Оба эти произведения имеют важные интерпретации. Предположим, что A — это матрица объектов $n \times d$, состоящая из n строк, представляющих элементы или точки, и d столбцов, представляющих наблюдаемые признаки этих элементов.

- $C = AA^T$ — это матрица размером $n \times n$ скалярных произведений, измеряющая "синхронность" среди точек. В частности, C_{ij} является мерой того, насколько элемент i похож на элемент j.
- $D = A^TA$ — это матрица размером $d \times d$ скалярных произведений, измеряющая "синхронность" среди столбцов или элементов. Здесь D_{ij} представляет сходство между признаком i и признаком j.

Эти матрицы достаточно часто используются, чтобы заработать собственное имя — *ковариационные матрицы* (covariance matrices). Этот термин нередко встречается в разговорах между специалистами по данным, так что вы должны знать о нем. Формула ковариации, которую мы дали при вычислении коэффициента корреляции, была такова

$$Cov(X,Y) = \sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y}).$$

Таким образом, наши две матрицы, строго говоря, являются ковариационными матрицами только в том случае, если строки или столбцы A имеют нулевое среднее значение. Но независимо от этого, величины матричного про-

изведения отражают степень, в которой значения конкретных пар строк или столбцов совпадают.

На рис. 8.5 представлены ковариационные матрицы мемориала Линкольна. Более темные пятна определяют строки и столбцы на изображении с наибольшим сходством. Постарайтесь понять, откуда берутся видимые структуры в этих ковариационных матрицах.

На рис. 8.5 (в центре) представлена ковариационная матрица строк MM^T. Большой темный прямоугольник в середине представляет большие скалярные произведения, полученные из любых двух рядов, проходящих через все белые колонны мемориала. Эти полосы света и тьмы жестко коррелируют, а интенсивно темные области способствуют образованию больших скалярных произведений. Светлые ряды, соответствующие небу, фронтону и лестнице, одинаково коррелированы и когерентны, но не имеют темных областей, чтобы сделать их скалярные произведения достаточно большими.

Правое изображение представляет ковариационную матрицу столбцов M^TM. Все пары столбцов матрицы сильно коррелируют друг с другом, как положительно, так и отрицательно, но столбцы матрицы через белые колонны здания имеют малый вес, а следовательно, небольшое скалярное произведение. Вместе они формируют шахматную доску из чередующихся темных и светлых полос.

Матричное умножение и пути

Квадратные матрицы могут быть умножены сами на себя без транспонирования. Действительно, $A^2 = A \times A$ называется *квадратом* матрицы A. В более общем смысле A^k называют k-й степенью матрицы.

Степени матрицы A имеют вполне естественную интерпретацию, когда A представляет *матрицу смежности* (adjacency matrix) графа или сети. Матрица смежности $A[i, j] = 1$, когда (i, j) является ребром в сети. В противном случае, когда i и j не являются прямыми соседями, $A[i, j] = 0$.

Для таких матриц 0/1 произведение A^2 дает число путей длиной два в A. В частности:

$$A^2[i, j] = \sum_{k=1}^{n} A[i,k]\, A[k, j].$$

Существует только один путь длиной два от i до j для каждой промежуточной вершины k, так как (i, k) и (k, j) оба являются ребрами в графе. Сумма этих путей рассчитывается по скалярному произведению выше.

Но вычислительные возможности матриц имеют смысл даже для более общих матриц. Они моделируют эффекты диффузии, распределяя вес каждого элемента среди связанных элементов. Такие вещи происходят в знаменитом алгоритме Google PageRank и других итерационных процессах, таких как распространение инфекции.

Умножение матриц и перестановки

Матричное умножение зачастую используется только для того, чтобы изменить порядок элементов в конкретной матрице. Напомним, что высокопроизводительные процедуры умножения матриц являются ослепительно быстрыми, достаточно того, что зачастую они способны выполнять такие операции быстрее, чем логика специального программирования. Они также обеспечивают способ описания таких операций в терминах алгебраических формул, сохраняя тем самым компактность и удобочитаемость.

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad M = \begin{pmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{pmatrix} \quad PM = \begin{pmatrix} 31 & 32 & 33 & 34 \\ 11 & 12 & 13 & 14 \\ 41 & 42 & 43 & 44 \\ 21 & 22 & 23 & 24 \end{pmatrix}$$

Рис. 8.6. Умножение матрицы на матрицу перестановок переставляет ее строки и столбцы

Самая известная матрица перестановок вообще ничего не делает. *Единичная матрица* (identity matrix) — это матрица $n \times n$, состоящая из всех нулей, кроме тех, которые расположены вдоль главной диагонали. Для $n = 4$

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Убедитесь, что $AI = IA = A$, т.е. что умножение на единичную матрицу коммутативно.

Обратите внимание: в каждой строке и столбце I содержится ровно один ненулевой элемент. Матрицы с этим свойством называются *матрицами перестановок* (permutation matrix), поскольку ненулевой элемент в позиции (i, j) можно интерпретировать как то, что элемент i находится в позиции j перестановки. Например, перестановка (2, 4, 3, 1) определяет следующую матрицу перестановок:

$$P_{(2431)} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Обратите внимание, что единичная матрица соответствует перестановке (1, 2, …, n).

Ключевым моментом здесь является то, что мы можем умножить A на соответствующую матрицу перестановок, чтобы переставить строки и столбцы, как нам угодно. На рис. 8.7 показано, что происходит, когда мы умножаем наше изображение на "обратную" матрицу перестановок r, где единицы лежат вдоль малой диагонали. Поскольку матричное умножение обычно не коммутативно, мы получаем разные результаты для Ar и rA. Убедитесь сами почему.

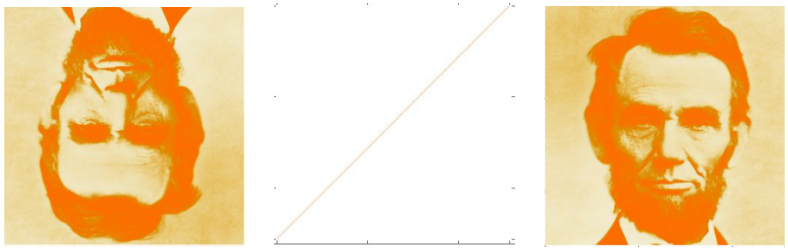


Рис. 8.7. Умножение матрицы Линкольна M на матрицу обратной перестановки r (в центре). Произведение rM переворачивает Линкольна вверх ногами (слева), в то время как Mr переносит пробор волос на другую сторону головы (справа)

Вращение точек в пространстве

Умножение чего-либо на правильную матрицу может иметь магические свойства. Мы видели, как набор из n точек на плоскости (т.е. в двух измерениях) может быть представлен $(n \times 2)$-мерной матрицей S. Умножение таких точек на правильную матрицу может привести к естественным геометрическим преобразованиям.

Матрица вращения (rotation matrix) R_θ осуществляет преобразование точек, поворачивая их вокруг начала координат на угол θ. В двух измерениях R_θ определяется как

$$R_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

В частности, после соответствующего умножения/поворота точка (x, y) переходит в

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R_\theta \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos(\theta) - y\sin(\theta) \\ x\sin(\theta) + y\cos(\theta) \end{bmatrix}$$

Для $\theta = 180° = \pi$ радиан, $\cos(\theta) = -1$ а $\sin(\theta) = 0$, так что это сводится к $(-x, -y)$, делая правильную вещь и помещая точку в противоположный квадрант.

Для нашей $(n \times 2)$-мерной скалярной матрицы S мы можем использовать функцию транспонирования для правильной ориентации матрицы. Попытайтесь доказать, что

$$S' = (R_\theta S^T)^T$$

делает именно то, что нам нужно.

Естественные обобщения R_θ существуют для поворота точек в произвольных измерениях. Кроме того, произвольные последовательности преобразований могут быть реализованы в ходе умножения цепочек матриц вращения, расширения и отражения, что дает компактное описание сложных манипуляций.

8.2.4. Единичные матрицы и инверсия

Единичные операции играют большую роль в алгебраических структурах. Для числового сложения единичным элементом является нуль, так $0 + x = x + 0 = x$. Такую же роль играет единица при умножении, так $1 \cdot x = x \cdot 1 = x$.

Единичный элемент при умножении матриц — это единичная матрица со всеми единицами по главной диагонали. Умножение на единичную матрицу коммутативно, поэтому $IA = AI = A$.

Операция *инверсии* (inverse) сводится к приведению элемента x к его единичному элементу. Для численного сложения обратная величина x равна $(-x)$, поскольку $x + (-x) = 0$. Инверсной операцией умножения является *деление*. Мы можем инвертировать число, умножив его на обратное, таким образом $x(1/x) = 1$.

Люди обычно не говорят о делении матриц. Однако они очень часто инвертируют их. Мы говорим, что A^{-1} — это *мультипликативная инверсия* (multiplicative inverse) матрицы A, если $AA^{-1} = I$, где I — это единичная матрица. Инверсия является важным частным случаем деления, поскольку $AA^{-1} = I$ подразумевает $A^{-1} = I/A$. Это на самом деле эквивалентные операции, поскольку $A/B = AB^{-1}$.

На рис. 8.8 (слева) демонстрируется инверсия изображения Линкольна, которая очень похожа на случайный шум. Но ее умножение на изображение дает тонкую главную диагональ единичной матрицы, хотя и наложенную на фон числовой ошибки. Вычисления с плавающей точкой по своей сути неточны, а такие алгоритмы, как инверсия, которые подразумевают многократные сложения и умножения, нередко накапливают ошибки в процессе.

Как мы можем вычислить обратную матрицу? Существует замкнутая форма для нахождения инверсии A^{-1} матрицы A размером 2×2, а именно:

$$A^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

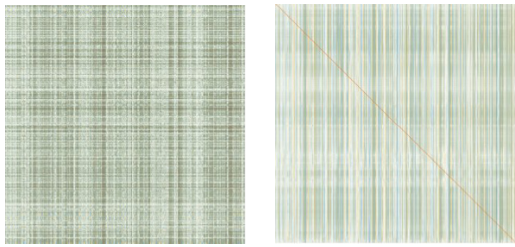


Рис. 8.8. Инверсия Линкольна не очень похожа на человека (слева), но MM^{-1} создает единичную матрицу по модулю небольших ненулевых членов из-за проблем с числовой точностью

В более общем смысле, существует подход к инвертированию матриц в ходе решения линейной системы с использованием исключения по Гауссу.

Заметим, что эта замкнутая форма для инверсии делится на нуль всякий раз, когда произведения диагоналей равны, т.е. $ad = bc$. Это говорит нам о том, что такие матрицы не являются обратимыми, или *сингулярными* (singular), что означает отсутствие инверсии. Так же, как мы не можем делить числа на нуль, мы не можем инвертировать сингулярные матрицы.

Матрицы, которые мы можем инвертировать, называются *несингулярными* (non-singular), и жизнь становится лучше, когда наши матрицы обладают этим свойством. Проверка матрицы на сингулярность заключается в выяснении, является ли ее *определитель* (determinant) ненулевым. Для матриц 2×2 определителем является разность между произведением ее диагоналей, точнее, знаменатель в формуле инверсии.

Кроме того, определитель определен только для квадратных матриц, поэтому только квадратные матрицы инвертируемы. Стоимость вычисления этого детерминанта составляет $O(n^3)$, поэтому он дорого обходится в больших матрицах, и действительно настолько же дорог, как попытка инвертировать саму матрицу с использованием исключения по Гауссу.

8.2.5. Инверсия матриц и линейные системы

Линейные уравнения определяются суммой переменных, взвешенных по постоянным коэффициентам:

$$y = c_0 + c_1 x_1 + c_2 x_2 + \ldots c_{m-1} x_{m-1}.$$

Таким образом, коэффициенты, определяющие систему из n линейных уравнений, могут быть представлены в виде матрицы C размером $n \times m$. Здесь каждая строка представляет уравнение, а каждый из m столбцов — коэффициенты отдельной переменной.

Мы можем аккуратно оценить все n этих уравнений на конкретном входном векторе X размером $m \times 1$, умножив CX. Результатом будет вектор $n \times 1$, сообщающий значение $f_i(X)$ для каждого из n линейных уравнений, $1 \le i \le n$. Частным случаем здесь является аддитивный член c_0. Для правильной интерпретации соответствующий столбец в векторе X должен содержать все единицы.

Если мы обобщим X как матрицу $m \times p$, содержащую p различных точек, наше произведение Cx приведет к матрице $n \times p$, оценивающей каждую точку по каждому уравнению в умножении одной матрицы.

Но основной операцией в системах из n уравнений является их решение, т.е. определение вектора X, необходимого для получения целевого значения Y для каждого уравнения. Предположим, дан вектор $n \times 1$ значений решений Y и матрица коэффициентов C, мы ищем X, так что $CX = Y$. Инверсия матриц может использоваться для решения линейных систем. Умножение обеих сторон $CX = Y$ на инверсное значение C дает:

$$(C^{-1}C)X = C^{-1}Y \rightarrow X = C^{-1}Y.$$

Таким образом, система уравнений может быть решена за счет инверсии C с последующим умножением C^{-1} на Y.

Исключение по Гауссу (Gaussian elimination) — это еще один подход к решению линейных систем, который, я надеюсь, вы видели раньше. Напомним, что он решает уравнения, выполняя операции сложения/вычитания строк, чтобы упростить матрицу C уравнений до тех пор, пока она не сведется к единичной матрице. Это упрощает считывание значений переменных, поскольку каждое уравнение приведено к виду $X_i = Y'_i$, где Y' — это результат применения этих же операций со строками к исходному целевому вектору Y.

$$[A \mid I] = \left[\begin{array}{ccc|ccc} 6 & 4 & 1 & 1 & 0 & 0 \\ 10 & 7 & 2 & 0 & 1 & 0 \\ 5 & 3 & 1 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & 0 & 1 & -2 \\ 5 & 3 & 1 & 0 & 0 & 1 \end{array}\right] =$$

$$= \left[\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & -1 & 1 \\ 0 & 1 & 0 & 0 & 1 & -2 \\ 5 & 3 & 1 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & -1 & 1 \\ 0 & 1 & 0 & 0 & 1 & -2 \\ 0 & 0 & 1 & -5 & 2 & 2 \end{array}\right]$$

$$\rightarrow A^{-1} = \left[\begin{array}{ccc} 1 & -1 & 1 \\ 0 & 1 & -2 \\ -5 & 2 & 2 \end{array}\right]$$

Рис. 8.9. Инверсия матриц может быть вычислена с использованием исключения по Гауссу

Вычисление инверсной матрицы может быть выполнено таким же образом, как показано на рис. 8.9. Мы выполняем операции со строками, чтобы упростить матрицу коэффициентов до единичной матрицы *I*, чтобы создать инверсную. Я рассматриваю это как алгоритм Дориана Грея: матрица коэффициентов *C* украшает единичную матрицу, а цель *I* стареет в обратном направлении.[2]

Поэтому мы можем использовать инверсию матриц для решения линейных систем и решатели линейных систем — для инвертирования матриц. Таким образом, две проблемы в некотором смысле эквивалентны. Вычисление инверсии удешевляет оценку нескольких векторов *Y* для данной системы *C*, сводя ее к умножению на одну матрицу. Но это может быть сделано еще более эффективно с использованием *разложения LU матрицы* (LU-decomposition), обсуждаемого в разделе 8.3.2. Исключение по Гауссу оказывается более численно устойчивым, чем инверсия и, как правило, является предпочтительным методом при решении линейных систем.

8.2.6. Ранг матриц

Система уравнений правильно *определена* (determined), когда существует *n* линейно независимых уравнений и *n* неизвестных. Например, линейная система

$$2x_1 + 1x_2 = 5,$$
$$3x_1 - 2x_2 = 4.$$

определена правильно. Единственным решением является точка ($x_1 = 2, x_2 = 1$).

Напротив, системы уравнений *недоопределены* (underdetermined), если есть строки (уравнения), которые могут быть выражены как линейные комбинации других строк. Линейная система

$$2x_1 + 1x_2 = 5,$$
$$4x_1 + 2x_2 = 10.$$

недоопределена, поскольку второй ряд в два раза больше первого. Должно быть понятно, что для решения неопределенной системы линейных уравнений недостаточно информации.

Ранг (rank) матрицы измеряет количество линейно независимых строк. Матрица $n \times n$ должна иметь ранг *n* для всех операций, которые должны быть на ней правильно определены.

Ранг матрицы можно вычислить, применив метод исключения по Гауссу. Если она недоопределена, то некоторые переменные исчезнут в ходе операций сокращения строк. Существует также связь между недоопределенными

[2] В романе Оскара Уайльда “Портрет Дориана Грея” главный герой остается красивым, в то время как его портрет со временем стареет.

системами и сингулярными матрицами: напомним, что они были идентифицированы по нулевому детерминанту. Вот почему разница в перекрестном произведении здесь $(2 \cdot 2 - 4 \cdot 1)$ равна нулю.

Матрицы характеристик зачастую имеют более низкий ранг, чем мы могли бы пожелать. Файлы примеров, как правило, содержат повторяющиеся записи, что приводит к идентичности двух строк матрицы. Также вполне возможно, что несколько столбцов эквивалентны: представьте, что каждая запись содержит рост, измеренный в футах и метрах, например.

Такие вещи, безусловно, случаются, и плохо, когда это происходит. На нашем изображении мемориала Линкольна некоторые алгоритмы потерпели неудачу численно. Оказалось, что наше изображение 512×512 имело ранг только 508, поэтому не все строки были линейно независимыми. Чтобы сделать его полноценной матрицей, вы можете добавить небольшое количество случайного шума к каждому элементу, что повысит ранг без серьезных искажений изображения. Эта нелепость может позволить вашим данным проходить через алгоритм без предупреждающего сообщения, но это указывает на количество проблем, которые могут возникнуть.

Линейные системы могут быть "почти" низкого ранга, что приводит к большей опасности потери точности из-за числовых проблем. Это формально фиксируется матричным инвариантом, *коэффициентом перекоса* (condition number), который в случае линейной системы измеряет, насколько чувствительно значение X к небольшим изменениям Y в $Y = AX$.

Помните о капризах числовых расчетов при оценке ваших результатов. Например, рекомендуется вычислять AX для любого предполагаемого решения X и видеть, насколько AX *действительно* хорош по сравнению с Y. Теоретически разница будет равна нулю, но на практике вы можете быть удивлены тем, насколько грубыми являются вычисления.

8.3. Разложение матриц

Разложение (factoring) матрицы A в матрицы B и C представляет собой особый аспект деления. Мы видели, что любая несингулярная матрица M имеет инверсию M^{-1}, поэтому единичная матрица I может быть разложена как $I = MM^{-1}$. Это доказывает, что некоторые матрицы (например, I) могут быть разложены, а кроме того, они могут иметь много различных разложений. В этом случае каждая возможная несингулярная M определяет разные разложения.

Разложение матриц является важной абстракцией в науке о данных, приводящей к кратким представлениям функций и идеям, таким как тематическое моделирование. Она играет важную роль в решении линейных систем за счет специальных разложений, таких как разложение LU матрицы.

К сожалению, найти такие разложения проблематично. Разложение *целых чисел* — это сложная проблема, хотя эта сложность исчезает, когда вам доступны числа с плавающей запятой. Разложение матриц оказывается сложнее: для конкретной матрицы точное разложение может быть невозможно, особенно если мы ищем разложение $M = XY$, где X и Y имеют заданные размеры.

8.3.1. Разложение матрицы признаков

Многие важные алгоритмы машинного обучения можно рассматривать с точки зрения разложения матрицы. Предположим, нам дана матрица признаков A размером $n \times m$, где в соответствии с обычным соглашением строки представляют элементы/примеры, а столбцы представляют признаки примеров.

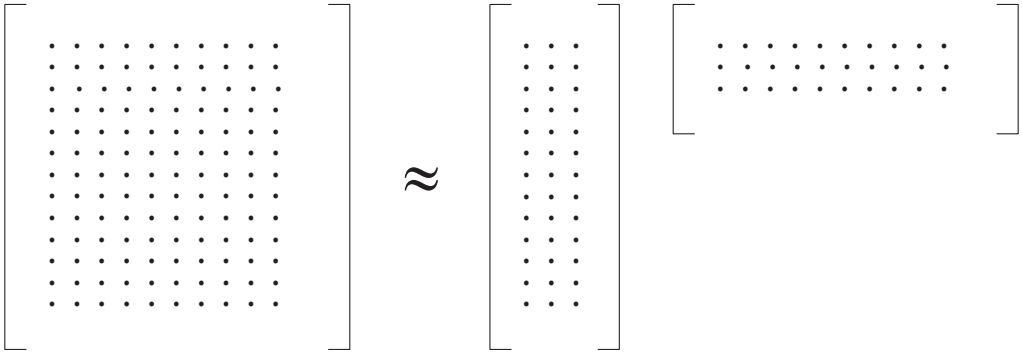


Рис. 8.10. Разложение матрицы признаков $A \approx BC$ дает B как более краткое представление элементов и C как более сжатое представление признаков A^T

Теперь предположим, что мы можем *разложить* (factor) матрицу A, т.е. выразить ее как произведение $A \approx BC$, где B — это матрица $n \times k$, а C — матрица $k \times m$. С учетом, что $k < \min(n, m)$, как показано на рис. 8.10, это хорошо по нескольким причинам.

- *Совместно B и C обеспечивают сжатое представление матрицы A*. Матрицы признаков, как правило, большие и неудобные для работы. Разложение обеспечивает способ кодирования всей информации большой матрицы в две меньшие матрицы, которые вместе будут меньше исходной.
- *B служит меньшей матрицей признаков для элементов, заменяя A*. Разложение матрицы B имеет n строк, точно так же, как исходная матрица A. Но у нее существенно меньше столбцов, поскольку $k < m$. Это означает, что "большая часть" информации в A теперь закодирована в B. Меньшее количество столбцов означает меньшую матрицу и меньшее количество параметров, подходящих для любой модели, построенной с использованием этих новых признаков. Эти более абстрактные признаки могут

также представлять интерес для других приложений, поскольку являются краткими описаниями строк набора данных.

- *C^T служит меньшей матрицей признаков, заменяя A^T.* Транспонирование матрицы признаков превращает столбцы/признаки в строки/элементы. Разложенная матрица C^T имеет m строк и k столбцов признаков, представляющих их. Во многих случаях m оригинальных "признаков" стоит моделировать сами по себе.

Рассмотрим репрезентативный пример из анализа текста. Возможно, мы хотим представить n документов, каждый твит или другое сообщение в социальной сети с точки зрения используемого им словаря. Каждый из m наших признаков будет соответствовать отдельному словарному слову, и $A[i, j]$ будет записывать, как часто словарное слово w_j (скажем, *cat* (кот)) появлялось в сообщении номер i. Рабочий словарный запас в английском языке довольно большой с длинным хвостом, поэтому, вероятно, мы можем ограничить его $m = 50\ 000$ наиболее часто используемых слов. Большинство сообщений будут короткими, не более нескольких сотен слов. Таким образом, наша матрица признаков A будет очень разреженной, изобилующей огромным количеством нулей.

Теперь предположим, что мы можем разложить $A = BC$, где внутреннее измерение k относительно мало. Скажем, $k = 100$. Теперь каждое сообщение будет представлено строкой B, содержащей только сто чисел вместо полных 50 000. Это значительно упрощает сравнение текстов на предмет сходства. Эти k измерений можно рассматривать как аналог "тем" в документах, поэтому все сообщения о спорте должны использовать другой набор тем, нежели те, что касаются отношений.

Теперь матрица C^T может рассматриваться как содержащая вектор признаков для каждого из словарных слов. Это интересно. Мы ожидаем, что слова, которые применяются в одинаковых контекстах, будут иметь похожие тематические векторы. Названия цветов, такие как *yellow* (желтый) и *red* (красный), скорее всего, будут выглядеть примерно одинаково в пространстве тем, в то время как *baseball* (бейсбол) и *sex* (секс) должны иметь довольно отдаленные отношения.

Обратите внимание, что эта матрица "слово-тема" потенциально полезна в любой задаче, стремящейся использовать язык в качестве признака. Связь с сообщениями социальных сетей в значительной степени исчезла, поэтому она будет применима к другим областям, таким как книги и новости. Действительно, такие сжатые *вектора представления слов* (word embedding) являются очень мощным инструментом в *обработке естественного языка* (Natural Language Processing — NLP), как будет обсуждаться в разделе 11.6.3.

8.3.2. Разложение LU матрицы и детерминанты

Разложение LU матрицы (LU decomposition) — это конкретное матричное разложение, которое делит квадратную матрицу A на нижнюю и верхнюю треугольные матрицы L и U, так что $A = LU$.

Матрица является *треугольной* (triangular), если она содержит все нулевые слагаемые либо выше, либо ниже основной диагонали. Нижняя треугольная матрица L имеет все ненулевые члены ниже главной диагонали. Другим разложением, U, является верхняя треугольная матрица. Поскольку главная диагональ L состоит из всех единиц, мы можем упаковать все разложение в то же пространство, что и исходная матрица $n \times n$.

Основное значение разложения LU матрицы заключается в том, что оно оказывается полезным при решении линейных систем $AX = Y$, особенно при решении нескольких задач с одним и тем же A, но разными Y. Матрица L — это результат очистки всех значений выше главной диагонали с помощью исключения по Гауссу. Оказавшись в этой треугольной форме, остальные уравнения могут быть упрощены непосредственно. Матрица U отражает, какие строковые операции произошли в ходе построения L. Упрощение U и применение L к Y требует меньше работы, чем решение A с нуля.

Другое значение разложения LU матрицы заключается в получении алгоритма для вычисления детерминанта матрицы. *Детерминант* (determinant) A является произведением основных диагональных элементов U. Как мы видели, нулевой детерминант означает, что матрица не имеет полного ранга.

На рис. 8.11 иллюстрируется разложение LU матрицы мемориала Линкольна. Существует четкая текстура, видимая для двух треугольных матриц. Эта конкретная функция разложения LU матрицы (в системе Mathematica) использовала тот факт, что уравнения в системе можно переставлять без потери информации. То же самое не относится к изображениям, но мы видим точные реконструкции белых колонн мемориала, хотя и не на своем месте.

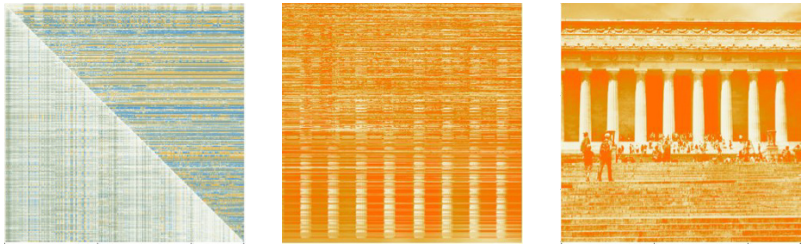


Рис. 8.11. Разложение LU матрицы мемориала Линкольна (слева) с произведением $L \cdot U$ (в центре). Строки матрицы LU были переставлены во время расчета, но при правильном порядке полностью реконструировали изображение (справа)

8.4. Собственные значения и собственные векторы

Умножение вектора U на квадратную матрицу A может иметь тот же эффект, что и умножение его на скаляр l. Рассмотрим следующую пару примеров. Действительно, проверьте их вручную:

$$\begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \end{bmatrix} = -6 \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = -1 \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Оба эти равенства содержат произведения с одинаковым вектором U размером 2×1 как слева, так и справа. С одной стороны U умножается на матрицу A, а с другой — на скаляр λ. В подобных случаях, когда $AU = \lambda U$, мы говорим, что λ — это *собственное значение* (eigenvalue) матрицы A, а U — это связанный с ней *собственный вектор* (eigenvector).

Такие пары собственных векторов и собственных значений являются любопытной вещью. То, что скаляр λ может сделать то же самое с U, что и вся матрица A, говорит нам, что они должны быть специальными. Собственный вектор U и собственное значение λ должны кодировать много информации об A.

Кроме того, обычно существует несколько таких пар собственных векторов для любых матриц. Обратите внимание, что второй пример выше работает с той же матрицей A, но дает иное, чем U и λ.

8.4.1. Свойства собственных значений

Теория собственных значений ведет нас куда глубже в дебри линейной алгебры, чем я готов идти в этой книге. Однако мы можем резюмировать свойства, которые являются важными для нас.

- С каждым собственным значением связан собственный вектор. Они всегда ходят парами.
- В общем случае, для каждой матрицы $n \times n$ полного ранга существует n пар собственных векторов.
- Каждая пара собственных векторов симметричной матрицы взаимно *ортогональна* (orthogonal), так же, как ортогональны оси X и Y на плоскости. Два вектора ортогональны, если их скалярное произведение равно нулю. Заметьте, что $(0, 1) \cdot (1, 0) = 0$, как и $(2, -1) \cdot (1, 2) = 0$ из предыдущего примера.

- В результате этого собственные векторы могут играть роль измерений или *баз* (base) в некотором *n*-мерном пространстве. Это открывает много геометрических интерпретаций матриц. В частности, любая матрица может быть закодирована так, что каждое собственное значение представляет величину своего связанного собственного вектора.

8.4.2. Вычисление собственных значений

Можно найти *n* различных собственных значений матрицы ранга *n*, разложив ее *характеристическое уравнение* (characteristic equation). Начнем с определения равенства $AU = \lambda U$. Убедитесь, что оно остается неизменным, когда мы умножаем на единичную матрицу I, поэтому

$$AU = \lambda IU \rightarrow (A - \lambda I)U = 0.$$

Для нашего примера матрицы мы получаем

$$A - \lambda I = \begin{bmatrix} -5-\lambda & 2 \\ 2 & -2-\lambda \end{bmatrix}$$

Обратите внимание, что наше равенство $(A - \lambda I)U = 0$ остается верным, если мы умножим вектор U на любое скалярное значение c. Это подразумевает, что существует бесконечное количество решений, а следовательно, линейная система должна быть недоопределена.

В такой ситуации детерминант матрицы должен быть равен нулю. С матрицей 2×2 детерминант является просто перекрестным произведением $ad - bc$, поэтому

$$(-5-\lambda)(-2-\lambda) - 2 \cdot 2 = \lambda^2 + 7\lambda + 6 = 0.$$

Решение для λ с квадратичной формулой дает $\lambda = -1$ и $\lambda = -6$. В более общем смысле детерминант $|A - \lambda I|$ является многочленом степени n, а следовательно, корни этого характеристического уравнения определяют собственные значения A.

Вектор, связанный с любым заданным собственным значением, может быть вычислен в результате решения линейной системы. Согласно нашему примеру, мы знаем, что

$$\begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

для любого собственного значения λ и ассоциированного собственного вектора $U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$. После того как мы зафиксировали значение λ, у нас будет система из

n уравнений и n неизвестных, и, таким образом, мы можем решить его для значений U. Для $\lambda = -1$

$$-5u_1 + 2u_2 = -1u_1 \rightarrow -4u_1 + 2u_2 = 0,$$
$$2u_1 + -2u_2 = -1u_2 \rightarrow 2u_1 + -1u_2 = 0.$$

это имеет решение $u_1 = 1$, $u_2 = 2$, что дает соответствующий собственный вектор.

Для $\lambda = -6$ получаем

$$-5u_1 + 2u_2 = -6u_1 \rightarrow 1u_1 + 2u_2 = 0,$$
$$2u_1 + -2u_2 = -6u_2 \rightarrow 2u_1 + 4u_2 = 0.$$

Эта система недоопределена, поэтому $u_1 = 2$, $u_2 = -1$ и любая постоянная, кратная этому, квалифицируется как собственный вектор. Это имеет смысл: поскольку U находится с обеих сторон равенства $AU = \lambda U$, для любой константы c вектор $U' = c\,U$ одинаково удовлетворяет определению.

Более быстрые алгоритмы для вычисления собственного значения/вектора основаны на таком подходе разложения матрицы, как *разложение QR* (QR decomposition). Другие алгоритмы пытаются избежать решения полной линейной системы. Например, альтернативный подход многократно использует $U' = (AU)/\lambda$, чтобы вычислять лучшие и лучшие приближения к U, пока он не сходится. Когда условия правильные, это может быть намного быстрее, чем решение полной линейной системы.

Наибольшие собственные значения и связанные с ними векторы, по правде говоря, более важны, чем остальные. Почему? Потому что они вносят больший вклад в аппроксимацию матрицы A. Таким образом, высокопроизводительные системы линейной алгебры используют специальные процедуры для нахождения k наибольших (и наименьших) собственных значений, а затем итерационные методы для восстановления векторов для каждого.

8.5. Разложение по собственным значениям

Любая симметричная матрица M размером $n \times n$ может быть разложена на сумму из n своих собственных векторных произведений. Представим n собственных пар (λ_i, U_i) для $1 \le i \le n$. По договоренности сортировка осуществляется по размеру, поэтому $\lambda_i \ge \lambda_{i-i}$ для всех i.

Поскольку каждый собственный вектор U_i является матрицей $n \times 1$, его умножение на транспозицию дает матричное произведение $U_i U_i^T$ размером $n \times n$. Оно имеет те же размеры, что и исходная матрица M. Мы можем вычислить линейную комбинацию этих матриц, взвешенную по соответствующему

собственному значению. Фактически это восстанавливает исходную матрицу, поскольку:

$$M = \sum_{i=1}^{n} \lambda_i U_i U_i^T .$$

Этот результат сохраняется только для симметричных матриц, поэтому мы не можем использовать его для кодирования нашего изображения. Но ковариационные матрицы всегда симметричны, и они кодируют основные характеристики каждой строки и столбца матрицы.

Таким образом, ковариационная матрица может быть представлена ее разложением по собственным значениям. Это занимает немного больше места, чем исходная матрица: n собственных векторов длиной n, плюс n собственных значений против $n(n + 1)/2$ элементов в верхнем треугольнике симметричной матрицы плюс главная диагональ.

Однако, используя только векторы, связанные с наибольшими собственными значениями, мы получаем хорошее приближение матрицы. Меньшие размеры вносят очень малый вклад в значения матрицы, поэтому их можно исключить с небольшой ошибкой. Этот метод уменьшения размеров очень полезен для создания меньших, более эффективных наборов объектов.

На рис. 8.12 (слева) показана реконструкция ковариационной матрицы M мемориала Линкольна по ее единственному наибольшему собственному вектору, т.е. $U_1 U_1^T$, вместе с ассоциированной с ней матрицей ошибок $M - U_1 U_1^T$. Даже один собственный вектор делает весьма существенную работу при реконструкции, восстанавливая такие функции, как большой центральный блок.

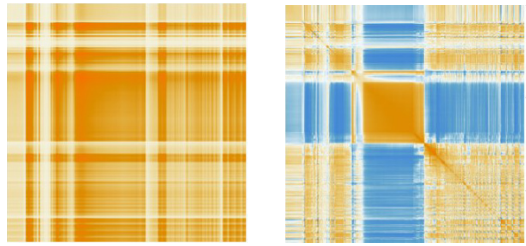


Рис. 8.12. Наибольшего собственного вектора мемориала Линкольна достаточно, чтобы захватить большую часть деталей его ковариационной матрицы

График на рис. 8.12 (справа) показывает, что ошибки возникают в неоднородных областях, поскольку для кодирования более тонких деталей требуются дополнительные векторы. На рис. 8.13 показан график ошибок при использовании одного, пяти и пятидесяти самых больших собственных векторов. Области ошибок становятся меньше, когда мы восстанавливаем более мелкие

детали, а величина ошибок уменьшается. Поймите, что даже 50 собственных векторов составляют менее 10% от 512, необходимых для восстановления идеальной матрицы, но этого достаточно для очень хорошего приближения.

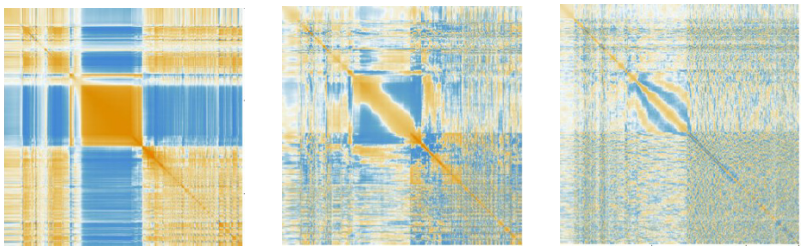


Рис. 8.13. Ошибка в реконструкции мемориала Линкольна по одному, пяти и пятидесяти самым большим собственным векторам

8.5.1. Разложение по сингулярному значению

Разложение по собственным значениям — это очень хорошая вещь. Но она работает только для симметричных матриц. *Разложение по сингулярным значениям* (singular value decomposition) является более общим подходом разложения матрицы, который аналогичным образом сводит матрицу к сумме других матриц, определенных векторами.

Разложение по сингулярным значениям действительной матрицы M размером $n \times m$ делит ее на три матрицы U, D и V с размерами $n \times n$, $n \times m$ и $m \times m$ соответственно. Это разложение имеет вид[3]

$$M = UDV^T.$$

Центральная матрица D обладает свойством *диагональной* матрицы, т.е. все ненулевые значения лежат на главной диагонали, как единичная матрица I.

Не беспокойтесь о том, как мы находим это разложение. Вместо этого давайте сосредоточимся на том, что это значит. Произведение UD имеет эффект умножения $U[i, j]$ на $D[j, j]$, поскольку все члены D равны нулю, кроме главной диагонали. Таким образом, D можно интерпретировать как меру относительной важности каждого столбца U или, через DV^T, важности каждой строки V^T. Эти весовые значения D являются *сингулярными значениями* (singular value) M.

Пусть X и Y — это векторы размерами $n \times 1$ и $1 \times m$ соответственно. *Внешним произведением* (outer product) $P = X \otimes Y$ матрицы является матрица $n \times m$, где $P[j, k] = X[j]Y[k]$. Традиционное матричное умножение $C = AB$ может быть выражено как сумма этих внешних произведений, а именно:

[3] Если M содержит выпуклые числа, то это обобщается до $M = UDV^*$, где V^* означает *сопряженное транспонирование* (conjugate transpose) V.

$$C = AB = \sum_k A_k \otimes B_k^T .$$

где A_k — это вектор, определенный k-м столбцом A, а B_k^T — это вектор, определенный k-й строкой B.

Таким образом, матрица M может быть выражена как сумма внешних произведений векторов, возникающих в результате разложения по сингулярным числам, а именно $(UD)_k$ и $(V^T)_k$ для $1 \le k \le m$. Кроме того, сингулярные значения D определяют, какой вклад каждое внешнее произведение вносит в M, поэтому достаточно взять только векторы, связанные с наибольшим сингулярным значением, чтобы получить приближение к M.

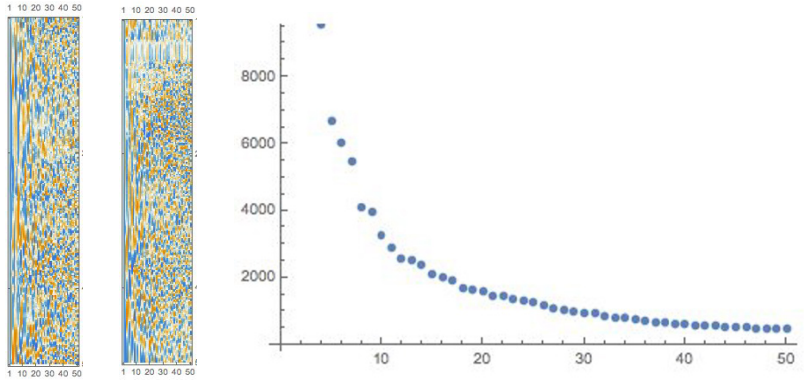


Рис. 8.14. Матрицы сингулярных значений в разложении Линкольна для 50 сингулярных значений

На рис 8.14 (слева) представлены векторы, связанные с первыми 50 сингулярными значениями портрета Линкольна. Если вы внимательно посмотрите, то увидите, что первые 50 векторов значительно более блочные, чем последующие, что указывает на то, что ранние векторы пролистывают базовую структуру матрицы, а последующие векторы добавляют больше деталей. На рис. 8.14 (справа) показано, как уменьшается среднеквадратичная ошибка между матрицей и ее реконструкцией при добавлении дополнительных векторов.

Эти эффекты становятся еще более яркими, когда мы смотрим на сами восстановленные изображения. На рис. 8.15 (слева) показано лицо Линкольна с пятью самыми сильными векторами, что составляет менее 1% от того, что доступно для идеальной реконструкции. Но даже в этот момент вы вполне сможете узнать его на опознании в полиции. На рис. 8.15 (в центре) демонстрируется бо́льшая детализация, когда мы включаем 50 векторов. Это выглядит так же хорошо, как исходное изображение, при печати, хотя график ошибок (рис. 8.15 (справа)) подчеркивает недостающие детали.

Рис. 8.15. Лицо Линкольна реконструировано из 5 (слева) и 50 (в центре) сингулярных значений с погрешностью для k = 50 (справа)

> *На заметку. Разложение по сингулярным значениям* (Singular Value Decomposition — SVD) — это мощный метод уменьшения размерности любой матрицы признаков.

8.5.2. Анализ основных компонентов

Анализ основных компонентов (Principal Components Analysis — PCA) — это техника, тесно связанная с уменьшением размерности наборов данных. Подобно SVD, мы будем определять векторы для представления набора данных. Как и SVD, мы будем упорядочивать их по важности, чтобы можно было восстановить приблизительное представление, используя несколько компонентов. PCA и SVD настолько тесно связаны, что их невозможно отличить для наших задач. Они делают то же самое одинаково, но с разных сторон.

Основные компоненты определяют оси эллипсоида, наилучшим образом подходящие к точкам. Источником этого набора осей является центр тяжести точек. PCA начинает с определения направления, на которое нужно проецировать точки, чтобы объяснить максимальное количество отклонений. Это линия, проходящая через центр тяжести, которая в некотором смысле лучше всего подходит для точек, что делает ее аналогом линейной регрессии. Затем мы можем спроецировать на эту линию каждую точку, причем точка пересечения определяет конкретную позицию на линии относительно центроида (centroid). Эти проецируемые позиции теперь определяют первое измерение (или основной компонент) нашего нового представления, как показано на рис. 8.16.

Для каждого последующего компонента мы ищем линию l_k, которая ортогональна всем предыдущим линиям и объясняет наибольшее количество оставшейся дисперсии. То, что каждое измерение ортогонально друг другу, означает, что они действуют как оси координат, устанавливая связь с собственными векторами. Каждое последующее измерение постепенно становится менее

важным, чем предшествующее ему, потому что сначала мы выбрали наиболее перспективные направления. Более поздние компоненты лишь постепенно вносят более мелкие детали, и, следовательно, мы можем остановиться, когда они станут достаточно малы.

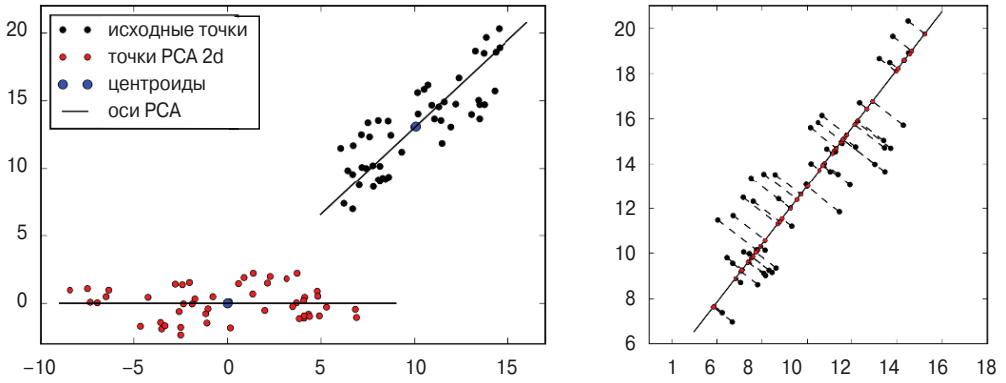


Рис. 8.16. PCA проецирует черные точки на ортогональную ось, повернутую так, чтобы получить альтернативное представление в красном (слева). Значения каждого компонента задаются проекцией каждой точки на соответствующую ось (справа)

Предположим, что размеры x и y практически идентичны. Мы ожидаем, что линия регрессии будет проецироваться вниз до $y = x$ в этих двух измерениях, поэтому их можно будет в значительной степени заменить одним измерением. PCA конструирует новые измерения как линейные комбинации исходных, объединяя те, которые сильно коррелируют, в пространство меньшего измерения. *Статистический факторный анализ* (statistical factor analysis) — это метод, который идентифицирует наиболее важные ортогональные измерения (измеряемые с помощью корреляции), которые объясняют бо́льшую часть дисперсии.

Относительно небольшого числа компонентов достаточно, чтобы охватить базовую структуру набора точек. Остаток, вероятно, является шумом, и как правило лучше удалить его из данных. После уменьшения размеров с помощью PCA (или SVD) мы должны получить более чистые данные, а не просто меньшее количество измерений.

> *На заметку*. PCA и SVD — это два разных подхода к вычислению одного и того же. Они должны служить одинаково хорошо для малоразмерных приближений матрицы признаков.

8.6. Случай из жизни: человеческий фактор

Впервые я был поражен мощью методов уменьшения размеров, таких как PCA и SVD, в ходе анализа исторических фигур для нашей книги *Who's Bigger*. Помните (раздел 4.7 "Случай из жизни"), как мы анализировали структуру и содержание Википедии, выделив в конечном итоге полдюжины функций, таких как PageRank, и длину статьи, для каждой из 800 000 + статей о людях в английском издании? Это свело каждую из персон к шестимерному вектору признаков, который мы проанализировали, чтобы оценить их относительную значимость.

Но все оказалось не так просто, как мы думали. Существенно разные люди были оценены выше по каждой конкретной переменной. Было непонятно, как их интерпретировать.

"В наших функциях так много различий и случайных шумов, — заметил мой соавтор Чарльз. — Давайте определим основные факторы, лежащие в основе этих наблюдаемых переменных, которые действительно показывают, что происходит".

Решением Чарльза был *факторный анализ* (factor analysis), являющийся вариантом PCA, который, в свою очередь, является вариантом SVD. Все эти методы сжимают матрицы признаков в меньший набор переменных, или факторов, так, чтобы эти факторы объясняли бо́льшую часть различий в полной матрице признаков. Мы ожидали, что факторный анализ извлечет один основной фактор, определяющий индивидуальную значимость. Но вместо этого наши входные переменные дали *два* независимых фактора, объясняющих данные. Оба объяснили примерно одинаковые пропорции дисперсии (31% и 28%), что означает, что эти скрытые переменные были примерно одинаковой важности. Но самое интересное то, что показали эти факторы.

Факторы (или единичные векторы, или основные компоненты) являются просто линейными комбинациями оригинальных входных функций. Как правило, у них нет имен, поэтому обычно вы просто описываете их как Фактор 1 и Фактор 2. Но два наших фактора были настолько разными, что Чарльз дал им названия *gravitas* (авторитетность) и *celebrity* (известность), и вы можете понять на рис. 8.17 почему.

Наш фактор *gravitas* в значительной степени происходит от (или "загружается" в статистическом смысле) двух форм PageRank. Авторитетность, кажется, точно отражает понятия признания, основанного на достижениях. Фактор *celebrity*, напротив, сильнее влияет на количество просмотров страниц, правок и длин статей. Фактор известности лучше отражает популярные (некоторые могут сказать — вульгарные) представления о репутации. Популярность певцов, актеров и других артистов лучше измеряется известностью, чем авторитетностью.

Наивысший рейтинг по авторитетности

Личность	Авт.	Знач.	Изв./Авт.
Наполеон	8	2	C G
Карл Линней	13	31	C G
Платон	23	25	C G
Аристотель	27	8	C G
Ф. Д. Рузвельт	30	43	C G
Плутарх	32	258	C G
Карл II	33	78	C G
Елизавета II	35	132	C G
Королева Виктория	38	16	C G
Уильям Шекспир	42	4	C G
Плиний Старший	43	212	C G
Тацит	52	300	C G
Геродот	58	123	C G
Карл V	61	84	C G
Георг V	64	235	C G

Наивысший рейтинг по известности

Личность	Изв.	Знач.	Изв./Авт.
Гробовщик (рестлер)	2	2172	C G
Виджай	8	4456	C G
Эдж	10	2603	C G
Кейн	13	2229	C G
Джон Сина	16	2277	C G
Бейонсе	19	1519	C G
Triple H	26	1596	C G
Рей Мистерио	36	2740	C G
Бритни Спирс	37	689	C G
Энн Коултер	45	3376	C G
Джесси Маккартни	48	4236	C G
Роджер Федерер	57	743	C G
Эшли Тисдейл	60	4445	C G
Майкл Джексон	75	180	C G
Дуэйн Джонсон	78	1446	C G

Рис. 8.17. Факторы gravitas и celebrity отлично справляются с разделением двух типов известных людей

Чтобы почувствовать разницу между авторитетностью и известностью, сравните наши самые высокие показатели для каждого фактора на рис. 8.17. Верхние фигуры слева — это явно старомодные тяжеловесы, люди высокого ранга и достижений. Это философы, короли и государственные деятели. Имена, перечисленные на рис. 8.17 (справа), настолько известны, что четверке лучших достаточно на этой земле только имени. Это профессиональные борцы, актеры и певцы. Весьма показательно, что только две фигуры изображают авторитетность на нашем измерителе знаменитости, это *Бритни Спирс* (1981–) [566] и *Майкл Джексон* (1958–2009) [136], оба среди платоновских идеалов современной знаменитости.

Я нахожу удивительным, что эти методы обучения без учителя были способны разделить два разных типа славы без каких-либо помеченных обучающих примеров или даже предвзятого мнения о том, что они искали. Факторы/векторы/компоненты просто отражают то, что было в данных, которые будут найдены.

Этот континуум известности-авторитетности служит поучительным примером силы методов уменьшения размеров. Все факторы/векторы/компоненты должны по определению быть ортогональными друг другу. Это означает, что каждый из них измеряет разные вещи так, как не делают две коррелированные входные переменные. Вам стоит провести некоторый предварительный анализ данных по основным компонентам, чтобы попытаться выяснить, что они на самом деле означают в контексте вашего приложения. Вы можете присваивать

имена по своему усмотрению, хотите — кошки, хотите — собаки, поэтому выбирайте те имена, с которыми вам будет приятно жить.

8.7. Дополнительная информация

Существует много популярных учебников, посвященных линейной алгебре, в том числе [75, 76, 77]. Кляйн [78] представляет интересное введение в линейную алгебру для информатики с упором на программирование и приложения, такие как теория кодирования и компьютерная графика.

8.8. Упражнения

Основы линейной алгебры

8.1. *[3]* Придумайте пару квадратных матриц A и B так, чтобы:
(a) $AB = BA$ (коммутативно).
(b) $AB \neq BA$ (не коммутативно).
В общем случае умножение матриц не является коммутативным.

8.2. *[3]* Докажите, что сложение матриц ассоциативно, т.е. $(A + B) + C = A + (B + C)$ для совместимых матриц A, B и C.

8.3. *[5]* Докажите, что матричное умножение ассоциативно, т.е. $(AB)C = A(BC)$ для совместимых матриц A, B и C.

8.4. *[3]* Докажите, что $AB = BA$, если A и B — это диагональные матрицы одного порядка.

8.5. *[5]* Докажите, что если $AC = CA$ и $BC = CB$, то $C(AB + BA) = (AB + BA)C$.

8.6. *[3]* Являются ли матрицы MM^T и M^TM квадратными и симметричными? Объясните.

8.7. *[5]* Докажите, что $(A^{-1})^{-1} = A$.

8.8. *[5]* Докажите, что $(A^T)^{-1} = (A^{-1})^T$ для любой несингулярной матрицы A.

8.9. *[5]* Является ли LU разложение матрицы уникальной? Обоснуйте ответ.

8.10. *[3]* Объясните, как решить матричное уравнение $Ax = b$.

8.11. *[5]* Покажите, что если M является квадратной матрицей, которая не является обратимой, то либо L, либо U в LU-разложении $M = L\,U$ имеет нуль на своей диагонали.

Собственные значения и собственные векторы

8.12. *[3]* Пусть $M = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$. Найдите все собственные значения M. Имеет ли M два линейно независимых собственных вектора?

8.13. *[3]* Докажите, что собственные значения A и A^T одинаковы.

8.14. *[3]* Докажите, что собственные значения диагональной матрицы равны диагональным элементам.

8.15. *[5]* Предположим, что матрица A имеет собственный вектор v с собственным значением λ. Покажите, что v является также собственным вектором для A^2, и найдите соответствующее собственное значение. Как насчет A^k, для $2 \leq k \leq n$?

8.16. *[5]* Предположим, что A — это обратимая матрица с собственным вектором v. Покажите, что v является также собственным вектором для A^{-1}.

8.17. *[8]* Покажите, что собственные значения MM^T такие же, как у M^TM. Их собственные векторы также одинаковы?

Реализация проектов

8.18. *[5]* Сравните скорость библиотечной функции для умножения матриц с собственной реализацией алгоритма вложенных циклов.

- Насколько быстрее библиотека на произведениях случайных матриц $n \times n$ как функция от n, когда n становится большим?
- Как насчет произведения матрицы $n \times m$ и $m \times n$, где $n \ll m$?
- Насколько вы повышаете производительность своей реализации для вычисления $C = AB$, сначала транспонируя внутреннее значение B, чтобы все скалярные произведения вычислялись по строкам матриц для повышения производительности кеша?

8.19. *[5]* Реализация исключения по Гауссу для решения систем уравнений, $CX = Y$. Сравните вашу реализацию с популярной библиотечной подпрограммой по следующим параметрам.

(a) *Скорость*. Как сравнивается время выполнения для матриц с плотными и разреженными коэффициентами?

(b) *Точность*. Каковы размеры числовых остатков $CX - Y$, особенно при увеличении числа условий матрицы.

(c) *Стабильность*. Происходит ли сбой вашей программы в сингулярной матрице? Как насчет почти сингулярных матриц, созданных в результате добавления небольшого случайного шума к сингулярной матрице?

Вопросы на интервью

8.20. *[5]* Почему векторизация считается мощным методом оптимизации числового кода?

8.21. *[3]* Что такое разложение по сингулярным числам? Что такое сингулярное значение? Что такое сингулярный вектор?

8.22. *[5]* Объясните разницу между “длинным” и “широким” форматом данных. Когда каждый из них может применяться на практике?

Конкурсы Kaggle

8.23. Расскажите, кто на что смотрит, исходя из анализа их мозговых волн.
`https://www.kaggle.com/c/decoding-the-human-brain`

8.24. Решите, будет ли конкретный студент правильно отвечать на заданный вопрос.
`https://www.kaggle.com/c/WhatDoYouKnow`

8.25. Определите пользователей мобильных телефонов по данным акселерометра.
`https://www.kaggle.com/c/accelerometer-biometric-competition`

Глава 9

Линейная и логистическая регрессии

> Неискушенный специалист по прогнозированию использует статистику, как пьяный использует фонарные столбы — для поддержки, а не для освещения.
>
> — Эндрю Лэнг (Andrew Lang)

Линейная регрессия является наиболее представительным методом "машинного обучения" для построения моделей, прогнозирования и классификации значений на основе учебных данных. Это допускает обучение на контрастах.

- Линейная регрессия имеет красивую теоретическую основу, но на практике эта алгебраическая формулировка обычно отбрасывается в пользу более быстрой и более эвристической оптимизации.
- Модели линейной регрессии по определению являются линейными. Это позволяет засвидетельствовать ограничения таких моделей, а также разработать интеллектуальные методы для обобщения в другие формы.
- Линейная регрессия одновременно стимулирует построение модели с сотнями переменных и методы регуляризации, чтобы гарантировать, что большинство из них будут игнорироваться.

Линейная регрессия — это метод моделирования, который должен служить базовым подходом к построению моделей, управляемых данными. Эти модели, как правило, легко построить, легко интерпретировать, и на практике они довольно хороши. При достаточных навыках и труде более передовые методы машинного обучения могут привести к лучшей производительности, но возможная отдача зачастую не стоит этих усилий. Сначала создайте свои модели линейной регрессии, а затем решите, стоит ли работать усерднее для достижения лучших результатов.

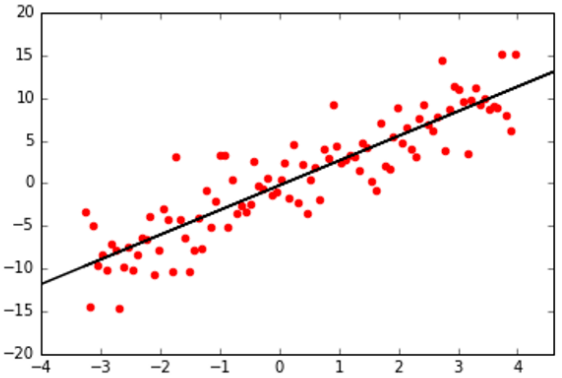


Рис. 9.1. Линейная регрессия создает линию, наилучшим образом соответствующую набору точек

9.1. Линейная регрессия

Когда дан набор из n точек, линейная регрессия стремится найти линию, которая наилучшим образом *аппроксимирует* или соответствует этим точкам, как показано на рис. 9.1. Есть много причин, по которым мы можем захотеть сделать это. Один класс задач подразумевает упрощение и сжатие: мы можем заменить большой набор зашумленных точек данных в плоскости xy аккуратной линией, которая их описывает, как показано на рис. 9.1. Эта линия регрессии удобна для визуализации, показывая основную тенденцию в данных и выделяя местоположение и величину выбросов.

Однако нас больше всего интересует регрессия как метод прогнозирования значений. Мы можем представить, что каждая наблюдаемая точка $p = (x, y)$ является результатом функции $y = f(x)$, где x представляет переменные функции, а y — независимую целевую переменную. Когда дан набор из n таких точек $\{p_1, p_2, \ldots, p_n\}$, мы ищем функцию $f(x)$, которая лучше всего объясняет эти точки. Такая функция $f(x)$ интерполирует или моделирует точки, предоставляя способ оценить значение y', связанное с любым возможным x', а именно $y' = f(x')$.

9.1.1. Линейная регрессия и двойственность

Между регрессией и решением линейных уравнений существует связь, что интересно исследовать. При решении линейных систем мы ищем единственную точку, которая лежит на пересечении n данных линий. В регрессии нам вместо этого дают n точек, и мы ищем линию, которая лежит на "всех" точках. Здесь есть два различия: (a) взаимообмен точек для линий и (b) поиск наилучшего соответствия при ограничениях приводят к полностью ограниченной задаче ("все" против всех).

Различие между точками и линиями оказывается тривиальным, поскольку они на самом деле одно и то же. В двумерном пространстве обе точки (s, t) и прямые $y = mx + b$ определяются двумя параметрами: $\{s, t\}$ и $\{m, b\}$ соответственно. Кроме того, при соответствующем преобразовании *двойственности* (duality) эти линии эквивалентны точкам в другом пространстве. В частности, рассмотрим следующее преобразование

$$(s, t) \leftrightarrow y = sx - t$$

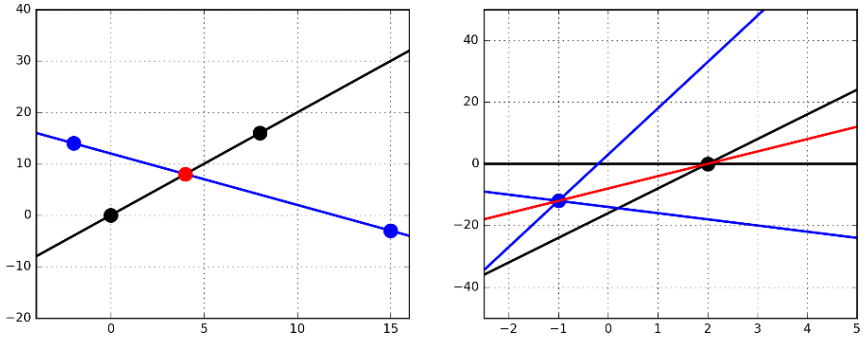


Рис. 9.2. Точки эквивалентны линиям при преобразовании двойственности. Точка (4, 8) красного цвета (слева) отображается на красную линию $y = 4x - 8$ справа. Оба набора из трех коллинеарных точек слева соответствуют трем линиям, проходящим через одну и ту же точку справа

Теперь любой набор точек, которые лежат на одной линии, сопоставляется с набором линий, которые пересекают сингулярную точку, поэтому поиск линии, которая попадает во все множество точек, является алгоритмически тем же, что и поиск точки, которая попадает во все линии набора.

Пример показан на рис. 9.2. Точка пересечения на рис. 9.2 (слева) — это $p = (4, 8)$, и она соответствует красной линии $y = 4x - 8$ справа. Эта красная точка p определяется пересечением черных и синих линий. В двойственном пространстве эти линии превращаются в черные и синие точки, лежащие на красной линии. Три коллинеарные точки слева (красная с двумя черными или двумя синими) отображают три линии, проходящие через общую точку справа: одну красную и две одинакового цвета. Это преобразование двойственности меняет роль точек и линий таким образом, что все имеет смысл.

Главная разница в определении линейной регрессии заключается в том, что мы ищем линию, *максимально приближенную* ко всем точкам. Чтобы выполнить эту работу, мы должны быть осторожны и правильно измерить погрешность.

9.1.2. Ошибка в линейной регрессии

Остаточная ошибка (residual error) подобранной линии $f(x)$ — это разница между прогнозируемыми и фактическими значениями. Как показано на рис. 9.3, для конкретного вектора признаков x_i и соответствующего целевого значения y_i определяется остаточная ошибка r_i:

$$r_i = y_i - f(x_i).$$

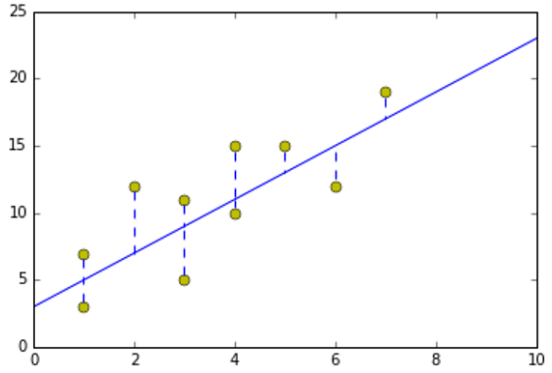


Рис. 9.3. Остаточная ошибка в наименьших квадратах — это проекция $y_i - f(X)$ на X, а не кратчайшее расстояние между линией и точкой

Это то, о чем мы позаботимся, но учтите, что это не единственный способ определения ошибки. Ближайшее расстояние до линии на самом деле определяется перпендикуляром-биссектором через целевую точку. Но мы стремимся спрогнозировать значение y_i из x_i, поэтому остаток является правильным понятием ошибки для наших целей.

Регрессия наименьших квадратов (least squares regression) минимизирует сумму квадратов остаточных ошибок для всех точек. Эта метрика была выбрана потому, что (1) возведение в квадрат остатка игнорирует знак ошибки, поэтому положительные и отрицательные остатки не компенсируют друг друга, и (2) это приводит к удивительно хорошей замкнутой форме для нахождения коэффициентов наилучшей соответствующей линии.

9.1.3. Нахождение оптимального соответствия

Линейная регрессия ищет линию $y = f(x)$, которая минимизирует сумму квадратов ошибок по всем учебным точкам, т.е. вектор коэффициентов w, который минимизирует

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 \text{ , где } f(x) = w_0 + \sum_{i=1}^{m-1} w_i x_i.$$

Предположим, мы пытаемся найти соответствие набору из n точек, каждая из которых m-мерна. Первыми $m - 1$ измерениями каждой точки является вектор признаков $(x_1, \ldots, x_{m-1})$, причем последнее значение $y = x_m$ служит целевой или зависимой переменной.

Мы можем закодировать эти n векторов признаков в виде матрицы $n \times (m - 1)$. Мы можем сделать ее матрицей A размером $n \times m$, добавив в нее столбец единиц. Этот столбец можно рассматривать как "постоянный" признак, который при умножении на соответствующий коэффициент становится y-пересечением подобранной линии. Кроме того, n целевых значений могут быть удобно представлены в виде вектора b размером $n \times 1$.

Оптимальная линия регрессии $f(x)$, которую мы ищем, определяется вектором $m \times 1$ коэффициентов $w = \{w_0, w_1, \ldots, w_{m-1}\}$. Вычисление этой функции по данным точкам является в точности произведением Aw, создающим вектор $n \times 1$ предсказаний целевого значения. Таким образом, $(b - Aw)$ является вектором остаточных значений.

Как мы можем найти коэффициенты наилучшей соответствующей линии? Вектор w определяется как

$$w = (A^T A)^{-1} A^T b.$$

Во-первых, давайте попробуем это понять, прежде чем пытаться продолжить. Размеры члена справа таковы

$$((m \times n)(n \times m))(m \times n)(n \times 1) \rightarrow (m \times 1),$$

что точно соответствует размерам целевого вектора w, так что это хорошо. Во-вторых, $A^T A$ определяет ковариационную матрицу для столбцов/признаков матрицы данных, и ее инвертирование похоже на решение системы уравнений. Член $A^T b$ вычисляет скалярные произведения значений данных и целевых значений для каждого из m объектов, обеспечивая меру того, насколько коррелирован каждый объект с целевыми результатами. Мы еще не понимаем, почему это работает, но должно быть ясно, что это уравнение состоит из значимых компонентов.

> *На заметку.* То, что линия регрессии наименьших квадратов определяется как $w = (A^T A)^{-1} A^T b$, означает, что решение задач регрессии сводится к инвертированию и умножению матриц. Эта формула отлично работает для небольших матриц, но алгоритм градиентного спуска (см. раздел 9.4) окажется более эффективным на практике.

Рассмотрим случай одной переменной x, где мы ищем наиболее подходящую линию вида $y = w_0 + w_1 x$. Наклон этой линии определяется как

$$w_1 = \sum_{i=1}^{n} \frac{(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} = r_{xy} \frac{\sigma_x}{\sigma_y}.$$

с $w_0 = \bar{y} - w_1 \bar{x}$ благодаря наблюдению, что наиболее подходящая линия проходит через $(\bar{x}, \bar{y})$.

Связь с коэффициентом корреляции (r_{xy}) здесь очевидна. Если x не был связан с y ($r_{xy} = 0$), то w_1 действительно должно быть нулем. Даже при идеальной корреляции ($r_{xy} = 1$) мы должны масштабировать x, чтобы привести его в правильный размерный диапазон y. Это роль σ_y/σ_x.

Откуда берется формула линейной регрессии? Должно быть ясно, что в линии наилучшего соответствия мы не можем изменить ни один из коэффициентов w и надеемся получить лучшее соответствие. Это означает, что вектор ошибки $(b - Aw)$ должен быть ортогональным с вектором, связанным с каждой переменной x_i, иначе был бы способ изменить коэффициент так, чтобы он подходил лучше.

Ортогональные векторы имеют нулевые произведения. Поскольку i-й столбец A^T имеет произведение нулевой точки с вектором ошибки, $A^T(b - Aw) = \bar{0}$, где $\bar{0}$ — это вектор всех нулей. Тогда простая алгебра дает

$$w = (A^T A)^{-1} A^T b.$$

9.2. Лучшие регрессионные модели

Если дана матрица A из n точек, каждая из $m - 1$ измерений и целевого массива b размером $n \times 1$, мы можем инвертировать и умножить соответствующие матрицы, чтобы получить матрицу желаемых коэффициентов w. Это определяет модель регрессии. Готово!

Тем не менее можно предпринять несколько шагов, которые могут привести к лучшим регрессионным моделям. Некоторые из них включают манипулирование входными данными для повышения вероятности точной модели, но другие требуют более концептуальных вопросов о том, как *должна* выглядеть наша модель.

9.2.1. Удаление выбросов

Линейная регрессия осуществляет поиск линии $y = f(x)$, которая минимизирует сумму квадратов ошибок по всем учебным точкам, т.е. вектор коэффициентов w, который минимизирует

$$\sum_{i=1}^{n} (y_i - f(x_i))^2, \text{ где } f(x) = w_0 + \sum_{i=1}^{m-1} w_i x_i.$$

Из-за квадратичного веса остатков отдаленные точки могут сильно повлиять на соответствие. Точка на расстоянии 10 от ее прогноза оказывает в 100 раз большее влияние на ошибку обучения, чем точка всего в 1 единице от подобранной линии. Кто-то может возразить, что это вполне уместно, но должно быть ясно, что выбросы имеют большое влияние на форму линии наилучшего соответствия. Это создает проблему, когда точки выброса отражают шум, а не сигнал, поскольку линия регрессии изо всех сил старается приспособить плохие данные, вместо того, чтобы подогнать хорошие.

Впервые мы столкнулись с этой проблемой еще на рис. 6.3 с квартетом Энскомба — набором из четырех небольших наборов данных с идентичными итоговыми статистическими данными и линиями регрессии. Два из этих наборов точек осуществили свою магию из-за одиночных точек выброса. Удалите выбросы, и линия соответствия будит проходит через середину данных.

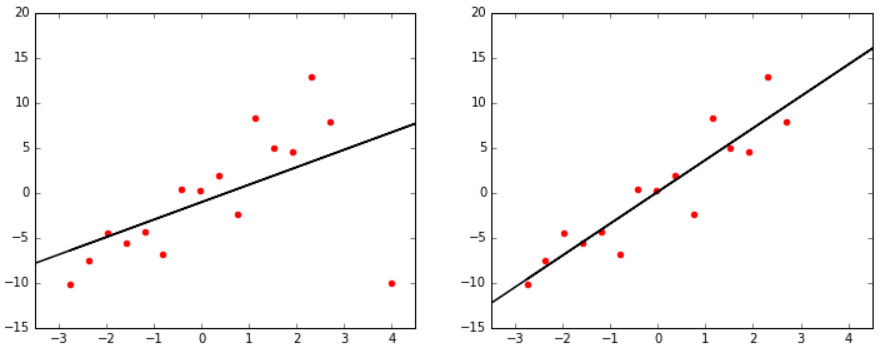


Рис. 9.4. Удаление точек выброса (слева) может привести к гораздо более значимым соответствиям (справа)

На рис. 9.4 показана наиболее подходящая линия регрессии с точкой выброса (слева) и без нее (справа) в правом нижнем углу. Подгонка справа намного лучше: с r^2, равным 0,917 без выброса, по сравнению с 0,548 с выбросом.

Таким образом, выявление отдаленных точек и их удаление может привести к принципиально более надежной подгонке. Самый простой подход заключается в том, чтобы подогнать весь набор точек, а затем использовать величину остатка $r_i = (y_i - f(x_i))^2$, чтобы решить, является ли точка p_i выбросом. Однако, прежде чем удалять их, важно убедиться, что эти точки *действительно* представляют собой ошибки. В противном случае у вас останется впечатляюще линейная подгонка, которая хорошо работает только на примерах.

9.2.2. Поиск соответствия нелинейных функций

Линейные отношения легче понять, чем нелинейные, и в целом они подходят в качестве стандартного допущения при отсутствии лучших данных.

Многие явления имеют *линейный* характер, причем зависимая переменная растет примерно пропорционально входным переменным.

- Оплата растет примерно линейно пропорционально количеству отработанного времени.
- Цена дома растет примерно линейно пропорционально размерам его жилой площади.
- Вес людей увеличивается примерно линейно пропорционально количеству съеденной пищи.

Линейная регрессия прекрасно работает, когда она пытается найти соответствие данным, в основе которых лежат на самом деле линейные отношения. Но, по правде говоря, ни одна интересная функция не является *абсолютно* линейной. Существует старое правило статистики, которое гласит, что, если вы хотите, чтобы функция была линейной, измеряйте ее только по двум точкам.

Мы можем значительно увеличить репертуар моделируемых форм, если выйдем за пределы линейных функций. Линейная регрессия соответствует линиям, а не кривым высокого порядка. Но мы можем найти квадратичное соответствие, добавив дополнительную переменную со значением x^2 в нашу матрицу данных, в дополнение к x. Модель $y = w_0 + w_1 x + w_2 x^2$ является квадратичной, но обратите внимание, что она является линейной функцией своих нелинейных входных значений. Мы можем подбирать произвольно сложные функции, добавляя правильные переменные высшего порядка в нашу матрицу данных и формируя их линейные комбинации. Мы можем подогнать произвольные многочлены и экспоненты/логарифмы, явно включив в нашу матрицу данных правильные переменные компонента, такие как $\sqrt{x}$, $\lg(x)$, x^3 и $1/x$.

Дополнительные функции также могут быть использованы для фиксации нелинейных взаимодействий между парами входных переменных. Площадь прямоугольника A вычисляется как длина, умноженная на ширину, что означает невозможность получить точное приближение к A как линейной комбинации длины и ширины. Но, как только мы добавим элемент площади в нашу матрицу данных, это нелинейное взаимодействие может быть зафиксировано с помощью линейной модели.

Однако явное включение всех возможных нелинейных членов быстро становится неразрешимым. Сложение всех степеней x_i для $1 \leq i \leq k$ приведет к увеличению матрицы данных в k раз. Включение всех пар произведений среди n переменных еще хуже, что делает матрицу в $n(n+1)/2$ раза больше. Нужно быть осмотрительным в отношении того, какие нелинейные члены следует учитывать для роли в модели. Действительно, одним из преимуществ более мощных методов обучения, таких как метод опорных векторов, будет возможность включать нелинейные члены без явного перечисления.

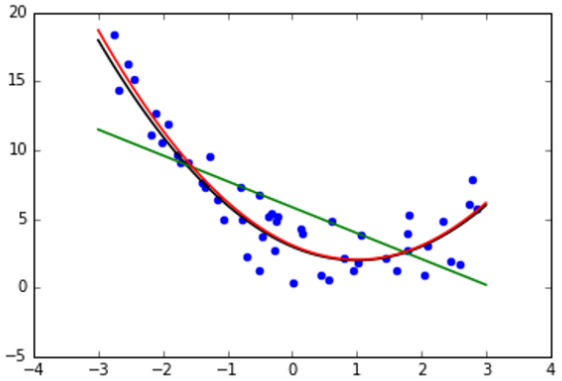


Рис. 9.5. Модели более высокого порядка (верхняя кривая линия) могут привести к лучшему соответствию, чем линейные модели (прямая линия)

9.2.3. Функция и целевое масштабирование

В принципе, линейная регрессия может найти лучшую линейную модель, подходящую для любого набора данных. Но мы должны сделать все возможное, чтобы помочь ему найти правильную модель. Обычно это подразумевает предварительную обработку данных, чтобы оптимизировать их для выразительности, интерпретируемости и числовой стабильности. Проблема здесь заключается в том, что функции, которые варьируются в широких числовых диапазонах, требуют коэффициентов в одинаково широких диапазонах для их объединения.

Предположим, что мы хотели построить модель для прогнозирования валового национального продукта стран в долларах в зависимости от численности их населения x_1 и уровня грамотности x_2. Оба фактора кажутся разумными компонентами такой модели. Действительно, оба фактора могут в равной степени влиять на объем экономической активности. Но они работают в совершенно разных масштабах: население страны варьируется от десятков тысяч до более миллиарда человек, а доля людей, умеющих читать, по определению составляет от нуля до единицы. Можно было бы предположить, что полученная модель соответствия выглядит примерно так:

$$\text{ВВП} = 10\,000x_1 + 10\,000\,000\,000\,000x_2 \text{ долл.}$$

Это очень плохо по нескольким причинам.

- *Нечитаемые коэффициенты*. Скажите быстро, каков коэффициент x_2 в приведенном выше уравнении? Нам трудно иметь дело с величиной таких чисел (это 10 триллионов), и нам трудно сказать, какая переменная

вносит более важный вклад в результат, учитывая их диапазоны. Это x_1 или x_2?

- *Числовая неточность.* Алгоритмы числовой оптимизации имеют проблемы, когда значения находятся в пределах многих порядков. Дело не только в том, что числа с плавающей запятой представлены конечным количеством битов. Более важно то, что многие алгоритмы машинного обучения параметризуются константами, которые должны выполняться одновременно для всех переменных. Например, использование фиксированных размеров шагов при поиске с градиентным спуском (что будет обсуждаться в разделе 9.4.2) может привести к резкому превышению скорости в определенных направлениях, а в других — к недостаточности.
- *Неуместные формулировки.* Приведенная выше модель для прогнозирования ВВП глупа на первый взгляд. Предположим, я решил создать свою собственную страну, в которой был бы ровно один человек, который мог бы читать. Неужели мы действительно думаем, что ВВП в Скиеналенде должен составлять 10 000 000 010 000 долл.?

Лучшей моделью может быть что-то вроде:

$$\text{ВВП} = 20\,000 x_1 x_2 \text{ долл.},$$

что может быть истолковано как каждый из x_1 людей, накапливающих богатство со скоростью, модулируемой их грамотностью. Это обычно требует заполнения матрицы данных соответствующими членами произведения. Но есть вероятность, что при правильном (логарифмическом) масштабировании цели эта модель может выпасть из линейной регрессии.

Рассмотрим три разные формы масштабирования, которые решают различные типы проблем.

Функция масштабирования: Z-оценки

Ранее мы уже обсуждали Z-оценки, которые масштабируют значения каждого объекта в отдельности, так что среднее значение равно нулю, а диапазоны сравнимы. Пусть μ будет средним значением данного признака, а σ стандартным отклонением. Тогда Z-оценка x равна $Z(x) = (x - \mu)/\sigma$.

Использование Z-оценок в регрессии решает вопрос интерпретируемости. Поскольку все характеристики будут иметь одинаковые средние значения и отклонения, величина коэффициентов будет определять относительную важность этих факторов для прогноза. Действительно, в надлежащих условиях эти коэффициенты будут отражать коэффициент корреляции каждой переменной с целью. И то, что эти переменные находятся теперь в пределах одной величины, упрощает работу для алгоритма оптимизации.

Сублинейное масштабирование функций

Рассмотрим линейную модель для прогнозирования количества лет обучения y ребенка в зависимости от дохода семьи. Уровень образования может варьироваться от 0 до 12 + 4 + 5 = 19 лет, поскольку мы учитываем возможность докторантуры. Уровень дохода семьи x может варьироваться от 0 до Билла Гейтса. Однако заметьте, что нет модели в форме

$$y = w_1 x + w_0,$$

способной дать разумные ответы как для моих детей, так и для детей Билла Гейтса. Реальное влияние дохода на уровень образования, по-видимому, находится на более низком уровне: дети, находящиеся за чертой бедности, в среднем не могут выйти за рамки средней школы, в то время как дети из среднего класса обычно ходят в колледж. Однако нет способа зафиксировать это в линейно-взвешенной переменной, не обрекая детей Гейтса на сотни или тысячи лет в школе.

Огромный разрыв между наибольшим/наименьшим и медианным значениями означает, что ни один коэффициент не может использовать эту функцию без увеличения больших значений. Уровень дохода распределен по степенному закону, и Z-оценки таких переменных по степенному закону не могут помочь, поскольку они являются просто линейными преобразованиями. Ключ в том, чтобы заменить/дополнить такие функции x сублинейными функциями, такими как $\log(x)$ и $\sqrt{x}$. Z-оценки этих преобразованных переменных окажутся гораздо более значимыми для построения моделей.

Сублинейное целевое масштабирование

Мелкомасштабные переменные нуждаются в мелкомасштабных целях, чтобы быть реализованными с использованием мелкомасштабных коэффициентов. Попытка предсказать ВВП по Z-оценочным переменным потребует чрезвычайно больших коэффициентов. Как еще можно получить триллионы долларов от линейной комбинации переменных в диапазоне от –3 до +3?

Возможно, было бы полезно изменить целевое значение от долларов до миллиардов долларов, но здесь есть более глубокая проблема. Когда ваши функции распределены нормально, вы можете хорошо справиться только с регрессом к аналогично распределенной цели. Статистические данные, такие как ВВП (GDP), вероятно, распределены по степенному закону: есть много небольших бедных стран по сравнению с очень немногими крупными и богатыми. Любая линейная комбинация нормально распределенных переменных не может эффективно реализовать цель, распределенную по степенному закону.

Решение здесь заключается в том, что попытка предсказать *логарифм* ($\log_c y$) целевого значения степенного закона y обычно лучше, чем предсказание самого y. Конечно, значение $c^{f(x)}$ может затем использоваться для оценки y, но теперь существует возможность делать значимые прогнозы по всему диапазону значений. Применение степенной функции с логарифмом обычно приводит к лучшему поведению и более нормальному распределению.

Это также позволяет нам неявно реализовывать более широкий спектр функций. Предположим, что "правильная" функция прогнозирования валового внутреннего продукта была фактически такой

$$\text{ВВП} = 20\,000 x_1 x_2 \text{ долл.}$$

Это не может быть реализовано с помощью линейной регрессии без переменных взаимодействия. Однако заметьте, что

$$\log \text{ВВП} = \log(20\,000 x_1 x_2) = \log 20\,000 + \log x_1 + \log x_2.$$

Таким образом, логарифмы произвольных произведений взаимодействий могут быть реализованы при условии, что матрица признаков содержит также логарифмы исходных входных переменных.

9.2.4. Работа с сильно коррелирующими признаками

Последним подводным камнем, который мы обсудим, является проблема сильно коррелирующих признаков. Замечательно иметь признаки, которые сильно коррелируют с целью: они позволяют нам строить модели с высокой степенью прогнозирования. Однако наличие множества признаков, которые тесно связаны *друг с другом*, может вызывать проблемы.

Предположим, в вашей матрице данных есть два идеально коррелирующих признака, скажем, высота объекта в футах (x_1) и его высота в метрах (x_2). Поскольку 1 метр равен 3,28 футам, эти две переменные прекрасно коррелируют. Но наличие обеих этих переменных на самом деле не может помочь нашей модели, поскольку добавление идеально коррелирующих признаков не дает никакой дополнительной информации для прогнозирования. Если бы такие дублирующие признаки действительно имели для нас значение, это означало бы, что мы могли бы создавать все более точные модели, просто делая дополнительные копии столбцов из любой матрицы данных!

Но коррелирующие признаки вредны для моделей, а не просто нейтральны. Предположим, что наша зависимая переменная является функцией высоты. Обратите внимание, что одинаково хорошие модели могут быть построены в зависимости только от x_1, или только от x_2, или от любой произвольной линейной комбинации x_1 и x_2. Какая модель подходит для ответа?

Это сбивает с толку, но могут случиться и худшие вещи. Строки в ковариационной матрице будут взаимозависимы, поэтому для вычисления $w = (A^T A)^{-1} A^T b$ теперь потребуется инвертировать сингулярную матрицу! Численные методы для вычисления регрессии могут потерпеть неудачу.

Решение здесь заключается в том, чтобы идентифицировать пары признаков, которые сильно коррелируют, в ходе вычисления подходящей ковариационной матрицы. Если они прячутся, вы можете устранить любую переменную с небольшой потерей мощности. Лучше полностью исключить эти корреляции, комбинируя функции. Это одна из проблем, решаемых в ходе уменьшения размерности с использованием таких методов, как разложение по сингулярным числам, которые мы обсуждали в разделе 8.5.1.

9.3. Случай из жизни: водитель такси

Я горжусь многими вещами в своей жизни, но, возможно, больше всего из них тем, что я житель Нью-Йорка. Я живу в самом захватывающем городе на земле, истинном центре вселенной. Астрономы, по крайней мере хорошие, скажут вам, что каждый новый год начинается, когда шар падает на Таймс-сквер, а затем излучается из Нью-Йорка со скоростью света в остальной мир.

Таксистов Нью-Йорка уважают во всем мире за их ум и знание улиц. Водителю за каждую поездку принято давать чаевые, однако нет устоявшейся традиции по их размеру. В нью-йоркских ресторанах "правильная" сумма чаевых официанту — это удвоенная такса, но я не знаю ни одной такой эвристики для чаевых в такси. Мой алгоритм заключается в округлении до ближайшего доллара и последующем добавлении пары долларов в зависимости от того, как быстро он меня туда привез. Но я всегда чувствовал себя неуверенно. Я жадный пассажир? Или, может быть, транжира?

Набор данных о такси, рассмотренный в разделе 1.2.4, обещал дать ответ. Он содержал более 80 миллионов записей, с полями для даты, времени, мест посадки и высадки, пройденного расстояния, стоимости проезда и, конечно, чаевых. Люди несоразмерно платят за более длинные или короткие поездки? Поздно ночью или по выходным? Другие вознаграждают быстрых водителей, как я? Все должно быть в данных.

Мой студент Алексей Старов принял вызов. Мы добавили соответствующие функции в набор данных, чтобы охватить некоторые из этих понятий. Чтобы явно фиксировать такие условия, как поздняя ночь и выходные, мы установили двоичные переменные-индикаторы, где 1 будет означать, что поездка была поздней ночью, а 0 — в другое время дня. Коэффициенты нашего окончательного уравнения регрессии были таковы:

Переменная	Коэффициент LR
(intercept)	0,08370835
duration	0,00000035
distance	0,00000004
fare	0,17503086
tolls	0,06267343
surcharge	0,01924337
weekends	–0,02823731
business day	0,06977724
rush hour	0,01281997
late night	0,04967453
# of passengers	–0,00657358

Результаты здесь можно объяснить просто. Только одна переменная действительно имеет значение: тариф на счетчике (fare). Эта модель дает 17,5% от общей стоимости проезда с незначительными корректировками для других вещей. Модель с одним параметром, дающая 18,3% от каждого тарифа, оказалась почти такой же точной, как и модель с десятью факторами.

Были очень сильные корреляции между тарифом, пройденным расстоянием (distance) (0,95) и продолжительностью поездки (duration) (0,88), но оба последних фактора являются частью формулы, по которой рассчитываются тарифы. Эти корреляции с тарифом настолько сильны, что ни одна из переменных не может дать много дополнительной информации. К нашему разочарованию, мы не могли реально выявить влияние времени суток или чего-либо еще, потому что эти корреляции были слишком слабыми.

Более глубокий взгляд на данные показал, что каждая поездка в базе данных была оплачена с кредитной карты, а не наличными. Ввод суммы чаевых в счетчик после каждой транзакции утомителен и отнимает много времени, особенно когда вы стремитесь сделать как можно больше поездок за каждую 12-часовую смену. Более того, настоящие нью-йоркские таксисты достаточно сообразительны, чтобы не платить налоги с чаевых, о которых никто не знает.

Я всегда плачу за проезд наличными, но люди, которые оплачивают их с помощью кредитной карты, видят меню, предлагающее им выбрать, какие чаевые оставить. Данные ясно показали, что большинство из них бездумно нажимают среднюю кнопку, вместо того, чтобы ввести собственный выбор и отразить качество обслуживания.

Использование 80 миллионов записей о тарифах для выработки простой линейной регрессии по десяти переменным является излишним. Лучше использовать эти данные для создания сотен или даже тысяч различных моделей,

каждая из которых предназначена для определенного класса поездок. Возможно, мы могли бы построить отдельную модель для поездок между каждой парой городских адресов. В самом деле, давайте вспомним нашу карту щедрости чаевых, представленную на рис. 1.7.

Алгоритму решения потребовалось несколько минут, чтобы найти наилучшее соответствие для такого большого набора данных, но его окончательное завершение означало, что некоторый алгоритм должен был работать быстрее и надежнее, чем инверсия матриц. Эти алгоритмы рассматривают регрессию как проблему подбора параметров, как мы обсудим в следующем разделе.

9.4. Регрессия как подбор параметров

Замкнутая форма формулы для линейной регрессии, $w = (A^T A)^{-1} A^T b$, компактна и элегантна. Однако у нее есть некоторые проблемы, которые делают ее неоптимальной для практических вычислений. Инверсия матриц медленна для больших систем и подвержена числовой нестабильности. Кроме того, формулировка довольно хрупкая: магию линейной алгебры здесь трудно распространить на более общие задачи оптимизации.

Но существует альтернативный способ формулирования и решения задач линейной регрессии, который на практике оказывается лучше. Этот подход ведет к более быстрым алгоритмам, более надежным числам и может быть легко адаптирован к другим алгоритмам обучения. Он моделирует линейную регрессию как задачу *подбора параметров* и использует алгоритмы поиска, чтобы найти наилучшие возможные значения для этих параметров.

Для линейной регрессии мы ищем линию, которая наилучшим образом соответствует точкам, по всем возможным наборам коэффициентов. В частности, мы ищем линию $y = f(x)$, которая минимизирует сумму квадратов ошибок по всем учебным точкам, т.е. вектор коэффициентов w, который минимизирует

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 \text{, где } f(x) = w_0 + \sum_{i=1}^{m-1} w_i x_i .$$

Давайте начнем со случая, когда мы пытаемся смоделировать y как линейную функцию от одной переменной или свойства x, поэтому $y = f(x)$ означает $y = w_0 + w_1 x$. Чтобы определить нашу линию регрессии, мы ищем пару параметров (w_0, w_1), которая минимизирует ошибку или стоимость, или *потери* (loss), а именно сумму квадратов отклонения между значениями точек и линией.

Каждая возможная пара значений для (w_0, w_1) будет определять *некоторую* строку, но нам действительно необходимы значения, которые минимизировали бы ошибки или *функцию потерь* (loss function) $J(w_0, w_1)$, где

$$J(w_0, w_1) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - f(x_i))^2 =$$

$$= \frac{1}{2n}\sum_{i=1}^{n}(y_i - (w_0 + w_1 x_i))^2.$$

Сумма квадратов ошибок должна быть понятна, но откуда берется $1/(2n)$? $1/n$ превращает это в среднюю ошибку на строку, а $1/2$ — это общепринятое соглашение по техническим причинам. Но следует иметь в виду, что $1/(2n)$ никоим образом не влияет на результаты оптимизации. Этот множитель будет одинаковым для каждой пары (w_0, w_1), и поэтому не может сказать, какие параметры выбраны.

Итак, как мы можем найти правильные значения для w_0 и w_1? Мы могли бы попробовать несколько пар случайных значений и сохранить ту, которая набрала лучшие результаты, т.е. с минимальными потерями $J(w_0, w_1)$. Но, похоже, вряд ли удастся найти лучшее или даже достойное решение. Для более систематического поиска нам нужно воспользоваться специальным свойством, скрывающимся в функции потерь.

9.4.1. Выпуклые пространства параметров

Результатом вышеприведенного обсуждения является то, что функция потерь $J(w_0, w_1)$ определяет поверхность в (w_0, w_1)-пространстве, при этом нас интересует точка в данном пространстве с наименьшим значением z, где $z = J(w_0, w_1)$.

Давайте начнем с того, что сделаем все еще проще, заставив нашу линию регрессии проходить через начало координат, установив $w_0 = 0$. Это оставляет нам только один свободный параметр для поиска, а именно наклон линии w_1. Определенные уклоны будут выполнять работу по выравниванию точек, показанных на рис. 9.6 (слева), значительно лучше, чем другие, причем линия $y = x$ явно соответствует желаемому подбору.

На рис. 9.6 (справа) показано, как погрешность подбора (потери) зависит от w_1. Интересно, что функция ошибок имеет форму, похожую на параболу. Она достигает единственного минимального значения в нижней части кривой. Значение x этой минимальной точки определяет наилучший наклон w_1 для линии регрессии, который равен $w_1 = 1$.

Любая выпуклая поверхность имеет ровно один локальный минимум. Кроме того, для любого выпуклого пространства поиска найти этот минимум совсем не сложно: просто продолжайте идти в направлении вниз, пока не дойдете до него. Из любой точки на поверхности мы можем сделать небольшой шаг к ближайшей точке на поверхности. Некоторые направления приведут нас к более высокому значению, но другие приведут нас вниз. Если мы сможем

определить, какой шаг приведет нас к снижению, мы приблизимся к минимумам. И такое направление есть всегда, кроме тех случаев, когда мы стоим на самой нижней точке!

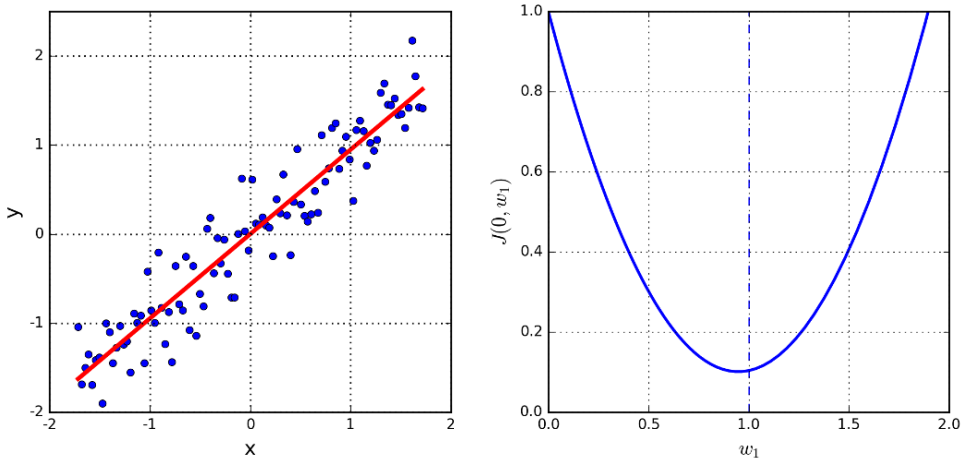


Рис. 9.6. Наилучшая возможная линия регрессии $y = w_1 x$ (слева) может быть найдена за счет определения w_1, который минимизирует ошибку подбора, определяемую минимумами выпуклой функции

На рис. 9.7 показана поверхность, которую мы получаем для задачи полной регрессии в (w_0, w_1)-пространстве. Функция потерь $J(w_0, w_1)$ выглядит как чаша с единственным наименьшим z-значением, которое определяет оптимальные значения для двух параметров линии. Самое замечательное то, что эта функция потерь $J(w_0, w_1)$ снова выпукла и действительно остается выпуклой для любой задачи линейной регрессии в любом количестве измерений.

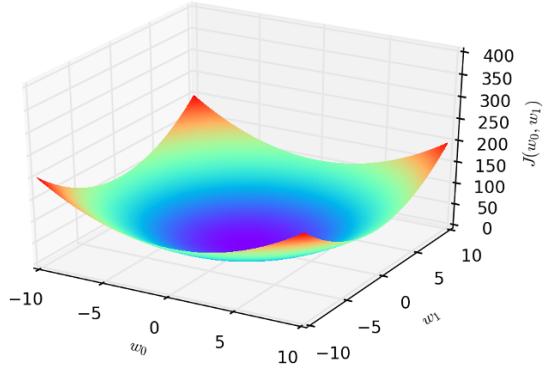


Рис. 9.7. Линейная регрессия определяет выпуклое пространство параметров, где каждая точка представляет возможную линию, а минимальная точка определяет лучшую подходящую линию

Как мы можем определить, является ли данная функция выпуклой? Вспомните, как изучали исчисление с одной переменной, x. Вы узнали, как взять *производную* $f'(x)$ от функции $f(x)$, которая соответствует значению наклона поверхности $f(x)$ в каждой точке. Всякий раз, когда эта производная была нулевой, это означало, что вы достигли какой-то интересной точки, будь то локальные максимумы или минимумы. Вспомним *вторую производную* $f''(x)$, которая была производной функцией производной $f'(x)$. В зависимости от знака второй производной $f''(x)$, вы можете определить, достигли вы максимума или минимума.

Таким образом: анализ подобных производных может сказать нам, какие функции являются выпуклыми, а какие нет. Здесь мы не будем углубляться. Но как только было установлено, что наша функция потерь является выпуклой, мы знаем, что можем доверять такой процедуре, как *поиск с градиентным спуском* (gradient descent search), который возвращает нас к глобальному оптимуму, спускаясь вниз.

9.4.2. Поиск с градиентным спуском

Мы можем найти минимумы выпуклой функции, просто начав с произвольной точки и поэтапно двигаясь вниз. Есть только одна точка, где нет пути вниз: сам глобальный минимум. И именно эта точка определяет параметры наиболее подходящей линии регрессии.

Но как мы можем найти направление, которое ведет нас вниз по склону? Опять же, давайте сначала рассмотрим случай с одной переменной, когда мы ищем наклон w_1 наиболее подходящей линии, где $w_0 = 0$. Предположим, что наш текущий кандидат на наклон равен x_0. В этих ограниченных одномерных условиях мы можем двигаться только влево или вправо. Попробуйте сделать небольшой шаг в каждом направлении, т.е. значения $x_0 - \varepsilon$ и $x_0 + \varepsilon$. Если значение $J(0, x_0 - \varepsilon) < J(0, x_0)$, то мы должны двигаться влево, чтобы спуститься вниз. Если $J(0, x_0 + \varepsilon) < J(0, x_0)$, то мы должны двигаться вправо. Если ни то, ни другое не соответствует действительности, это значит, что нам некуда идти, чтобы уменьшить J, поэтому мы, должно быть, нашли минимум.

Направление вниз в точке $f(x_0)$ определяется наклоном касательной в этой точке. Положительный наклон означает, что минимумы должны лежать слева, а отрицательный наклон — справа. Величина наклона описывает крутизну этой капли: насколько $J(0, x_0 - \varepsilon)$ будет отличаться от $J(0, x_0)$?

Этот наклон может быть аппроксимирован при нахождении уникальной линии, которая проходит через точки $(x_0, J(0, x_0))$ и $(x_0, J(0, x_0 - \varepsilon))$, как показано на рис. 9.8. Это именно то, что делается при вычислении производной, которая в каждой точке указывает касательную к кривой.

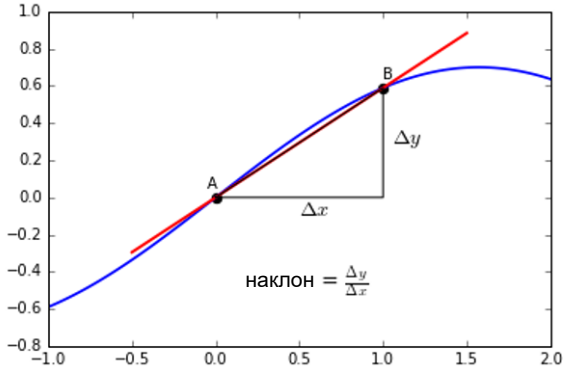


Рис. 9.8. Касательная линия аппроксимирует производную в точке

Когда мы выходим за пределы одного измерения, мы получаем свободу движения в большем диапазоне направлений. Диагональные перемещения позволяют задействовать сразу несколько измерений. Но, в принципе, мы можем получить тот же эффект, сделав несколько шагов вдоль каждого отдельного измерения в осевом направлении. Подумайте о сетке улиц Манхэттена, где мы можем попасть куда угодно, двигаясь по комбинации шагов на север-юг и восток-запад. Нахождение этих направлений требует вычисления *частной производной* целевой функции по каждому измерению, а именно:

$$\frac{\partial}{\partial w_j} = \frac{2}{\partial w_j}\frac{1}{2n}\sum_{i=1}^{n}(f(x_i)-b_i)^2 = \frac{2}{\partial w_j}\frac{1}{2n}\sum_{i=1}^{n}(w_0+(w_1x_i)-b_i)^2$$

Но зигзаги по размерностям кажутся медленными и неуклюжими. Как и Супермен, мы хотим прыгать через здания сразу за один проход. Величина частных производных определяет крутизну в каждом направлении, а результирующий вектор (скажем, три шага на запад для каждого шага на север) определяет самый быстрый путь вниз от этой точки.

Поиск градиентного спуска в двух измерениях

Повторите до схождения {

$$w_0^{t+1} := w_0^t - \alpha\frac{\partial}{\partial w_0}J(w_0^t, w_1^t)$$

$$w_1^{t+1} := w_1^t - \alpha\frac{\partial}{\partial w_1}J(w_0^t, w_1^t)$$

}

Рис. 9.9. Псевдокод для регрессии при поиске градиентного спуска. Переменная t обозначает номер итерации вычисления

9.4.3. Какова правильная скорость обучения?

Производная функции потерь указывает нам правильное направление к минимумам, которые определяют параметры для решения нашей регрессионной задачи. Но это не говорит нам, как далеко идти. С расстоянием значение этого направления уменьшается. Это правда, что самый быстрый способ добраться до Майами из Нью-Йорка — это отправиться на юг, но в какой-то момент вам понадобятся более подробные инструкции.

Поиск по градиентному спуску происходит по кругу: найдите лучшее направление, сделайте шаг, а затем повторяйте, пока мы не достигнем цели. Размер нашего шага называется *скоростью обучения* (learning rate), и он определяет скорость, с которой мы находим минимумы. Делая крошечные детские шаги и постоянно обращаясь к карте (т.е. частным производным), мы действительно попадем туда, куда нужно, но только очень медленно.

Однако больше — не всегда лучше. Если скорость обучения слишком высока, мы можем перепрыгнуть через минимумы, как показано на рис. 9.10 (справа). Это может означать медленное продвижение к минимуму, когда мы отскакиваем от него на каждом шаге, или даже отрицательный прогресс, когда мы получаем значение $J(w)$ выше, чем там, где мы были раньше.

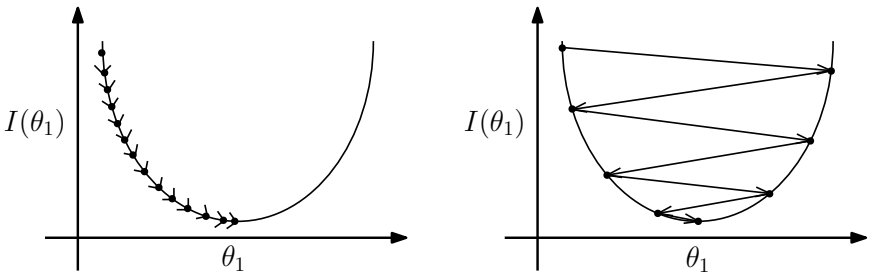


Рис. 9.10. Эффект скорости/размера шага обучения. При слишком маленьком размере шага требуется много итераций для сходимости, в то время как слишком большой размер шага приводит к перескакиванию через минимум

В принципе, мы хотим иметь высокую скорость обучения в начале поиска, но она должна уменьшаться по мере приближения к нашей цели. В ходе оптимизации необходимо отслеживать значение нашей функции потерь. Если процесс становится слишком медленным, мы можем увеличить размер шага на мультипликативный коэффициент (скажем, 3) или отказаться: принимать текущие значения параметров для нашей линии соответствия как достаточно хорошие. Но если значение $J(w)$ увеличивается, это означает, что мы перешагнули

нашу цель. Таким образом, размер нашего шага был слишком большим, поэтому мы должны уменьшить скорость обучения с помощью мультипликативного коэффициента: скажем, на 1/3.

Детали этого беспорядочные, эвристические и специальные. Но, к счастью, библиотечные функции для поиска градиентного спуска имеют встроенные алгоритмы для настройки скорости обучения. Предположительно эти алгоритмы были тщательно настроены и обычно должны делать то, что должны делать.

Но форма поверхности имеет большое значение для того, насколько успешно поиск градиентного спуска находит глобальный минимум. Если бы наша чашеобразная поверхность была относительно ровной, как пластина, то действительно самая низкая точка могла бы быть скрыта облаком шума и числовой погрешностью. Даже если мы в конце концов найдем минимум, это может занять очень много времени.

Тем не менее еще хуже, когда наша функция потерь не выпуклая, т.е. возможно несколько локальных минимумов, как на рис. 9.11. Теперь это не может иметь место для линейной регрессии, но действительно происходит у многих других интересных проблем машинного обучения, с которыми мы столкнемся.

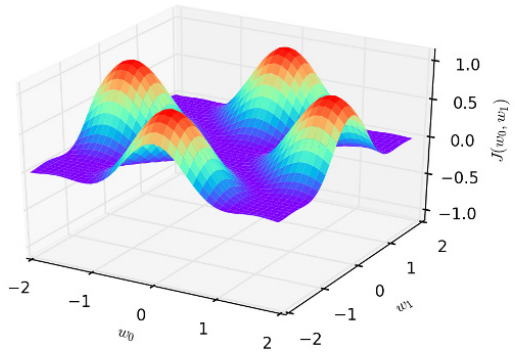


Рис. 9.11. Поиск по градиентному спуску находит локальные минимумы для невыпуклых поверхностей, но не гарантирует глобально оптимального решения

Локальная оптимизация может легко застрять в локальных минимумах невыпуклых функций. Предположим, мы хотим добраться до вершины лыжного склона из нашего дома в долине. Если мы начнем с того, что подойдем на второй этаж дома, мы попадем в ловушку навсегда, если не будет какого-то механизма, позволяющего сделать шаги назад, чтобы освободить нас от местных оптимумов. В этом ценность поисковых эвристик, таких как имитация отжига, которая дает выход из небольших локальных оптимумов, чтобы мы продвигались к глобальной цели.

На заметку. Поиск градиентного спуска остается полезным на практике для невыпуклой оптимизации, хотя он больше не гарантирует оптимального решения. Вместо этого имеет смысл начать несколько раз с разных отправных точек и использовать лучшие локальные минимумы, которые мы найдем, чтобы определить наше решение.

9.4.4. Стохастический градиентный спуск

Алгебраическое определение нашей функции потерь скрывает нечто очень дорогое:

$$\frac{\partial}{\partial w_j} = \frac{2}{\partial w_j}\frac{1}{2n}\sum_{i=1}^{n}(f(x_i)-b_i)^2 = \frac{2}{\partial w_j}\frac{1}{2n}\sum_{i=1}^{n}(w_0+(w_1x_i)-b_i)^2.$$

Это суммирование. Чтобы вычислить наилучшее направление и скорость изменения для каждого измерения j, мы должны пройти через *все* n наших учебных точек. Оценка каждой частной производной требует времени, линейно пропорционального числу примеров, для каждого шага! Для линейной регрессии в нашем обширном наборе данных такси это означает 80 миллионов вычислений квадратов разностей, чтобы определить абсолютное наилучшее направление для продвижения на шаг вперед к цели.

Это безумие. Вместо этого мы можем попробовать приближение, которое использует только небольшое количество примеров для оценки производной, и надеяться, что результирующее направление действительно указывает вниз. В среднем так и должно быть, поскольку каждый пункт в итоге получит право голоса при выборе направления.

Стохастический градиентный спуск (stochastic gradient descent) — это оптимизационный подход, основанный на выборке небольшой партии учебных точек, в идеале случайных, и использовании их для оценки производной в нашей текущей позиции. Чем меньше размер пакета, который мы используем, тем быстрее будет оценка, хотя мы должны скептически относиться к правильному расчетному направлению. Оптимизация скорости обучения и размера пакета для градиентного спуска приводит к очень быстрой оптимизации для выпуклых функций, причем детали благословенно скрываются при вызове библиотечной функции.

Случайный выбор на каждом этапе поиска может быть очень дорог. Лучше один раз рандомизировать порядок обучающих примеров, чтобы избежать систематических артефактов в том, как они представлены, а затем составить наши партии, просто пройдя по списку. Таким образом, мы можем гарантировать,

что все n наших обучающих экземпляров в конечном итоге вносят свой вклад в поиск, в идеале несколько раз, так как мы постоянно просматриваем все примеры в ходе оптимизации.

9.5. Упрощение моделей с помощью регуляризации

Линейная регрессия рада определить *наилучшее возможное* линейное соответствие любой совокупности из n точек данных, каждая из которых определяется $m - 1$ независимыми переменными и заданным целевым значением. Но "лучшее" соответствие может быть не тем, что мы *действительно* хотим.

В этом и проблема. Большинство из $m - 1$ возможных признаков могут быть не связаны с целью и, таким образом, не имеют реальной предсказательной силы. Как правило, они будут отображаться как переменные с небольшими коэффициентами. Однако алгоритм регрессии будет использовать эти значения, чтобы подтолкнуть линию и уменьшить ошибку наименьших квадратов в приведенных обучающих примерах. Использование шума (некоррелированные переменные) для подгонки (остаток от простой модели по действительно коррелированным переменным) вызывает проблемы.

В качестве примера можно привести наш опыт использования модели чаевых в такси, подробно описанной в разделе "Случай из жизни". Модель полной регрессии с использованием десяти переменных имела среднеквадратическую ошибку 1,5448. Модель регрессии с одной переменной, работающая только на плате за проезд, работала немного хуже, с ошибкой 1,5487. Но эта разница просто шум. Модель с одной переменной, очевидно, лучше согласно бритве Оккама или кого-либо еще.

При использовании неограниченной регрессии возникают другие проблемы. Мы видели, как сильно коррелированные функции вносят неоднозначность в модель. Если характеристики A и B идеально коррелируют, использование обоих дает ту же точность, что и любой из них, что приводит к более сложным и менее интерпретируемым моделям.

Предоставление богатого набора функций регрессии — это хорошо, но помните, что "лучшее объяснение — самое простое". Самое простое объяснение опирается на наименьшее количество переменных, которые хорошо справляются с моделированием данных. В идеале наша регрессия выбрала бы наиболее важные переменные и соответствовала бы им, но обсуждаемая нами целевая функция только пытается минимизировать ошибку суммы квадратов. Нам нужно изменить нашу целевую функцию, используя магию регуляризации.

9.5.1. Гребневая регрессия

Регуляризация (regularization) — это трюк с добавлением вторичных членов к целевой функции для предпочтения моделей, в которых коэффициенты сохраняются небольшими. Предположим, мы обобщаем нашу функцию потерь со вторым набором членов, которые являются функцией коэффициентов, а не обучающих данных:

$$J(w) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda\sum_{j=1}^{m} w_j^2.$$

В этой формуле мы платим штраф (pay a penalty), пропорциональный сумме квадратов коэффициентов, используемых в модели. Возводя коэффициенты в квадрат, мы игнорируем знак и фокусируемся на величине. Константа λ определяет относительную силу ограничений регуляризации. Чем выше λ, тем сложнее будет работать оптимизация, чтобы уменьшить размер коэффициента за счет увеличения остатков. В конечном итоге становится более целесообразным установить коэффициент некоррелированной переменной в нуль, а не использовать его для переобучения обучающего набора.

Наложение штрафа в размере суммы квадратов коэффициентов, как в приведенной выше функции потерь, называется *гребневой регрессией* (ridge regression) или *регуляризацией Тихонова* (Tikhonov regularization). Предполагая, что все зависимые переменные были должным образом нормализованы, чтобы означать нуль, их величина коэффициента является мерой их значения для целевой функции.

Как мы можем оптимизировать параметры гребневой регрессии? Естественное продолжение формулировки наименьших квадратов делает свою работу. Пусть G будет нашей матрицей $n \times n$ "весовых коэффициентов штрафа" (coefficient weight penalty). Для простоты пусть G = I, единичная матрица. Функция потерь суммы квадратов, которую мы стремимся минимизировать, становится такой

$$|| Aw - b ||^2 + || \lambda \Gamma w ||^2 .$$

Форма записи $|| v ||$ обозначает *норму* v, функцию расстояния для вектора или матрицы. Норма $||\Gamma w||^2$ в точности равна сумме квадратов коэффициентов, когда $\Gamma = I$. При таком подходе правдоподобной замкнутой формой для оптимизации w можно считать

$$w = (A^T A + \lambda \Gamma^T \Gamma)^{-1} A^T b .$$

Таким образом, уравнение нормальной формы может быть обобщено, чтобы справиться с регуляризацией. Но альтернативно мы можем вычислить частные производные этой функции потерь и использовать поиск по градиентному

спуску, чтобы быстрее выполнять работу на больших матрицах. В любом случае библиотечные функции для гребневой регрессии и ее двоюродного брата регрессии LASSO будут легко доступны для использования в вашей задаче.

9.5.2. Регрессия LASSO

Гребневая регрессия оптимизирует выбор небольших коэффициентов. Благодаря функции суммы квадратов стоимости она особенно штрафует самые большие коэффициенты. Это позволяет избежать моделей вида $y = w_0 + w_1 x_1$, где w_0 — большое положительное число, а w_1 — большое отрицательное число.

Хотя гребневая регрессия эффективна при уменьшении величины коэффициентов, эти критерии на самом деле не сводят их к нулю и полностью исключают переменную из модели. Альтернативный подход здесь заключается в том, чтобы попытаться минимизировать сумму абсолютных значений коэффициентов, которая так же склонна снижать наименьшие коэффициенты, как и большие.

Регрессия LASSO (LASSO regression, Least Absolute Shrinkage and Selection Operator) вполне соответствует этому критерию: минимизировать метрику L_1 по коэффициентам вместо метрики L_2 гребневой регрессии. В LASSO мы указываем явное ограничение t того, какой может быть сумма коэффициентов, а оптимизация сводит к минимуму ошибку суммы квадратов при этом ограничении:

$$J(w,t) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - f(x_i))^2 \text{ с учетом } \sum_{j=1}^{m}\left|w_j\right| \le t.$$

Задание меньшего значения t сжимает LASSO, дополнительно ограничивая величины коэффициентов w.

В качестве примера того, как LASSO обнуляет малые коэффициенты, рассмотрим, что она сделала с моделью чаевых такси для определенного значения t:

Переменная	Коэффициент LR	LASSO
(intercept)	0,08370835	0,079601141
duration	0,00000035	0,00000035
distance	0,00000004	0,00000004
fare	0,17503086	0,17804921
tolls	0,06267343	0,00000000
surcharge	0,01924337	0,00000000
weekends	–0,02823731	0,00000000
business day	0,06977724	0,00000000
rush hour	0,01281997	0,00000000
late night	0,04967453	0,00000000
# of passengers	–0,00657358	0,00000000

Как можно заметить, LASSO обнуляет большинство коэффициентов, что приводит к более простой и надежной модели, которая соответствует данным почти так же, как и линейная регрессия без ограничений.

Но почему LASSO активно сводит коэффициенты к нулю? Это связано с формой круга метрики L_1. Как мы увидим на рис. 10.2, форма круга L_1 (набор точек, равноудаленных от начала координат) не является круглой, она имеет вершины и другие низкоуровневые объекты, такие как ребра и грани. Ограничение наших коэффициентов w, лежащих на поверхности радиуса t круга L_1, означает, что он может попасть в одну из этих низкоуровневых функций, т.е. неиспользуемые измерения получат нулевые коэффициенты.

Что работает лучше, LASSO или гребневая регрессия? Ответ зависит от ситуации. Оба метода должны присутствовать в вашей любимой библиотеке оптимизации, поэтому опробуйте каждый из них и посмотрите, что будет.

9.5.3. Компромисс между точностью соответствия и сложностью

Как установить правильное значение для нашего параметра регуляризации, будь то λ или t? Использование достаточно малой λ или достаточно большой t обеспечивает небольшой штраф за выбор коэффициентов, чтобы минимизировать ошибку обучения. Напротив, использование очень большого λ или очень малого t обеспечивает малые коэффициенты, даже за счет существенной ошибки моделирования. Настройка этих параметров позволяет нам найти компромисс между недостаточным и избыточным соответствием.

Оптимизируя эти модели в широком диапазоне значений для соответствующего параметра регуляризации t, мы получаем график ошибки оценки как функцию от t. Хорошее соответствие учебным данным с небольшим/малым параметром является более надежным, чем чуть более точным соответствием по многим параметрам.

Выбор этого компромисса во многом зависит от вкуса. Тем не менее было разработано несколько метрик, чтобы помочь с выбором модели. Наиболее значимыми являются *информационные критерии Акаике* (Akaike Information Criteria — AIC) и *Байеса* (Baysian Information Criteria — BIC). Мы не будем углубляться в них глубже упоминания названия, поэтому на данный момент можете считать эти показатели неким вуду. Но ваша система оптимизации/оценки вполне может выводить их для соответствующих моделей, которые они производят, что позволяет сравнивать модели с различным количеством параметров.

Даже если LASSO/гребневая регрессия штрафует коэффициенты на основе величины, они явно не устанавливают их в нуль, если вы хотите точно k параметров. Вы должны удалять ненужные переменные из вашей модели. Методы автоматического выбора признаков могут решить обнулить малые коэффициенты, но явное построение моделей из всех возможных подмножеств признаков обычно невозможно с вычислительной точки зрения.

В первую очередь должны быть удалены элементы с (а) малыми коэффициентами, (б) низкой корреляцией с целевой функцией, (в) высокой корреляцией с другим признаком в модели и (г) без очевидной оправданной связи с целью. Например, одно известное исследование показало сильную корреляцию между валовым национальным продуктом США и годовым объемом производства масла в Бангладеш. Мудрый разработчик может отклонить эту переменную как нелепую, для автоматических методов это невозможно.

9.6. Классификация и логистическая регрессия

Мы часто сталкиваемся с проблемой присвоения элементам правильной метки в соответствии с предопределенным набором классов.

- Является транспортное средство на изображении легковым или грузовым автомобилем? Является данный образец ткани анализом злокачественной или доброкачественной опухоли?
- Интересно пользователю данное сообщение электронной почты или это спам?
- Анализ социальных сетей направлен на выявление свойств людей по связанным данным. Является данный человек мужчиной или женщиной? Будут ли они склонны голосовать за демократов или республиканцев?

Классификация (classification) — это проблема прогнозирования правильной метки для данной исходной записи. Задача отличается от регрессии тем, что метки являются дискретными объектами, а не значениями непрерывных функций. Попытка выбрать правильный ответ из двух возможных вариантов может показаться проще, чем прогнозирование бесконечных значений, но так также намного проще получить неправильный ответ, совершив ошибку.

В этом разделе будут разработаны подходы к построению систем классификации с использованием линейной регрессии, но это только начало. Классификация — это сложная проблема в науке о данных, и мы увидим несколько других подходов в следующих двух главах.

9.6.1. Регрессия для классификации

Мы можем применить линейную регрессию к задачам классификации, преобразовав имена классов обучающих примеров в числа. А пока давайте ограничимся вниманием к двум классовым задачам или *двоичной классификации*. Мы рассмотрим задачи с несколькими классами в разделе 9.7.2.

Нумерация этих классов как 0/1 прекрасно работает для двоичных классификаторов. По соглашению "положительный" класс получает значение 0, а "отрицательный" — 1:

- мужчина = 0/женщина = 1;
- демократ = 0/республиканец = 1;
- спам = 1/не спам = 0;
- рак = 1/доброкачественная = 0.

Отрицательный/1 класс обычно обозначает более редкий или более частный случай. Здесь нет никакого оценочного суждения, подразумевающего положительное/отрицательное: действительно, когда классы имеют одинаковый размер, выбор делается произвольно.

Мы могли бы рассмотреть обучение линии регрессии $f(x)$ для нашего вектора функций x, где целевыми значениями являются метки 0/1, как показано на рис. 9.12. Здесь есть какая-то логика. Экземпляры, похожие на примеры положительного обучения, должны получить более низкие оценки, чем те, которые ближе к отрицательным. Мы можем ограничить значение, возвращаемое функцией $f(x)$, чтобы интерпретировать его как метку: $f(x) \leq 0{,}5$ означает, что x классифицируется положительно. Когда $f(x) > 0{,}5$, мы вместо этого назначаем отрицательную метку.

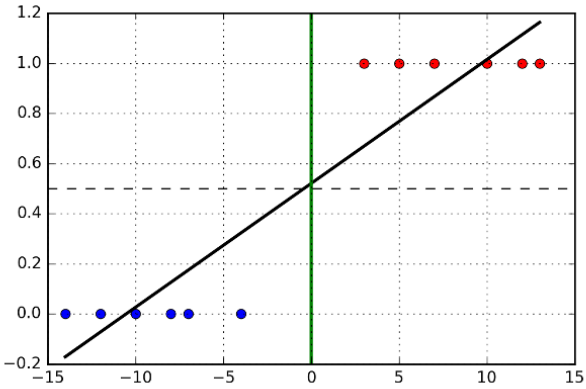


Рис. 9.12. Оптимальная линия регрессии пронизывает классы, даже если существует идеальная разделительная линия ($x = 0$)

Но с этой формулировкой есть проблемы. Предположим, мы добавили ряд "крайне отрицательных" примеров к учебным данным. Линия регрессии сместится в сторону этих примеров, подвергая риску правильную классификацию более правильных примеров. Это прискорбно, поскольку в любом случае мы бы уже правильно классифицировали эти крайне отрицательные моменты. Мы действительно хотим, чтобы линия проходила между классами и служила границей, но не выглядела, как кардиограмма спринтера сразу после забега.

9.6.2. Границы принятия решений

Правильный подход к классификации заключается в разделении пространства объектов на регионы так, что всем точкам в любом данном регионе назначается одна и та же метка. Регионы определяются своими границами, поэтому мы хотим, чтобы регрессия находила разделительные линии вместо соответствия.

На рис. 9.13 (слева) показано, что обучающие примеры для двоичной классификации можно рассматривать как цветные точки в пространстве признаков. Наши надежды на точную классификацию опираются на региональную согласованность между точками. Это значит, что близлежащие точки, как правило, имеют одинаковые метки, и что границы между регионами обычно являются четкими, а не размытыми.

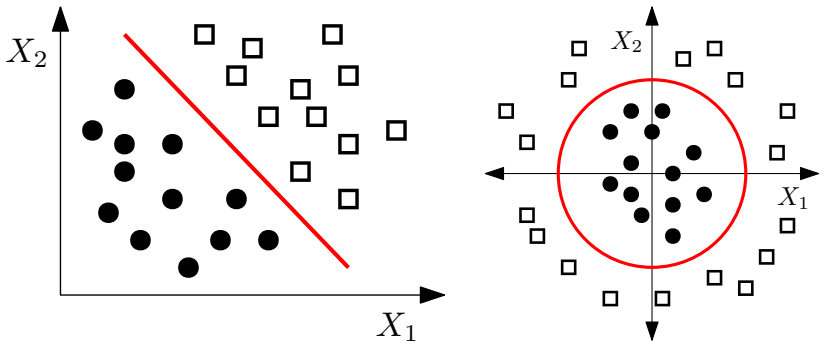


Рис. 9.13. Разделительная линия разграничивает два класса в пространстве признаков (слева). Тем не менее нелинейные разделители лучше подходят для определенных тренировочных наборов (справа)

В идеале наши два класса должны хорошо различаться в пространстве признаков, поэтому линия может легко их разделить. Но в более общем плане будут присутствовать выбросы. Нам нужно судить о нашем классификаторе по "чистоте" результирующего разделения, штрафуя за неправильную классификацию точек, лежащих не на той стороне линии.

Любой набор точек может быть идеально разделен, если мы спроектируем достаточно сложную границу, которая отклоняется и захватывает все экземпляры с данной меткой. Посмотрите на рис. 9.14. Такие сложные разделители обычно отражают перетренировку на учебном наборе. Линейные разделители обладают простотой и надежностью и, как мы увидим, могут быть эффективно созданы с использованием *логистической регрессии* (logistic regression).

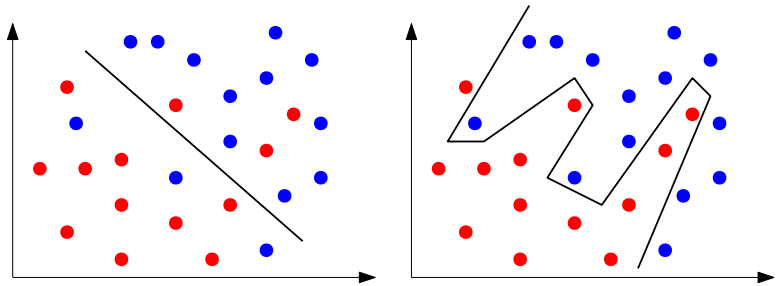


Рис. 9.14. Линейные классификаторы не всегда могут разделить два класса (слева). Тем не менее идеальное разделение, достигаемое с использованием сложных границ, обычно отражает больше переобучение, чем понимание (справа)

В более общем плане нас могут интересовать нелинейные, но не сложные границы решения, если они лучше разделяют границы классов. Идеальная разделяющая кривая на рис. 9.13 (справа) — это не линия, а круг. Однако его можно найти как линейную функцию квадратичных функций, таких как x_1^2 и $x_1 x_2$. Мы можем использовать логистическую регрессию для нахождения нелинейных границ, если матрица данных заполнена нелинейными элементами, как обсуждалось в разделе 9.2.2.

9.6.3. Логистическая регрессия

Вспомним логит-функцию $f(x)$, которую мы ввели в разделе 4.4.1:

$$f(x) = \frac{1}{1+e^{-cx}}.$$

Эта функция принимает на входе действительное значение $-\infty \le x \le \infty$ и выдает значение в диапазоне [0,1], т.е. вероятность. На рис. 9.15 показана логит-функция $f(x)$ — сигмоидальная кривая, плоская с обеих сторон, но с крутым подъемом в середине.

Форма логит-функции делает ее особенно подходящей для интерпретации границ классификации. В частности, пусть x будет оценкой, которая отражает

расстояние, на котором конкретная точка p лежит выше/ниже или слева/справа от линии l, разделяющей два класса. Мы хотим, чтобы $f(x)$ измеряла вероятность того, что p заслуживает отрицательной метки.

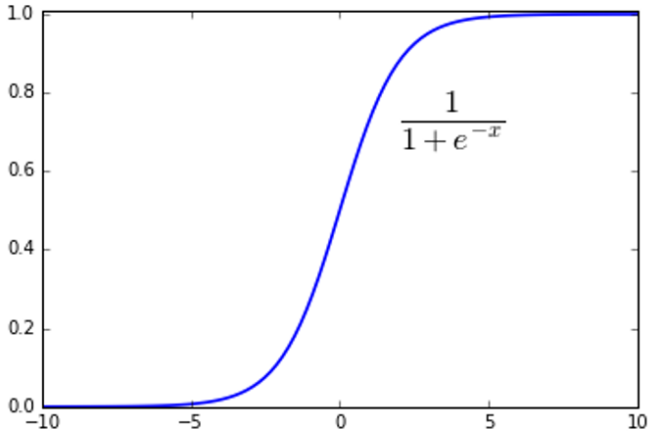


Рис. 9.15. Логит-функция соотносит оценку с вероятностью

Логит-функция соотносит оценки с вероятностью, используя только один параметр. Важными являются случаи в средней и конечных точках. Когда логит-функция говорит, что $f(0) = 1/2$, это значит, что метка точки лежит на границе, по сути, это монета, упавшая на ребро. Это так и должно быть. Чем более однозначное решение может быть принято, тем больше наше расстояние от этой границы, поэтому $f(\infty) = 1$ и $f(-\infty) = 0$.

Наша уверенность как функция расстояния модулируется масштабирующей константой c. Значение c, близкое к нулю, дает очень постепенный переход от положительного к отрицательному. Напротив, мы можем превратить логит-функцию в лестничную ступеньку, назначив достаточно большое значение c, что означает, что небольшие расстояния от границы приводят к значительному увеличению достоверности классификации.

Чтобы эффективно использовать логит-функцию для классификации, нужны три вещи.

- Расширение $f(x)$ за пределы одной переменной до полного $(m - 1)$-мерного входного вектора x.
- Пороговое значение t устанавливает среднюю точку распределения оценок (здесь нуль).
- Значение постоянной масштабирования c, регулирующей крутизну перехода.

Мы можем достичь всех трех, подбирая линейную функцию $h(x, w)$ к данным, где

$$h(x, w) = w_0 + \sum_{i=1}^{m-1} w_i x_i$$

может быть затем применен к логистической функции для получения классификатора:

$$f(x) = \frac{1}{1 + e^{-h(x,w)}}.$$

Обратите внимание, что коэффициенты $h(x, w)$ достаточно богаты, чтобы задать пороговые значения ($t = w_0$) и крутизну (c — это, по сути, среднее значение от w_1 до w_{m-1}) параметров.

Единственный оставшийся вопрос — как подогнать вектор коэффициента w к обучающим данным. Напомним, что для каждого входного вектора x_i задана метка класса y_i типа нуль/один, где $1 \le i \le n$. Нам нужна штрафная функция, которая назначает соответствующие штрафы результату $f(x_i)$ как вероятности того, что класс y_i является положительным, т.е. $y_i = 1$.

Давайте сначала рассмотрим случай, когда y_i *на самом деле* равен 1. В идеале $f(x_i) = 1$, поэтому мы хотим оштрафовать его за то, что он меньше 1. Действительно, мы хотим оштрафовать его агрессивно, когда $f(y_i) \to 0$, поскольку это означает, что классификатор заявляет: у элемента i мало шансов быть в классе 1, когда это действительно так.

Логарифмическая функция $cost(x_i, 1) = -\log f(x_i)$ оказывается хорошей штрафной функцией, когда $y_i = 1$. Напомним определение логарифма (или обратной экспоненциальной функции) из раздела 2.4, а именно:

$$y = \log_b x \to b^y = x.$$

Как показано на рис. 9.16, $\log 1 = 0$ для любого разумного основания, поэтому при $f(x_i) = 1$ начисляется нулевой штраф, как и должно быть для правильной идентификации $y_i = 1$. Начиная с $b^{\log_b x} = x$, $\log x \to -\infty$ при $x \to 0$. Это делает $cost(x_i, 1) = -\log f(x_i)$ все более суровым штрафом, чем больше мы неверно классифицируем y_i.

Теперь рассмотрим случай, когда $y_i = 0$. Мы хотим штрафовать классификатор за высокие значения $f(x_i)$, т.е. больше чем $f(x_i) \to 1$. Небольшое размышление должно убедить вас, что правильный штраф теперь $cost(x_i, 0) = -\log(1 - f(x_i))$.

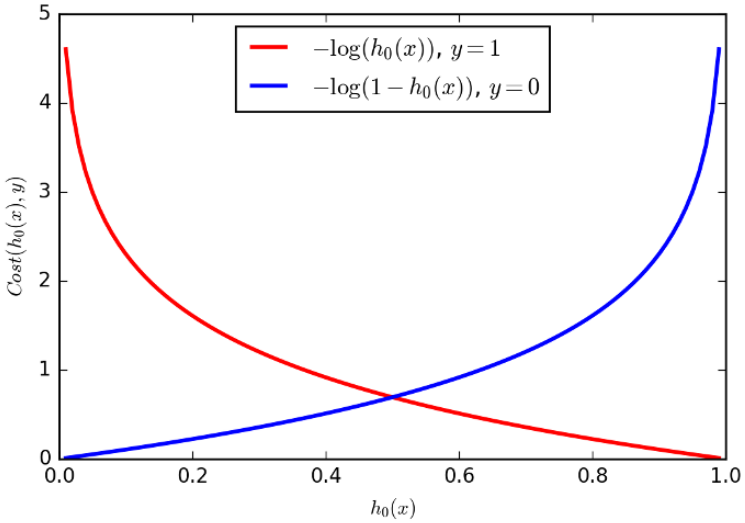


Рис. 9.16. Стоимость штрафов за положительный (справа) и отрицательный (слева) элементы. Штраф равен нулю, если правильная метка назначена с вероятностью 1, но увеличивается как функция неоправданной уверенности

Чтобы связать все вместе, обратите внимание, что происходит, когда мы умножаем $cost(x_i, 1)$ на y_i. Есть только два возможных значения, а именно $y_i = 0$ или $y_i = 1$. Это имеет желаемый эффект, поскольку штраф обнуляется в случае, если он не применяется. Аналогично умножение на $(1-y_i)$ имеет противоположный эффект: обнуление штрафа при $y_i = 1$ и применение его при $y_i = 0$. Умножение *трат* (*cost*) на соответствующие индикативные переменные позволяет нам определить функцию потерь для логистической регрессии как алгебраическую формулу:

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} cost(f(x_i, w), y_i) =$$

$$= -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log f(x_i, w) + (1-y_i)\log(1-f(x_i, w))\right].$$

Замечательной особенностью этой функции потерь является то, что она выпуклая, а это означает, что мы можем найти параметры w, которые лучше всего соответствуют обучающим примерам, используя градиентный спуск. Таким образом, мы можем использовать логистическую регрессию, чтобы найти лучший линейный разделитель между двумя классами, обеспечивая естественный подход к двоичной классификации.

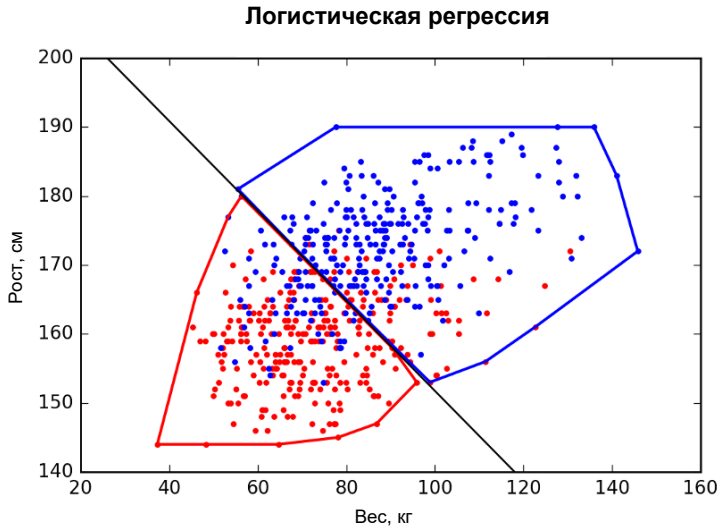


Рис. 9.17. Классификатор логистической регрессии лучше всего разделяет мужчин и женщин в пространстве вес-рост. Красная область (ниже прямой) содержит 229 женщин и только 63 мужчины, в то время как синяя область (выше прямой) содержит 223 мужчины и 65 женщин

9.7. Проблемы логистической классификации

В построении эффективных классификаторов есть несколько нюансов, а также проблем, имеющих отношение как к логистической регрессии, так и к другим методам машинного обучения, которые мы рассмотрим в следующих двух главах. К ним относятся управление несбалансированными размерами классов, многоклассовая классификация и построение истинных распределений вероятностей из независимых классификаторов.

9.7.1. Сбалансированные учебные классы

Рассмотрим следующую проблему классификации, которая представляет большой интерес для правоохранительных органов любой страны. Учитывая данные о конкретном человеке p, решите, является он террористом или нет.

Качество доступных вам данных в конечном итоге определит точность вашего классификатора, но тем не менее в этой проблеме есть кое-что, что делает ее очень сложной. Дело в том, что в общей численности населения недостаточно террористов.

В Соединенных Штатах мы наделены общим миром и безопасностью. Меня не удивило бы, если бы во всей стране было всего около 300 настоящих террористов. В стране с населением 300 миллионов человек это означает, что только один человек из каждого миллиона является активным террористом.

Есть два основных следствия из этого дисбаланса. Первое, *любой* значимый классификатор обречен на множество ложных срабатываний. Даже если бы наш классификатор оказался верным в 99,999% случаев, он бы классифицировал 3000 ни в чем не повинных людей как террористов, что в десять раз превышает количество плохих парней, которых мы поймаем. Подобные вопросы обсуждались в разделе 7.4.1, когда речь шла о точности и отзыве.

Второе следствие этого дисбаланса заключается в том, что не может быть многих примеров реальных террористов, на которых можно обучаться. У нас могли бы быть десятки тысяч невинных людей, которые могли бы служить положительными примерами (классом 0), но только несколько десятков известных террористов были бы отрицательными примерами (класса 1).

Подумайте, что будет делать логистический классификатор в таком случае. Даже неправильная классификация *всех* террористов как невинных не может внести слишком большой вклад в функцию потерь, по сравнению со стоимостью того, как мы относимся к большему классу. Скорее всего, он проведет разделительную линию так, чтобы оправдать всех, и не охотиться на террористов. Мораль здесь в том, что лучше использовать одинаковое количество положительных и отрицательных примеров.

Но для одного класса может быть трудно найти примеры. Итак, каковы наши варианты по созданию лучшего классификатора?

- *Формирование сбалансированных классов за счет отбрасывая членов большего класса.* Это самый простой способ реализации сбалансированных учебных классов. Это вполне оправданно, если у вас достаточно элементов редкого класса, чтобы построить достойный классификатор. Отбрасывая лишние экземпляры, которые нам не нужны, мы создаем более сложную проблему, которая не подходит для большинства классов.
- *Репликация элементов меньшего класса, в идеале с вариациями.* Простой способ получить больше обучающих примеров — это клонировать террористов, вставляя их точные копии в учебный набор под разными именами. Эти повторяющиеся примеры, в конце концов, вполне *похожи* на террористов, и если добавить их достаточно, то классы будут сбалансированы. Но это хрупкая формула. Такие идентичные записи данных могут создавать числовую нестабильность и, безусловно, иметь тенденцию к переобучению, поскольку перемещение одного дополнительного настоящего террориста в правую сторону от границы также перемещает всех его

клонов. Возможно, было бы лучше добавить определенное количество случайных шумов к каждому клонированному примеру, что согласуется с дисперсией в общей популяции. Это утяжеляет работу классификатора по их поиску, и, таким образом, сводит переобучение к минимуму.

- *Взвесить редкие учебные примеры тяжелее, чем экземпляры большего класса*. Функция потерь для оптимизации параметров содержит отдельный член для ошибки каждого учебного экземпляра. Добавление коэффициента для придания большего веса наиболее важным экземплярам оставляет проблему выпуклой оптимизации, поэтому его все еще можно оптимизировать с помощью стохастического градиентного спуска.

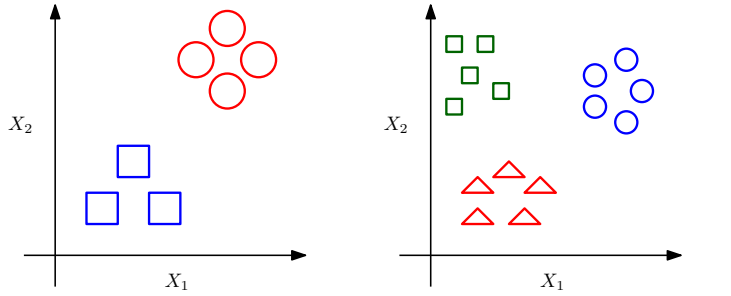


Рис. 9.18. Проблемы мультиклассовой классификации являются обобщением двоичной классификации

Проблема всех этих трех решений заключается в том, что мы смещаем классификатор, изменяя базовое распределение вероятностей. Важно уведомить классификатор о том, что террористы крайне редки среди населения, возможно, задав предварительное распределение по Байесу.

Конечно, лучшим решением было бы собрать больше учебных примеров из более редкого класса, но это не всегда возможно. Эти три подхода — лучшее из того, что мы можем использовать в качестве альтернативы.

9.7.2. Мультиклассовая классификация

Зачастую задачи классификации включают выбор из более чем двух разных меток. Рассмотрим проблему определения жанра данного фильма. Логические возможности включают *драму*, *комедию*, *анимацию*, *боевик*, *документальный* и *музыкальный*.

Естественный, но ошибочный подход к представлению k различных классов — добавить несколько классов, кроме 0/1. В задаче классификации цвета волос, возможно, мы могли бы присвоить *blond* = 0 (белый), *brown* = 1 (каштановый), *red* = 2 (рыжий), *black* = 4 (черный) и так далее, пока не исчерпаем

человеческие различия. Тогда мы могли бы использовать линейную регрессию, чтобы предсказать количество классов.

Но вообще это плохая идея. *Порядковые* шкалы (ordinal scale) определяются увеличением либо уменьшением значений. Если порядок ваших классов не отражает порядковую шкалу, нумерация классов будет бессмысленной задачей для регрессии.

Рассмотрим нумерацию цвета волос выше. Должны ли *рыжие* волосы лежать между *каштановыми* и *черными* (как определено в настоящее время) или между *белыми* и *каштановыми*? *Седые* волосы — это светлый оттенок *белого*, скажем, класс –1, или это не подлежащее сравнению состояние, главным образом из-за возраста? Предположительно факторы, способствующие появлению седых волос (возраст и количество детей подросткового возраста), полностью ортогональны таковым для белых волос (воздействие парикмахерской и североевропейское происхождение). Если это так, то нет никакой системы линейной регрессии, которая соответствовала бы цвету волос в качестве непрерывной переменной и была бы предназначена для отделения этих цветов от более темных волос.

Некоторые наборы классов правильно определяются порядковыми шкалами. Рассмотрим, например, такие классы, которые образуются при оценке людьми при таких опросах, как "На курсе Скиены слишком много работы" или "Сколько звезд вы даете этому фильму?"

Полностью согласен ↔ В основном согласен ↔
↔ Нейтрально ↔ В основном не согласен ↔ Полностью не согласен

Четыре звезды ↔ Три звезды ↔ Две звезды ↔ Одна звезда ↔ Ноль звезд

Классы, определяемые такими *шкалами Ликерта* (Likert scale), являются порядковыми, а следовательно, такие номера классов являются вполне разумной вещью, к которой применима регрессия. В частности, ошибочное присвоение элемента соседнему классу является гораздо меньшей проблемой, чем присвоение его неправильному концу шкалы.

Но, по правде говоря, метки классов не являются порядковыми. Лучшая идея по распознаванию нескольких классов заключается в создании множества классификаторов "один против всех", как показано на рис. 9.19. Для каждого из возможных классов C_i, где $1 \leq i \leq c$, мы обучаем логистический классификатор так, чтобы отличать элементы C_i от объединения элементов из всех других классов вместе взятых. Чтобы идентифицировать метку, связанную с новым элементом x, мы проверяем его по всем c этих классификаторов и возвращаем метку i, которая имеет наибольшую ассоциированную вероятность.

Этот подход должен казаться простым и разумным, но учтите, что проблема классификации усложняется с увеличением количества классов. Рассмотрим

обезьяну. Подбрасывая монету, обезьяна должна быть в состоянии правильно обозначить 50% примеров в любой проблеме двоичной классификации. Но теперь предположим, что есть сто классов. Обезьяна будет правильно угадывать только в 1% случаев. Задача сейчас очень сложная, и даже отличному классификатору будет очень трудно добиться хороших результатов.

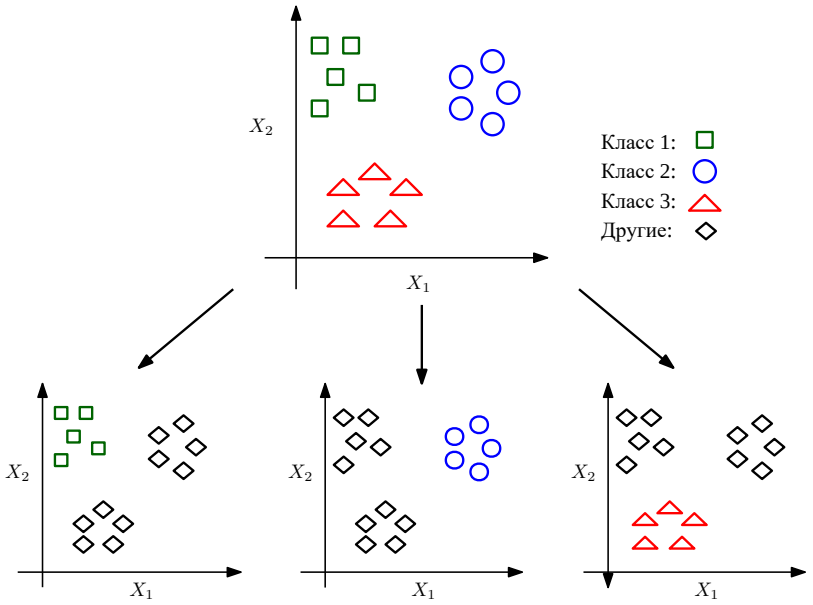


Рис. 9.19. Выборы между несколькими классификаторами "один против остальных" — это, как правило, лучший способ осуществить классификацию для нескольких классов

9.7.3. Иерархическая классификация

Когда ваша задача подразумевает большое количество классов, имеет смысл сгруппировать их в дерево или иерархию, чтобы повысить как точность, так и эффективность. Предположим, мы построили двоичное дерево, где каждая отдельная категория представлена листовым узлом. Каждый внутренний узел представляет собой классификатор для различения левых потомков и правых потомков.

Чтобы использовать эту иерархию для классификации нового элемента x, мы начнем с корня. Запуск корневого классификатора для x определит его как принадлежность к левому или правому поддереву. Двигаясь вниз на один уровень, мы сравниваем x с классификатором нового узла и продолжаем повторять так, пока не достигнем листа, который определяет метку, присвоенную x. Требуемое время пропорционально высоте дерева, и в идеале логарифмически пропорционально количеству классов c, а не линейно пропорционально, если мы явно сравниваем с каждым классом. Основанные на этом подходе

классификаторы являются *деревьями решений* (decision tree) и будут обсуждаться в разделе 11.2.

В идеале эта иерархия может быть построена из знаний предметной области, гарантируя, что категории, представляющие сходные классы, сгруппированы вместе. Это имеет два преимущества. Во-первых, это повышает вероятность того, что в результате неправильной классификации будут по-прежнему производиться ярлыки из аналогичных классов. Во-вторых, это означает, что промежуточные узлы способны определять понятия более высокого порядка, которые могут быть распознаны более точно. Предположим, что сто категорий в задаче классификации изображений включали "автомобиль", "грузовик", "лодка" и "велосипед". Когда все эти категории являются потомками промежуточного узла, называемого "транспортным средством", мы можем интерпретировать путь к этому узлу как классификатор с более низким разрешением и более высокой точностью.

Существует еще одна независимая опасность с классификацией, которая становится все острее по мере роста количества классов. Члены определенных классов (например, "студенты колледжа") гораздо более многочисленны, чем другие, например "рок-звезды". Относительное расхождение между размерами самых больших и самых маленьких классов обычно увеличивается вместе с количеством классов.

Для этого примера давайте согласимся, что "рок-звезды", как правило, угрюмые, запущенные мужчины, что является полезным признаком для любого классификатора. Однако только небольшая часть угрюмых, запущенных мужчин являются рок-звездами, поскольку очень мало людей, которые преуспели в этой требовательной профессии. Системы классификации, которые не имеют надлежащего смысла предварительного распределения по меткам, обречены на множество ложных срабатываний, поскольку слишком редко назначают редкие метки.

В этом суть *байесовского анализа* (Baysian analysis): обновление нашего текущего (предшествующего) понимания распределения вероятностей перед лицом новых доказательств. Здесь доказательством является результат классификатора. Если мы включим обоснованное предварительное распределение в наши рассуждения, мы сможем гарантировать, что элементы требуют особенно веских доказательств, чтобы их можно было отнести к редким классам.

9.7.4. Функции разбиения и полиномиальная регрессия

Напомним, что к нашим предпочтительным средствам мультиклассовой классификации относилось обучение независимых классов вместо всех логи-

стических классификаторов $F_i(x)$, где $1 \le i \le c$, а c — это количество различных меток. Осталась одна небольшая проблема. Вероятности, которые мы получаем из логистической регрессии, на самом деле не являются вероятностями. Превращение их в реальные вероятности требует концепции *функции разбиения* (partition function).

Для любого конкретного элемента x суммирование "вероятностей" по всем возможным меткам для x *должно* привести к $T = 1$, где

$$T = \sum_{i=1}^{c} F_i(x).$$

Но это не значит, что так и есть. Все эти классификаторы были обучены независимо, а следовательно, ничто не заставляет их суммировать до $T = 1$.

Решение заключается в том, чтобы разделить все эти вероятности на соответствующую константу, а именно $F'(x) = F(x)/T$. Это может звучать как бред, поскольку это так. Но именно это и делают физики, когда говорят о функциях разбиения, которые служат знаменателями, превращающими нечто пропорциональное вероятностям в реальные вероятности.

Полиномиальная регрессия (multinomial regression) является более принципиальным методом обучения независимых классов вместо всех классификаторов, так что вероятности работают правильно. Это включает в себя использование правильной функции разделения для коэффициентов логарифмов, которые вычисляются с экспонентами результирующих значений. Более того, имеет смысл поискать функцию полиномиальной регрессии в вашей любимой библиотеке машинного обучения и посмотреть, как она работает, столкнувшись с проблемой регрессии нескольких классов.

Понятие, связанное с функцией разбиения, возникает в анализе Байеса. Мы часто сталкиваемся с проблемой определения наиболее вероятной маркировки предмета, скажем, A, в зависимости от доказательств E. Напомним, что теорема Байеса утверждает, что

$$P(A|E) = \frac{P(E|A)P(A)}{P(E)}.$$

Вычисление этого как реальной вероятности требует знания знаменателя $P(E)$, который может быть довольно сложно вычисляемым. Однако сравнение $P(A|E)$ с $P(B|E)$ для определения того, является ли метка A более вероятной, чем метка B, не требует знания $P(E)$, поскольку оно одинаково в обоих выражениях. Как физики, мы можем отказаться от него, бормоча о "функции разбиения".

9.8. Дополнительная информация

Линейная и логистическая регрессия являются стандартными темами в статистике и оптимизации. К учебникам по линейной/логистической регрессии и ее приложениям относятся [21, 79].

Эндрю Ын вдохновлял подход градиентного спуска к решению регрессии, представленный в его курсе Coursera машинного обучения. Я настоятельно рекомендую его видеолекции для тех, кто заинтересован в более тщательном рассмотрении предмета.

Открытие того, что производство масла в Бангладеш точно прогнозировало фондовый индекс S&P 500, связано с Лейнвебером [80]. К сожалению, как и большинство ложных корреляций, она была разоблачена сразу после своего открытия и больше не обладает предсказательной силой.

9.9. Упражнения

Линейная регрессия

9.1. *[3]* Постройте пример по $n \geq 6$ точкам, где оптимальная линия регрессии равна $y = x$, хотя ни одна из входных точек не лежит непосредственно на этой линии.

9.2. *[3]* Предположим, мы подбираем линию регрессии, чтобы предсказать срок годности яблока в зависимости от его веса. Для конкретного яблока мы прогнозируем срок годности 4,6 дня. Остаток яблок составляет –0,6 дня. Мы переоценили срок годности яблока? Объясните свои рассуждения.

9.3. *[3]* Предположим, мы хотим найти наиболее подходящую функцию $y = f(x)$, где $y = w^2x + wx$. Как мы можем использовать линейную регрессию, чтобы найти наилучшее значение w?

9.4. *[3]* Предположим, у нас есть возможность выбирать между использованием модели наилучшего соответствия вида $y = f(x)$, где $y = w^2x$ или $y = wx$, для постоянного коэффициента w. Что-то из этого является более общим, или они идентичны?

9.5. *[5]* Объясните, что такое распределение с длинными хвостами, и приведите три примера соответствующих явлений. Почему они важны в задачах классификации и регрессии?

9.6. *[5]* Используя библиотеку/пакет линейной алгебры, реализуйте регрессор в замкнутой форме $w = (A^TA)^{-1}A^Tb$. Насколько хорошо он работает по сравнению с существующим решателем?

9.7. *[3]* Установите влияние, которое различные значения для константы c логит-функции оказывают на вероятность классификации, составляющую 0,01; 1; 2 и 10 единиц от границы.

Эксперименты с линейной регрессией

9.8. *[5]* Поэкспериментируйте с влиянием подбора нелинейных функций с линейной регрессией. Для заданного набора данных (x, y) постройте наилучшую линию соответствия, где набор переменных составляет $\{1, x, \ldots, x^k\}$, для диапазона различных k. Модель улучшается или ухудшается в ходе этого процесса, как с точки зрения ошибки подбора, так и общей устойчивости?

9.9. *[5]* Поэкспериментируйте с влиянием масштабирования параметров в линейной регрессии. Для данного набора данных, по крайней мере с двумя параметрами (измерениями), умножьте все значения одного параметра на 10^k, для $-10 \leq k \leq 10$. Приводит ли эта операция к потере числовой точности при подборе?

9.10. *[5]* Поэкспериментируйте с влиянием сильно коррелированных параметров в линейной регрессии. Для заданного набора данных (x, y) скопируйте значение x с небольшим, но увеличивающимися количеством случайного шума. Что возвращается, когда новый столбец идеально соотносится с оригиналом? Что происходит с увеличением количества случайного шума?

9.11. *[5]* Поэкспериментируйте с влиянием выбросов на линейную регрессию. Для заданного набора данных (x, y) постройте наилучшую линию подбора. Периодически удаляйте точку с наибольшим остатком и совершайте подбор заново. Является ли последовательность предсказанных наклонов относительно стабильной для большей части этого процесса?

9.12. *[5]* Поэкспериментируйте с влиянием регуляризации на линейную/логистическую регрессию. Для заданного многомерного набора данных постройте линию наилучшего соответствия (а) без регуляризации, (б) гребневой регрессии и (в) регрессии LASSO; последние два с диапазоном значений ограничений. Как изменяется точность модели при уменьшении размера и количества параметров?

Реализация проектов

9.13. *[5]* Используйте линейную/логистическую регрессию, чтобы построить модель для одного из следующих конкурсов *The Quant Shop*.

(a) *Miss Universe*.
(b) *Movie gross*.
(c) *Baby weight*.
(d) *Art auction price*.
(e) *White Christmas*.
(f) *Football champions*.

(g) *Ghoul pool.*
(h) *Gold/oil prices.*

9.14. *[5]* Эта история о прогнозировании результатов турнира NCAA по баскетболу среди колледжей поучительна: `http://www.nytimes.com/2015/03/22/opinion/sunday/making-march-madness-easy.html`. Реализуйте такой классификатор логистической регрессии и распространите его на другие виды спорта, такие как футбол.

Вопросы на интервью

9.15. *[8]* Предположим, что мы обучаем модель с использованием стохастического градиентного спуска. Как мы узнаем, что приближаемся к решению?
9.16. *[5]* Методы градиентного спуска всегда сходятся к одной и той же точке?
9.17. *[5]* Какие предположения необходимы для линейной регрессии? Что, если некоторые из этих предположений не выполняются?
9.18. *[5]* Как мы обучаем модель логистической регрессии? Как мы интерпретируем ее коэффициенты?

Конкурсы Kaggle

9.19. Определите, что готовится, учитывая список ингредиентов.
`https://www.kaggle.com/c/whats-cooking`
9.20. Какие клиенты довольны своим банком?
`https://www.kaggle.com/c/santander-customer-satisfaction`
9.21. К чему работник должен иметь доступ, чтобы выполнять свою работу?
`https://www.kaggle.com/c/amazon-employee-access-challenge`

Глава 10

Методы измерения расстояний и сетей

Когда мера становится целью, она перестает быть мерой.

— Чарльз Гудхарт (Charles Goodhart) (Закон Гудхарта)

Матрица данных $n \times d$, состоящая из n примеров/строк, каждая из которых определяется d объектами/столбцами, естественно определяет набор из n точек в d-мерном геометрическом пространстве. Интерпретация примеров, как точек в пространстве, позволяет думать о них, как о звездах на небесах. Какие звезды находятся ближе всего к нашему Солнцу, т.е. каковы наши ближайшие соседи? Галактики — это естественные группы звезд, идентифицируемые группировкой данных. Какие звезды находятся на Млечном пути вместе с нашим солнцем?

Существует тесная связь между наборами точек в пространстве и вершинами в сетях. Часто мы строим сети из геометрических точек, соединяя близкие пары точек по ребрам. Наоборот, мы можем строить наборы точек из сетей, встраивая вершины в пространство, так что пары связанных вершин расположены рядом друг с другом.

Несколько важных проблем с геометрическими данными легко обобщаются как сетевые данные, включая классификацию ближайших соседей и кластеризацию. Таким образом, в этой главе мы рассматриваем обе темы вместе, чтобы лучше использовать синергизм между ними.

10.1. Измерение расстояний

Самая основная проблема в геометрии точек p и q в d измерениях заключается в том, как лучше всего измерить расстояние между ними. Возможно, неочевидно, что здесь есть какая-то проблема, о которой можно говорить, поскольку традиционная *Евклидова* геометрия вполне *очевидно* определяет то, как измеряется расстояние. Евклидова матрица определяется как

$$d(p,q) = \sqrt{\sum_{i=1}^{d} | p_i - q_i |^2}.$$

Но есть и другие разумные представления о расстоянии. Действительно, что такое метрика расстояния? Чем она отличается от произвольной функции оценки?

10.1.1. Метрики расстояния

Меры расстояния вполне очевидно отличаются от показателей сходства, таких как коэффициент корреляции, по направлению их увеличения. Меры расстояния уменьшаются по мере того, как предметы становятся более похожими, в то время как для функций подобия верно обратное.

У любой разумной меры расстояния есть некоторые полезные математические свойства, которые мы предполагаем. Мы говорим, что мера расстояния является *метрикой*, если она удовлетворяет следующим свойствам.

- *Положительность.* $d(x, y) \geq 0$ для всех x и y.
- *Идентичность.* $d(x, y) = 0$ тогда и только тогда, когда $x = y$.
- *Симметрия.* $d(x, y) = d(y, x)$ для всех x и y.
- *Треугольное неравенство.* $d(x, y) \leq d(x, z) + d(z, y)$ для всех x, y и z.

Эти свойства важны для рассуждения о данных. Действительно, многие алгоритмы работают правильно только тогда, когда функция расстояния является метрикой.

Евклидово расстояние является метрикой, поэтому эти условия кажутся нам такими естественными. Однако другие одинаково естественные меры подобия не являются метриками расстояния.

- *Коэффициент корреляции.* Не является положительным, поскольку варьируется от –1 до 1. Он также не идентичен, поскольку корреляция последовательности с самой собой равна 1.
- *Косинусный коэффициент/скалярное произведение.* Подобно коэффициенту корреляции, он не является ни положительным, ни идентичным по той же причине.
- *Время пути в направленной сети.* В мире с улицами с односторонним движением расстояние от x до y не обязательно совпадает с расстоянием от y до x.
- *Самые дешевые авиабилеты.* Это часто нарушает неравенство треугольника, поскольку самый дешевый способ перехода от y к x может привести к объезду через z из-за причудливых ценовых стратегий авиакомпаний.

В отличие от этого, не сразу очевидно, что некоторые известные функции расстояния являются метриками, такими как *редакторское расстояние* (edit distance), используемыми при распознавании строк. Вместо того чтобы делать

предположения, докажите или опровергните каждое из четырех основных свойств, чтобы убедиться, что вы понимаете, с чем работаете.

10.1.2. Метрика расстояния L_k

Евклидово расстояние — это всего лишь частный случай более общего семейства функций расстояния, известного как норма или метрика расстояния L_k:

$$d_k(p,q) = \sqrt[k]{\sum_{i=1}^{d} | p_i - q_i |^k} = \left(\sum_{i=1}^{d} |p_i - q_i|^k \right)^{1/k}.$$

Параметр k обеспечивает некий компромисс между наибольшей и полной разницей размеров. Значение для k может быть любым числом от 1 до ∞ с такими особенно популярными значениями.

- *Манхэттенское расстояние ($k = 1$).* Если игнорировать такие исключения, как Бродвей, все улицы Манхэттена проходят с востока на запад и с севера на юг, определяя регулярную сетку. Расстояние между двумя точками представляет собой сумму разниц между севером и югом и между востоком и западом, поскольку высокие здания исключают любую возможность сократить путь.

  Аналогично L_1, или *манхэттенское расстояние*, — это общая сумма отклонений между измерениями. Здесь все линейно, поэтому разница в 1 в каждом из двух измерений равна разнице в 2 только в одном измерении. Поскольку мы не можем использовать преимущества диагональных сокращений, как правило, существует много возможных кратчайших путей между двумя точками, как показано на рис. 10.1.

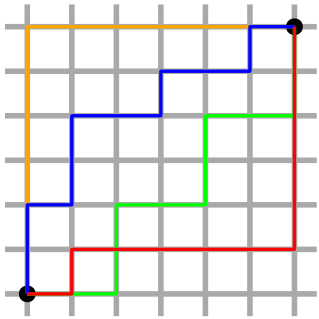


Рис. 10.1. Множество разных путей через сетку имеют одинаковую манхэттенское расстояние (L_1)

- *Евклидово расстояние ($k = 2$).* Это самая популярная метрика расстояния, предлагающая больший вес наибольшему отклонению в размерах, не подавляя меньшие измерения.
- *Предельная метрика ($k = \infty$).* По мере увеличения значения k меньшие различия в размерах теряют свою актуальность. Если $a > b$, то $a^k \gg b^k$. Взятие k-го корня из $a^k + b^k$ приближается к a как $b^k/a^k \rightarrow 0$.

  Рассмотрим отдаление точек $p_1 = (2, 0)$ и $p_2 = (2, 1{,}99)$ от начала координат:

  – при $k = 1$ расстояния составляют 2 и 3,99 соответственно;

  – при $k = 2$ они равны 2 и 2,82136;

  – при $k = 1000$ они равны 2 и 2,00001;

  – при $k = \infty$ они равны 2 и 2.

  Метрика L_∞ возвращает наибольшую одномерную разницу как расстояние.

Нам комфортно с евклидовой метрикой потому, что мы живем в евклидовом мире. Мы верим в истинность теоремы Пифагора, что стороны прямоугольного треугольника подчиняются соотношению $a^2 + b^2 = c^2$. В мире метрики L_k теорема Пифагора будет выглядеть как $a^k + b^k = c^k$.

Мы также довольны тем, что круги круглые. Напомним, что круг определяется как набор точек, которые находятся на расстоянии r от начальной точки p. Измените определение расстояния, и вы измените форму круга.

Форма "круга" L_k определяет, какие точки равны соседям относительно центральной точки p. На рис. 10.2 иллюстрируется, как развивается форма в зависимости от k. В манхэттенской метрике ($k = 1$) круг выглядит как ромб. При $k = 2$ это круглый объект, с которым мы знакомы. При $k = \infty$ этот круг растягивается до ориентированного по осям прямоугольника.

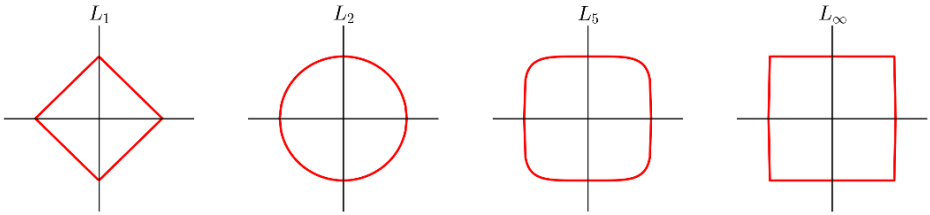


Рис. 10.2. Формы окружностей, определенные равными расстояниями при смене k

Происходит плавный переход от ромба к квадрату по мере изменения $1 \leq k \leq \infty$. Выбор значения k эквивалентен выбору того круга, который лучше всего подходит для нашей модели предметной области. Расстояния становятся особенно важными в пространствах более высокого измерения: нас интересуют отклонения во всех измерениях или в первую очередь самые большие?

На заметку. Выбор правильного значения k может оказать существенное влияние на значимость вашей функции расстояния, особенно в многомерных пространствах.

Взятие k-го корня из суммы членов k-й степени необходимо, чтобы результирующие значения "расстояния" удовлетворяли свойствам метрики. Однако во многих приложениях мы будем использовать расстояния только для сравнения: проверяем, является ли $d(x, p) \leq d(x, q)$, в отличие от использования значений в формулах или отдельно.

Поскольку мы берем абсолютное значение каждой размерности расстояния перед его увеличением до k-й степени, суммирование в функции расстояния всегда дает положительное значение. Функция k-й степени/корня является *монотонной*, что означает, что для $x, y, k \geq 0$

$$(x > y) \rightarrow (x^k > y^k).$$

Таким образом, порядок сравнения расстояний остается неизменным, если мы не берем k-й корень суммирования. Избегание вычисления k-го корня экономит время, что может оказаться очень важным при выполнении многих вычислений расстояния, как при поиске ближайшего соседа.

10.1.3. Работа в более высоких размерностях

Лично у меня нет геометрического представления о многомерных пространствах, где $d > 3$. Обычно лучшее, что мы можем сделать, — это рассматривать многомерную геометрию через линейную алгебру: уравнения, которые определяют наше понимание двух/трехмерной геометрии, легко обобщить для произвольного d, и именно так все и работает.

Мы можем развить некоторую интуицию о работе с многомерным набором данных с помощью *проекционных* методов, которые уменьшают размерность до уровней, которые мы можем понять. Зачастую полезно визуализировать двумерные проекции данных, полностью игнорируя другие измерения $d - 2$, а вместо этого изучать точечные графики размерных пар. С помощью таких методов уменьшения размерности, как анализ основных компонентов (см. раздел 8.5.2), мы можем комбинировать сильно коррелированные функции для получения более четкого представления. Конечно, некоторые детали теряются в процессе: будь то шум или нюанс, зависит от вашей интерпретации.

Должно быть ясно, что по мере увеличения количества измерений в нашем наборе данных мы неявно говорим, что каждое измерение является менее важной частью целого. При измерении расстояний между двумя точками в пространстве признаков следует понимать, что большое значение d означает, что

существует больше способов для точек быть близко (или далеко) друг от друга: мы можем представить, что они практически идентичны по всем измерениям, кроме одного.

Это делает выбор метрики расстояния наиболее важным в многомерных пространствах данных. Конечно, мы всегда можем придерживаться расстояния L_2, что является безопасным и стандартным выбором. Но если мы хотим премиальные баллы за близость во многих измерениях, мы предпочитаем метрику, которая больше склоняется к L_1. Если вместо этого вещи похожи, когда нет единых полей серьезного различия, возможно, нас должно интересовать нечто более близкое к L_∞.

Это можно рассматривать и так — беспокоит нас случайный добавочный шум в наших функциях или исключительные события, приводящие к большим артефактам. В первом случае L_1 нежелателен, поскольку метрика будет складывать шум ото всех размерностей расстояния. Но артефакты делают L_∞ подозрительным, поскольку существенная ошибка в любом отдельном столбце будет доминировать во всем расчете расстояния.

На заметку. Используйте свою свободу, чтобы выбрать лучшую метрику расстояния. Оцените, насколько хорошо работают различные функции, чтобы выявить сходство элементов в вашем наборе данных.

10.1.4. Размерный эгалитаризм

Все метрики расстояния L_k неявно взвешивают каждое измерение одинаково. Так не должно быть. Иногда мы сталкиваемся с такой проблемой понимания предметной области, где некоторые средства более важны для сходства, чем другие. Мы можем закодировать эту информацию, используя коэффициент c_i, чтобы указать различный вес для каждого измерения:

$$d_k(p,q) = \sqrt[k]{\sum_{i=1}^{d} c_i |p_i - q_i|^k} = \left(\sum_{i=1}^{d} c_i |p_i - q_i|^k \right)^{1/k}.$$

Мы можем рассматривать традиционное расстояние L_k как частный случай более общей формулы, где $c_i = 1$ для $1 \le i \le d$. Это взвешенное по размеру расстояние все еще удовлетворяет свойствам метрики.

Если у вас есть достоверные данные о желаемом расстоянии между определенными парами точек, то вы можете использовать линейную регрессию для подбора коэффициентов c_i, чтобы наилучшим образом соответствовать вашему учебному набору. Но, по правде говоря, взвешенное по размеру расстояние — это зачастую не очень хорошая идея. Если у вас нет подлинной причины знать,

что одни измерения важнее других, вы просто кодируете свои отклонения в формуле расстояния.

Однако появляются гораздо более серьезные предубеждения, если вы не нормализуете свои переменные до вычисления расстояний. Предположим, у нас есть возможность обозначить расстояние в метрах или километрах. Вклад 30-метровой разницы в функцию расстояния будет $30^2 = 900$ или $0{,}03 = 0{,}0009$, разница буквально в *миллион раз* по весу.

Правильный подход заключается в том, чтобы нормализовать значения каждого измерения по Z-оценкам *прежде*, чем вычислять ваше расстояние. Замените каждое значение x_i его Z-оценкой $z = (x - \mu_i)/\sigma_i$, где μ_i — среднее значение измерения i, а σ_i — его стандартное отклонение. Теперь ожидаемое значение x_i равно нулю для всех измерений, и разброс строго контролируется, если первоначально они распределялись нормально. Если определенное измерение подчиняется, например, степенному закону, придется предпринять более серьезные усилия. Соответствующие методы можно найти в разделе 4.3 о нормализации, например, прежде чем вычислять Z-оценку, можно применить логарифм.

> *На заметку.* Наиболее распространено использование взвешенных по размерам метрик расстояния — это маскировка того факта, что вы неправильно нормализовали свои данные. Не попадитесь в эту ловушку. Перед вычислением расстояний замените исходные значения Z-оценками, чтобы все размерности в равной степени влияли на результат.

10.1.5. Точки или векторы

Векторы и точки определяются массивами чисел, но концептуально — это разные звери для представления элементов в пространстве признаков. Векторы отделяют направление от величины, а поэтому могут рассматриваться как определяющие точки на поверхности единичной сферы.

Чтобы понять, почему это важно, рассмотрим проблему определения ближайших документов по количеству слов/тем. Предположим, что мы разделили словарь английского языка на n различных подмножеств на основе тем, поэтому каждое слово словаря находится точно в одной из тем. Мы можем представить каждую статью A как пакет слов, как точку p в n-мерном пространстве, где p_i равно числу встречающихся в статье A слов, которые взяты из темы i.

Если мы хотим, чтобы длинная статья о футболе была близка к короткой статье о футболе, величина этого вектора не может иметь значения, только его направление. Без нормализации по длине все крошечные документы длиной в

твит будут собираться рядом с источником, а не семантически кластеризоваться в пространстве так, как нам хочется.

Нормы (norm) являются мерами векторной величины, по существу, функциями расстояния, включающими только одну точку, поскольку вторая считается исходной. Векторы, по сути, являются нормализованными точками, где мы делим значение каждого измерения p на его L_2-норму $L_2(p)$, которая является расстоянием между p и началом O:

$$L_2(p) = \sqrt{\sum_{i=1}^{n} p_i^2}.$$

После такой нормализации длина каждого вектора будет равна 1, превращая его в точку на единичной сфере относительно начала координат.

У нас есть несколько возможных метрик расстояния, которые можно использовать при сравнении пар векторов. Первый класс определяется метриками L_k, включая евклидово расстояние. Это работает, потому что точки на поверхности сферы все еще являются точками в пространстве. Но, возможно, мы можем более осмысленно рассмотреть расстояние между двумя векторами в терминах угла, определенного между ними. Мы видели, что *косинусный коэффициент* (cosine similarity) между двумя точками p и q является их скалярным произведением, деленным на их L_2-нормы:

$$\cos(p, q) = \frac{pq}{\| p \| \| q \|}.$$

Функция косинуса здесь является функцией подобия, а не мерой расстояния, поскольку большие значения означают более высокое сходство. Определение *косинусного расстояния* (cosine distance) как $1 - |\cos(p, q)|$ действительно дает меру расстояния, которая удовлетворяет трем метрическим свойствам, кроме неравенства треугольника. Метрика истинного расстояния следует из *углового расстояния* (angular distance), где

$$d(p, q) = 1 - \frac{\arccos(\cos(p, q))}{\pi}.$$

Здесь $\arccos()$ — это обратная функция косинуса $\cos^{-1}()$, а π — это наибольший диапазон углов в радианах.

10.1.6. Расстояния между вероятностными распределениями

Вспомните критерий Колмогорова–Смирнова (раздел 5.3.3), который позволяет определить, были ли два набора выборок, вероятно, взяты из одного и того же базового распределения вероятностей.

Это говорит о том, что нам зачастую нужен способ сравнить пару распределений и определить меру сходства или расстояния между ними. Типичное применение заключается в измерении того, насколько близко одно распределение приближается к другому, и предоставляет способ определения лучших из множества возможных моделей.

Измерения расстояния, которые были описаны для точек, в принципе, могут применяться для измерения сходства двух распределений вероятности P и Q в заданном диапазоне дискретных переменных R.

Предположим, что R может принимать любое из ровно d возможных значений, скажем, $R = \{r_1, \ldots, r_d\}$. Пусть p_i (q_i) обозначает вероятность того, что $X = r_i$ при распределении P (Q). Поскольку и P, и Q являются вероятностными распределениями, мы знаем, что

$$\sum_{i=1}^{d} p_i = \sum_{i=1}^{d} q_i = 1.$$

Спектр значений p_i и q_i для $1 \le i \le d$ можно рассматривать как d-мерные точки, представляющие P и Q, расстояние между которыми можно вычислить с использованием евклидовой метрики.

Тем не менее существуют более специализированные меры, которые лучше оценивают сходство распределений вероятностей. Они основаны на теоретико-информационном понятии *энтропии*, которое определяет меру неопределенности для значения выборки, взятой из распределения. Это делает концепцию несколько аналогичной дисперсии.

Энтропия $H(P)$ распределения вероятностей P определяется как

$$H(P) = \sum_{i=1}^{d} p_i \log_2 \frac{1}{p_i} = -\sum_{i=1}^{d} p_i \log_2 p_i.$$

Как и расстояние, энтропия всегда неотрицательная величина. Две суммы выше отличаются только тем, как они этого достигают. Поскольку p_i является вероятностью, обычно она меньше 1, а следовательно, $\log p_i$, как правило, является отрицательным. Таким образом, получения обратной величины вероятностей перед взятием логарифма, либо смены знака каждого члена на отрицательный достаточно, чтобы сделать $H(P) \ge 0$ для всех P.

Энтропия — это мера неопределенности. Рассмотрим распределение, где $p_1 = 1$ и $p_i = 0$, для $2 \le i \le d$. Это похоже на бросок полностью смещенного кубика, поэтому, несмотря на наличие d-сторон, нет никакой неопределенности относительно результата. Конечно, $H(P) = 0$, поскольку либо p_i, либо $\log_2 1$ обнуляют каждый член суммы. Теперь рассмотрим распределение, где $q_i = 1/d$ для $1 \le i \le d$. Это представляет бросок идеального кубика, максимально неопределенное распределение, где $H(Q) = \log_2 d$ бит.

Противоположность неопределенности — это информация. Энтропия $H(P)$ соответствует тому, сколько информации вы узнаете после того, как пример из P изучен. Вы ничего не узнаете, когда некто говорит вам то, что вы уже знаете.

Стандартные меры расстояния по вероятностным распределениям основаны на энтропии и теории информации. Дивергенция *Кульбака — Лейблера* (Kullback-Leibler — KL) измеряет полученную неопределенность или потерю информации при замене распределения P на Q. В частности,

$$KL(P \,||\, Q) = \sum_{i=1}^{d} p_i \log_2 \frac{p_i}{q_i}.$$

Предположим, что $P = Q$. Тогда ничто не должно быть ни получено, ни потеряно, и $KL(P, P) = 0$ поскольку $\lg 1 = 0$. Но чем хуже замена Q на P, тем больше $KL(P \,||\, Q)$ поднимает до ∞, когда $p_i > q_i = 0$.

Дивергенция KL напоминает меру расстояния, но не является метрикой, поскольку она не симметрична ($KL(P \,||\, Q) \neq KL(Q \,||\, P)$) и не удовлетворяет неравенству треугольника. Тем не менее она составляет основу дивергенции *Дженсена-Шеннона* (Jensen-Shannon — JS) (P, Q):

$$JS(P, Q) = \frac{1}{2} KL(P \,||\, M) + \frac{1}{2} KL(Q \,||\, M),$$

где распределение M является средним значением P и Q, т.е. $m_i = (p_i + q_i)/2$.

Дивергенция JS (P, Q) явно симметрична и сохраняет при этом другие свойства дивергенции KL. Далее, $\sqrt{JS(P, Q)}$ магическим образом удовлетворяет неравенству треугольника, превращая его в истинную метрику. Это правильная функция для измерения расстояния между распределениями вероятностей.

10.2. Классификация ближайших соседей

Функции расстояния позволяют определить, какие точки находятся ближе всего к заданной цели. Они обладают большой мощью и лежат в основе механизма *классификация методом ближайшего соседа* (nearest neighbor

classification). Когда дан набор маркированных учебных примеров, мы ищем такой учебный пример, который больше всех похож на немаркированную точку p, а затем берем для p метку класса у ближайшего маркированного соседа.

Идея здесь проста. Мы используем ближайшего маркированного соседа к заданной точке q в качестве ее представителя. Если мы имеем дело с проблемой классификации, мы назначим q ту же метку, что и у ближайшего соседа (соседей). Если мы имеем дело с проблемой регрессии, присвойте q среднее значение/медиану ближайшего соседа (соседей). Эти прогнозы легко оправдать, если предположить, что (1) пространство признаков когерентно фиксирует свойства рассматриваемых элементов, и (2) функция расстояния осмысленно распознает похожие линии/точки, когда они встречаются.

Библия призывает нас любить ближнего своего. У методов ближайшего соседа при классификации есть три больших преимущества.

- *Простота.* Методы ближайшего соседа — это не ракетостроение; здесь нет математики более пугающей, чем метрика расстояния. Это важно, потому что это означает, что мы можем точно знать, что происходит, и не стать жертвой ошибок или неправильных представлений.
- *Интерпретируемость.* Изучение ближайших соседей заданной точки q объясняет, почему именно классификатор принял свое решение. Если вы не согласны с этим результатом, вы можете систематически отлаживать результаты. Были ли соседние точки помечены неправильно? Ваша функция расстояния не смогла выбрать элементы, которые были логической группой равных для q?
- *Нелинейность.* У классификаторов ближайших соседей есть границы принятия решений, которые являются кусочно-линейными, но могут произвольно извиваться, следуя набору обучающих примеров, как показано на рис. 10.3. Из вычислений мы знаем, что кусочно-линейные функции приближаются к гладким кривым, когда кусочки становятся достаточно маленькими. Таким образом, классификаторы ближайших соседей позволяют реализовать очень сложные границы принятия решений, их поверхности действительно настолько сложные, что они не имеют краткого представления.

Существует несколько аспектов построения эффективных классификаторов ближайших соседей, в том числе технические вопросы, связанные с надежностью и эффективностью. Но прежде всего следует научиться ценить силу аналогии. Мы обсудим эти вопросы в разделах ниже.

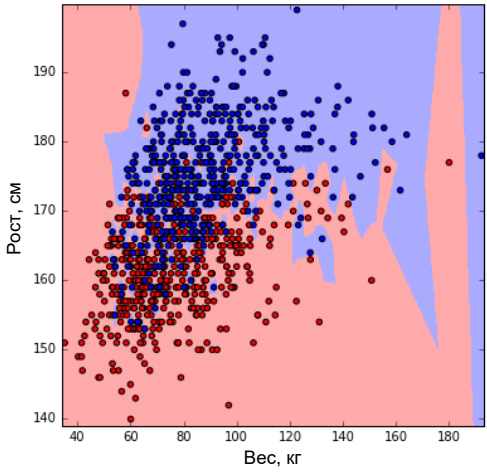


Рис. 10.3. Граница решения классификаторов по методу ближайших соседей может быть нелинейной

10.2.1. В поисках хороших аналогий

Некоторые интеллектуальные дисциплины опираются на силу аналогий. Юристы не ссылаются непосредственно на законы, они полагаются на прецеденты: результаты дел, рассмотренных ранее уважаемыми юристами. Правильное решение для текущего дела (я выигрываю или проигрываю) — это функция, которая может продемонстрировать, что предыдущие дела наиболее фундаментально похожи на рассматриваемый вопрос.

Точно так же большая часть медицинской практики опирается на опыт. Старый деревенский врач вспоминает своих предыдущих пациентов, которым удалось выжить, чтобы припомнить случаи с симптомами, похожими на ваши, а затем дает вам то же, что и им. Моему нынешнему врачу (доктору Лирнеру) сейчас за восемьдесят, но я доверяю ему больше всех тех молодых, которые полагаются только на последние знания, полученные в медицинской школе.

Извлечение максимальной выгоды из методов ближайшего соседа включает в себя обучение уважению аналогических рассуждений. Как правильно предсказать цену дома? Мы можем описать каждое свойство с точки зрения таких характеристик, как площадь участка и количество спален, а также назначить каждому вес в долларах, который будет добавлен с помощью линейной регрессии. Или мы можем искать “льготы”, находя сопоставимые объекты в похожих кварталах и прогнозировать цену, аналогичную той, которую мы видим. Второй подход — аналогичное рассуждение.

Я рекомендую вам воспользоваться набором данных, в предметной области которых вы обладаете знаниями и интересами, и провести некоторые экс-

перименты по поиску ближайших соседей. Один из ресурсов, который меня всегда вдохновляет, — это `http://www.baseball-reference.com`, который сообщает о десяти ближайших соседях для каждого игрока, основываясь на их текущей статистике. Я нахожу эти аналогии удивительно запоминающимися: идентификации игрока зачастую играют сходные роли и применяют стили, которые не должны быть явно отражены статистикой. Но все же они есть.

Попробуйте сделать это с другой предметной областью, которая вам небезразлична: книги, фильмы, музыка или что-то еще. Попробуйте почувствовать силу методов ближайшего соседа и аналогий.

> *На заметку.* Идентифицируйте десять ближайших соседей по известным вам точкам, что предоставляет отличный способ понять сильные и слабые стороны данного набора данных. Визуализация подобных аналогий должна стать вашим первым шагом в работе с любым многомерным набором данных.

10.2.2. *k* ближайших соседей

Чтобы классифицировать заданную точку q, методы ближайшего соседа возвращают метку q' — ближайшую маркированную точку к q. Это разумная гипотеза, предполагающая, что сходство в пространстве признаков подразумевает сходство в пространстве меток. Но эта классификация основана только на одном обучающем примере.

Более надежная классификация, или интерполяция, следует из голосования по нескольким близким соседям. Предположим, что мы находим k точек, наиболее близких к нашему запросу, где k обычно является некоторым значением в диапазоне от 3 до 50 в зависимости от размера n. Расположение помеченных точек в сочетании с выбором k делит пространство признаков на регионы, причем всем точкам в конкретной заданной области назначается одна и та же метка.

Рассмотрим рис. 10.4, на котором делается попытка построить гендерный классификатор на основе данных о росте и весе. По правде говоря, женщины ниже и легче мужчин, но есть много исключений, особенно вблизи границы принятия решений. Как показано на рис. 10.4, увеличение k приводит к созданию более крупных регионов с более плавными границами, что обеспечивает более надежные решения. Однако, чем больше мы делаем k, тем более общими будут наши решения. Выбор $k = n$ — это просто другое имя для классификатора большинства, где мы присваиваем каждой точке наиболее распространенную метку независимо от ее индивидуальных особенностей.

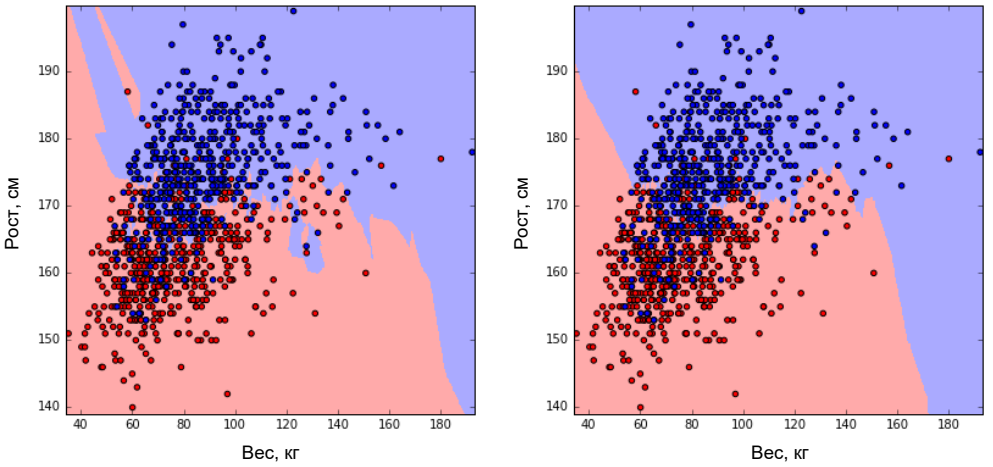


Рис. 10.4. Влияние k на границу решения для гендерной классификации методом k ближайших соседей. Сравните k = 3 (слева) и k = 10 (справа) с k = 1 на рис. 10.3

Правильный способ установить *k* заключается в том, чтобы назначить часть помеченных обучающих примеров в качестве оценочного набора, а затем поэкспериментировать с различными значениями параметра *k*, чтобы увидеть, где достигается наилучшая производительность. Эти оценочные значения могут быть затем возвращены в учебный/целевой набор, как только было выбрано *k*.

Для задачи бинарной классификации мы хотим, чтобы *k* было нечетным числом, поэтому решение никогда не будет равносильным. По правде говоря, разницу между количеством положительных и отрицательных голосов можно интерпретировать как показатель нашей уверенности в принятии решения.

Существуют потенциальные асимметрии относительно геометрически ближайших соседей. Каждая точка имеет ближайшего соседа, но для точек выброса эти ближайшие соседи могут не быть особенно близки. Эти точки выброса на самом деле могут играть огромную роль в классификации, определяя ближайшего соседа для огромного объема пространства признаков. Однако, если вы правильно выбрали учебные примеры, это должна быть в основном необитаемая территория, область в пространстве признаков, где точки встречаются редко.

Идея классификации ближайших соседей может быть обобщена для интерполяции функций за счет усреднения значений *k* ближайших точек. Предположительно это делается на сайтах недвижимости, таких как `www.zillow.com`, для прогнозирования цен на жилье от ближайших соседей. Такие схемы усреднения могут быть обобщены с помощью неоднородных весов, оценивая точки по-разному в зависимости от ранга или величины расстояния. Подобные идеи работают для всех методов классификации.

10.2.3. Поиск ближайших соседей

Возможно, наибольшим ограничением методов классификации ближайших соседей является их стоимость во время выполнения. Сравнение точки q в d измерениях с n учебными точками наиболее очевидно выполняется в ходе n явных сравнений расстояний ценой $O(nd)$. Из-за наличия тысяч или даже миллионов доступных учебных точек этот поиск может внести заметную задержку в любую систему классификации.

Один из подходов к ускорению поиска подразумевает использование геометрических структур данных. Популярные варианты таковы.

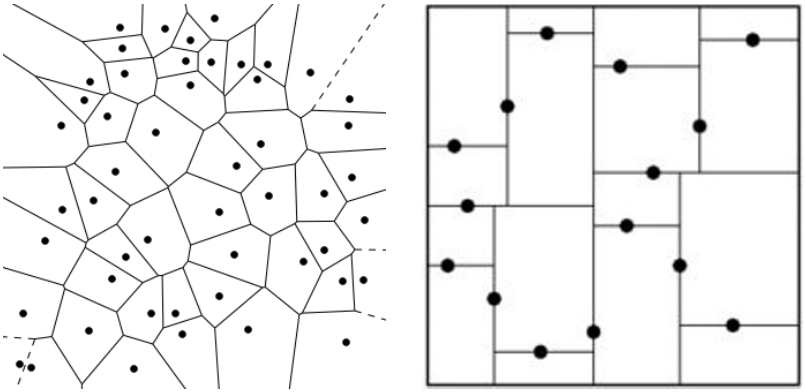


Рис. 10.5. К структурам данных для поиска ближайшего соседа относятся диаграммы Вороного (слева) и kd-деревья (справа)

- *Диаграммы Вороного*. Для набора целевых точек мы бы хотели разделить пространство вокруг них на ячейки так, чтобы каждая ячейка содержала ровно одну целевую точку. Кроме того, мы хотим, чтобы целевая точка каждой ячейки была ближайшим целевым соседом для всех местоположений в ячейке. Такое разбиение, *диаграмма Вороного*, показано на рис. 10.5 (слева).

  Границы диаграмм Вороного определяются перпендикулярными биссектрисами между парами точек (a, b). Каждая биссектриса разделяет пространство пополам: одна половина содержит a, а другая — b, так что все точки на половине a ближе к a, чем b, и наоборот.

  Диаграммы Вороного являются прекрасным инструментом для размышлений о данных и обладают множеством полезных свойств. Существуют эффективные алгоритмы их построения и поиска, особенно в двух измерениях. Однако эти процедуры быстро усложняются по мере увеличения размерности, что делает их, как правило, непрактичными за пределами двух или трех измерений.

- *Индексные сетки.* Мы можем разбить пространство на d-мерные блоки, разделив диапазон каждого измерения на r интервалов или сегментов. Рассмотрим, например, двумерное пространство, в котором каждая ось представляет собой вероятность, варьируясь, таким образом, от 0 до 1. Этот диапазон можно разделить на r равных интервалов, так что i-й интервал находится в диапазоне $[(i-1)/r, i/r]$.

  Эти интервалы определяют регулярную сетку в пространстве, поэтому мы можем связать каждую из учебных точек с ячейкой сетки, которой она принадлежит. Поиск теперь становится проблемой определения правильной ячейки сетки для точки q с помощью поиска в массиве или двоичного поиска, а затем сравнения q со всеми точками в этой ячейке для определения ближайшего соседа.

  Такие индексные сетки могут быть эффективными, но есть потенциальные проблемы. В первую очередь, учебные точки могут быть неравномерно распределены, и многие ячейки могут быть пустыми, как показано на рис. 10.6. Установление неоднородной сетки может привести к более сбалансированному расположению, но это затруднит быстрый поиск ячейки, содержащей q. Но также нет гарантии того, что ближайший сосед q фактически находится в той же ячейке, что и q, особенно если q лежит очень близко к границе ячейки. Это означает, что мы должны обыскать и соседние ячейки, чтобы убедиться, что мы нашли абсолютного ближайшего соседа.

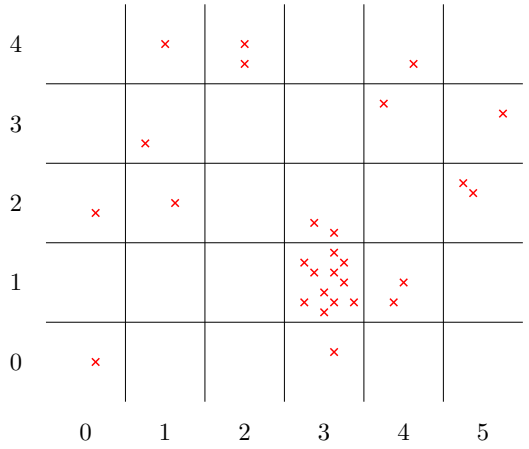


Рис. 10.6. Структура данных индексных сеток обеспечивает быстрый доступ к ближайшим соседям, когда точки распределены равномерно, но она может быть неэффективной, когда точки в определенных регионах плотно кластеризованы

- *Kd-деревья*. Существует большой класс древовидных структур данных, которые разделяют пространство, используя иерархию делений, облегчающую поиск. Начиная с произвольного измерения в качестве корня, каждый узел *kd*-дерева определяет срединную линию/плоскость, которая разделяет точки поровну в соответствии с этим измерением. Конструкция повторяется на каждой стороне, используя разные измерения, и так далее, пока область, определенная узлом, не будет содержать только одну обучающую точку.

  Эта конструкция иерархии идеально подходит для поддержки поиска. Начиная с корня, мы проверяем, находится ли точка запроса q слева или справа от средней линии/плоскости. Это определяет, на какой стороне лежит q, а следовательно, к какой стороне дерева следует вернуться. Время поиска составляет $\log n$, так как мы разделяем заданную точку пополам с каждым шагом вниз по дереву.

  Существует множество таких древовидных структур поиска с пространственным разделением, причем одна или несколько из них реализованы в библиотеке функций вашего любимого языка программирования. Некоторые предлагают более быстрое время поиска по проблемам, таким как ближайший сосед, возможно, с компромиссом по точности и скорости.

Хотя эти методы действительно могут ускорить поиск ближайших соседей в скромных количествах измерений (скажем, $2 \leq d \leq 10$), они становятся все менее эффективными по мере увеличения размерности. Причина в том, что количество способов, благодаря которыми две точки можно считать расположенными близко друг к другу, быстро увеличивается с ростом размерности, что затрудняет удаление областей, которые не имеют шансов содержать ближайшего соседа к q. Детерминированный поиск ближайшего соседа в конечном итоге сводится к линейному поиску для данных с достаточно высокой размерностью.

10.2.4. Локальное хеширование

Чтобы добиться более быстрого времени работы, мы должны отказаться от идеи поиска точного ближайшего соседа и сделать правильный выбор. Мы хотим сгруппировать близлежащие точки в сегменты по сходству и быстро найти наиболее подходящий сегмент B для нашей точки q. Вычисляя только расстояние между q и точками в сегменте, мы экономим время поиска, когда $||B|| \ll n$.

Это была основная идея индекса сетки, описанная в предыдущем разделе, но на практике структуры поиска становятся громоздкими и несбалансированными. Лучший подход основан на хешировании.

Локальное хеширование (Locality Sensitive Hashing — LSH) определяется хеш-функцией $h(p)$, которая в качестве входных данных получает точку или

вектор и выдает число или код в качестве выходных данных, так что вполне вероятно, что $h(a) = h(b)$, если a и b расположены близко друг к другу, и $h(a) \neq h(b)$, если они расположены далеко друг от друга.

Такие локальные хеш-функции легко выполняют ту же роль, что и индексная сетка, но без суеты. Мы можем просто поддерживать таблицу точек, сгруппированных по этому одномерному хеш-значению, а затем, выполняя поиск $h(q)$, искать потенциальные совпадения для точки q.

Как мы можем построить такие локальные хеш-функции? Поначалу эту идею проще всего понять, ограничиваясь векторами, а не точками. Напомним, что множества d-мерных векторов можно рассматривать как точки на поверхности сферы, т.е. как круг, когда $d = 2$.

Рассмотрим произвольную прямую l_1, проходящую через начало этого круга и разрезающую круг пополам, как показано на рис. 10.7. Действительно, мы можем случайным образом выбрать линию l_1, просто выбрав случайный угол $0 \leq \Theta_1 < 2\pi$. Этот угол определяет наклон прямой, проходящей через начало O, и вместе Θ_1 и O полностью определяют l_1. Будучи выбранной случайным образом, линия l_1 должна разделить векторы, поместив около половины из них слева, а остальные справа.

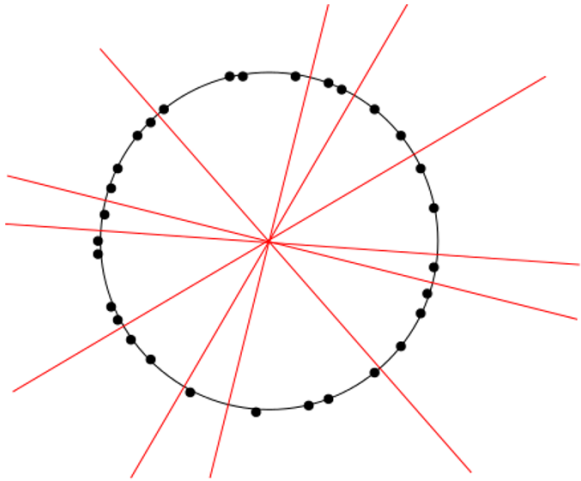


Рис. 10.7. Ближайшие точки на окружности обычно лежат на одной стороне случайных линий через начало координат. Локальные хеш-коды для каждой точки могут быть составлены как последовательность тестов на односторонность (слева или справа) для любой конкретной последовательности линий

Теперь добавьте второй случайный делитель l_2, который должен иметь те же свойства. Затем разделите все векторы между четырьмя областями {*LL*, *LR*, *RL*, *RR*}, определенными их статусом относительно этих делителей l_1 и l_2.

Ближайший сосед любого вектора v должен лежать в той же области, что и вектор v, если только нам не повезло, и линии l_1 либо l_2 их не разлучили. Но вероятность $p(v_1, v_2)$, что оба v_1 и v_2 находятся на одной стороне l, зависит от угла между v_1 и v_2. В частности, $p(v_1, v_2) = 1 - \theta(v_1, v_2)/\pi$.

Таким образом, мы можем вычислить точную вероятность того, что близкие соседи сохранятся для n точек и m случайных плоскостей. Шаблон L и R в этих m плоскостях определяет m-битный локальный хэш-код $h(v)$ для любого вектора v. По мере того как мы переходим от двух плоскостей нашего примера к более длинным кодам, ожидаемое количество точек в каждом интервале достигает $n/2^m$, хотя с повышенным риском того, что одна из m плоскостей отделяет вектор от своего истинного ближайшего соседа.

Обратите внимание, что этот подход можно легко обобщить за пределы двух измерений. Пусть гиперплоскость определяется ее вектором нормали r, перпендикулярным направлению к плоскости. Знак $s = vr$ определяет, на какой стороне находится вектор v. Напомним, что скалярное произведение двух ортогональных векторов равно 0, поэтому $s = 0$, если v лежит точно на разделяющей плоскости. Кроме того, s положительно, если v выше этой плоскости, и отрицательно, если ниже нее. Таким образом, i-я гиперплоскость вносит ровно один бит в хеш-код, где $h_i(q) = 0$, если $vr_i \leq 0$.

Такие функции могут быть обобщены за пределы векторов на произвольные множества точек. Кроме того, их точность может быть улучшена за счет создания нескольких наборов кодовых слов для каждого элемента, включая различные наборы случайных гиперплоскостей. До тех пор пока q разделяет хотя бы одно кодовое слово со своим истинным ближайшим соседом, мы в конечном итоге столкнемся с сегментом, содержащим обе эти точки.

Обратите внимание, что LSH имеет совершенно противоположную цель по сравнению с традиционными хеш-функциями, используемыми для криптографических приложений или для управления хеш-таблицами. Традиционные хеш-функции стремятся к тому, чтобы пары схожих элементов приводили к совершенно разным хеш-значениям, поэтому мы можем распознавать изменения и использовать весь диапазон таблицы. LSH, напротив, хочет, чтобы подобные элементы получали тот же хеш-код, чтобы мы могли распознать сходство. При LSH ближайшие соседи принадлежат одному интервалу.

Локальное хеширование имеет другие применения в науке о данных, помимо поиска ближайших соседей. Возможно, наиболее важным является создание сжатых представлений объектов из сложных объектов, например видео или музыкальных потоков. Коды LSH, построенные из интервалов этих потоков, определяют числовые значения, потенциально подходящие в качестве функций для сопоставления с образцом или построения модели.

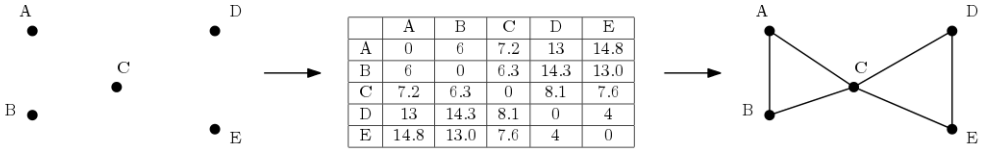
10.3. Графы, сети и расстояния

Граф (graph) $G = (V, E)$ определен на множестве *вершин* (vertices) V и содержит множество *ребер* (edges) E упорядоченных или неупорядоченных пар вершин из V. При моделировании дорожной сети вершины могут представлять города или перекрестки, некоторые пары которых напрямую связаны дорогами/ребрами. При анализе человеческих взаимодействий вершины обычно представляют людей с ребрами, соединяющими пары связанных душ.

Многие другие современные наборы данных естественным образом моделируются в виде графов или сетей.

- *Всемирная паутина (WWW).* Здесь есть вершина в графе для каждой веб-страницы с направленным ребром (x, y), если веб-страница x содержит гиперссылку на веб-страницу y.
- *Сети товаров/клиентов.* Они возникают в любой компании, у которой много клиентов и видов товаров: будь то Amazon, Netflix или даже продуктовый магазин на углу. Существует два типа вершин: один для клиентов и другой для товаров. Ребро (x, y) обозначает товар y, купленный клиентом x.
- *Генетические сети.* Здесь вершины представляют различные гены/белки в определенных организмах. Считайте это списком запчастей для животного. Ребро (x, y) обозначает наличие взаимодействия между частями x и y. Возможно, ген x регулирует ген y, или белки x и y связываются вместе, образуя более крупный комплекс. Такие сети взаимодействия кодируют важную информацию о том, как работает базовая система.

Граф и набор точек являются тесно связанными объектами. Оба состоят из отдельных объектов (точек или вершин), представляющих элементы в наборе. Оба кодируют важные понятия расстояния и взаимоотношений, либо близко-далеко, либо независимо друг от друга. Наборы точек могут быть осмысленно представлены графами, а графы — наборами точек.



	A	B	C	D	E
A	0	6	7.2	13	14.8
B	6	0	6.3	14.3	13.0
C	7.2	6.3	0	8.1	7.6
D	13	14.3	8.1	0	4
E	14.8	13.0	7.6	4	0

Рис. 10.8. Попарные расстояния между множеством точек в пространстве (слева) определяют полный взвешенный граф (в центре). Обработка пороговых значений с помощью отсечения больших расстояний удаляет все длинные ребра, оставляя разреженный график, который отражает структуру точек (справа)

10.3.1. Взвешенные графы и индуцированные сети

Ребра в графах фиксируют *бинарные отношения* (binary relations), где каждое ребро (x, y) представляет наличие связи между x и y. Иногда наличие этих отношений — все, что нужно знать, как, например, о связи между веб-страницами или фактом, что кто-то приобрел определенный товар.

Но часто есть некая мера силы или близости отношений. Конечно, мы видим это в дорожных сетях: у каждого сегмента дороги есть длина или время в пути, которое необходимо знать, чтобы найти оптимальный маршрут для движения между двумя точками. Мы говорим, что граф *взвешен* (weighted), если с каждым ребром связано числовое значение.

Зачастую (но не всегда) этот вес естественно интерпретируется как расстояние. Так, можно интерпретировать набор данных из n точек в пространстве как полный взвешенный граф из n вершин, где вес ребра (x, y) — это геометрическое расстояние между точками x и y в пространстве. Для многих приложений этот граф кодирует всю необходимую информацию о точках.

Графы наиболее естественно представлены *матрицами смежности* (adjacency matrix) $n \times n$. Определим *нереберный* (non-edge) символ x. Матрица M представляет граф $G = (V, E)$, когда $M[i, j] \neq x$, если и только если вершины i, $j \in V$ соединены ребром $(i, j) \in E$. Для невзвешенных сетей обычно реберный символ равен 1, когда $x = 0$. Для графов, взвешенных по расстоянию, вес ребра (i, j) — это стоимость перемещения между ними, поэтому параметр $x = \infty$ означает отсутствие какой-либо прямой связи между i и j.

Это матричное представление для сетей имеет значительную мощь, поскольку для работы с ними мы можем использовать все наши инструменты из линейной алгебры. К сожалению, это сопряжено с затратами, поскольку хранение матриц размера $n \times n$ может оказаться безнадежно дорогостоящим, когда сети выходят за пределы нескольких сотен вершин. Существуют более эффективные способы хранения больших *разреженных графов* (sparse graph) со множеством вершин, но относительно небольшим количеством пар, соединенных ребрами. Я не буду обсуждать здесь детали алгоритмов графов, но с уверенностью отсылаю вас к моей книге *The Algorithm Design Manual* [1], чтобы вы могли узнать больше.

Изображения графов/сетей нередко создаются в ходе назначения каждой вершине точки на плоскости и рисования линий между этими вершинами для представления ребер. Такие *диаграммы связности узлов* (node-link diagram) чрезвычайно полезны для визуализации структуры сетей, с которыми вы работаете. Они могут быть алгоритмически построены с использованием *метода взаимодействия сил* (force-directed layout), где ребра действуют как пружины, сближая соседние пары вершин, а несмежные вершины отталкивая друг друга.

Такие рисунки устанавливают связь между графическими структурами и точками. *Вложение* (embedding) — это точечное представление вершин графа, которое отражает некоторый аспект его структуры. Выполнение сжатия объекта, такого как разложение по собственному значению или сингулярному значению (см. раздел 8.5) на матрицу смежности графа, приводит к представлению более низкой размерности, которое служит точечным представлением каждой вершины. К другим подходам к векторному представлению графов относится DeepWalk, который будет обсуждаться в разделе 11.6.3.

> *На заметку.* Наборы точек могут быть осмысленно представлены графами/матрицами расстояний, а графы/матрицы расстояний осмысленно представлены наборами точек (векторных представлений).

Геометрические графики, определяемые расстояниями между точками, представляют класс графов, которые я называю *индуцированными сетями* (induced network), где ребра определяются механическим способом из какого-либо внешнего источника данных. Это общий источник сетей в науке о данных, поэтому важно следить за тем, как ваш набор данных может быть преобразован в граф.

Функции расстояний или подобия зачастую используются для построения сетей по наборам элементов. Обычно нас интересуют ребра, соединяющие каждую вершину с ее k ближайшими/похожими вершинами. Мы получаем разреженный граф, сохраняя k небольшим, скажем, $k \approx 10$, что означает, что с ним легко работать даже при больших значениях n.

Но есть и другие типы индуцированных сетей. Вполне нормально было бы соединять вершины x и y всякий раз, когда они имеют значимый общий атрибут. Например, мы можем построить индуцированную социальную сеть из людей и их резюме, связывая любых двух человек, которые работали в той же компании или посещали ту же школу в тот же период. Такие сети, как правило, имеют блочную структуру, где существуют большие подмножества вершин, образующих полностью связанные группировки. В конце концов, если x окончил тот же колледж, что и y, а y окончил тот же колледж, что и z, то это означает, что (x, z) также должно быть ребром в графе.

10.3.2. Классификация графов

Существует терминология графов, которую важно знать для работы с ними. Разговор — это важно. Несколько фундаментальных свойств графов влияют на то, что они представляют, и как мы можем их использовать. Таким образом,

первым шагом в любой задаче с графами является определение разновидностей графов, с которыми придется иметь дело.

- *Направленный или ненаправленный.* Граф $G = (V, E)$ является *ненаправленным* (undirected), если ребро $(x, y) \in E$ подразумевает, что (y, x) также находится в E. Если нет, мы говорим, что граф *направлен* (directed). Дорожные сети *между* городами, как правило, не направлены, поскольку любая большая дорога имеет полосы, идущие в обоих направлениях. Уличные сети *внутри* городов почти всегда направлены, поскольку где-то скрывается по крайней мере несколько улиц с односторонним движением. Графы веб-страниц, как правило, направлены, поскольку ссылка со страницы x на страницу y не должна отвечать взаимностью.
- *Взвешенный или невзвешенный.* Как обсуждалось в разделе 10.3.1, каждому ребру (или вершине) во *взвешенном* (weighted) графе G назначается числовое значение или вес. Ребра графа дорожной сети могут быть взвешены с учетом их длины, времени прохождения или ограничения скорости, в зависимости от применения. В *невзвешенных* (unweighted) графах нет разницы в стоимости между различными ребрами и вершинами.

  Графы расстояний по своей природе взвешены, в то время как социальные сети или веб-сети, как правило, не взвешены. Разница определяет, являются ли векторы объектов, связанные с вершинами, 0/1 или числовыми значениями важности, которые, возможно, должны быть нормализованы.
- *Простой или непростой.* Некоторые типы ребер усложняют работу с графами. *Петля* (self-loop) — это ребро (x, x), имеющее только одну вершину. Ребро (x, y) является *многореберным* (multiedge), если оно встречается в графе более одного раза.

  Обе эти структуры требуют особой осторожности при предварительной обработке для генерации элементов. Следовательно, любой граф, который их избегает, называется *простым* (simple). В начале анализа мы часто стремимся удалить как петли, так и многореберники.
- *Плотный или разреженный.* Графы *разрежены* (sparse), когда только небольшая часть из всех возможных пар вершин ($\binom{n}{2}$ для простого неориентированного графа на n вершин) фактически имеет ребра, определенные между ними. Графы, в которых большая часть пар вершин определяют ребра, считаются *плотными* (dense). Не существует официальной границы между тем, что считается разреженным, и тем, что считается плотным, но обычно плотные графы имеют квадратичное количество ребер, в то время как разреженные графы имеют линейный размер.

Разреженные графы обычно разрежены по причинам, связанным с их применением. Дорожные сети должны иметь разреженные графы из-за дорожных развязок. Самым ужасным перекрестком, о котором я когда-либо слышал, была конечная точка девяти различных дорог. Графы ближайших соседей имеют ровно k-ю степень вершин. Разреженные графы допускают гораздо более эффективное представление пространства, чем матрицы смежности, что позволяет представлять гораздо большие сети.

- *Вложенный или топологический.* Граф является *вложенным* (embedded), если вершинам и ребрам назначены геометрические позиции. Таким образом, любой рисунок графа является *вложением* (embedding), которое может иметь или не иметь алгоритмическое значение.

  Иногда структура графа полностью определяется геометрией его вложения, как мы видели в определении графа расстояний, где веса определяются евклидовым расстоянием между каждой парой точек. Малоразмерные представления матриц смежности с помощью SVD также квалифицируются как вложения, точечные представления, которые собирают бо́льшую часть информации о связности графа.

- *Маркированный или немаркированный.* В *маркированном* (labeled) графе каждой вершине присваивается уникальное имя, или идентификатор, чтобы отличать его от всех других вершин. В *немаркированных* (unlabeled) графах таких различий не делается.

  Графы, возникающие в приложениях для обработки данных, часто имеют естественную и содержательную маркировку, например названия городов в транспортной сети. Они полезны в качестве идентификаторов для репрезентативных примеров, а также для обеспечения связи с внешними источниками данных, где это необходимо.

10.3.3. Теория графов

Теория графов является важной областью математики, имеющей дело с фундаментальными свойствами сетей и методами их вычисления. Большинство студентов, изучающих информатику, знакомятся с теорией графов на курсе по дискретным структурам или алгоритмам.

Классические алгоритмы поиска кратчайших путей, связанных компонентов, связующих деревьев, сегментации, сопоставлений и топологической сортировки могут быть применены к любому разумному графу. Но я не видел, чтобы эти инструменты применялись в науке о данных в целом, как мне кажется. Одна из причин заключается в том, что графы в науке о данных обычно являются очень большими, что ограничивает сложность того, что можно с ними

сделать. Однако во многом это просто близорукость: люди не видят, что матрица расстояний или сходств на самом деле является просто графом, чем могут воспользоваться другие инструменты.

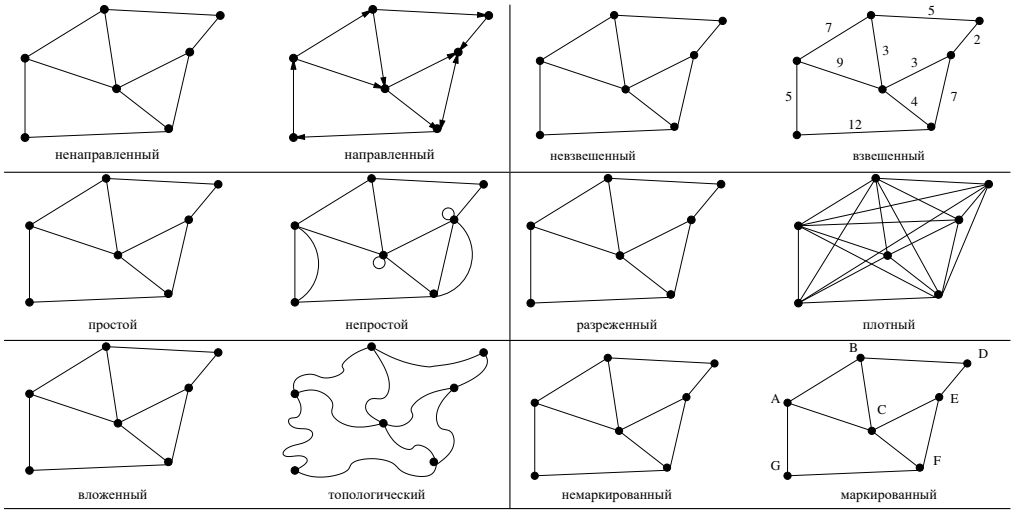


Рис. 10.9. Важные свойства/особенности графов

Я пользуюсь этой возможностью, чтобы рассмотреть связи этих фундаментальных проблем с наукой о данных и призвать заинтересованного читателя углубить свое понимание с помощью моей книги по алгоритмам [1].

- *Кратчайшие пути.* В "матрице" расстояний m значение $m[i, j]$ должно отражать минимальную длину пути между вершинами i и j. Обратите внимание, что независимые оценки попарных расстояний зачастую противоречивы и не обязательно удовлетворяют условию неравенства треугольника. Но когда $m'[i, j]$ отражает *кратчайшее расстояние* пути от i до j в любой матрице m, оно *должно* удовлетворять условиям метрики. Это может представить лучшую матрицу для анализа, чем оригинал.
- *Связные компоненты.* Каждый непересекающийся участок графа называется *связным компонентом* (connected component). Важно определить, состоит ли ваш граф из одного компонента или нескольких частей. Любые алгоритмы, которые вы запускаете, будут иметь лучшую производительность, если вы будете работать с компонентами независимо. Отдельные компоненты могут быть независимыми по веским причинам, например, из-за океана нет пересечения дорог между Соединенными Штатами и Европой. Но отдельные компоненты могут указывать на проблемы, такие как обработка артефактов или недостаточное подключение для работы.

- *Минимальные остовные деревья. Остовное дерево* (spanning tree) — это минимальное множество ребер, связывающих все вершины графа, по сути, доказательство того, что граф связен. Минимальный вес связующего дерева служит наиболее разреженным представлением структуры графа, что делает его полезным для визуализации. Действительно, в разделе 10.5 мы покажем, что минимальные остовные деревья играют важную роль в алгоритмах кластеризации.
- *Сегментация графа.* Кластер в графе определяется подмножеством вершин c, обладающим тем свойством, что (а) существует значительное сходство между парами вершин в c, и (б) существует слабая связность между вершинами в c и вне c. Ребра (x, y), где $x \in c$ и $y \notin c$, определяют сегментацию, отделяющую кластер от остальной части графа, что делает поиск таких сегментаций важным аспектом кластерного анализа.
- *Паросочетание.* Брак каждой вершины с похожим, верным партнером может быть полезен во многих отношениях. После такого паросочетания становятся возможными интересные типы сравнений. Например, просмотр всех близких пар, различающихся по одному признаку (скажем, по полу), может пролить свет на то, как эта переменная влияет на конкретную переменную результата (например, доход или продолжительность жизни). Паросочетания также предоставляют способы уменьшить эффективный размер сети. Заменив каждую сочетающуюся пару вершиной, представляющей ее центроид, мы можем построить граф с половиной вершин, но все же являющийся представителем целого.
- *Топологическая сортировка.* Проблемы ранжирования (см. главу 4) налагают на коллекцию элементов порядок кеширования в соответствии с некоторыми критериями качества. *Топологическая сортировка* (topological sorting) ранжирует вершины *ориентированного ациклического графа* (Directed Acyclic Graph — DAG) так, что ребро (i, j) подразумевает, что i занимает место выше j в порядке извлечения. Учитывая совокупность наблюдаемых ограничений вида "*я должен стоять выше j*", топологическая сортировка определяет порядок элементов, соответствующий этим наблюдениям.

10.4. PageRank

Нередко полезно классифицировать относительную важность вершин в графе. Возможно, самое простое понятие основано на *степени* вершины, количестве ребер, соединяющих вершину v с остальной частью графа. Чем более связана вершина, тем она, вероятно, важнее.

Степень вершины v дает хорошую функцию для представления элемента, связанного с v. Но еще лучше — *PageRank* [81], оригинальный секретный соус для поисковой системы Google. PageRank игнорирует текстовое содержимое веб-страниц, чтобы сосредоточиться только на структуре гиперссылок между страницами. Конечно, более важные страницы (вершины) должны иметь более высокую степень, чем страницы меньшего размера. Но важность страниц, которые ссылаются на вас, также имеет значение. Большое количество контактов, рекомендующих вас на работу, — это здорово, но еще лучше, когда один из них в настоящее время исполняет обязанности президента Соединенных Штатов.

Лучше всего рассматривать PageRank в контексте случайных прогулок по сети. Предположим, что мы начинаем с произвольной вершины, а затем случайным образом выбираем исходящую ссылку из набора возможных. Теперь повторите процесс отсюда, переходя к случайному соседу нашего текущего местоположения на каждом шаге. PageRank вершины v является мерой вероятности того, что, начиная со случайной вершины, вы достигнете v после длинного ряда таких случайных шагов. Основная формула для PageRank из $v(PR(v))$ такова:

$$PR_j(v) = \sum_{(u,v)\in E} \frac{PR_{j-1}(u)}{out-degree(u)}$$

Это рекурсивная формула с j в качестве номера итерации. Мы инициализируем $PR_0(v_i) = 1/n$ для каждой вершины v_i в сети, где $1 \le i \le n$. Значения PageRank будут меняться при каждой итерации, но неожиданно быстро сходятся к стабильным значениям. Для неориентированных графов эта вероятность, по сути, такая же, как у каждой вершины в градусах, но с ориентированными графами случаются гораздо более интересные вещи.

По сути, PageRank опирается на идею, что если все дороги ведут в Рим, то Рим должен быть довольно важным местом. Это пути *к* вашей странице, которая имеет значение. Это то, что делает PageRank трудным для игр: на вашу веб-страницу должны ссылаться другие люди, а то, что вы кричите о себе, не имеет значения.

Есть несколько настроек, которые можно внести в эту базовую формулу PageRank, чтобы сделать результаты более интересными. Мы можем позволить переход к произвольной вершине (вместо связанного соседа), чтобы обеспечить более быструю диффузию в сети. Пусть p — это вероятность перехода по ссылке на следующем этапе, также известном как *коэффициент затухания* (damping factor). Тогда

$$PR_j(v) = \sum_{u,v\in E} \frac{(1-p)}{n} + p\frac{PR_{j-1}(u)}{out\text{-}degree(u)},$$

где n — количество вершин в графе. Другие улучшения включают внесение изменений в саму сеть. Добавляя ребра из каждой вершины в одну супервершину, мы гарантируем, что случайные блуждания не могут привести в какой-то маленький угол сети. Самостоятельные петли и параллельные ребра (многогранники) могут быть удалены, чтобы избежать смещения из-за повторения.

Есть также линейная алгебраическая интерпретация PageRank. Пусть M — это матрица вероятностей перехода из вершины в вершину, поэтому $M_{i,j}$ — это вероятность того, что наш следующий шаг от i будет к j. Ясно, что $M_{i,j} = 1/outdegree(i)$, если существует направленное ребро от i до j, и нуль в противном случае. Оценка j-го раунда для вектора PageRank PR_j может быть вычислена как

$$PR_j = MPR_{j-1}.$$

После того как эта оценка сходится, $PR = MPR$ или $\lambda U = MU$, где $\lambda = 1$, а U представляет вектор PageRank. Это определяющее уравнение для собственных значений, поэтому вектор значений PageRank $n \times 1$ оказывается основным собственным вектором матрицы вероятности перехода, определяемой ссылками. Таким образом, итерационные методы для вычисления собственных векторов и быстрого умножения матриц приводят к эффективным вычислениям PageRank.

Насколько хорош PageRank в выкуривании наиболее близко расположенных к центру вершин? Чтобы обеспечить некоторое представление, мы запустили PageRank в сети ссылок из английского издания Википедии, сосредоточив внимание на страницах, связанных с людьми. В табл. 10.1 (слева) перечислены десять исторических личностей с самым высоким рейтингом PageRank.

Таблица 10.1. Исторические личности с самым высоким рейтингом PageRank в английской Википедии 2010 года, приведенные на полном графе Википедии (слева) и ограниченные ссылками на других людей (справа)

	PageRank PR1 (все страницы)
1	Наполеон
2	Джордж Буш
3	Карл Линней
4	Иисус
5	Барак Обама
6	Аристотель
7	Уильям Шекспир
8	Елизавета II
9	Адольф Гитлер
10	Билл Клинтон

	PageRank PR2 (только люди)
1	Джордж Буш
2	Билл Клинтон
3	Уильям Шекспир
4	Рональд Рейган
5	Адольф Гитлер
6	Барак Обама
7	Наполеон
8	Ричард Никсон
9	Франклин Д. Рузвельт
10	Елизавета II

Личности, обладающие высокими значениями PageRank, легко признаются очень значимыми людьми. Наименее знакомым среди них, вероятно, является Карл Линней (1707–1778) [46], "отец таксономии" в биологии, система Линнея (роды и виды; например, *Homo sapiens*) используется для классификации всей жизни на Земле. Он был великим ученым, но почему он так высоко ценится PageRank? Страницы Википедии со всеми видами растений и животных, которые он впервые классифицировал, ссылаются на него, поэтому тысячи форм жизни вносят заметный вклад в его страницу.

Пример Линнея указывает на возможную слабость PageRank: мы действительно хотим, чтобы растения и другие неодушевленные предметы голосовали за того, кто самый выдающийся человек? На рис. 10.10 (слева) показано подмножество полного графа PageRank для Барака Обамы [91]. Обратите внимание, например, что, хотя ссылки, связанные с Обамой, кажутся очень разумными, мы все же можем достичь Обамы всего за два клика со страницы Википедии о динозаврах. Должны ли вымершие звери вносить вклад в центральное место президента?

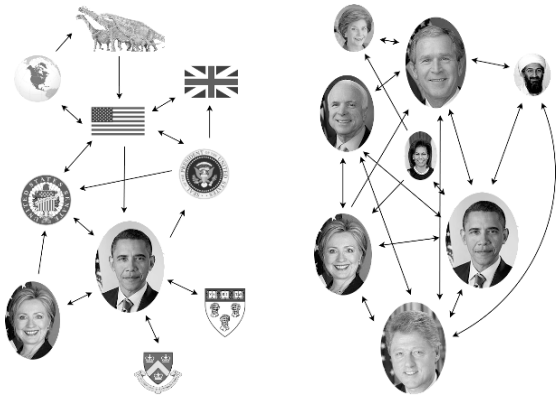


Рис. 10.10. Граф PageRank для Барака Обамы по всем страницам Википедии (слева) и только для людей (справа)

Добавление и удаление наборов ребер из данной сети приводит к появлению разных сетей, некоторые из которых лучше раскрывают основополагающее значение с помощью PageRank. Предположим, мы вычислили рейтинг PageRank (обозначенный как PR2), используя только ребра Википедии, связывающие людей. Это вычисление будет игнорировать любой вклад от мест, организаций и низших организмов. На рис. 10.10 (справа) показан образец графа PageRank для Барака Обамы [91], когда мы ограничиваем его только людьми.

PageRank на этом графе поддерживает несколько иную группу людей, как показано в табл. 10.1 (справа). Иисуса, Линнея и Аристотеля (384–322 гг. до н.э.) [8] теперь нет, их заменили три недавних президента США, которые

явно имеют прямые связи со многими важными людьми. Так какая версия PageRank лучше — PR1 или PR2? Обе, кажется, отражают разумные понятия центральности с существенной, но не подавляющей корреляцией (0,68), поэтому обе имеют смысл в качестве потенциальных возможностей в наборе данных.

10.5. Кластеризация

Кластеризация (clustering) — это проблема группировки точек по сходству. Элементы нередко поступают из небольшого количества логических "источников" или "объяснений", и кластеризация является хорошим способом выявления этих источников. Подумайте, что произойдет, если инопланетяне обнаружат данные о росте и весе большого количества людей. Предположительно, они выяснят, что существует два кластера, представляющих разные группы населения, один из которых последовательно больше другого. Если бы инопланетяне были действительно на высоте, они могли бы назвать эти группы "мужчинами" и "женщинами". Действительно, два кластера роста-веса на рис. 10.11 сильно сконцентрированы в одном конкретном поле.

Рис. 10.11. Кластеризация людей в пространстве вес-рост с использованием 2-групповой кластеризации. В левом кластере 240 женщин и 112 мужчин, а в правом кластере 174 мужчины и 54 женщины. Сравните это с классификатором логистической регрессии, обученным на том же наборе данных, см. рис. 9.17

Шаблоны на двумерном точечном графике, как правило, довольно легко заметить, но мы нередко имеем дело с многомерными данными, которые люди не могут визуализировать эффективно. Теперь нам нужны алгоритмы, которые найдут эти шаблоны за нас. Кластеризация, пожалуй, — это первое, что нужно сделать с любым интересным набором данных. Ее применения таковы.

- *Выработка гипотезы*. Знание того, что в вашем наборе данных (скажем) четыре отдельные группы населения, должно поднять вопрос о том, почему они существуют. Если эти кластеры достаточно компактны и хорошо отделены друг от друга, должна быть причина, и найти ее — ваше дело. Присвоив каждому элементу метку кластера, вы можете изучить несколько представителей одного кластера, чтобы выяснить, что у них общего, или посмотреть на пары элементов из разных кластеров и определить, почему они отличаются.
- *Моделирование более мелких подмножеств данных*. Наборы данных часто содержат очень большое количество строк (n) относительно количества столбцов функций (m): представьте, что данные такси содержат записи о 80 миллионах поездок с десятью полями на каждую поездку. Кластеризация обеспечивает логический способ разделения большого набора записей на, скажем, сто различных подмножеств, каждое из которых упорядочено по сходству. Каждый из этих кластеров по-прежнему содержит более чем достаточно записей, чтобы соответствовать модели прогнозирования, и результирующая модель может быть более точной по этому ограниченному классу элементов, чем общая модель, обученная по всем элементам. Создание прогноза теперь включает в себя определение соответствующего кластера, к которому принадлежит элемент q, за счет поиска ближайшего соседа и последующее использование соответствующей модели для этого кластера, чтобы вызвать q.
- *Сжатие данных*. Обработка или визуализация миллионов или миллиардов записей может быть сложной. Рассмотрим вычислительную стоимость определения ближайшего соседа заданной точки или попытки понять точечный график с миллионом точек. Одним из методов является кластеризация точек по сходству с последующим назначением центроида каждого кластера для представления всего кластера. Такие модели ближайших соседей могут быть достаточно надежными, поскольку вы сообщаете метку консенсуса кластера, и она следует с естественной мерой достоверности: корректностью этого консенсуса по всему кластеру.
- *Обнаружение выбросов*. Некоторые элементы, полученные в результате любой процедуры сбора данных, будут отличаться от всех остальных. Возможно, они отражают ошибки ввода данных или неправильные

измерения. Возможно, они сигнализируют о лжи или других проступках. Или, может быть, они возникли в результате неожиданного смешения популяций, когда несколько странных яблок потенциально испортили всю корзину.

Обнаружение выбросов (outlier detection) — это проблема избавления набора данных от неподходящих элементов, чтобы остаток лучше отражал желаемую совокупность. Кластеризация — это полезный первый шаг для выявления выбросов. Элементы кластера, наиболее удаленные от назначенного им центра кластера, не очень хорошо вписываются в него, но и не подходят лучше в другой. Это делает их кандидатами на выброс. Поскольку интервенты из другой популяции склонны объединяться в группы, мы вполне можем высказывать подозрения в отношении небольших кластеров, центры которых расположены необычно далеко от всех других центров кластеров.

Кластеризация является изначально плохо определенной проблемой, так как правильные кластеры зависят от контекста и взгляда зрителя. Посмотрите на рис. 10.12. Сколько разных кластеров вы там видите? Одни видят три, другие видят девять, а остальные голосуют за практически любое число между ними.

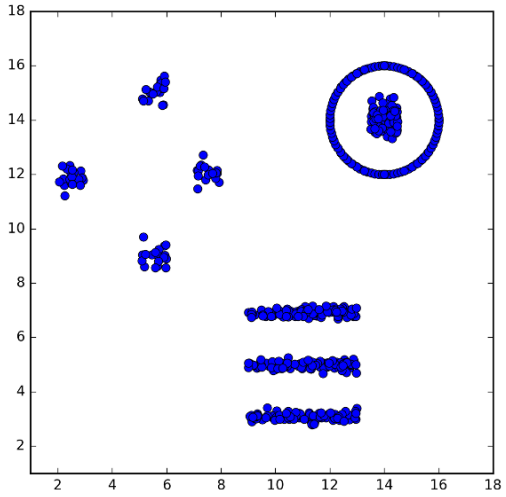


Рис. 10.12. Сколько кластеров вы здесь видите?

То, сколько кластеров вы видите, зависит от того, сколько кластеров вы хотите увидеть. Людей можно объединить в две группы: *объединители* (lumpers) и *разъединители* (splitters), в зависимости от их склонности проводить четкие различия. Разъединители смотрят на собак и видят пуделей, терьеров и кокер-спаниелей. Объединители смотрят на собак и видят млекопитающих.

Разъединители делают более захватывающие выводы, в то время как объединители менее склонны перегонять свои данные. Какой образ мышления является наиболее подходящим, зависит от вашей задачи.

Было разработано много различных алгоритмов кластеризации, и мы рассмотрим наиболее известные методы (метод *k*-средних, агломеративные кластеры и спектральные кластеры) в следующих разделах. Однако увлечься различиями между методами слишком легко. Если ваши данные содержат достаточно сильные кластеры, любой метод найдет что-то подобное. Но когда алгоритм возвращает кластеры с очень плохой согласованностью, обычно виноват ваш набор данных, а не сам алгоритм.

> *На заметку.* Убедитесь, что вы используете метрику расстояния, которая точно отражает те сходства, которые вы хотите найти. Конкретный выбор алгоритма кластеризации обычно оказывается гораздо менее важным, чем мера сходства/расстояния, лежащая в его основе.

10.5.1. Кластеризация методом *k*-средних

Мы не очень точно определили, *что именно* должен дать алгоритм кластеризации в качестве ответа. Одна возможность заключается в том, чтобы пометить каждую точку именем кластера, в котором она находится. Если имеется *k* кластеров, эти метки могут быть целыми числами от 1 до *k*, где точка метки *p* с *i* означает, что она находится в *i*-м кластере. Эквивалентное выходное представление может быть *k* отдельными списками точек, где список *i* представляет все точки в *i*-м кластере.

Но более абстрактное понятие сообщает о *центральной точке* (center point) каждого кластера. Обычно мы думаем о природных кластерах как о компактных гауссовоподобных областях, где есть идеальный центр, определяющий местоположение, где должны быть точки. Учитывая набор этих центров, кластеризация точек становится легкой: просто назначьте каждую точку *p* ближайшей к ней центральной точке C_i. *I*-й кластер состоит из всех точек, ближайший центр которых C_i.

Кластеризация методом *k*-средних — это быстрый, простой для понимания и в целом эффективный подход к кластеризации. Он начинается с предположения о том, где могут находиться кластерные центры, оценивает качество этих центров, а затем уточняет их, чтобы сделать более точные оценки центров.

Алгоритм начинается с предположения, что в данных будет ровно *k* кластеров, а затем переходит к выбору начальных центров для каждого кластера. Возможно, это означает, что случайным образом выбирается *k* точек из набора *S*, состоящего из *n* точек, и назначаются их центрами или выбирается *k*

случайных точек из ограничительной рамки S. Теперь протестируйте каждую из n точек со всеми k центрами и назначьте каждой точке в S ближайший текущий центр. Теперь мы можем вычислить лучшую оценку центра каждого кластера как центра тяжести точек, назначенных ему. Повторяйте так до тех пор, пока назначения кластера не станут достаточно стабильными, предположительно, если они не изменились со времени предыдущего поколения. На рис. 10.13 представлен псевдокод этой процедуры метода k-средних.

Кластеризация методом k-средних

Выберите k точек в качестве начальных центров кластеров $C_1, \ldots, C_k$.
Повторять до схождения {
  Для $1 \leq i \leq k$ назначить точек p_i ближайший центр кластера C_j
  Вычислить центроид C_j' для точек, ближайших к C_j, для $1 \leq j \leq k$
  Для всех $1 \leq j \leq k$ *установить* $C_j = C_j'$
}

Рис. 10.13. Псевдокод алгоритма кластеризации методом k-средних

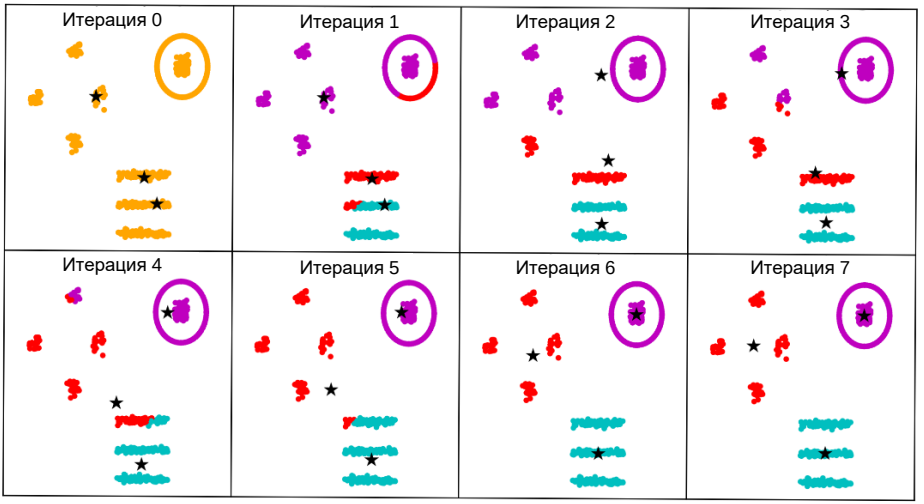


Рис. 10.14. Итерации методом k-средних (для k = 3), так как они сходятся на стабильной и точной кластеризации. Из-за неудачного размещения трех начальных центров кластеров вблизи логического центра требуется всего семь итераций

На рис. 10.14 представлена анимация метода k-средних в действии. Первоначальные предположения о центрах кластеров действительно плохие, и начальные назначения точек центрам разъединяют реальные кластеры, а не объединяют их. Но ситуация быстро улучшается, когда центроиды перемещаются в позиции, которые разделяют точки желаемым образом. Обратите внимание, что процедура k-средних не обязательно заканчивается наилучшим возможным набором из k центров — только при локально-оптимальном решении, которое обеспечивает логическую точку остановки. Рекомендуется повторить всю процедуру несколько раз с разными случайными инициализациями и принять кластеризацию, лучшую из всех найденных. *Среднеквадратичная ошибка* (mean squared error) представляет собой сумму квадратов расстояния между каждой точкой P_i и ее центром C_j, деленную на количество точек n. Лучшая из двух группировок может быть идентифицирована как имеющая меньшую среднеквадратичную ошибку или некоторую другую разумную статистику ошибок.

Центры или центроиды?

Существует как минимум два возможных критерия для вычисления новой оценки центральной точки как функции от набора S' точек. *Центроид* (centroid) C набора точек вычисляется по среднему значению каждого измерения. Для d-го измерения

$$C_d = \frac{1}{|S'|} \sum_{p \in S'} p[d].$$

Центроид служит *центром масс* (center of mass) набора S', т.е. местом, где сумма векторов, определенных через эту точку, является нулевой. Этот критерий баланса определяет естественный и уникальный центр для любого S'. Скорость вычислений — это еще одно преимущество использования центроида. Для n d-мерных точек в S' это занимает время $O(nd)$, что означает линейную зависимость от количества входных точек.

Для числовых точек данных использование центроида для соответствующей метрики L_k (например, евклидового расстояния) должно работать очень хорошо. Однако центроиды не очень эффективны при кластеризации записей данных с нечисловыми атрибутами, такими как категориальные данные. Каков центр тяжести у 7 блондинок, 2 рыжих и 6 седых людей? Мы обсудили, как построить значимые функции расстояния по категориальным записям. Проблема здесь не столько в измерении сходства, сколько в создании представительного центра.

Существует естественное решение, алгоритм *k-медоидов* (k-mediods). Предположим, что мы определяем вместо центроида как представитель кластера

центральную точку C в S'. Это точка, которая минимизирует сумму расстояний до всех других точек в кластере:

$$C = \arg\min_{c \in S'} \sum_{i=1}^{n} d(c, p_i).$$

Преимущество использования центральной точки для определения кластера заключается в том, что кластер получает потенциальное имя и идентификатор, предполагая, что входные точки соответствуют элементам с идентифицируемыми именами.

Используя ближайший к центру входной пример в качестве центра, мы можем применить кластеризацию методом *k*-средних, если у нас есть значимая функция расстояния. Кроме того, мы не особенно теряем точность, выбирая центральную точку вместо центроида. В действительности, сумма чисел, проходящих через центральную точку, в два раза больше, чем у центроида, на численных примерах, где центроид может быть вычислен. Большая победа центроида в том, что он может быть вычислен быстрее, чем центральная вершина, с коэффициентом *n*.

Использование *центральных вершин* (center vertices) для представления кластеров позволяет естественным образом распространить метод *k*-средних на графы и сети. Для взвешенных графов естественно использовать алгоритм кратчайшего пути для построения такой матрицы D, где $D[i, j]$ — это длина кратчайшего пути в графе от вершины i до вершины j. После построения матрицы D метод *k*-средних можно продолжить, считывая расстояния из этой матрицы, а не вызывая функции расстояния. У невзвешенных графов для вычисления расстояний по требованию можно эффективно использовать линейный алгоритм времени, такой как поиск в ширину.

Сколько кластеров?

Интерпретации кластеризации *k*-средних присуща идея *смешанной модели* (mixture model). Вместо всех наших данных наблюдений, поступающих из одного источника, мы предполагаем, что наши данные поступают из *k* различных групп населения или источников. Каждый источник генерирует точки, примерно из его центра, но с некоторой степенью изменения или ошибки. Вопрос о том, сколько кластеров в наборе данных имеет основополагающее значение: сколько разных групп населения было выбрано при выборке?

Первым шагом в алгоритме *k*-средних является инициализация *k*, количества кластеров в данном наборе данных. Иногда у нас есть предварительное представление о том, сколько кластеров мы хотим видеть: возможно, два или три для баланса или визуализации, либо 100 или 1000 в качестве прокси для "многих" при разбиении большого входного файла на меньшие наборы для отдельного моделирования.

Однако, по правде говоря, это проблема, поскольку "правильное" количество кластеров обычно неизвестно. Главной причиной кластеризации является наше ограниченное понимание структуры набора данных.

Самый простой способ найти правильное значение k — это попробовать их все, а затем выбрать лучшее. Начиная с $k = 2$ и до того уровня, на который, как вы считаете, у вас есть время, выполните метод k-средних и оцените результирующую кластеризацию в соответствии со среднеквадратической ошибкой (MSE) точек от их центров. Построение графика дает кривую ошибки, как показано на рис. 10.16. Кривая ошибки для случайных центров также предоставляется.

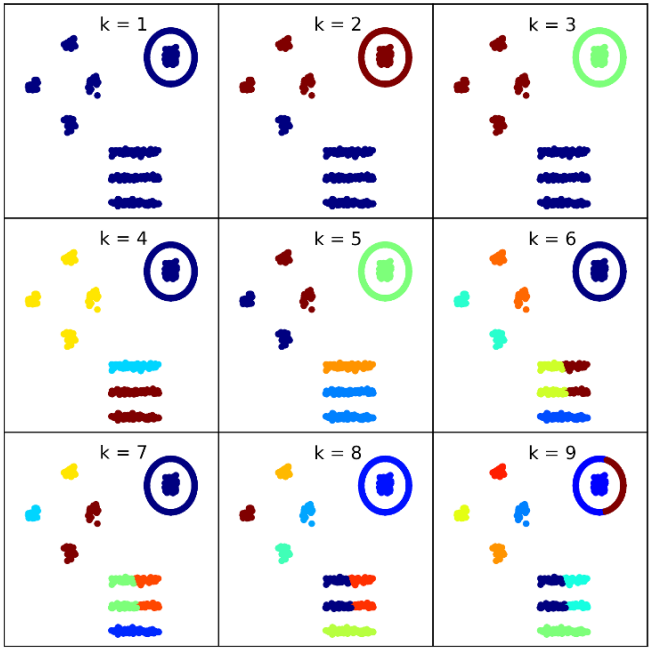


Рис. 10.15. Выполнение метода k-средних для k = 1 до k = 9. "Правильная" кластеризация найдена для k = 3, но алгоритм не может правильно различить вложенные круговые кластеры и длинные тонкие кластеры для больших k

Обе кривые ошибок показывают, что среднеквадратическая ошибка точек от их центров уменьшается, поскольку мы допускаем все больше и больше кластерных центров. Но неправильная интерпретация заключалась бы в том, чтобы предположить, что нам нужно k как можно большего размера, поскольку среднеквадратическая ошибка должна уменьшаться при разрешении большего количества центров. Вставка нового центра в случайной позиции r в предыдущее решение k-средних может только уменьшить среднеквадратичную

ошибку, поскольку она оказывается ближе к нескольким исходным точкам, чем их предыдущий центр. Это создает новый кластер вокруг r, но, вероятно, еще лучшая кластеризация была бы обнаружена с помощью k-средних с нуля на $(k + 1)$ центрах.

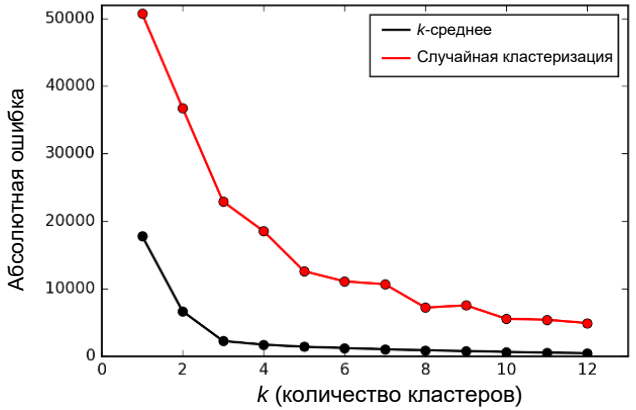


Рис. 10.16. Кривая ошибки для кластеризации k-средних в наборе точек на рис. 10.12, демонстрирующая изгиб, отражающий три основных кластера в данных. Кривая ошибки для случайных центров кластеров показана для сравнения

То, что мы ищем по кривой ошибок на рис. 10.16, — это значение k, где *скорость* изменения наклона уменьшается, поскольку мы превысили количество истинных источников, и поэтому каждый дополнительный центр действует как случайная точка в предыдущем обсуждении. Кривая ошибки должна выглядеть примерно так, как полусогнутая рука: она быстро снижается вниз от плеча к локтю, а затем медленнее от локтя до запястья. Мы хотим, чтобы k было расположено точно в локте. Эту точку легче определить, если сравнивать с аналогичным графиком среднеквадратических ошибок для случайных центров, поскольку относительная скорость уменьшения ошибок для случайных центров должна быть аналогична той, которую мы видим за локтем. Медленный нисходящий дрейф говорит нам о том, что дополнительные кластеры не делают для нас ничего особенного.

Каждый новый центр кластера добавляет в модель d параметров, где d — это размерность набора точек. Бритва Оккама говорит нам, что лучшая модель — это самая простая, что является философской основой для использования изгиба в локте для выбора k. Существуют формальные критерии качества, которые включают в себя как количество параметров, так и погрешность прогнозирования для оценки моделей, например *информационный критерий Акаике* (Akaike Information Criteria — AIC). Однако на практике вы должны быть уверены в том, что сделаете разумный выбор для k на основе формы кривой ошибки.

Максимизация ожидания

Алгоритм *k*-средних является наиболее ярким примером класса алгоритмов обучения, основанных на *максимизации ожидания* (Expectation Maximization — EM). Детали требуют более формальной статистики, чем я готов описывать здесь, но принцип можно наблюдать в двух логических этапах алгоритма *k*-средних: (а) назначение точек предполагаемому центру кластера, который находится ближе всего к ним, и (б) использование этих присвоенных точек для улучшения оценки центра кластера. Операция присваивания является ожиданием, или *E-шагом* (E-step), алгоритма, в то время как вычисление центроида является максимизацией параметра, или *M-шагом* (M-step).

Термины *ожидание* и *максимизация* не имеют для меня особого смысла с точки зрения алгоритма *k*-средних. Однако общая форма итерационного алгоритма подбора параметров, который улучшает параметры в раундах на основе ошибок предыдущих моделей, кажется разумной вещью. Например, мы, возможно, могли бы частично пометить классификационные данные, где относительно мало обучающих примеров, уверенно приписанными к правильному классу. Мы можем построить классификаторы на основе этих обучающих примеров и использовать их для назначения немаркированных точек классам кандидатов. Предположительно это определяет большие обучающие наборы, поэтому мы должны быть в состоянии подобрать лучшую модель для каждого класса. Теперь переназначение точек и повторение итераций должны привести к лучшей модели.

10.5.2. Агломерационная кластеризация

Многие источники данных генерируются в ходе процесса, определенного базовой иерархией, или таксономией. Зачастую это результат эволюционного процесса: в начале было одно, что неоднократно раздваивалось для создания богатой вселенной элементов. Все виды животных и растений являются результатом эволюционного процесса, как и человеческие языки и культурно-этические группы. В меньшей, но все же в реальной степени, такие же элементы, как фильмы и книги. Эту книгу можно описать как "Учебник по науке о данных", что является новым поджанром, происходящим от "Учебника по компьютерным наукам", который логически восходит к "Учебнику по инженерным дисциплинам", "Учебнику", "Нехудожественной литературе". В конце концов мы вернемся к первоисточнику: возможно, "Книга".

В идеале в ходе кластеризации элементов мы будем реконструировать эти эволюционные истории. Эта цель очевидна в *агломерационной кластеризации* (agglomerative clustering), коллекции восходящих методов, которые многократно объединяют два ближайших кластера в более крупный суперкластер,

определяя корневое дерево, листья которого являются отдельными элементами, а корень определяет статистическую совокупность.

На рис. 10.17 иллюстрируется агломерационная кластеризация, применяемая к данным экспрессии генов. Здесь каждый столбец представляет определенный ген, а каждая строка — результаты эксперимента, измеряющего, насколько активен каждый ген в определенном состоянии. В качестве аналогии скажем, что каждая из колонок представляла разных людей, и один отдельный ряд оценивал их настроение сразу после выборов. Поклонники победившей стороны будут более взволнованы, чем обычно (зеленый), в то время как избиратели проигравшей команды будут подавлены (красный). Большинству остального мира все равно (черный). Как с людьми, так и с генами: разные вещи включают и выключают их, и анализ данных экспрессии генов может выявить, что заставляет их тикать.

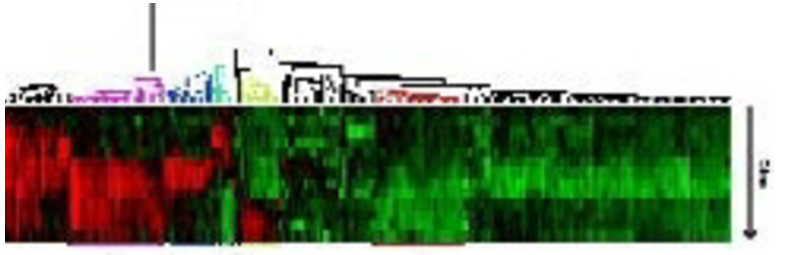


Рис. 10.17. Агломерационная кластеризация данных генной экспрессии

Итак, как нам прочитать рис. 10.17? Из проверки видно, что существуют блоки столбцов, которые ведут себя одинаково, включаются и выключаются в одинаковых условиях. Обнаружение этих блоков отражается в дереве над матрицей: области большого сходства связаны с небольшими ветвлениями. Каждый узел дерева представляет собой объединение двух кластеров. Высота узла пропорциональна расстоянию между двумя объединяемыми кластерами. Чем выше ребро, тем хитрее представление о том, что эти кластеры *должны* быть объединены. Столбцы матрицы переставлены для отражения этой древовидной организации, что позволяет нам визуализировать сотни генов, количественно измеренных в четырнадцати измерениях (каждая строка определяет отдельное измерение).

Биологические кластеры нередко связаны с такими *дендограммами* (dendograms) или *филогенетическими деревьями* (phylogenic trees), поскольку они являются результатом эволюционного процесса. Действительно, наблюдаемые здесь кластеры схожего поведения экспрессии генов являются результатами развития организмом новой функции, которая изменяет реакцию определенных генов на конкретное состояние.

Использование агломерационных деревьев

Агломерационная кластеризация возвращает дерево из группировок элементов. После обрезки самых длинных ребер в этом дереве остаются непересекающиеся группы элементов, созданные с помощью алгоритмов кластеризации, таких как *k*-средних. Но это дерево — изумительная вещь, обладающая способностями, выходящими далеко за рамки разделения элементов.

- *Организация кластеров и подкластеров.* Каждый внутренний узел в дереве определяет определенный кластер, состоящий из всех элементов листового узла под ним. Но дерево описывает иерархию среди этих кластеров, от наиболее специфических кластеров около листьев до самых общих кластеров около корня. В идеале узлы дерева определяют именуемые понятия: естественные группировки, которые эксперт по предметной области может объяснить, если его попросят. Эти различные уровни детализации важны, поскольку они определяют структурные концепции, которые мы, возможно, не заметили до кластеризации.
- *Визуализация процесса кластеризации.* Рисунок этого дерева агломерации многое говорит нам о процессе кластеризации, особенно если этот рисунок отражает стоимость каждого шага слияния. В идеале около корня дерева должны быть очень длинные ребра, показывающие, что кластеры самого высокого уровня хорошо разделены и относятся к разным группам. Мы можем сказать, являются ли группировки сбалансированными, или же группы высокого уровня имеют существенно разные размеры. Длинные цепочки слияния небольших кластеров в большой кластер, как правило, являются плохим признаком, хотя выбор критериев слияния (будет обсуждаться ниже) может повлиять на форму дерева. Выбросы хорошо видны на филогенном дереве в виде одноэлементных элементов или небольших скоплений, которые соединяются около корня через длинные ребра.
- *Естественная мера кластерного расстояния.* Интересным свойством любого дерева T является то, что между любыми двумя узлами x и y существует ровно один путь в T. Каждая внутренняя вершина в дереве агломерационной кластеризации имеет связанный с ней вес — стоимость объединения двух поддеревьев под ним. Мы можем вычислить "кластерное расстояние" между любыми двумя листами по сумме затрат на слияние по пути между ними. Если дерево хорошее, это может быть более значимым, чем евклидово расстояние между записями, связанными с x и y.
- *Эффективная классификация новых элементов.* Одной из важнейших областей применения кластеризации является классификация. Предположим, что продукты в магазине сгруппированы агломеративно, чтобы построить таксономию кластеров. Теперь поступает новый товар. К какой категории его следует отнести?

Для k-средних каждый из c кластеров классифицируется по их центроидам, поэтому классификация нового элемента q сводится к вычислению расстояния между q и всеми c центроидами для определения ближайшего кластера. Иерархическое дерево обеспечивает потенциально более быстрый метод. Предположим, что мы предварительно вычислили центроиды всех листьев на левом и правом поддеревьях под каждым узлом. Определение правильной позиции в иерархии для нового элемента q начинается со сравнения q с центроидами левого и правого поддеревьев корня. Ближайший из двух центроидов к q определяет соответствующую сторону дерева, поэтому мы возобновляем поиск там на один уровень вниз. Этот поиск занимает время, пропорциональное высоте дерева, а не количеству листьев. Обычно это усовершенствование от n до $\log n$, что намного лучше.

Поймите, что двоичные деревья слияния могут быть нарисованы многими различными способами, которые отражают одну и ту же структуру, поскольку не существует понятия о том, кто является левым потомком, а кто правым. Это означает, что возможно 2^{n-1} различных перестановок из n листьев в ходе переключения направления любого подмножества из $n-1$ внутренних узлов в дереве. Осознайте это, когда пытаетесь прочесть такую таксономию: два элемента, которые выглядят весьма удаленными в лево-правом порядке, вполне могли бы быть соседями, если бы такое разделение было сделано по-другому. И крайний справа узел левого поддерева может быть рядом с крайним слева узлом правого поддерева, даже если они действительно довольно далеко друг от друга в таксономии.

Создание агломерационных кластерных деревьев

Основной алгоритм агломерационной кластеризации достаточно прост, чтобы быть описанным в двух предложениях. Первоначально каждый элемент назначается своему кластеру. Объедините два ближайших кластера в один, поместив над ними корень, и повторяйте так, пока не останется только один кластер.

Осталось только указать, как рассчитать расстояние между кластерами. Когда кластеры содержат отдельные элементы, ответ прост: используйте вашу любимую метрику расстояния, такую как L_2. Но есть несколько разумных ответов для расстояния между двумя нетривиальными кластерами, которые приводят к различным деревьям на одном входе и могут оказать глубокое влияние на форму получающихся кластеров. Ведущими кандидатами, показанными на рис. 10.18, являются:

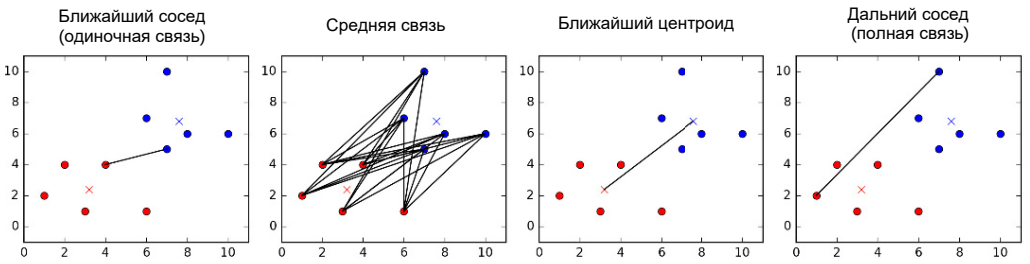


Рис. 10.18. Четыре меры расстояния для определения ближайшей пары кластеров

- *Ближайший сосед (одиночная связь)*. Здесь расстояние между кластерами C_1 и C_2 определяется ближайшей парой точек, охватывающих их:

$$d(C_1, C_2) = \min_{x \in C_1, y \in C_2} \| x - y \|.$$

Использование этой метрики называется *кластеризацией с одной связью* (single link clustering), поскольку решение о слиянии основывается исключительно на одной ближайшей связи между кластерами.

Минимальное остовное дерево (Minimum Spanning Tree — MST) графа G — это дерево, нарисованное по ребрам графа G, соединяющим все вершины с наименьшей общей стоимостью. Агломеративная кластеризация с критериями одиночной связи, по сути, аналогична *алгоритму Краскала* (Kruskal's algorithm), который создает минимальное остовное дерево графа за счет многократного добавления оставшегося ребра с наименьшим весом, что не создает цикл в получаемом дереве.

Связь между MST (с n узлами и $n - 1$ ребрами) и деревом кластера (с n листьями, $n - 1$ внутренними узлами и $2n - 2$ ребрами) несколько тонкая: порядок вставки ребер в MST от наименьшего к наибольшему описывает порядок слияния в дереве кластеров, как показано на рис. 10.20.

Платоновский идеал кластеров — это такие компактные круговые области, которые обычно излучаются из центроидов, как в кластеризации k-средних. В отличие от этого односвязная кластеризация имеет тенденцию создавать относительно длинные узкие кластеры, поскольку решение о слиянии основывается только на близости граничных точек. Кластеризация по одной связи быстра, но она склонна к ошибкам, так как точки выброса могут легко впитать два четко определенных кластера.

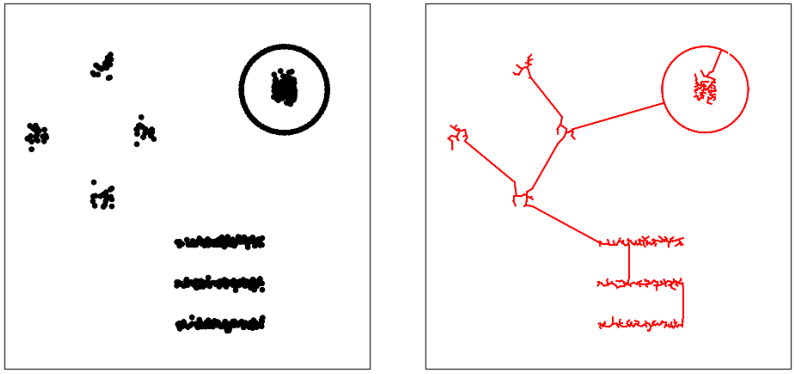


Рис. 10.19. Кластеризация с одной связью эквивалентна нахождению минимального связующего дерева сети

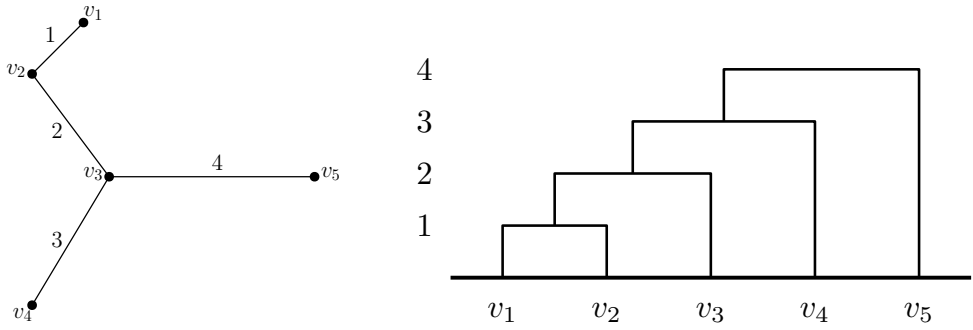


Рис. 10.20. Алгоритм Краскала для минимального связующего дерева действительно представляет собой агломеративную кластеризацию с одним звеном, как показано справа на дереве кластеров

- *Средняя связь*. Здесь мы вычисляем расстояние между всеми парами точек области кластера и усредняем их для более надежных критериев слияния, чем для одной связи:

$$d(C_1, C_2) = \frac{1}{|C_1||C_2|}\sum_{x \in C_1}\sum_{y \in C_2} \| x - y \|.$$

  Это, как правило, позволяет избегать узких кластеров одиночной связи, но с большими вычислительными затратами. Простая реализация кластеризации средней связи составляет $O(n^3)$, поскольку каждое из n слияний может потребовать касания $O(n^2)$ ребер для пересчета ближайшего оставшегося кластера. Это в n раз медленнее, чем кластеризация по одиночной связи, которая может быть реализована за $O(n^2)$ раз.

- *Ближайший центроид*. Здесь мы обеспечиваем центроид каждого кластера и объединяем пару кластеров с ближайшими центроидами. Это

имеет два основных преимущества. Во-первых, он имеет тенденцию создавать кластеры, аналогичные средней связи, поскольку точки выброса в кластере перегружаются при увеличении размера кластера (количества точек). Во-вторых, сравнивать центроиды двух кластеров намного быстрее, чем проверять все $|C_1 \| C_2|$ пары точек в простейшей реализации. Конечно, центроиды могут быть вычислены только для записей со всеми числовыми значениями, но алгоритм может быть адаптирован для использования центральной точки в каждом кластере (медоид) в качестве представителя в общем случае.

- *Дальний сосед (полная связь)*. Здесь стоимость объединения двух кластеров — это самая дальняя пара точек между ними:

$$d(C_1, C_2) = \max_{x \in C_1, y \in C_2} \| x - y \|.$$

Это звучит как безумие, но данный критерий лучше всего работает для удержания кластеров, штрафуя слияния с отдаленными элементами.

Какой из них лучше? Как всегда, это зависит от обстоятельств. Для очень больших наборов данных мы больше всего заинтересованы в использовании самых быстрых алгоритмов, которые обычно представляют собой одиночную связь или ближайший центроид с соответствующими структурами данных. В небольших и скромных наборах данных нас больше всего заботит качество, что делает более привлекательными надежные методы.

10.5.3. Сравнение кластеров

Вполне обычной практикой является использование нескольких алгоритмов кластеризации для одного набора данных и применение того из них, который лучше всего подходит для наших целей. Кластеризация, создаваемая двумя разными алгоритмами, должна быть довольно схожей, если оба алгоритма делают разумные вещи, но зачастую интересно измерить, насколько они похожи. Это означает, что нам нужно определить меру сходства или различия для кластеров.

Каждый кластер определяется подмножеством элементов, будь то точки или записи. *Жаккарово подобие* (Jaccard similarity) $J(s_1, s_2)$ множеств s_1 и s_2 определяется как отношение их пересечения и объединения:

$$J(s_1, s_2) = \frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}.$$

Поскольку пересечение двух множеств всегда не больше, чем объединение их элементов, $0 \le J(s_1, s_2) \le 1$, Жаккарово подобие — это, как правило, полезная мера, о которой нужно знать, например, при сравнении подобия k точек

ближайших соседей по двум разным метрикам расстояния или того, как часто верхние элементы по одному критерию соответствуют верхним элементам по другой метрике.

Эту меру подобия можно превратить в правильную метрику расстояния $d(s_1, s_2)$, известную как *расстояние Жаккара* (Jaccard distance), где

$$d(s_1, s_2) = 1 - J(s_1, s_2).$$

Эта функция расстояния принимает значения только от 0 до 1, но удовлетворяет всем свойствам метрики, включая неравенство треугольника.

Каждая кластеризация описывается разделом универсального набора и может иметь много частей. *Индекс Рэнда* (Rand index) является естественной мерой сходства между двумя кластеризациями c_1 и c_2. Если кластеризации совместимы, то любая пара элементов в одном и том же подмножестве c_1 должна быть в одном и том же подмножестве c_2, а любые пары в разных кластерах c_1 должны быть разделены в c_2. Индекс Рэнда подсчитывает количество таких непротиворечивых пар элементов и делит его на общее количество пар $\binom{n}{2}$, чтобы создать соотношение от 0 до 1, где 1 обозначает идентичные кластеризации.

10.5.4. Подобие графов и кластеризация на основе сегментации

Вспомните наше первоначальное обсуждение кластеризации, где я спросил, сколько кластеров вы видите в наборе точек, повторенном на рис. 10.21. Чтобы придумать разумный ответ из девяти кластеров, ваш внутренний алгоритм кластеризации должен был обладать такими трюками, как классификация кольца вокруг центрального объекта в качестве двух отдельных кластеров, и избегать слияния двух линий, которые располагаются подозрительно близко друг к другу. У *k*-среднего нет шансов сделать это, как показано на рис. 10.21 (слева), потому что он всегда ищет круговые кластеры и с удовольствием разделяет длинные цепочки. Из процедур агломерационной кластеризации только одиночная связь с совершенно правильным порогом могла бы иметь шанс сделать все правильно, но ее легко обмануть объединением двух кластеров одной парой близких точек.

Кластеры не всегда круглые. Признание других требует достаточно высокой плотности точек, которые достаточно *смежны* (contiguous), чтобы мы не испытывали искушения разрезать кластер на две части. Мы ищем кластеры, которые *связаны* (connected) в соответствующем графе подобия.

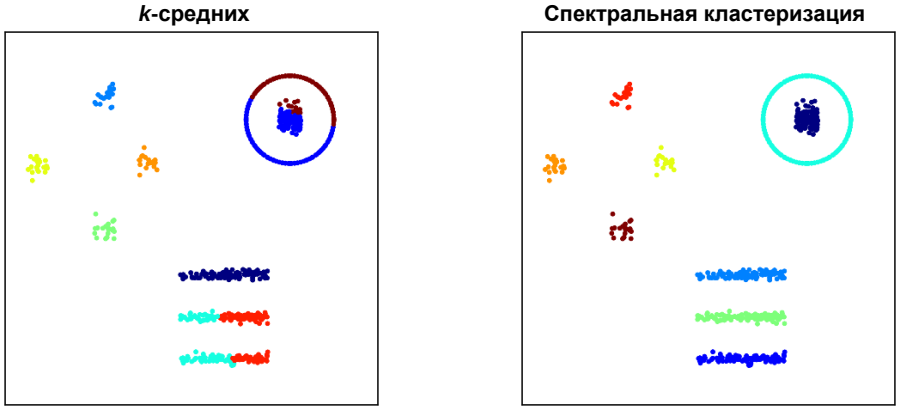


Рис. 10.21. Результаты k-средних (слева) и спектральной кластеризации на основе сегментации (справа) в нашем примере с 9 кластерами. Здесь спектральная кластеризация правильно находит связанные кластеры, чего k-средние не могут

Матрица подобия (similarity matrix) S размером $n \times n$ оценивает, насколько похожа каждая пара элементов p_i и p_j. Сходство по сути является подобным расстоянию: когда p_i близко к p_j, то элемент, связанный с p_i, должен быть похож на элемент p_j. Вполне естественно измерять сходство по шкале от 0 до 1, где 0 означает абсолютно разные значения, а 1 — одинаковые. Это можно реализовать, сделав $S[i, j]$ обратной экспоненциальной функцией расстояния, регулируемой параметром b:

$$S[i, j] = e^{-\beta \|p_i - p_j\|}.$$

Это работает, потому что $e^0 = 1$ и $e^{-x} = 1/e^x \to 0$ при $x \to \infty$.

Граф подобия (similarity graph) имеет взвешенное ребро (i, j) между каждой парой вершин i и j, отражающее сходство чисел p_i и p_j. Это точно то же, что и матрица сходства, описанная выше. Однако мы можем сделать этот граф разреженным, обнулив все маленькие члены ($S[i, j] \le t$ для некоторого порога t). Это значительно уменьшает количество ребер в графе. Мы даже можем превратить его в невзвешенный граф, установив вес 1 для всех $S[i, j] > t$.

Сегментации в графиках

Реальные кластеры в графах подобия имеют вид плотных областей, которые слабо связаны с остальной частью графа. Кластер C имеет вес, который является функцией ребер внутри кластера:

$$W(C) = \sum_{x \in C} \sum_{y \in C} S[i, j].$$

Ребра, соединяющие C с остальной частью графа, определяют *сегментацию* (cut), т.е. набор ребер, у которых одна вершина находится в C, а другая — в остальной части графа ($V - C$). Вес этой сегментации $W'(C)$ определяется как:

$$W'(C) = \sum_{x \in C} \sum_{y \in V-C} S[i, j].$$

В идеале кластеры должны иметь большой вес $W(C)$, а сегментация $W'(C)$ — небольшой, как показано на рис. 10.22. *Проводимость* (conductance) кластера C представляет собой отношение массы сегментации к внутренней массе $W'(C)/W(C)$, причем лучшие кластеры имеют более низкую проводимость.

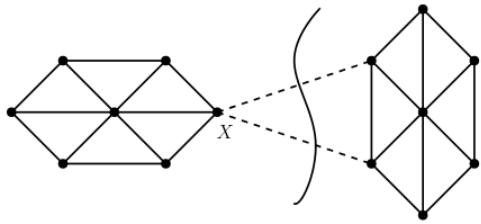


Рис. 10.22. Легковесная сегментация в графах сходства идентифицируют естественные кластеры

Поиск кластеров с низкой проводимостью является сложной задачей. Удивительно, но помощь приходит от линейной алгебры. Матрица подобия S является симметричной матрицей, что означает, что она имеет разложение по собственным значениям, как обсуждалось в разделе 8.5. Мы видели, что ведущий собственный вектор приводит к блочному приближению к S, с вкладом дополнительных собственных векторов, постепенно улучшающим приближение. Удаление самых маленьких собственных векторов удаляет либо детали, либо шум, в зависимости от интерпретации.

Обратите внимание, что идеальная матрица подобия является блочной матрицей, потому что внутри каждого кластера мы ожидаем плотную связь очень похожих пар с небольшим перекрестным взаимодействием с вершинами других кластеров. Это предполагает использование собственных векторов S для определения надежных признаков при кластеризации вершин. Выполнение кластеризации k-средних в этом преобразованном пространстве признаков восстановит хорошие кластеры.

Это подход *спектральной кластеризации* (spectral clustering). Мы строим соответственно нормированную матрицу подобия, *матрицу Лапласа* (Laplacian matrix), где $L = D - S$, а D — взвешенная по степени единичная матрица, т.е. $D[i, j] = \sum_j S[i, j]$. K наиболее важных собственных векторов L определяют

матрицу признаков $n \times k$. Любопытно, что наиболее ценные собственные векторы для кластеризации здесь, как оказалось, имеют наименьшие ненулевые собственные значения благодаря особым свойствам матрицы Лапласа. Выполнение кластеризации методом k-средних в этом пространстве признаков генерирует сильно связанные кластеры.

На заметку. Какой правильный алгоритм кластеризации использовать для ваших данных? Есть много возможностей для рассмотрения, но самые важные решения таковы.

- Какую функцию расстояния использовать?
- Что вы делаете для правильной нормализации переменных?
- Выглядят ли ваши кластеры разумно при соответствующей визуализации? Поймите, что кластеризация никогда не бывает идеальной, потому что алгоритм не может читать ваши мысли. Но достаточно ли все хорошо?

10.6. Случай из жизни: кластерная бомбардировка

Во время моего творческого отпуска моим руководителем в исследовательских лабораториях крупной медиатехнологической компании была Аманда Стент (Amanda Stent), лидер группы по *обработке естественного языка* (Natural Language Processing — NLP). Она исключительно эффективна, чрезвычайно вежлива и обычно невозмутима. Но и у ее терпения есть предел, однажды я услышал раздражение в ее голосе, когда она пробормотала: "Люди продукта!"

Частично ее задача в лаборатории заключалась во взаимодействии с группами продуктов компании, которые нуждались в знании языковых технологий. Виноватым здесь оказался новостной продукт, ответственный за показ пользователям последних статей, представляющих для них интерес. Модуль кластеризации статей был важной частью этих усилий, потому что он группировал все статьи, написанные об одной и той же истории/событии. Пользователи не хотели читать десять разных статей об одной бейсбольной игре или интернет-новости. Отображение пользователям повторяющихся историй из одного кластера статей оказалось очень раздражающим и отпугивало их от нашего сайта.

Но кластеризация статей помогает только тогда, когда сами кластеры точны.

"Это уже третий раз, когда они приходят ко мне с жалобами на кластеризацию. Они никогда не дают мне конкретных примеров того, что не так, просто

жалуются, что кластеры не достаточно хороши. Они продолжают присылать мне ссылки на сообщения, которые они находят в *Stack Overflow* о новых алгоритмах кластеризации, и спрашивают, должны ли мы использовать их вместо этого".

Я согласился поговорить с ними за нее.

Во-первых, я позаботился о том, чтобы специалисты по продукту поняли, что кластеризация — это плохо определенная проблема, и что независимо от того, какой алгоритм они использовали, будут случайные ошибки, с которыми им придется мириться. Это не означало, что по сравнению с их нынешним алгоритмом кластеризации нет места для улучшения, но им придется умерить любые мечты о совершенстве.

Во-вторых, я сказал им, что мы не можем надеяться решить проблему, пока нам не дадут четких примеров того, что именно идет не так. Я попросил у них двадцать примеров пар статей, которые были сгруппированы по алгоритму, но не должны были. И еще двадцать примеров пар статей, которые принадлежали к одному кластеру, но это сходство не было распознано алгоритмом.

Это дало желаемый эффект. Они с готовностью согласились, что мои требования были разумными и необходимыми для диагностики проблемы. Они сказали мне, что получат право на это. Но это требовало работы с их стороны, а все были слишком заняты. Поэтому я так и не получал от них ответа, и мне удалось провести остаток творческого отпуска в продуктивном мире.

Несколько месяцев спустя Аманда сказала мне, что она снова говорила с людьми, производящими этот продукт. Кто-то обнаружил, что их модуль кластеризации использует в качестве функций слова только из заголовков и игнорирует все содержание самой статьи. В самом алгоритме не было ничего плохого, только в функциях, и вскоре он заработал намного лучше, поскольку ему был предоставлен более богатый набор функций.

Какова мораль этой сказки? Человек должен знать свои пределы, как и алгоритм кластеризации. Зайдите на новости Google (Google News) прямо сейчас и внимательно изучите кластеры статей. Если у вас проницательный взгляд, вы обнаружите несколько небольших ошибок и, возможно, что-то действительно смущающее. Но более удивительным является то, насколько хорошо это работает на общем фоне, и что вы можете производить информативную, а не избыточную ленту новостей, алгоритмически созданную из тысяч различных источников. Эффективная кластеризация никогда не бывает идеальной, но может быть очень ценной.

Вторая мораль заключается в том, что инженерные функции и функции расстояния имеют значение для кластеризации гораздо больше, чем конкретный алгоритмический подход. Эти люди мечтали о мощном алгоритме, который ре-

шал бы все их проблемы, но только по заголовкам. Заголовки предназначены для привлечения внимания, а не для объяснения истории. Лучшие газетные заголовки в истории, такие как "Безголовое тело, найденное в баре" и "Форд — городу: отвяжись!", было бы невозможно сопоставить с более трезвыми людьми, связанными с теми же историями.

10.7. Дополнительная информация

Расчеты расстояния являются основой области вычислительной геометрии, изучения алгоритмов и структур данных для манипулирования множествами точек. Отличное введение в вычислительную геометрию — это [82, 83].

Самет [84] — это лучший справочник по kd-деревьям и другим структурам пространственных данных для поиска ближайших соседей. Все основные (и многие второстепенные) варианты разрабатываются достаточно подробно. Доступен также и более короткий обзор [85]. Индык (Indyk) [86] умело рассматривает последние результаты в области приближенного поиска ближайшего соседа в больших измерениях, основываясь на методах случайной проекции.

Теория графов — это изучение абстрактных свойств графов, и Вест [87] служит отличным введением. Сети представляют собой эмпирические связи между сущностями реального мира информацией о них. Исли и Клейнберг [88] обсуждают основы науки о сетях в обществе.

Кластеризация, известная также как *кластерный анализ* (cluster analysis), является классической темой в статистике и информатике. Типичные способы обработки излагают Эверитт и др. [89], а также Джеймс и др. [21].

10.8. Упражнения

Метрики расстояния

10.1. *[3]* Докажите, что евклидово расстояние на самом деле является метрикой.

10.2. *[5]* Докажите, что расстояние L_p является метрикой для всех $p \geq 1$.

10.3. *[5]* Докажите, что взвешенное по размеру расстояние L_p является метрикой для всех $p \geq 1$.

10.4. *[3]* Поэкспериментируйте с данными, чтобы убедиться, что (a) косинусное расстояние не является истинной метрикой расстояния и (b) угловое расстояние является метрикой расстояния.

10.5. *[5]* Докажите, что редакторское расстояние текстовых строк определяет метрику.

10.6. *[8]* Покажите, что ожидаемое расстояние между двумя точками, выбранными равномерно и независимо от линии длиной 1, составляет 1/3. Установите убедительные верхние и нижние границы этого ожидаемого расстояния для частичного кредита.

Классификация ближайших соседей

10.7. *[3]* Какое максимальное количество ближайших соседей может иметь данная точка p в двух измерениях, если предположить возможность связей?

10.8. *[5]* В продолжение предыдущего вопроса, каково максимальное количество различных точек, которые могут иметь данную точку p в качестве ближайшего соседа, опять же в двух измерениях?

10.9. *[3]* Постройте набор из точек $n \geq 10$ для двух классов в двух измерениях, где каждая точка будет ошибочно классифицирована в соответствии с ближайшим соседом.

10.10. *[5]* Повторите предыдущий вопрос, но теперь мы классифицируем каждую точку в соответствии с ее тремя ближайшими соседями ($k = 3$).

10.11. *[5]* Предположим, что двухклассный классификатор ближайших соседей с $k = 1$ обучен как минимум на трех положительных и как минимум трех отрицательных точках.

(a) Может ли этот классификатор пометить все новые примеры как положительные?

(b) Что, если $k = 3$?

Сети

10.12. *[3]* Дайте объяснение того, какими могут быть узлы с наибольшими входящей и исходящей степенями узла в следующих графах:

(a) граф телефонной сети, где ребро (x, y) означает, что x вызывает y;

(b) граф Твиттера, где ребро (x, y) означает, что x следует за y.

10.13. *[3]* Распределение по степенному закону степеней вершин в сетях обычно является результатом механизма *предпочтительного присоединения* (preferential attachment), с помощью которого новые ребра с большей вероятностью соединяются с узлами высокой степени. Для каждого из следующих графов предложите, каково распределение степеней их вершин, и, если они распределены по степенному закону, опишите, каким может быть механизм предпочтительного присоединения.

(a) Такие социальные сети, как Facebook или Instagram.

(b) Сайты во всемирной паутине (WWW).

(c) Дорожные сети, соединяющие города.

(d) Абонементные/торговые сети, такие как Amazon или Netflix.

10.14. *[5]* Для каждого из следующих теоретико-графических свойств приведите пример реальной сети, которая удовлетворяет этому свойству, и сети, которая не удовлетворяет ему.
(a) Направленный или ненаправленный.
(b) Взвешенный или невзвешенный.
(c) Простой или непростой.
(d) Разреженный или плотный.
(e) Вложенный или топологический.
(f) Маркированный или немаркированный.

10.15. *[3]* Докажите, что в любом простом графе всегда есть четное количество вершин с нечетной степенью вершины.

10.16. *[3]* Реализуйте простую версию алгоритма PageRank и проверьте его на своей любимой сети. Какие вершины выделены как ближайшие к центру?

Кластеризация

10.17. *[5]* Для набора данных с точками в позициях (4, 10); (7, 10); (4, 8); (6, 8); (3, 4); (2, 2); (5, 2); (9, 3); (12, 3); (l 1, 4); (10, 5) и (12, 6) продемонстрируйте кластеризацию, которая возникает в результате:
(a) кластеризации с одной связью;
(b) кластеризации средней связи;
(c) кластеризации дальнего соседа (полной связи).

10.18. *[3]* Для каждого из следующих конкурсов прогнозирования *The Quant Shop* предложите доступные данные, которые позволят использовать для их решения методы ближайшего соседа и аналогий.
- *Miss Universe.*
- *Movie gross.*
- *Baby weight.*
- *Art auction price.*
- *White Christmas.*
- *Football champions.*
- *Ghoul pool.*
- *Gold/oil prices.*

10.19. *[3]* Выполните кластеризацию методом k-средних вручную по следующим точкам для $k = 2$:

$$S = \{(1, 4); (1, 3); (0, 4); (5, 1); (6, 2); (4, 0)\}$$

Отобразите точки и окончательные кластеры.

10.20. *[5]* Реализуйте две версии простого алгоритма k-средних: один из них использует числовые центроиды в качестве центров, а другой ограничивает центры входными точками из набора данных. Затем поэксперимен-

тируйте. Какой алгоритм сходится быстрее в среднем? Какой алгоритм осуществляет кластеризацию с меньшей абсолютной и среднеквадратичной ошибкой и насколько?

10.21. *[5]* Предположим, что s_1 и s_2 являются случайно выбранными подмножествами из универсального набора с n элементами. Каково ожидаемое значение Жаккарова подобия $J(s_1, s_2)$?

10.22. *[5]* Определите набор данных об элементах, где у вас есть представление о природных кластерах, которые должны быть получены, будь то люди, университеты, компании или фильмы. Осуществите кластеризацию с помощью одного или нескольких алгоритмов, возможно, k-средних и агломерационной кластеризации. Затем оцените полученные кластеры на основе ваших знаний в области. Хорошо ли сделана работа? Что пошло не так? Можете ли вы объяснить, почему алгоритм не воссоздал то, что было в вашей голове?

10.23. *[5]* Предположим, что мы пытаемся объединить $n = 10$ точек в одном измерении, где точка p_i имеет позицию $x = i$. Объясните, что такое дерево агломерационной кластеризации для этих точек при:

(a) кластеризации одиночной связи;

(b) кластеризации средней связи;

(c) кластеризации полной связи/дальнего соседа.

10.24. *[5]* Предположим, что мы пытаемся объединить $n = 10$ точек в одном измерении, где точка p_i имеет позицию $x = 2^i$. Объясните, что такое дерево агломерационной кластеризации для этих точек при:

(a) кластеризации одиночной связи;

(b) кластеризации средней связи;

(c) кластеризации полной связи/дальнего соседа.

Реализация проектов

10.25. *[5]* Проведите эксперименты по изучению влияния критериев слияния (одиночная связь, центроид, средняя связь, самая дальняя связь) на свойства результирующего дерева кластеров. Что приводит к самым высоким деревьям? Самым сбалансированным? Какова разница во времени работы? Какой метод дает результаты, наиболее совместимые с кластеризацией методом k-средних?

10.26. *[5]* Поэкспериментируйте с производительностью различных алгоритмов/структур данных для поиска ближайшего соседа точки q среди n точек в d измерениях. Каково максимальное значение d, для которого каждый метод остается жизнеспособным? Насколько быстрее эвристические методы, основанные на LSH, чем методы, которые гарантируют точность ближайшего соседа, с какой потерей точности?

Вопросы на интервью

10.27. *[5]* Что такое проклятие размерности? Как это влияет на меры расстояния и сходства?

10.28. *[5]* Что такое кластеризация? Опишите пример алгоритма, который выполняет кластеризацию. Как мы можем узнать, произвел ли он удовлетворительные кластеры в нашем наборе данных?

10.29. *[5]* Как мы можем оценить правильное количество кластеров для использования с данным набором данных?

10.30. *[5]* В чем разница между обучением с учителем и без учителя?

10.31. *[5]* Как вы можете работать с коррелированными функциями в вашем наборе данных, уменьшая размерность данных.

10.32. *[5]* Объясните, что такое локальный оптимум. Почему это важно в кластеризации методом k-средних?

Конкурсы Kaggle

10.33. Какие люди наиболее влиятельны в данной социальной сети?
`https://www.kaggle.com/c/predict-who-is-more-influential-in-a-social-network`

10.34. Кому суждено стать друзьями в социальной сети?
`https://www.kaggle.com/c/socialNetwork`

10.35. Предскажите, какой товар скорее всего купит покупатель.
`https://www.kaggle.com/c/coupon-purchase-prediction`

Глава 11

Машинное обучение

> Любая достаточно передовая форма мошенничества неотличима от обучения.
>
> — Ян Шауманн (Jan Schaumann)

Большую часть своей карьеры я очень подозрительно относился к важности машинного обучения. За эти годы я провел много переговоров с грандиозными претензиями и очень скудными результатами. Однако ситуация явно изменилась. Самая интересная работа в области информатики сегодня связана с машинным обучением — как с новыми мощными алгоритмами, так и с захватывающими новыми приложениями.

Эта революция произошла по нескольким причинам. В первую очередь, объем доступных данных и вычислительной мощности перешел магический порог, когда системы машинного обучения начали делать интересные вещи, даже используя старые подходы. Это вдохновило на активную деятельность по разработке методов, которые лучше масштабируются, и на увеличение инвестиций в ресурсы данных и развитие системы. Культура программного обеспечения с открытым исходным кодом заслуживает того, чтобы снять перед ней шляпу, поскольку новые идеи удивительно быстро превращаются в доступные инструменты. Машинное обучение сегодня — это взрывоопасная область, вокруг которой сейчас много шума.

До сих пор мы обсуждали два способа построения моделей на основе данных: линейная регрессия и подход ближайших соседей, причем оба достаточно подробно. Для многих приложений это и все, что вам нужно знать. Если у вас достаточно маркированных учебных данных, все методы могут дать хорошие результаты. И если у вас их нет, все методы могут потерпеть неудачу. Наличие лучшего алгоритма машинного обучения может иметь значение, но, как правило, только в деталях. Я чувствую, что цель моей книги — это научить вас ходить, а более специализированные книги научат вас бегать.

Тем не менее было разработано множество интересных и важных алгоритмов машинного обучения. В данной главе мы рассмотрим эти методы, чтобы вы понимали преимущества и недостатки каждого из них, а также затронем нескольких важных аспектов производительности.

- *Выразительность*. Методы машинного обучения отличаются богатством и сложностью моделей, которые они поддерживают. Линейная регрессия соответствует линейным функциям, в то время как методы ближайших соседей определяют кусочно-линейные границы разделения с достаточным количеством частей для аппроксимации произвольных кривых. Большая выразительность обеспечивает возможность создания более точных моделей, а также создает опасность переобучения.
- *Интерпретируемость*. Такие мощные методы, как глубокое обучение, зачастую создают совершенно непроницаемые модели. Они могли бы обеспечивать очень точную классификацию на практике, но не могли бы объяснить человеку, почему они принимают те или иные решения. В модели линейной регрессии, напротив, самые большие коэффициенты идентифицируют самые сильные особенности, а идентичности ближайших соседей позволяют нам независимо определять нашу уверенность в этих аналогиях.

  Я лично считаю, что интерпретируемость является важным свойством модели, и, как правило, приятнее взять менее эффективную модель, которую я понимаю, чем чуть более точную, но совершенно непонятную. Это мнение может быть и не общепринятым, но вам действительно имеет смысл понимать свою модель и конкретную область применения.
- *Простота применения*. Некоторые методы машинного обучения имеют относительно немного параметров или решений, т.е. они работают прямо из коробки. И линейная регрессия, и классификация ближайших соседей в этом отношении довольно просты. Напротив, такие технологии, как *метод опорных векторов* (Support Vector Machine — SVM), предоставляют гораздо больше возможностей для оптимизации производительности алгоритма с правильными настройками. Я чувствую, что доступные инструменты для машинного обучения будут и дальше совершенствоваться: они проще в использовании и мощнее. Но пока некоторые методы позволяют пользователю взять их на поводок, если они не знают, что делать.
- *Скорость обучения*. Методы сильно различаются по тому, насколько быстро они находят соответствие необходимым параметрам модели, которая определяет, сколько учебных данных вы можете себе позволить использовать на практике. Традиционные методы линейной регрессии могут быть дорогими для больших моделей. В отличие от этого, поиск ближайшего соседа почти не требует времени на обучение, за исключением времени, необходимого для построения соответствующей структуры данных поиска.

- *Скорость прогнозирования*. Методы отличаются тем, насколько быстро они принимают решения о классификации нового запроса q. Линейная/логистическая регрессия быстра, ей достаточно вычислить взвешенную сумму полей во входных записях. Поиск ближайшего соседа, напротив, требует явного сравнения q с существенным количеством учебных тестов. В общем, со скоростью обучения есть дилемма: вы можете заплатить сейчас или заплатить позже.

На рис. 11.1 представлены мои примерные субъективные оценки того, как обсуждаемые в этой главе подходы соответствуют параметрам производительности. Эти рейтинги не являются абсолютной истинной, и у разумных людей могут быть разные мнения.

Метод	Выразительность	Простота интерпретации	Простота использования	Скорость обучения	Скорость прогнозирования
Линейная регрессия	5	9	9	9	9
Ближайший сосед	5	9	8	10	2
Наивный байесовский	4	8	7	9	8
Деревья решений	8	8	7	7	9
Метод опорных векторов	8	6	6	7	7
Бустинг	9	6	6	6	6
Графические модели	9	8	3	4	4
Глубокое обучение	10	3	4	3	7

Рис. 11.1. Субъективное ранжирование подходов к машинному обучению по пяти измерениям и шкале от 1 до 10, причем, чем выше, тем лучше

Надеюсь, что эти ранги будут полезны для обзора алгоритмов машинного обучения. Конечно, ни один метод машинного обучения не доминирует над всеми остальными. Это наблюдение формализовано в соответствующей теореме *no free lunch* (бесплатных завтраков не бывает), которая доказывает, что не существует ни одного алгоритма машинного обучения, который был бы лучше всех остальных по всем задачам.

Тем не менее методы все же можно ранжировать в соответствии с приоритетом использования на практике. Мое упорядочение методов в этой книге (см. рис. 11.1) начинается с методов, которые просты в использовании и настройке, но имеют меньшую дискриминирующую способность, чем самые передовые методы. По правде говоря, я призываю вас начать с простых методов и идти дальше по списку, если потенциальные улучшения в точности действительно оправдывают это.

Материал, который я представлю в этой главе, легко использовать неправильно, поскольку существует вполне естественное искушение опробовать все возможные алгоритмы машинного обучения и выбрать ту модель, которая дает наивысшую точность или $F1$-оценку. Довольно наивно полагать, что

с помощью одного обращения к библиотеке можно облегчить эту задачу, и вы, вероятно, обнаружите, что все модели примерно одинаково работают с вашими учебными данными. Кроме того, любые различия в производительности между ними, которые вы обнаружите, скорее связаны с дисперсией, чем с пониманием. Подобные эксперименты — это то, для чего была изобретена проверка достоверности.

Наиболее важным фактором, который будет определять качество ваших моделей, является качество ваших функций. В главе 3 мы много говорили об очистке данных, которая заключается в правильной подготовке вашей матрицы данных. Мы углубимся в разработку функций в разделе 11.5.4, прежде чем обсуждать методы глубокого обучения, которые стремятся создать свои собственные функции.

И последний комментарий. Специалисты по обработке данных, как правило, имеют предпочтительный подход машинного обучения, так же, как и любимый язык программирования или спортивную команду. По большей части это означает, что, поскольку они лучше всего знакомы с некой конкретной реализацией, она лучше всего работает в их руках. Отчасти это мышление связано с тем фактом, что они на нескольких примерах заметили, что одна библиотека немного опережает другие, и сделали ложное обобщение.

Не попадитесь в эту ловушку. Выберите те методы, которые наилучшим образом соответствуют потребностям вашего приложения, основываясь на вышеуказанных критериях, и получите достаточный опыт работы с их разнообразными регуляторами и рычагами для оптимизации производительности.

11.1. Наивный байесовский классификатор

Напомним, что два события A и B *независимы*, если $p(A \text{ И } B) = p(A)p(B)$. Если A — это событие, когда "моя любимая спортивная команда выигрывает сегодня", а B — "фондовый рынок сегодня растет", то, вероятно, A и B независимы. Но это не совсем так. Рассмотрим случай, если A — это событие, когда "в этом семестре я получаю A по науке о данных", а B — " в этом семестре я получаю A по другому курсу". Между этими событиями существуют зависимости: наличие энтузиазма к учебе или выпивке непосредственно влияет на успеваемость. В общем случае

$$p(A \text{ И } B) = p(A)p(B|A) = p(A) + P(B) - p(A \text{ ИЛИ } B).$$

Если бы все было независимо, мир вероятности был бы намного проще. Алгоритм наивного байесовского классификатора скрещивает пальцы и предполагает независимость, чтобы избежать необходимости вычислять все эти беспорядочные условные вероятности.

11.1.1. Формулировка

Предположим, мы хотим классифицировать вектор $X = (x_1, \ldots, x_n)$ на один из m классов $C_1, \ldots, C_m$. Мы стараемся вычислить вероятность каждого возможного класса для данного X, чтобы мы могли присвоить X метку класса с наибольшей вероятностью. По теореме Байеса,

$$p(C_i|X) = \frac{p(C_i)p(\mathrm{X}|C_i)}{p(X)}.$$

Давайте разберем это уравнение. Член $p(C_i)$ — это *априорная* вероятность, вероятность маркировки класса без каких-либо конкретных доказательств. Я знаю, что читатели чаще имеют черные волосы, чем рыжие, поскольку в мире больше темноволосых людей, чем рыжих.[1]

Знаменатель $P(X)$ дает вероятность увидеть заданный входной вектор X по всем возможным входным векторам. Установление точного значения $P(X)$ кажется несколько рискованным, но, к счастью, обычно в этом нет необходимости. Обратите внимание, что этот знаменатель одинаков для всех классов. Мы только пытаемся установить метку класса для X, поэтому значение $p(X)$ не влияет на наше решение. Выбор класса с наибольшей вероятностью означает

$$C(X) = \arg\max_{i=1,\ldots,m} \frac{p(C_i)p(\mathrm{X}|C_i)}{p(X)} = \arg\max_{i=1,\ldots,m} p(C_i)p(\mathrm{X}|C_i).$$

Оставшийся член $p(X|C_i)$ — это вероятность увидеть входной вектор X, если мы знаем, что класс элемента — это C_i. Это также кажется несколько рискованным. Какова вероятность, что кто-то весит 150 фунтов (68 кг) и имеет рост 5 футов 8 дюймов (172,72 см), учитывая, что он мужчина? Должно быть ясно, что $p(X|C_i)$, как правило, будет очень маленьким: существует огромное пространство возможных входных векторов, соответствующих классу, только один из которых соответствует данному элементу.

Но теперь предположим, что мы жили там, где все было независимо, т.е. вероятность события A и события B всегда была $p(A)p(B)$. Тогда

$$p(X|C_i) = \prod_{j=1}^{n} p(x_j|C_i).$$

Ныне любой, кто действительно верит в мир независимых вероятностей, довольно наивен, отсюда и название *наивный Байесовский классификатор* (naive Bayes). Но такое предположение действительно облегчает вычисления. Собираем все вместе:

---

[1] Википедия утверждает, что только 1-2% мирового населения являются рыжеволосыми.

$$C(X) = \arg\max_{i=1,\ldots,m} p(C_i)p(X \mid C_i) = \arg\max_{i=1,\ldots,m} p(C_i)\prod_{j=1}^{n} p(x_j \mid C_i).$$

Наконец, мы должны взять логарифм произведения, чтобы превратить его в сумму для лучшей числовой стабильности. Логарифмы вероятностей будут отрицательными числами, но менее вероятные события будут более отрицательными, чем обычные. Таким образом, полный наивный Байесовский алгоритм задается следующей формулой:

$$C(X) = \arg\max_{i=1,\ldots,m} (\log p(C_i) + \sum_{j=1}^{n} \log p(x_j \mid C_i)).$$

Как рассчитать $p(x_j \mid C_i)$, вероятность наблюдения x_j по данной метке класса i? Это легко сделать из учебных данных, особенно если x_j является категориальной переменной, например "имеет рыжие волосы". Мы можем просто выбрать все экземпляры класса i в учебном наборе и вычислить ту долю из них, которые имеют свойство x_j. Эта дробь определяет разумную оценку $p(x_j \mid C_i)$. Но когда x_j является числовой переменной, например "age = 18" или " в данном документе слово *dog* встречалось шесть раз", понадобится немного больше воображения, но в принципе вычисляется по тому, как часто это значение наблюдается в учебном наборе.

День	Прогноз	Т-ра	Влажность	Пляж?
1	Солнечно	Высокая	Высокая	Да
2	Солнечно	Высокая	Нормальная	Да
3	Солнечно	Низкая	Нормальная	Нет
4	Солнечно	Умеренная	Высокая	Да
5	Дождь	Умеренная	Нормальная	Нет
6	Дождь	Высокая	Высокая	Нет
7	Дождь	Высокая	Нормальная	Нет
8	Облачно	Низкая	Высокая	Нет
9	Облачно	Высокая	Нормальная	Да
10	Облачно	Высокая	Нормальная	Нет

P(*X* | Класс)	Вероятность в классе	
Прогноз	Пляж	Не пляж
Солнечно	3/4	1/6
Дождь	0/4	3/6
Облачно	1/4	2/6
Температура	Пляж	Не пляж
Высокая	3/4	2/6
Умеренная	1/4	2/6
Низкая	0/4	2/6
Влажность	Пляж	Не пляж
Высокая	2/4	2/6
Нормальная	2/4	4/6
P(Пляж День)	4/10	6/10

Рис. 11.2. Вероятности для поддержки расчета наивным Байесовским классификатором того, хорош ли сегодняшний день для посещения пляжа: таблица событий (слева) с распределениями предельной вероятности (справа)

На рис. 11.2 иллюстрируется простая Байесовская процедура. Слева представлена таблица из десяти наблюдений за погодными условиями, и было ли каждое наблюдение сделано в тот день, когда стоило пойти на пляж, а не остаться дома. Справа эта таблица была разделена так, чтобы получить условные вероятности погодных условий с учетом активности. Исходя из этих

вероятностей, мы можем использовать теорему Байеса для следующего вычисления.

P(Пляж|(Солнечно, Умеренная, Высокая)) =
$= (P$(Солнечно|Пляж)$\cdot P$(Умеренная|Пляж)$\cdot P$(Высокая|Пляж)$\cdot P$(Пляж) =
$= (3/4)\cdot(1/4)\cdot(2/4)\cdot(4/10) = 0{,}0375.$

P(Не пляж|(Солнечно, Умеренная, Высокая)) =
$= (P$(Солнечно|Нет)$\cdot P$(Умеренная|Нет)$\cdot P$(Высокая|Нет))$\cdot P$(Нет) =
$= (1/6)\cdot(2/6)\cdot(2/6)\cdot(6/10) = 0{,}0111.$

Начиная с 0,0375 > 0,0111, наивный Байесовский классификатор говорит нам, что можно идти на пляж. Обратите внимание: не имеет значения, что именно эта комбинация (Солнечно, Умеренная, Высокая) появилась в учебных данных. Мы основываем наше решение на совокупных вероятностях, а не на одной строке, как в классификации ближайших соседей.

11.1.2. Как справиться с нулевым счетом (дисконтирование)

Есть тонкая, но важная проблема подготовки функций, в частности связанная с наивным Байесовским алгоритмом. Наблюдаемые показатели не очень точно отражают частоту редких событий, для которых обычно характерен длинный хвост.

Впервые этот вопрос был поднят математиком Лапласом, который спросил: какова вероятность того, что солнце взойдет завтра? Это значение может быть близко к единице, но это не совсем 1,0. Хотя солнце восходило, как часы, каждое утро около 36,5 миллионов дней или около того с тех пор, как человек начал замечать такие вещи, оно не будет существовать вечно. Придет время, когда взорвется земля или солнце, а потому существует небольшая, но ненулевая вероятность того, что это случится сегодня ночью.

Всегда могут быть события, которых вы еще не видели ни в одном конечном наборе данных. У вас вполне могут быть записи на сотню людей, ни у кого из которых нет рыжих волос. Заключение о том, что вероятность появления рыжих волос 0/100 = 0, потенциально катастрофично, когда нас попросят классифицировать кого-то с *рыжими волосами*, поскольку вероятность того, что он окажется в нужном классе, будет равна нулю. Было бы еще хуже, если бы во всем учебном наборе был бы ровно один рыжий, скажем, помеченный классом C_2. Наш наивный Байесовский классификатор решит, что каждый последующий рыжий просто должен быть в классе C_2, независимо от других доказательств.

Дисконтирование (discounting) — это статистический метод, позволяющий корректировать счет для еще невиданных событий, явно оставляя для них доступную массу вероятности. Самым простым и популярным методом является *дисконтирование "добавь единицу"* (add-one discounting), когда мы добавляем единицу к частоте всех результатов, включая невиданные. Предположим, например, что мы тянем шары из лотерейного барабана. После появления пяти красных и трех зеленых, какова вероятность того, что мы увидим новый цвет при следующем розыгрыше? Если мы используем дисконтирование "добавь единицу",

$$P(\text{красный}) = (5 + 1)/((5 + 1) + (3 + 1) + (0 + 1)) = 6/11,$$

$$P(\text{зеленый}) = (3 + 1)/((5 + 1) + (3 + 1) + (0 + 1)) = 4/11,$$

оставляя для нового цвета массу вероятности

$$P(\text{новый\_цвет}) = 1/((5 + 1) + (3 + 1) + (0 + 1)) = 1/11.$$

Для небольшого количества выборок или большого количества известных классов дисконтирование вызывает нетривиальное демпфирование вероятностей. Наша оценка вероятности увидеть красный шар изменяется с 5/8 = 0,625 до 6/11 = 0,545, когда мы применяем дисконтирование "добавь единицу". Но это более безопасная и более правдивая оценка, и после того, как мы увидим достаточно примеров, различия исчезнут начисто.

Вы должны знать, что были разработаны и другие методы дисконтирования, но их добавление может быть и не лучшим из возможных методов оценки во всех ситуациях. Тем не менее *отсутствие* учета дисконтирования вызывает проблемы, и никто не будет уволен за использование метода "добавь единицу".

Дисконтирование становится особенно важным при обработке естественного языка, где традиционное представление *наборов слов* (bag of words) моделирует документ как вектор частот слов во всем словаре языка, скажем, из 100 000 слов. Поскольку частота использования слов подчиняется степенному закону (закон Ципфа), слова в хвосте встречаются довольно редко. Вы когда-нибудь ранее видели английское слово *defenestrate*?[2] Еще хуже то, что документы меньше, чем книга, слишком короткие, чтобы содержать 100 000 слов, поэтому мы обречены видеть нули, куда бы мы ни смотрели. Дисконтирование "добавь единицу" превращает эти векторы счетов в векторы разумной вероятности с ненулевой вероятностью увидеть редкие и до сих пор не встречавшиеся слова.

[2] Это значит "выгнать кого-то с работы".

11.2. Классификаторы дерева решений

Дерево решений (decision tree) — это двоичная ветвящаяся структура, используемая для классификации произвольного входного вектора X. Каждый узел в дереве содержит простое сравнение признаков с некоторым полем $x_i \in X$, например "справедливо ли, что $x_i \geq 23{,}7$?" Результатом каждого такого сравнения является либо истина, либо ложь, определяя, следует ли нам двигаться дальше к левому или правому потомку данного узла. Эти структуры иногда называют *деревьями классификации и регрессии* (Classification And Regression Trees — CART), потому что они могут быть применены к более широкому классу проблем.

Дерево решений разделяет обучающие примеры на группы относительно однородного состава классов, поэтому решение становится легким. На рис. 11.3 представлен пример дерева решений, предназначенного для прогнозирования ваших шансов на выживание при кораблекрушении "Титаника". Для классификации каждая строка/экземпляр проходит уникальный путь от корня к листу. Корневое сравнение отражает здесь морскую традицию спасать в первую очередь женщин и детей: 73% женщин выжили, поэтому одной этой функции достаточно, чтобы сделать прогноз для женщин. Второй уровень дерева в первую очередь отражает детей: любой мужчина 10 лет и старше считается неудачником. Даже младшие должны преодолеть одно последнее препятствие: они обычно добираются до спасательной шлюпки, только если у них есть братья и сестры, чтобы поддержать их.

Какова точность этой модели на учебных данных? Это зависит от того, какая фракция примеров заканчивается на каждом листе и насколько чисты эти примеры листьев. Для примера на рис. 11.3, дополненного процентом охвата и долей выживания (чистотой) в каждом узле, точность классификации A этого дерева составляет:

$$A = 0{,}35 \cdot 73\% + 0{,}61 \cdot 83\% + 0{,}02 \cdot 95\% + 0{,}02 \cdot 89\% = 78{,}86\%.$$

Точность 78,86% неплоха для такой простой процедуры принятия решения. Мы могли бы довести его до 100%, укомплектовав дерево так, чтобы у каждого из 1317 пассажиров был свой лист, обозначающий этот узел своей судьбой. Возможно, 23-летние мужчины второго класса выжили с большей вероятностью, чем мужчины 22- или 24-летнего возраста. Это наблюдение можно использовать для повышения точности обучения. Но такое сложное дерево было бы чрезмерным, создавая структуру, которая не имеет смысла. Дерево на рис. 11.3 является вполне интерпретируемым, устойчивым и достаточно точным. Кроме того, каждый сам за себя.

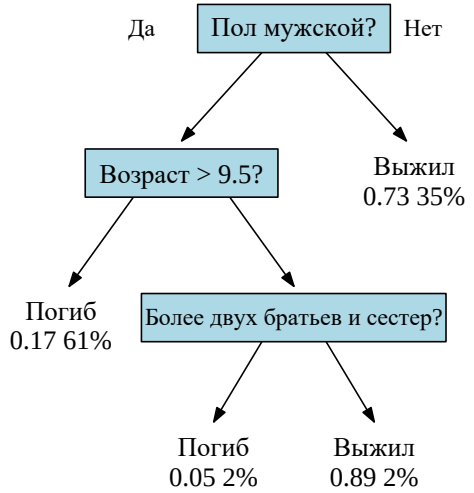


Рис. 11.3. Простое дерево решений для прогнозирования смертности на "Титанике"

Преимущества деревьев решений таковы.

- *Нелинейность.* Каждый лист представляет фрагмент пространства для принятия решений, но достигаемый потенциально сложным путем. Эта логическая цепочка позволяет деревьям решений представлять очень сложные границы решений.
- *Поддержка категориальных переменных.* Деревья решений естественным образом используют категориальные переменные, такие как "если цвет волос = рыжий", в дополнение к числовым данным. Категориальные переменные менее комфортно вписываются в большинство других методов машинного обучения.
- *Интерпретируемость.* Деревья решений объяснимы; вы можете прочитать их и понять, в чем их смысл. Таким образом, алгоритмы дерева решений могут рассказать вам что-то о вашем наборе данных, чего вы раньше не видели. Кроме того, интерпретируемость позволяет вам проверить, доверяете ли вы решениям, которые он примет: принимает ли он решения по правильным причинам?
- *Надежность.* Количество возможных деревьев решений растет в геометрической прогрессии по количеству функций и возможных тестов, т.е. мы можем построить их столько, сколько захотим. Построение множества деревьев случайных решений (алгоритм CART (Classification and Regression Tree)) и использование результата каждого в качестве голоса для данной метки повышает надежность и позволяет нам оценить достоверность нашей классификации.

- *Применимость регрессии.* Подмножество элементов, которые следуют аналогичным путем по дереву решений, вероятно, схожи по свойствам, отличным от просто метки Для каждого такого подмножества мы можем использовать линейную регрессию для построения специальной модели прогнозирования числовых значений таких конечных элементов. Это, вероятно, будет работать лучше, чем более общая модель, обученная на всех элементах.

Самым большим недостатком деревьев решений определенно является отсутствие элегантности. Методы обучения, такие как логистическая регрессия и метод опорных векторов, используют *математику.* Передовая теория вероятностей, линейная алгебра, многомерная геометрия — это, как вы знаете, *математика.*

Деревья решений, напротив, являются хакерской игрой. В процессе обучения есть много разных ручек, которые нужно подкрутить, и сравнительно мало теории, которая поможет вам подкрутить их правильно.

Но в том-то и дело, что модели дерева решений на практике работают очень хорошо. *Градиентные деревья решений* (Gradient Boosted Decision Tree — GBDT) в настоящее время являются наиболее часто используемым методом машинного обучения для победы на конкурсах *Kaggle.* Мы будем работать над этим поэтапно. Сначала деревья решений, а затем бустинг в следующем разделе.

11.2.1. Построение деревьев решений

Деревья решений строятся сверху вниз. Мы начинаем с заданного набора учебных элементов, каждый из которых имеет n признаков и помечен одним из m классов C_1, ..., C_m. Каждый узел в дереве решений содержит двоичный предикат, логическое условие, полученное из данной функции.

Признак с дискретным набором значений v_i можно легко превратить в двоичные предикаты с помощью проверки на равенство: "является ли признак $x_i = v_{i,j}$?" Таким образом, существует $|v_i|$ различных предикатов, связанных с x_i. Числовые признаки можно превратить в двоичные предикаты с добавлением порога t: "является ли признак $x_i \geq t$?"

Множество потенциально интересных порогов t определяется промежутками между наблюдаемыми значениями, которые x_i принимает в обучающем наборе. Если полный набор наблюдений x_i равен (10, 11, 11, 14, 20), значимые возможные значения составляют для $t \in$ (10, 11, 14) или, возможно, $t \in$ (10,5; 11,5; 17). Оба набора порогов дают одни и те же разделы наблюдений, но использование средних точек каждого раздела кажется более разумным при обобщении на будущие значения, невиданные при обучении.

Нам нужен способ оценить каждый предикат на предмет того, насколько хорошо он будет способствовать разделению набора S обучающих примеров, достижимых с этого узла. Идеальный предикат p — это *чистое разбиение* (pure partition) S, когда метки классов не пересекаются. В этом сне все члены S из каждого класса C_i будут появляться исключительно на одной стороне дерева, однако такая чистота обычно невозможна. Мы также хотим, чтобы предикаты создавали *сбалансированные разделения* (balanced split) S, когда левое поддерево содержит примерно столько же элементов из набора S, сколько и правое поддерево. Сбалансированные разделения ускоряют процесс классификации, а также потенциально более устойчивы. Установка порогового значения t на минимальное значение x_i отнимает от S одиночный элемент и создает совершенно чистое, но максимально разбалансированное разделение.

Таким образом, наши критерии отбора должны поощрять как баланс, так и чистоту, чтобы максимизировать то, что мы узнаем из теста. Один из способов измерения чистоты подмножества S — обратное выражение беспорядка, или *энтропии*. Обозначим через f_i долю из S, являющуюся классом C_i. Тогда теоретико-информационная энтропия для S, $H(S)$, может быть вычислена как:

$$H(S) = -\sum_{i=1}^{m} f_i \log_2 f_i.$$

Отрицательный знак здесь для того, чтобы сделать все числа положительными, поскольку логарифм *правильной дроби* (proper fraction) всегда отрицателен.

Давайте разберем эту формулу. Самый чистый возможный вклад происходит, когда все элементы принадлежат одному классу, что означает, что $f_j = 1$ для некоторого класса j. Вклад класса j в $H(S)$ равен $1 \log_2 1 = 0$, что идентично вкладу всех других классов: $0 \cdot \log_2 0 = 0$. Наиболее беспорядочная версия — это когда все m классов представлены одинаково, что означает, что $f_i = 1/m$. Тогда по вышеприведенному определению $H(S) = \log_2 m$. Чем меньше энтропия, тем лучше узел для классификации.

Значение потенциального разбиения, примененного к узлу дерева, — это то, насколько оно уменьшает энтропию системы. Предположим, что логический предикат p разбивает S на два непересекающихся подмножества, поэтому $S = S_1 \cup S_2$. Тогда *информационный прирост* (information gain) p определяется как

$$IG_p(S) = H(S) - \sum_{i=1}^{2} \frac{|S_i|}{|S|} H(S_i).$$

Мы ищем предикат p', который максимизирует этот выигрыш информации как лучший разделитель для S. Этот критерий неявно предпочитает сбалансированное разделение, так как обе стороны дерева вычисляются.

Были определены и используются на практике также и альтернативные показатели чистоты. *Коэффициент Джини* (Gini impurity) основан на другой величине $f_i(1 - f_i)$, которая равна нулю в обоих случаях чистого разделения, $f_i = 0$ или $f_i = 1$:

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i=1}^{m}(f_i - f_i^2) = \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} f_i^2 = 1 - \sum_{i=1}^{m} f_i^2$$

Критерии выбора предикатов для оптимизации коэффициента Джини могут быть определены аналогичным образом.

Нам нужно условие остановки для завершения эвристики. Когда узел достаточно чист, чтобы назвать его листом? Устанавливая порог ε для получения информации, мы прекращаем деление, когда вознаграждение за следующий тест меньше ε.

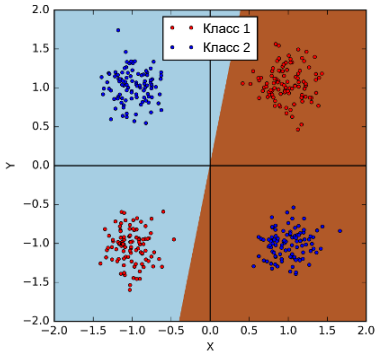
Альтернативная стратегия заключается в том, чтобы строить полное дерево до тех пор, пока все листья не станут полностью чистыми, а затем сократить его, удалив узлы, которые вносят наименьший информационный прирост. Как ни странно, у большой вселенной, возможно, нет хороших разделителей около корня, но лучшие из них появляются, когда набор живых объектов становится меньше. Преимущество этого подхода заключается в том, что сдается он не слишком рано.

11.2.2. Реализация исключающего ИЛИ

Некоторые формы границ решения может быть очень трудно или даже невозможно подогнать к конкретному подходу машинного обучения. Печальней всего то, что линейные классификаторы не могут использоваться для подгонки к некоторым простым нелинейным функциям, таким как исключающее ИЛИ (eXclusive OR — XOR). Логическая функция $A \oplus B$ определяется как

$$A \oplus B \ = \ (A \text{ ИЛИ } \overline{B}) \text{ ИЛИ } (\overline{A} \text{ ИЛИ } B).$$

Для точек (x, y) в двух измерениях мы можем определить предикаты так, что A означает "справедливо ли, что $x \geq 0$?", а B означает "справедливо ли, что $y \geq 0$?" Теперь есть две различные области, где $A \oplus B$ истинно, в противоположных квадрантах плоскости xy, показанной на рис. 11.4 (слева). Необходимость разделить две области одной линией объясняет, почему XOR невозможно для линейных классификаторов.



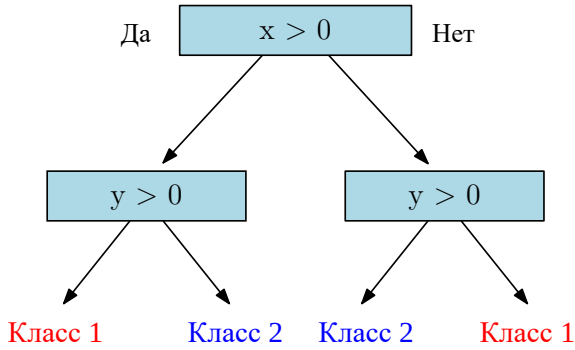


Рис. 11.4. Функция исключающего ИЛИ не подходит линейным классификаторам. Слева мы представляем четыре естественных кластера в пространстве x – y. Это демонстрирует полную неспособность логистической регрессии найти значимый разделитель, даже если небольшое дерево решений легко выполняет свою работу (справа)

Деревья решений достаточно мощные, чтобы распознавать XOR. Действительно, двухуровневое дерево на рис. 11.4 (справа) вполне справляется со своей работой. После корневых тестов, является A истиной или ложью, тесты второго уровня для B уже подготовлены к A, поэтому каждый из четырех листов может быть связан с отдельным квадрантом, что позволяет правильно их классифицировать.

Хотя деревья решений могут распознавать XOR, это не означает, что легко найти такое дерево, которое это делает. Что делает функцию XOR трудной, так это то, что вы не можете видеть, как вы продвигаетесь к лучшей классификации, даже если вы выбрали правильный корневой узел. В приведенном выше примере выбор корневого узла "справедливо ли, что $x > 0$?" Вызывает видимого улучшения чистоты класса с обеих сторон. Ценность этого теста становится очевидной, только если мы посмотрим вперед на другой уровень, поскольку прирост информации равен нулю.

С такими проблемами, как XOR, эвристика построения дерева решений терпит неудачу. Это говорит о ценности более сложных и вычислительно дорогих процедур построения дерева в сложных случаях, которые выглядят как компьютерные шахматные программы, оценивающие ценность хода p не сейчас, а несколькими шагами позже.

11.2.3. Ансамбли деревьев решений

Существует огромное количество возможных деревьев решений, которые могут быть построены на любом обучающем множестве S. Кроме того, каждый

из них отлично классифицирует *все* обучающие примеры, если мы продолжим уточнение до тех пор, пока все листья не станут чистыми. Это предполагает создание сотен или даже тысяч различных деревьев, а также оценку элемента q для каждого из них, чтобы получить возможную метку. Позволяя каждому дереву отдавать свой собственный независимый голос, мы получаем уверенность в том, что наиболее часто встречаемая метка будет правильной.

Чтобы избежать группового мышления, нам нужно, чтобы деревья были разнообразными. Повторное использование детерминированной процедуры построения, которая находит лучшее дерево, бесполезно, поскольку все они будут идентичны. Лучше было бы случайным образом выбрать новое измерение разбиения в каждом узле дерева, а затем найти наилучший из возможных порогов для этой переменной, чтобы определить предикат.

Но даже при случайном выборе размеров результирующие деревья нередко сильно коррелируют. Лучшим подходом является *бэггинг* (bagging), построение наилучших деревьев на относительно небольших случайных подмножествах предметов. Если все сделано правильно, результирующие деревья должны быть относительно независимыми друг от друга, обеспечивая разнообразие классификаторов для работы, способствуя мудрости толпы.

Использование ансамблей деревьев решений имеет еще одно преимущество, помимо надежности. Степень консенсуса между деревьями предлагает меру уверенности для любого решения классификации. Существует большая разница в маркере большинства, который появляется в 501 из 1000 деревьев против 947 из них.

Эта доля может быть интерпретирована как вероятность, но еще лучше может быть включение этого числа в логистическую регрессию для более мотивированной меры доверия. Предполагая, что у нас есть проблема двоичной классификации, пусть f_i обозначает долю деревьев, выбирающих класс C_1 на входном векторе X_i. Проведите весь обучающий набор через ансамбль дерева решений. Теперь определите задачу логистической регрессии, где f_i — это входная переменная, а класс X_i — выходная переменная. Результирующая логит-функция определит соответствующий уровень достоверности для любой наблюдаемой доли согласия.

11.3. Бустинг и ансамблевое обучение

Идея объединения большого количества шумных "предикторов" в один более мощный классификатор применима как к алгоритмам, так и к толпе. Зачастую многие различные признаки слабо коррелируют с зависимой переменной. Так как же лучше всего объединить их в один более мощный классификатор?

11.3.1. Голосование с классификаторами

Ансамблевое обучение (ensemble learning) — это стратегия объединения множества различных классификаторов в одну прогностическую единицу. Наивный байесовский подход из раздела 11.1 имеет некую схожесть, поскольку он использует каждый признак как отдельный относительно слабый классификатор, а затем перемножает их вместе. Линейная/логистическая регрессия имеет похожую интерпретацию в том смысле, что она присваивает вес каждому признаку, чтобы максимизировать предсказательную силу ансамбля.

Но в целом ансамблевое обучение вращается вокруг идеи *голосования* (voting). Мы видели, что деревья решений могут быть более мощными в совокупности, в результате построения сотен или тысяч из них на случайных подмножествах примеров. Мудрость толпы следует из триумфа разнообразия мыслей над индивидуальностью с величайшим опытом.

Демократия основывается на принципе "один человек, один голос". Ваше образованное, аргументированное суждение о лучшем образе действия считается равным голосу того идиота с громким голосом в зале. Демократия имеет смысл с точки зрения динамики общества: общие решения, как правило, влияют на идиота так же, как и вы, поэтому равенство требует, чтобы все люди заслуживали равного права голоса в этом вопросе.

Но тот же аргумент не относится к классификаторам. Наиболее естественный способ использования нескольких классификаторов — каждый голосует и получает метку большинства. Но почему каждый классификатор должен получить одинаковый голос?

На рис. 11.5 показана сложность назначения весов классификаторам. Пример состоит из пяти избирателей, каждый из которых классифицирует пять пунктов. Все избиратели довольно хороши, каждый набрал 60% правильных ответов, за исключением v_1, который набрал 80%. Вариант большинства оказывается не лучше, чем наихудший индивидуальный классификатор. Но идеальный классификатор получится, если мы отбрасываем избирателей v_4 и v_5, а остальных взвешиваем одинаково. Что делает избирателей v_2 и v_3 ценными, так это не их общая точность, а их производительность в самых сложных задачах (D и особенно E).

Кажется, что есть три основных способа присвоения весов классификатору/избирателям. Самым простым может быть придание большего веса голосам тех классификаторов, которые доказали свою точность в прошлом, возможно, присвоив v_i мультипликативный вес t_i/T, где t_i — это количество раз, которое v_i классифицировал правильно, и $T = \sum_{i=1}^{c} t_i$. Обратите внимание, что эта весовая схема будет работать не лучше, чем правило большинства в примере на рис 11.5.

Пункт/ избиратель	V_1	V_2	V_3	V_4	V_5	Большинство	Лучшие веса
A	*		*	*	*	*	*
B	*		*	*	*	*	*
C	*	*		*	*	*	*
D	*	*					*
E		*	*				*
Процент правильных	80	60	60	60	60	60	100
Лучший вес	1/3	1/3	1/3	0	0		

Рис. 11.5. Равномерное взвешивание голосов не всегда дает наилучший возможный классификатор, даже когда избиратели одинаково точны, поскольку некоторые проблемные случаи сложнее, чем другие (здесь D и E). "" означает, что данный избиратель правильно классифицировал данный элемент*

Вторым подходом может быть использование линейной/логистической регрессии для поиска наилучших возможных весов. В задаче бинарной классификации два класса будут обозначаться как 0 и 1 соответственно. Результаты 0–1 по каждому классификатору можно использовать как функцию для прогнозирования фактического значения класса. Эта формулировка позволит найти неоднородные весовые коэффициенты, которые благоприятствуют классификаторам, соотнесенным с правильными ответами, но явно не стремятся максимизировать количество правильных классификаций.

11.3.2. Алгоритмы бустинга

Третья идея — это *бустинг* (boosting). Ключевым моментом является взвешивание примеров в соответствии с тем, насколько трудно их получить, и вознаграждение классификаторов на основе веса примеров, которые они получают, а не только количества.

Чтобы установить веса классификатора, мы регулируем веса обучающих примеров. Простые обучающие примеры будут должным образом классифицированы большинством классификаторов: мы вознаграждаем классификаторы больше за правильное решение сложных случаев.

Типичным алгоритмом бустинга является *AdaBoost*, представленный на рис. 11.6. Мы не будем здесь останавливаться на деталях, в частности на особенностях корректировки веса на каждом цикле. Мы предполагаем, что наш классификатор будет построен как объединение нелинейных классификаторов в форме "справедливо ли, что $(v_i \geq t_i)$?", т.е. с использованием пороговых значений элементов в качестве классификаторов.

AdaBoost

Для t из $1 \ldots T$:

- Выбрать $f_t(x)$:
  - Найти слабого ученика $h_t(x)$ который минимизирует ϵ_t, ошибку взвешенной суммы ошибок для неправильно классифицированных точек $\epsilon_t = \sum_i w_{i,t}$
  - Выбрать $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- Добавить в ансамбль:
  - $F_t(x) = F_{t-1}(x) + \alpha_t h_t(x)$
- Обновить веса:
  - $w_{i,t+1} = (w_{i,t}) e^{-y_i \alpha_t h_t(x_i)}$ для всех i;
  - Перенормализировать $w_{i,t+1} \quad \sum_i w_{i,t+1} = 1$

Рис. 11.6. Псевдокод алгоритма AdaBoost

Алгоритм работает за T циклов, для $t = \{0, \ldots, T\}$. Первоначально все учебные примеры (точки) должны иметь одинаковый вес, поэтому $w_{i,0} = 1/n$ для всех точек $x_1, \ldots, x_n$. Мы рассматриваем все возможные классификаторы признаков/порогов и определяем функцию $f_t(x)$, которая минимизирует ε_t, сумму весов ошибочно классифицированных точек. Вес α_t нового классификатора зависит от того, насколько он точен для текущей заданной точки, и измеряется как

$$\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}.$$

Вес точек нормализован, поэтому $\sum_{i=1}^{n} w_i = 1$ всегда должно быть классификатором с ошибкой $\varepsilon_t \leq 0{,}5$.[3]

На следующем цикле вес ошибочно классифицированных точек увеличивается, чтобы сделать их более важными. Пусть $h_t(x_i)$ будет классом (–1 или 1), предсказанным для x_i, а y_i правильным классом для этой точки. Знак $h_t(x_i)y$ отражает, соответствующий это класс (положительный) или нет (отрицательный). Затем мы корректируем веса в соответствии с

$$w'_{i,t+1} = w_{i,t} e^{-y_i \alpha_t h_t(x_i)}$$

[3] Рассмотрим два классификатора, один из которых называет класс C_0, если $x_i \geq t_i$, а другой — класс C_1, если $x_i \geq t_i$. Первый классификатор прав именно тогда, когда второй неправ, поэтому один из двух должен быть не менее 50%.

перед повторной нормализацией, чтобы их сумма продолжала равняться 1, т.е.

$$C = \sum_{i=1}^{n} w'_{i,t+1} \text{ и } w_{i,t+1} = w'_{i,t+1} / C.$$

Пример на рис. 11.7 показывает окончательный классификатор как линейную сумму трех пороговых классификаторов с одной переменной. Считайте их простейшими возможными деревьями решений, каждое из которых имеет ровно один узел. Веса, назначенные алгоритмом AdaBoost, не являются одинаковыми, но и не настолько искажены в этом конкретном случае, так что они ведут себя иначе, чем классификатор большинства. Посмотрите на нелинейную границу решения, проистекающую из дискретного характера пороговых тестов/деревьев решений.

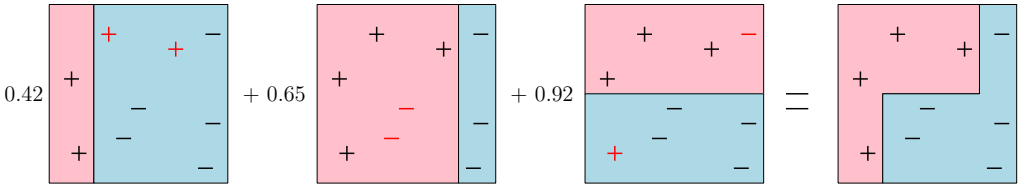


Рис. 11.7. Финальный классификатор представляет собой взвешенный ансамбль, который правильно классифицирует все точки, несмотря на ошибки в каждом составляющем классификаторе, которые выделены красным

Бустинг особенно ценен при применении к деревьям решений в качестве элементарных классификаторов. Популярный подход, *градиентно-бустинговое дерево решений* (Gradient Boosted Decision Tree — GBDT) обычно начинается с множества маленьких деревьев, каждое из которых может содержать от четырех до десяти узлов. Каждое такое дерево кодирует достаточно простую логику, чтобы не переобучить данные. Относительные веса, назначенные каждому из этих деревьев, следуют из процедуры обучения, которая стремится соответствовать ошибкам предыдущих циклов (остаткам) и увеличивает веса деревьев, которые правильно классифицируют более сложные примеры.

Бустинг усердно работает над правильной классификацией каждого обучающего экземпляра, а это означает, что особенно тяжело классифицировать самые сложные экземпляры. Существует пословица: “Трудные прециденты ведут к плохим законам” (hard cases make bad law), предполагающая, что трудно решаемые дела создают плохие прецеденты для последующего анализа. Это важный аргумент против бустинга, поскольку метод может показаться склонным к переобучению, хотя в целом он хорошо работает на практике.

Опасность переобучения особенно велика, когда учебные данные не являются идеальным золотым стандартом. Человеческие аннотации классов зачастую субъективны и противоречивы, что приводит к бустингу, чтобы усилить

шум при воздействии сигнала. Лучшие алгоритмы бустинга будут справляться с переобучением за счет регуляризации. Цель будет состоять в том, чтобы минимизировать количество ненулевых коэффициентов и избежать больших коэффициентов, которые слишком сильно верят в какой-либо один классификатор в ансамбле.

> *На заметку.* Бустинг может эффективно использовать слабые классификаторы. Тем не менее, когда часть ваших обучающих примеров аннотирована неправильно, он может повести себя особенно неправильно.

11.4. Метод опорных векторов

Метод опорных векторов (Support Vector Machine — SVM) является важным способом построения нелинейных классификаторов. Его можно считать родственниками логистической регрессии, которая искала линию/плоскость l, наилучшим образом разделяющую точки на два класса меток. Логистическая регрессия присваивает точке q метку своего класса в зависимости от того, лежит q выше или ниже этой линии l. Кроме того, она использует логит-функцию для преобразования расстояния от q до l в вероятность того, что q принадлежит указанному классу.

Вопросы оптимизации в логистической регрессии включали минимизацию суммы вероятностей ошибочной классификации по всем точкам. Метод опорных векторов, напротив, работает за счет поиска максимального предела *линейных* разделителей между двумя классами. На рис. 11.8 (слева) показаны красные и синие точки, разделенные линией. Эта линия стремится максимизировать расстояние d до ближайшей учебной точки, максимальный предел между красным и синим. Это естественная цель при построении границы принятия решений между двумя классами, поскольку, чем больше разница, тем дальше наши учебные точки от невозможности классификации. Классификатор максимального предела должен быть самым надежным разделителем между двумя классами.

Существует несколько свойств, которые помогают определить максимальный разделитель полей между наборами красных и синих точек.

- Оптимальная линия должна находиться посередине канала, на расстоянии d от ближайшей красной точки и ближайшей синей точки. Если бы это было не так, мы могли бы сдвигать линию до тех пор, пока она не поделит этот канал пополам, увеличивая таким образом поле.
- Действительный разделяющий канал определяется его контактом с небольшим количеством красных и синих точек, где "небольшое количе-

ство" означает количество, максимум в два раза превышающее размерность точек, для наборов точек с хорошим поведением, избегающих расположения $d + 1$ на любой d-мерной поверхности. Это отличается от логистической регрессии, где все точки способствуют выбору наилучшего положения линии. Эти контактные точки являются *опорными векторами* (support vector), определяющими канал.

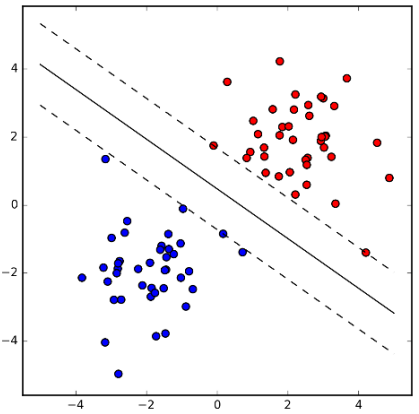


Рис. 11.8. SVM стремятся разделить два класса по наибольшему пределу, создавая канал вокруг разделительной линии

- Точки внутри выпуклого корпуса красного или синего абсолютно не влияют на разделитель максимального предела, поскольку нам нужно, чтобы все точки одного цвета находились на одной стороне границы. Мы могли бы удалить эти внутренние точки или переместить их, но разделитель максимального предела не изменится, пока одна из точек не покинет корпус и не войдет в разделительную полосу.
- Идеально отделить красный от синего с помощью прямой линии не всегда возможно. Представьте, что синяя точка лежит где-то внутри выпуклого корпуса красных точек. Невозможно будет отделить эту синюю точку от красной, используя только линию.

Совместно логистическая регрессия и метод опорных векторов создают разделительные линии между наборами точек. Они оптимизированы для разных критериев, а следовательно, могут быть разными, как показано на рис 11.9. Логистическая регрессия ищет разделитель, который максимизирует общее доверие к нашей классификации, суммируемой по всем точкам, в то время как разделитель SVM с широкими полями делает все возможное с ближайшими точками между наборами. Оба метода обычно дают похожие классификаторы.

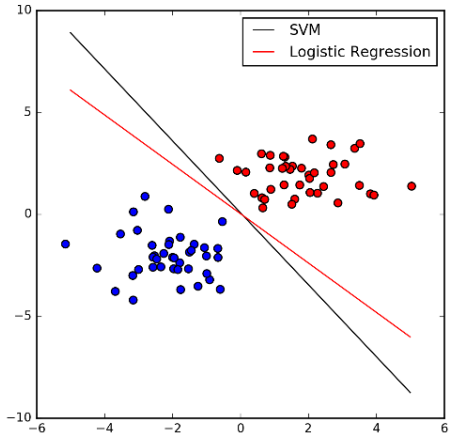


Рис. 11.9. И логистическая регрессия, и SVM дают разделительные линии между наборами точек, но оптимизированы они для разных критериев

11.4.1. Линейные SVM

Эти свойства определяют оптимизацию *линейного метода опорных векторов* (linear support vector machine). Разделительная линия/плоскость, как и любая другая линия/плоскость, может быть записана как

$$wx - b = 0$$

для вектора коэффициентов w, отмеченного вектором входных переменных x. Канал, разделяющий два класса, будет определен двумя параллельными ему равноудаленными линиями с обеих сторон, а именно: $wx - b = 1$ и $wx - b = 1$.

Фактическое геометрическое разделение между линиями зависит от w, а именно $2/\|w\|$. Рассмотрим наклонные линии в двух измерениях: эти линии расположены на расстоянии 2 друг от друга для горизонтальных линий, но весьма далеки друг от друга, если они почти вертикальны. Этот разделяющий канал должен быть лишен точек и действительно отделять красные точки от синих. Таким образом, мы должны добавить ограничения. Для каждой красной точки x_i (класс 1) мы настаиваем на том, что

$$wx - b \geq 1$$

в то время как каждая синяя точка x_i (класс – 1) должна удовлетворять требованию

$$wx - b \leq -1.$$

Если мы обозначим класс x_i как $y_i \in [-1, 1]$, то их можно объединить, чтобы получить задачу оптимизации

$$\max \|w\|, \text{ где } y_i(w x_i - b) \geq 1 \text{ для всех } 1 \leq i \leq n.$$

Это может быть решено с использованием методов, подобных линейному программированию. Обратите внимание, что канал должен быть определен точками, соприкасающимися с его границами. На эти векторы "опирается" (support) канал, от чего происходит название *метод опорных векторов* (support vector machine). Алгоритм оптимизации эффективных решателей, таких как LibLinear и LibSVM, осуществляет поиск по соответствующим небольшим подмножествам опорных векторов, которые потенциально определяют разделяющие каналы, чтобы найти самый широкий.

Существуют более общие критерии оптимизации для SVM, которые ищут линию, определяющую широкий канал, и штрафуют (но не запрещают) точки, которые неправильно классифицированы. Такого рода двойственная функция (расширение канала при неправильной классификации нескольких точек) может рассматриваться как форма регуляризации с константой для компромисса между двумя целями. Для решения этих общих проблем может быть использован поиск градиентного спуска.

11.4.2. Нелинейные SVM

SVM определяют гиперплоскость, которая разделяет точки двух классов. Плоскости — это линии в более высоких измерениях, которые легко определяются с помощью линейной алгебры. Так как же этот линейный метод может создать нелинейную границу решения?

Чтобы в заданной точке был установлен максимальный разделитель, два цвета должны быть сначала разделены линейно. Но, как мы видели, это не всегда так. Рассмотрим патологический случай на рис. 11.10 (слева), где кластер красных точек окружен кольцевым кластером черных точек. Как может возникнуть такая вещь? Предположим, мы разделяем пункты назначения на *однодневные* или *длительные*, в зависимости от того, достаточно ли они близки к нашему местоположению. Долгота и широта каждого возможного пункта назначения будут давать данные с точно такой же структурой, как на рис. 11.10 (слева).

Основная идея заключается в том, что мы можем проецировать наши d-мерные точки в пространство более высокого измерения, где будет больше возможностей для их разделения. Для n красных/синих точек вдоль линии в одном измерении существует только $n-1$ потенциально интересных способов их разделения, особенно с разрезом между i-й и $(i+1)$-й точками для $1 \leq i < n$. Но это приводит к примерно $\binom{n}{2}$ путям, когда мы перемещаемся в два измерения, потому что есть

больше свободы для разделения, когда мы увеличиваем размерность. На рис. 11.10 (справа) показано, как поднятие точек в ходе преобразования $(x, y) \rightarrow (x, y, x^2 + y^2)$ помещает их в параболоид и позволяет линейно разделять классы, которые были неразделимы в исходном пространстве.

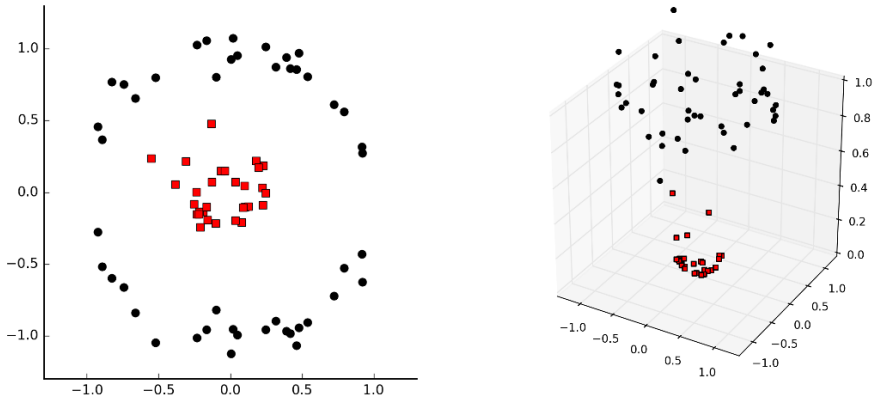


Рис. 11.10. Проецирование точек на более высокие размерности может сделать их линейно разделимыми

Если мы поднимем размерность любой двухклассной точки, установленной достаточно высоко, между красной и черной точками всегда будет разделительная линия. Если мы поместим n точек в n измерениях в ходе разумного преобразования, они всегда будут линейно разделены очень простым способом. Чтобы стало понятней, подумайте о частном случае двух точек (одна красная и одна синяя) в двух измерениях: очевидно, что должна быть разделяющая их линия. Проецирование этой разделительной плоскости вниз на исходное пространство приводит к некой форме изогнутой границы решения, и, следовательно, нелинейность SVM зависит от того, как именно был спроецирован вход на многомерное пространство.

Хороший способ превращения n точек в d измерениях в n точек в n измерениях заключается в представлении каждой точки ее расстояниями до всех n входных точек. В частности, для каждой точки p_i мы можем создать вектор v_i такой, что $v_{ij} = dist(i, j)$, расстояние от p_i до p_j. Вектор таких расстояний должен служить мощным набором признаков для классификации любой новой точки q, поскольку расстояние до членов фактического класса должно быть небольшим по сравнению с расстояниями другого класса.

Это пространство признаков действительно мощное, и можно легко представить написание функции для преобразования исходной матрицы признаков $n \times d$ в новую матрицу признаков $n \times n$ для классификации. Проблема здесь в пространстве, поскольку количество входных точек n обычно намного больше,

чем размерность d, в которой они находятся. Такое преобразование было бы возможно построить только для довольно небольших наборов точек, скажем, $n \leq 1000$. Далее, работа с такими многомерными точками должна быть очень дорога, поскольку каждая оценка расстояния занимает линейное время по числу точек n вместо измерения данных d. Но происходит нечто удивительное...

11.4.3. Ядра

Волшебство SVM состоит в том, что эту матрицу пространственных объектов *на самом деле* не нужно вычислять явно. Фактически оптимизация, присущая нахождению разделителя максимального предела, выполняет только скалярное произведение точек с другими точками и векторами. Таким образом, мы могли бы представить увеличение расстояния по оси y, когда связанная точка используется в сравнении. Следовательно, нет необходимости предварительно вычислять матрицу расстояний: мы можем расширять точки от d до n измерений по мере необходимости, выполнить вычисление расстояния, а затем отбросить расширения.

Это сработало бы для устранения узкого места в пространстве, но мы все равно заплатили бы высокую цену во время вычислений. Удивительно то, что есть такие функции, как *ядра* (kernel), которые возвращают то, что по сути является вычислением расстояния для большего вектора, без построения самого большего вектора. Выполнение SVM с ядрами позволяет нам найти лучший разделитель для множества нелинейных функций без особых дополнительных затрат. Математика выходит за рамки того, что я хотел бы охватить здесь.

> *На заметку.* Функции ядра — это то, что позволяет SVM разделять d-мерные точки проекта на n измерений, поэтому их можно разделить, *не тратя* на вычисления более d шагов.

Для эффективного использования метода опорных векторов требуют опыта. Помимо ядра расстояния, доступно много других функций ядра, которые я представил здесь. Каждая из них имеет преимущества для определенных наборов данных, поэтому для достижения максимальной производительности необходимо использовать такие инструменты, как LibSVM. Они лучше всего работают с наборами данных среднего размера, с тысячами, но не миллионами точек.

11.5. Степени контроля

Существует естественное различие между подходами машинного обучения, основанными на степени и характере *контроля* (supervision), используемого при

накоплении данных обучения и оценки. Как и в любой таксономии, на границах есть некоторая нечеткость, что делает крайне затруднительными попытки точно определить, что представляет собой данная система и чего она не делает. Однако, как и любая *хорошая* таксономия, она дает вам основу для мышления и предлагает подходы, которые могут привести к лучшим результатам.

Методы, обсуждаемые до сих пор в этой главе, предполагают, что нам дают учебные данные с метками классов или целевыми переменными, оставляя нашу задачу как единственную для обучения систем классификатора или регрессии. Однако добраться до точки маркировки данных, как правило, сложно. Алгоритмы машинного обучения работают тем лучше, чем больше данных вы можете им дать, но аннотации зачастую трудны и дороги. Модуляция уровня контроля позволяет повысить громкость, чтобы ваш классификатор мог слышать, что происходит.

11.5.1. Обучение с учителем

Обучение с учителем (supervised learning) — это парадигма, основанная на проблемах классификации и регрессии. Нам даны векторы функций x_i, каждый со связанной меткой класса или целевым значением y_i. Аннотации y_i представляют собой контроль, как правило, полученный из некоторого ручного процесса, который ограничивает потенциальный объем обучающих данных.

В некоторых задачах аннотация обучающих данных осуществляется из наблюдений при взаимодействии с миром или, по крайней мере, из его модели. Программа AlphaGo от Google стала первой компьютерной программой, обыгравшей чемпиона мира по Го. Функция оценки позиции — это функция оценки, которая получает позицию на доске и вычисляет число, оценивая, насколько она сильна. Функция оценки позиции AlphaGo была обучена на всех опубликованных играх мастеров-людей, но для этого потребовалось очень много данных. Решение, по сути, заключалось в построении функции оценки позиции в ходе обучения при игре против себя самого. Оценка положения существенно улучшена поиском на несколько шагов вперед перед вызовом функции оценки для каждого листа. Попытка предсказать оценку после поиска без самого поиска дает более сильную функцию оценки. Генерация этих обучающих данных является просто результатом вычислений: программа играет сама против себя.

Эта идея обучения, исходя из окружения, называется *обучением с подкреплением* (reinforcement learning). Его нельзя применять повсеместно, но всегда стоит искать разумные подходы для генерации механически аннотированных обучающих данных.

11.5.2. Обучение без учителя

Методы *обучения без учителя* (unsupervised learning) пытаются найти структуру в данных, предоставляя метки (кластеры) или значения (ранжирование) без какого-либо конкретного стандарта. Их лучше всего использовать для исследования, для извлечения смысла из набора данных.

Основой всех методов обучения без учителя является *кластеризация* (clustering), о которой мы подробно говорили в разделе 10.5. Обратите внимание, что кластеризация может использоваться при предоставлении учебных данных для классификации даже при отсутствии меток. Если предположить, что найденные кластеры представляют собой подлинное явление, мы можем использовать идентификатор кластера в качестве метки для всех элементов в данном кластере. Теперь они могут служить обучающими данными при построении классификатора для прогнозирования идентификатора кластера. Прогнозирование идентификаторов кластера может быть полезным, даже если с этими понятиями не связаны имена, предоставляя резонную метку для любой входной записи q.

Тематическое моделирование

Другим важным классом методов обучения без учителя является *тематическое моделирование* (topic modeling), обычно связываемое с документами, написанными с использованием заданного словаря. Документы пишутся по темам, обычно это разные темы. Эта книга разделена на главы, каждая из которых посвящена отдельной теме, но она затрагивает также различные темы, от бейсбола до свадеб. Но что это за тема? Обычно каждая тема связана с определенным набором словарных слов. В статьях о бейсболе упоминаются *hits*, *pitchers*, *strikeouts*, *bases* и *slugging* (хиты, питчеры, ауты, базы и слаггинг). С темой свадеб связаны такие слова, как *married*, *engaged*, *groom*, *bride*, *love* и *celebrate* (женатые, помолвленные, жених, невеста, любовь и празднование). Определенные слова могут представлять несколько тем. Например, *любовь* также связана с теннисом, а *хиты* с гангстерами.

Если у вас есть набор тем (t_1, ..., t_k) и слова, которые их определяют, задача идентификации конкретных тем, связанных с любым заданным документом d, кажется довольно простой. Мы подсчитываем количество вхождений слов в d в общем с t_i и сообщаем об успехе всякий раз, когда совпадение достаточно высоко. Если задан набор документов, помеченных вручную темами, представляется разумным подсчитать частоту каждого слова в каждом классе тем, чтобы составить список слов, наиболее тесно связанных с каждой темой.

Но это все очень строго контролируется. *Моделирование тем* — это подход обучения без учителя, который выводит темы и списки слов с нуля, для этого

достаточно немаркированного документа. Мы можем представить эти тексты с помощью частотной матрицы F размером $w \times d$, где w — это размер словаря, а d — количество документов, при этом $F[i, j]$ отражает, сколько раз слово i встречается в документе j. Предположим, что мы разлагаем F на $F \approx W \times D$, где W — это матрица слово-тема $w \times t$, а D — это матрица тема-документ $t \times d$. Самые большие записи в i-й строке W отражают темы, с которыми наиболее тесно связано слово w_i, в то время как самые большие записи в j-м столбце D отражают темы, лучше всего представленные в документе d_j.

Такое разложение будет представлять собой в точности форму обучения без учителя, за исключением указания желаемого количества тем t. Построение такого приблизительного разложения кажется довольно грязным процессом, но существует множество подходов, чтобы попытаться это сделать. Вероятно, самым популярным методом тематического моделирования является подход *латентного размещения Дирихле* (Latent Dirichlet Allocation — LDA), производящий аналогичный набор матриц W и D, хотя и не строго разложением.

На рис. 11.11 представлен пример LDA в действии. Были проанализированы три разные книги, чтобы показать, как они были организованы по трем скрытым темам. Алгоритм LDA определял эти темы способом без учителя, присваивая каждому слову весовые коэффициенты в зависимости от того, какой вклад оно вносит в каждую тему. Результаты здесь, как правило, эффективны: понятие каждой темы вытекает из ее наиболее важных слов (справа), а распределение слов в каждой книге может быть легко разделено на три скрытые темы (слева).

Текст	T_1	T_2	T_3
Библия	0,73	0,01	0,26
Наука о данных: учебный курс	0,05	0,83	0,12
Кто больше?	0,08	0,23	0,69

T_1		T_2		T_3	
Термин	**Вес**	**Термин**	**Вес**	**Термин**	**Вес**
Бог	0,028	CPU	0,021	Прошлос	0,013
Иисус	0,012	Компьютер	0,010	История	0,011
Молиться	0,006	Данные	0,005	Старый	0,006
Израиль	0,003	Программа	0,003	Война	0,004
Моисей	0,001	Математика	0,002	Книга	0,002

Рис. 11.11. Пример тематического моделирования (LDA). Три книги представлены распределением тем (слева). Каждая тема представлена списком слов, с весовой мерой ее важности для темы (справа). Документы состоят из слов: магия LDA заключается в том, что он одновременно выводит назначения тем и слов без учителя

Обратите внимание: такое мышление разложения может быть применено не только к документам, а к любой матрице признаков F. Подходы разложения матриц, которые мы ранее обсуждали, включая разложение сингулярных значений и анализ главных компонентов, в равной степени не контролируемы, индуцируя структуру, присущую наборам данных без нашего вмешательства в его поиск.

11.5.3. Обучение с частичным привлечением учителя

Разрыв между обучением с учителем и без учителя заполняет *обучение с частичным привлечением учителя* (semi-supervised learning), которое увеличивают небольшое количество маркированных обучающих данных до большего. Превращение небольшого количества примеров в большее зачастую называют *бутстрэпингом* (bootstrapping), исходя из поговорки "pulling yourself up from your bootstraps" (спасение утопающих — дело рук самих утопающих). Подходы обучения с частичным привлечением учителя олицетворяют хитрость, которую необходимо использовать для создания предметных учебных наборов.

Предположим, что нам дано небольшое количество помеченных примеров в виде пар (x_i, y_i), подкрепленных большим количеством вводов x_j неизвестных меток. Вместо того чтобы напрямую строить нашу модель из обучающего набора, мы можем использовать ее для классификации массы немаркированных экземпляров. Возможно, для классификации этих неизвестных мы используем подход ближайшего соседа или любой другой подход, который мы здесь обсуждали. Но, как только мы их классифицируем, мы предполагаем, что метки верны и переходим к большему набору.

Такие подходы сильно выигрывают от наличия надежного оценочного набора. Нам нужно установить, что модель, обученная на бутстрэпинговых примерах, работает лучше, чем модель, с которой мы начали. Добавление миллиардов обучающих примеров бесполезно, если метки являются мусором.

Существуют и другие способы генерации обучающих данных без аннотаций. Зачастую кажется, что легче найти положительные примеры, чем отрицательные. Рассмотрим проблему подготовки грамматического корректора, способного отличать правильные фрагменты текста от плохо сформированного материала. Легко получить большое количество правильных примеров английского языка: все, что публикуется в книгах и газетах, обычно считается хорошим. Но, кажется, труднее овладеть большим корпусом неправильного текста. Тем не менее мы можем заметить, что случайное добавление, удаление или замена произвольных слов в любом тексте почти всегда ухудшает его.[4] Помечая весь публикуемый текст как правильный, а все случайные возмущения как неправильные, мы можем создать настолько большой обучающий набор, насколько пожелаем, причем без найма кого-то для его аннотирования.

---

[4] Давайте попробуем где-нибудь на этой странице. Выберите слово наугад и замените его словом *red* (красный). Затем снова артиклем *the*. И, наконец, словом *defenestrate* (выгнать с работы). Мой оригинальный текст написан лучше, чем то, что вы получаете после такого изменения?

Как мы можем оценить такой классификатор? Зачастую можно получить достаточно подлинных аннотированных данных для оценки, поскольку обычно для этого нам нужно их гораздо меньше, чем для обучения. Мы также можем использовать наш классификатор для того, чтобы предложить, что аннотировать. Наиболее ценные примеры для проверки аннотатора — это те, в которых наш классификатор допускает ошибки: опубликованные предложения, помеченные как неправильные, или случайные мутации, которые проходят тест, стоит передать человеческому судье.

11.5.4. Проектирование признаков

Проектирование признаков (feature engineering) — это искусство применения знаний в предметной области для облегчения алгоритмам машинного обучения выполнения запланированной работы. В контексте нашей таксономии проектирование признаков может рассматриваться как важная часть обучения с учителем, где контроль применяется к векторам признаков x_i вместо связанных целевых аннотаций y_i.

Важно убедиться, что признаки представлены моделям таким образом, чтобы модель могла их правильно использовать. Включение специфических знаний области в данные вместо их изучения кажется жульничеством. Но профессионалы понимают, что есть вещи, которые не могут быть легко изучены, а следовательно, их лучше явно включить в набор признаков.

Рассмотрим модель прогнозирования цены на аукционах. Аукционные дома зарабатывают деньги, взимая комиссию с победителя торгов, помимо того, что они платят владельцу. Разные дома взимают разные ставки, и они могут быть весьма существенны. Поскольку общая стоимость для победителя слагается из цены покупки и комиссии, более высокие комиссионные могут значительно снизить цену покупки, сократив то, что покупатель может позволить себе заплатить владельцу.

Итак, как вы можете представить комиссионную цену в модели ценообразования? Я могу придумать как минимум три разных подхода, некоторые из которых могут иметь катастрофические последствия.

- *Указать процент комиссии в качестве признака.* Представление сокращения домов (скажем, 10%) в виде столбца в наборе признаков может быть невозможно использовать в линейной модели. Хит, полученный участником торгов, является произведением налоговой ставки и окончательной цены. Он имеет мультипликативный эффект, а не аддитивный, а следовательно, не может быть осмысленно использован, если диапазон цен на произведение искусства колеблется от 100 до 1 000 000 долл.
- *Включить фактическую комиссию в качестве признака.* Мошенничество... Если вы включаете комиссию в конечный выплачиваемый итог

как признак, вы загрязняете функции данными, не известными на момент проведения аукциона. В самом деле, если бы все картины продавались с 10%-ным налогом, а уплаченный налог был признаком, совершенно точная (и совершенно бесполезная) модель предсказывала бы цену в десять раз больше, чем уплаченный налог!

- *Установить целевую переменную регрессии равной общей выплачиваемой сумме.* Поскольку комиссионные ставки дома и дополнительные сборы известны покупателю до того, как он сделает ставку, правильной целевой переменной должна быть общая выплачиваемая сумма. Любой данный прогноз общей цены покупки может быть разделен позже на цену покупки, комиссию и налоги в соответствии с правилами дома.

Проектирование признаков можно рассматривать как зависящую от домена версию очистки данных, поэтому здесь применимы все методы, описанные в разделе 3.3. Наиболее важные из них будут рассмотрены в контексте, поскольку мы достигли наконец точки фактического построения моделей, управляемых данными.

- *Z-оценки и нормализация.* Нормально распределенные значения в сопоставимых числовых диапазонах дают лучшие характеристики. Чтобы сделать диапазоны сопоставимыми, превратите значения в Z-оценки, вычтя среднее значение и разделив на стандартное отклонение, $Z = (x - \mu)/\sigma$. Чтобы сделать переменную степенного закона более нормальной, измените x в наборе функций на $\log x$.
- *Вставка недостающих значений.* Убедитесь, что в ваших данных нет пропущенных значений, и, если это так, замените их значимым предположением или оценкой. Регистрация того, что чей-то вес равен –1, — это легкий способ испортить любую модель. Самый простой метод вставки — заменить каждое пропущенное значение средним значением данного столбца, и этого в целом достаточно, но более мощные методы обучают модель прогнозированию пропущенного значения на основе других переменных в записи (см. раздел 3.3.3).
- *Уменьшение размеров.* Напомним, что *регуляризация* (regularization) — это способ заставить модели отказаться от несущественных признаков, чтобы предотвратить переобучение. Куда эффективней исключить ненужные признаки перед созданием моделей, удалив их из набора данных. Когда признак x, скорее всего, не имеет отношения к вашей модели? Плохая корреляция с целевой переменной y, а также отсутствие какой-либо качественной причины, по которой вы можете указать, почему x может повлиять на y, являются отличными показателями.

Методы уменьшения размерности, такие как разложение по сингулярным числам, являются отличными способами уменьшить большие векторы признаков до более мощных и кратких представлений. К преимуществам относятся более быстрое обучение, меньшее переобучение и снижение шума от наблюдений

- *Явное включение нелинейных комбинаций.* Некоторые произведения или соотношения переменных признаков имеют естественную интерпретацию в контексте. Площадь или объем — это произведения длины, ширины и высоты, но они не могут быть частью линейной модели, если явно не указан столбец в матрице объектов. Итоговые суммы, такие как спортивные очки, полученные за карьеру, или общий объем заработанных долларов, обычно несопоставимы с предметами разного возраста или продолжительности. Но преобразование итогов в коэффициенты (например, количество очков за игру или доллары в час) обычно дает более значимые признаки.

  Для определения этих произведений и соотношений требуется информация, относящаяся к конкретному домену, и тщательно продуманный процесс разработки признаков. У вас гораздо больше шансов узнать правильные комбинации, чем у вашего нелинейного классификатора.

Не стесняйтесь. Разница между хорошей моделью и плохой моделью обычно сводится к качеству ее функциональности. Усовершенствованные алгоритмы машинного обучения очаровательны, но именно подготовка данных дает результаты.

11.6. Глубокое обучение

Алгоритмы машинного обучения, которые мы здесь изучали, не очень хорошо масштабируются до на самом деле *больших* массивов данных по нескольким причинам. Такие модели, как линейная регрессия, обычно имеют относительно немного параметров, скажем, один коэффициент на столбец, а следовательно, не могут реально извлечь выгоду из огромного количества обучающих примеров. Если данные имеют хорошее линейное соответствие, вы сможете найти их и с небольшим набором данных. А если нет, ну, в общем-то, вы действительно не хотели его искать.

Глубокое обучение (deep learning) — это невероятно захватывающая недавняя разработка в машинном обучении. Оно основано на *нейронных сетях* (neural network), популярном подходе 1980-х годов, который затем вышел из моды. Но за последние пять лет что-то произошло, и внезапно многослойные (глубокие) сети начали дико превосходить традиционные подходы к

классическим проблемам в области компьютерного зрения и обработки естественного языка.

Почему именно это произошло, остается загадкой. Похоже, что не было фундаментального алгоритмического прорыва, просто объем данных и скорость вычислений переступили порог, когда способность использовать огромные объемы обучающих данных превзошла методы, более эффективные при работе с ограниченным ресурсом. Но инфраструктура быстро развивается, чтобы использовать это преимущество: такие новые программные среды с открытым исходным кодом, как *TensorFlow* от Google, позволяют легко задавать сетевые архитектуры для процессоров специального назначения, предназначенных для ускорения обучения на порядки.

Что отличает глубокое обучение от других подходов, так это то, что оно обычно избегает проектирования признаков. Каждый слой в нейронной сети обычно принимает в качестве входных данных выходные данные своего предыдущего уровня, получая постепенно более высокоуровневые признаки по мере того, как мы продвигаемся к вершине сети. Это служит для определения иерархии понимания от исходного ввода до конечного результата, и, действительно, предпоследний уровень сети, предназначенной для одной задачи, часто предоставляет полезные функции высокого уровня для связанных задач.

Почему нейронные сети так успешны? Никто на самом деле не знает. Есть признаки того, что для многих задач не нужен полный вес этих сетей; то, что они делают, в конечном итоге будет сделано с использованием менее непрозрачных методов. Нейронные сети, кажется, работают за счет переобучения, находя способ использовать миллионы примеров, чтобы соответствовать миллионам параметров. Тем не менее им обычно удается избежать наихудшего поведения переобучения, возможно, используя менее точные способы кодирования знаний. Система, явно запоминающая длинные строки текста для расщепления по требованию, покажется хрупкой и неоправданной, в то время как система, представляющая такие фразы более свободным способом, может быть более гибкой и обобщаемой.

Эта область развивается достаточно быстро, хотя я бы хотел, чтобы мое повествование оставалось строго на уровне идей. Каковы основные свойства этих сетей? Почему они вдруг стали такими успешными?

> *На заметку.* Глубокое обучение — это очень интересная технология, у которая уже умет ходить, хотя она лучше всего подходит для областей с огромным количеством обучающих данных. Таким образом, большинство моделей науки о данных будет по-прежнему строиться с использованием традиционных алгоритмов классификации и регрессии, которые мы подробно описали ранее в этой главе.

11.6.1. Сети и глубина

На рис. 11.12 показана архитектура сети глубокого обучения. Каждый узел x представляет вычислительную единицу, которая вычисляет значение данной простой функции $f(x)$ по всем входам в нее. На данный момент ее можно рассматривать как простой сумматор, который суммирует все входные данные, а затем выводит сумму. Каждое направленное ребро (x, y) соединяет выход узла x со входом узла y выше в сети. Кроме того, каждому такому фронту соответствует коэффициент умножения $w_{x,y}$. Значение, фактически переданное y, представляет собой $w_{x,y} f(x)$, что означает, что узел y вычисляет взвешенную сумму своих входных данных.

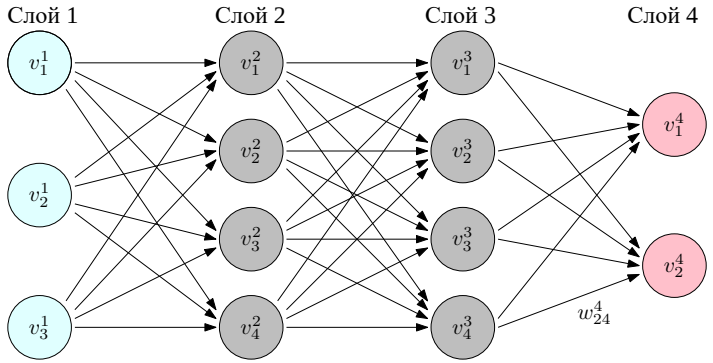


Рис. 11.12. Сети глубокого обучения имеют скрытые уровни параметров

В левом столбце рис. 11.12 представлен набор входных переменных, значения которых меняются всякий раз, когда мы просим сеть сделать прогноз. Считайте это интерфейсом к сети. Ссылки отсюда на следующий уровень распространяют это входное значение на все узлы, которые будут с ним работать. Справа находятся одна или несколько выходных переменных, представляющих окончательные результаты этого вычисления. Между этими входным и выходным уровнями располагаются *скрытые уровни* (hidden layer) узлов. Учитывая веса всех коэффициентов, структуру сети и значения входных переменных, вычисление становится простым: вычислить значения самого низкого уровня в сети, распространить их вперед и повторять со следующего уровня, пока вы не достигнете вершины.

Обучение (learning) сети означает установку весов параметров коэффициента $w_{x,y}$. Чем больше ребер, тем больше параметров мы должны изучить. В принципе, обучение означает анализ учебного корпуса пар (x_i, y_i) и корректировку весов параметров ребер таким образом, чтобы выходные узлы генерировали нечто близкое к y_i при подаче входных данных x_i.

Глубина сети

Глубина сети должна в некотором смысле соответствовать концептуальной иерархии, связанной с моделируемыми объектами. Изображение, которое мы должны получить, — это то, как входные данные последовательно преобразуются, фильтруются, увариваются и ужимаются все в лучшую и лучшую форму по мере продвижения по сети. По правде говоря, количество узлов должно постепенно уменьшаться по мере продвижения к более высоким уровням.

Мы можем рассматривать каждый слой как уровень абстракции. Рассмотрим проблему классификации изображений, когда необходимо решить, содержит изображение кота или нет. Размышляя в терминах последовательных уровней абстракции, можно сказать, что изображения созданы из пикселей, соседних участков, краев, текстур, областей, простых объектов, составных объектов и сцен. Это аргумент за то, что по крайней мере восемь уровней абстракции потенциально могут быть распознаны и использованы сетями для изображений. Подобные иерархии существуют в понятии документа (символы, слова, фразы, предложения, абзацы, разделы, документы) и любых других артефактов аналогичной сложности.

Действительно, сети глубокого обучения, подготовленные для конкретных задач, могут создавать ценные признаки общего назначения, выставляя выходные данные более низких уровней в сети как мощные признаки для традиционных классификаторов. Например, *Imagenet* — популярная сеть для распознавания объектов по изображениям. Один высокоуровневый слой из 1000 узлов измеряет уверенность в том, что изображение содержит объекты каждого из 1000 различных типов. Шаблоны того, какие объекты проявились и до какой степени, обычно полезны для других задач, таких как измерение сходства изображений.

Мы не навязываем никакого реального представления о том, что каждый из этих уровней должен представлять, только для того, чтобы соединить их так, чтобы был потенциал для распознавания такой сложности. Neighborhood patches (соседние участки) являются функциями небольших групп связанных пикселей, в то время как регионы будут состоять из небольшого количества связанных участков. В разработку топологии входит некоторый смысл того, что мы пытаемся распознать, но сеть делает то, что, по ее мнению, она должна делать во время обучения, чтобы минимизировать ошибки или *потери*.

Недостатки более глубоких сетей в том, что им становится все сложнее обучаться, чем они больше и глубже. Каждый новый уровень добавляет новый набор весовых параметров, увеличивая риск переобучения. Правильно предусмотреть влияние ошибок предсказания весов ребер становится все труднее, поскольку растет количество промежуточных уровней между листом и

наблюдаемым результатом. Тем не менее сети с более чем десятью уровнями и миллионами параметров были успешно обучены, и, по правде говоря, производительность распознавания увеличивается со сложностью сети.

По мере увеличения глубины сети также становятся более вычислительно дорогостоящими, поскольку вычисление занимает время, линейно пропорциональное количеству ребер в сети. Это не страшно, особенно потому, что все узлы на любом данном уровне могут обрабатываться параллельно на нескольких ядрах для сокращения времени прогнозирования. Время обучения — это реально узкое место вычислений.

Нелинейность

Фактор возрастания уровней абстракции в скрытых слоях сети, безусловно, является убедительным. Однако справедливо спросить, реально ли это. Действительно ли дополнительные слои в сети дают нам дополнительную вычислительную мощность, чтобы сделать то, чего мы не можем сделать при меньших затратах?

Пример на рис. 11.13, кажется, доказывает обратное. Он демонстрирует суммирующие сети, построенные с двумя и тремя уровнями узлов соответственно, но обе вычисляют точно ту же функцию на всех входах. Это говорит о том, что в дополнительном слое не было необходимости, за исключением, возможно, уменьшения инженерного ограничения степени узла и количества ребер, вводимых в качестве входных данных.

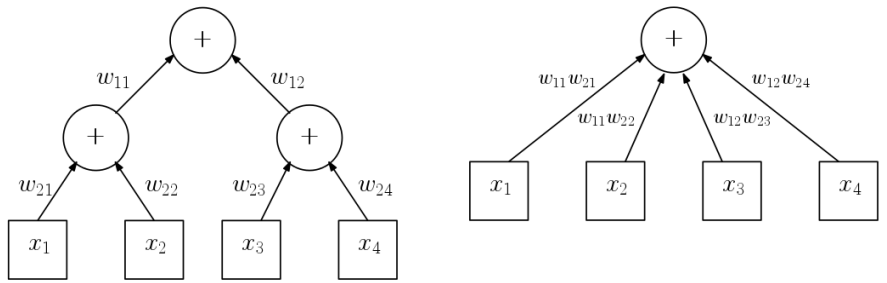


Рис. 11.13. Суммирующие сети не извлекают выгоды из глубины. Двухуровневая сеть (слева) вычисляет точно ту же функцию, что и эквивалентная однослойная сеть (справа)

На самом деле это показывает, что нам нужны более сложные нелинейные функции *активизации* (activation) узла $\phi(v)$, чтобы воспользоваться преимуществами глубины. Нелинейные функции не могут быть составлены так же, как суммирование может быть составлено для получения сложения. Эта нелинейная функция активизации $\phi(v_i)$ обычно работает с взвешенной суммой входов x, где

$$v_i = \beta + \sum_i w_i x_i.$$

Здесь β является константой для данного узла, возможно, для изучения в процессе обучения. Это называется *смещением* (bias) узла, поскольку оно определяет активизацию при отсутствии других входов.

То, что вычисление выходных значений уровня l включает применение функции активизации ϕ к взвешенным суммам значений из уровня $l - 1$, имеет важное значение для производительности. В частности, оценка нейронной сети в основном включает только одно умножение матриц на уровень, где взвешенные суммы получаются в результате умножения $|V_l| \times |V_{l-1}|$ весовой матрицы W на $|V_{l-1}| \times 1$ выходного вектора V_{l-1}. Далее каждый элемент результирующего вектора $|V_l| \times 1$ попадает в функцию ϕ, чтобы подготовить выходные значения для этого слоя. Быстрые библиотеки для умножения матриц могут очень эффективно выполнять такую оценку.

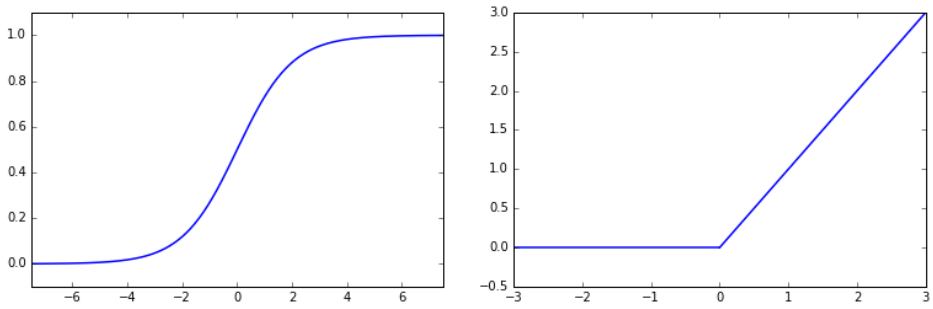


Рис. 11.14. Логистическая функция (слева) и функция ReLU (справа) активизации для узлов в нейронных сетях

Набор интересных нелинейных функций активизации был развернут в строительных сетях. Два из наиболее заметных, показанных на рис. 11.14, включают в себя следующее.

- *Логит-функция*. С *логистической функцией*, или логит-функцией, мы уже встречались при обсуждении логистической регрессии при классификации.

  $$f(x) = \frac{1}{1 + e^{-x}}.$$

  Эта секция обладает тем свойством, что выход ограничен диапазоном [0, 1], где $f(0) = 1/2$. Кроме того, функция является дифференцируемой, поэтому для обучения полученной сети может использоваться обратное распространение.

- *Линейный выпрямитель* (Rectified linear unit — ReLU). *Выпрямитель* (rectifier), или диод, в электрической цепи позволяет току течь только в одном направлении. Его функция отклика $f(x)$ линейна, когда значение x положительно, но равна нулю, когда значение x отрицательно, как показано на рис. 11.14 (справа).

$$f(x)=\begin{cases}x, x \geq 0;\\ 0, x < 0.\end{cases}$$

  Этого перегиба при $x = 0$ достаточно, чтобы убрать линейность из функции, и он обеспечивает естественный способ выключения устройства, приводя его в отрицательное значение. Функция ReLU остается дифференцируемой, но имеет совершенно иной отклик, чем логит-функция, монотонно возрастая и будучи неограниченной с одной стороны.

Я на самом деле не в курсе теории, объясняющей, почему определенные функции должны работать лучше в определенных контекстах. Некоторые функции активизации, по-видимому, стали популярными, потому что они хорошо работали в экспериментах, при этом выбор устройства — это то, что вы можете изменить, если чувствуете, что ваша сеть работает не так, как должна.

По правде говоря, добавление одного скрытого слоя значительно добавляет мощность сети, а дополнительные уровни страдают от убывающей отдачи. Теория показывает, что сети без каких-либо скрытых слоев способны распознавать линейно разделимые классы, но мы обратились к нейронным сетям для создания более мощных классификаторов.

> *На заметку*. Начните с одного скрытого уровня с несколькими узлами, количеством от размера входного до выходного слоев, поскольку они вынуждены изучать сжатые представления, обеспечивающие мощные признаки.

11.6.2. Обратное распространение

Обратное распространение (backpropagation) — это основная процедура обучения для нейронных сетей, которая достигает очень впечатляющих результатов, постепенно подбирая большое количество параметров на больших учебных наборах. Это очень напоминает случайный градиентный спуск, который мы ввели в разделе 9.4.

Наша основная проблема заключается в следующем. Дана нейронная сеть с предварительными значениями для каждого параметра w^l_{ij}, что означает множитель, который вывод узла v^{l-1}_j получает перед добавлением в узел v^l_i. Нам также дается обучающий набор, состоящий из n пар входной вектор — выходное значение (x_a, y_a), где $1 \leq a \leq n$. В нашей сетевой модели вектор x_i представляет

значения, которые должны быть назначены входному слою v^1, а y_i — это желаемый отклик выходного слоя v_l. Оценка текущей сети по x_i приведет к выходному вектору v_l. Ошибка E_l сети на уровне l может быть измерена, возможно, как

$$E_l = \| y_i - v^l \|^2 = \sum_j \left(\phi \left(\beta + \sum_i w_{ij}^l v_{ij}^{l-1} \right) - y_{ij} \right)^2 .$$

Мы хотели бы улучшить значения весовых коэффициентов w_{ij}^l, чтобы они лучше предсказывали y_i и минимизировали E_l. Вышеупомянутое уравнение определяет потери E_l как функцию весовых коэффициентов, поскольку входные значения из предыдущего слоя v^{l-1} являются фиксированными. Как и при стохастическом градиентном спуске, текущее значение w_{ij}^l определяет точку p на этой поверхности ошибки, а производная E_l в этой точке определяет направление наискорейшего спуска, уменьшая ошибки. Пройдя расстояние d в этом направлении, определяемом текущим размером шага, или *скоростью обучения* (learning rate), вы получите обновленные значения коэффициентов, у которых v_l лучше предсказывает y_a из x_a.

Но это только меняет коэффициенты в выходном слое. Чтобы перейти к предыдущему уровню, обратите внимание, что предыдущая оценка сети обеспечивала вывод для каждого из этих узлов как функцию ввода. Чтобы повторить ту же процедуру обучения, нам нужно целевое значение для каждого узла в слое $l-1$, чтобы играть роль y_a из нашего примера обучения. Учитывая y_a и новые весовые коэффициенты для вычисления v^l, мы можем вычислить значения для выходных данных этих слоев, которые отлично предскажут y_i. С этими целями мы можем изменять вес коэффициентов на данном уровне и продолжать распространяться в обратном направлении, пока не достигнем дна сети, на входном слое.

11.6.3. Векторное представление слов и графов

Есть одно конкретное применение технологии глубокого обучения без учителя, которое я нашел легко применимым к нескольким интересным проблемам. Оно имеет дополнительное преимущество, заключающееся в доступности для широкой аудитории, не знакомой с нейронными сетями. *Векторное представление слов* (word embedding) — это распространенное представление того, что слова на самом деле *означают* или *делают*.

Каждое слово обозначается одной точкой, скажем, в 100-мерном пространстве, так что слова, которые играют сходные роли, как правило, представлены соседними точками. На рис. 11.15 представлены пять ближайших соседей нескольких характерных английских слов в соответствии с векторным представлением слов *GloVe* [90], и я верю, что вы согласитесь, что они связывают поразительное количество значений каждого слова по ассоциации.

Источник	1	2	3	4	5
Apple	iPhone	iPad	apple	MacBook	iPod
apple	apples	blackberry	Apple	Iphone	fruit
car	cars	vehicle	automobile	Truck	Car
chess	Chess	backgammon	mahjong	Checkers	tournaments
dentist	dentists	dental	orthodontist	Dentistry	Dentist
dog	dogs	puppy	pet	Cat	puppies
Mexico	Puerto	Peru	Guatemala	Colombia	Argentina
red	blue	yellow	purple	Orange	pink
running	run	ran	runs	Runing	start
write	writing	read	written	Tell	Write

Рис. 11.15. Ближайшие соседи по векторному представлению слов замечают термины с похожими ролями и значением

Основная ценность встраивания слов заключается в общих чертах, применяемых в конкретных приложениях машинного обучения. Давайте пересмотрим проблему отличия спама от значимых почтовых сообщений. В традиционном представлении пакета слов каждое сообщение может быть представлено в виде разреженного вектора b, где $b[i]$ может сообщать о количестве вхождений словарного слова w_i в сообщении. Разумный размер словарного запаса v для английского языка составляет 100000 слов, превращая b в ужасное 100 000-мерное представление, которое не отражает сходства между родственными терминами. Представления векторов слов оказываются гораздо менее хрупкими из-за меньшей размерности.

Мы уже видели, что такие алгоритмы, как *разложение по сингулярным значениям* (Singular Value Decomposition — SVD) или *анализ основных компонентов* (principle components analysis), могут быть использованы для сжатия матрицы признаков M размером $n \times m$ в матрицу M' размером $n \times k$ (где $k \ll m$) таким образом, что M' сохраняет бо́льшую часть информации о M. Аналогично мы можем думать о встраивании слов как о сжатии $v \times t$ матрицы инцидентности текста M, где t — это количество документов в корпусе, а $M[i, j]$ измеряет релевантность слова я к документу j. Сжатие этой матрицы до $v \times k$ приведет к форме векторного представления слов.

Тем не менее нейронные сети являются наиболее популярным подходом к встраиванию слов. Представьте себе сеть, в которой входной слой принимает текущее векторное представление (скажем) из пяти слов, w_1, ..., w_5, соответствующих конкретной фразе из пяти слов нашего учебного корпуса по документам. Задача сети может заключаться в том, чтобы предсказать внедрение среднего слова w_3 из векторных представлений фланкирующих четырех слов. Благодаря обратному распространению мы можем отрегулировать вес узлов в сети так, чтобы повысить точность в этом конкретном примере. Ключевым

моментом здесь является то, что мы продолжаем обратное распространение до самого низкого уровня, чтобы изменить действительные входные параметры! Эти параметры представляли векторные представления для слов в данной фразе, поэтому этот шаг улучшает векторное представление для задачи прогнозирования. Повторение этого на большом количестве обучающих примеров дает значимое векторное представление для всего словаря.

Основной причиной популярности встраивания слов является *word2vec*, потрясающая реализация этого алгоритма, которая может быстро обучать встраиванию сотен тысяч слов словарного запаса в гигабайтах текста совершенно без контроля. Самый важный параметр, который вы должны установить, — это желаемое количество измерений d. Если d слишком мало, векторное представление не может полностью уловить значение данного символа. Если d слишком велико, представление становится излишне громоздким. По правде говоря, золотая середина лежит где-то между 50 и 300 измерениями.

Векторное представление графов

Предположим, нам дана матрица попарного сходства S размером $n \times n$, определенная в пространстве из n элементов. Мы можем построить матрицу смежности графа подобия G, объявляя ребро (x, y) всякий раз, когда сходство x и y в S достаточно велико. Эта большая матрица G может быть сжата с использованием разложения по сингулярным значениям (SVD) или анализа основных компонентов (PCA), но в больших сетях это оказывается слишком дорогостоящим.

Такие программы, как *word2vec*, отлично справляются с построением представлений из последовательностей символов в учебном корпусе. Ключом к их применению в новых доменах является сопоставление вашего конкретного набора данных со строками из интересующего словаря. *DeepWalk* — это подход к построению векторных представлений графов, точечных представлений для каждой вершины таким образом, чтобы "похожие" вершины располагались в пространстве близко друг к другу.

Наш словарь может быть выбран как набор различных идентификаторов вершин от 1 до n. Но что за текст может представлять граф в виде последовательности символов? Мы можем построить случайные блуждания по сети, начиная с произвольной вершины и многократно переходя к случайному соседу. Эти проходы можно рассматривать как "предложения" для нашего словаря вершин-слов. Получившиеся векторные представления после запуска *word2vec* на этих случайных блужданиях доказывают свою эффективность в приложениях.

DeepWalk — это превосходная иллюстрация того, как векторные представления слов могут использоваться для извлечения значения из любого крупномасштабного корпуса последовательностей, независимо от того, взяты ли они из естественного языка. Эта же идея играет важную роль в следующем разделе "Случай из жизни".

11.7. Случай из жизни: игра имен

Мой брат, когда ему нужен псевдоним для бронирования ресторана или заполнения онлайн-формы, использует имя *Тора Рабиновича* (Thor Rabinowitz). Чтобы понять смысл этого случая из жизни, следует сначала понять, почему это очень смешно.

- *Тор* — это имя древнескандинавского бога и более позднего персонажа-супергероя. В мире немного, но и немало людей по имени Тор, большинство из которых, по-видимому, норвежцы.
- *Рабинович* — это польско-еврейская фамилия, что означающая "сын раввина". В мире есть небольшое, но и не такое уж и малое количество людей по фамилии Рабинович, но, по сути, ни один из них не норвежец.

В результате никогда не было человека с таким именем, и этот факт вы можете легко проверить, прибегнув к поиску имени "Thor Rabinowitz" в Google. Упоминание этого имени должно вызвать когнитивный диссонанс у любого слушателя, потому что эти два названия культурно несовместимы.

Призрак Тора Рабиновича нависает над этой историей. Мой коллега Ю Фан Ху пытался найти способ доказать, что пользователь, входящий с подозрительной машины, действительно был тем, кем, по его словам, он был. Если следующая попытка входа в мою учетную запись неожиданно произойдет из Нигерии после многих лет в Нью-Йорке, действительно ли это я или плохой парень, пытающийся украсть мою учетную запись?

"Плохой парень не узнает, кто ваши друзья, — заметил Ю Фан. — Что, если мы предложим вам распознать имена двух настоящих друзей из контактов электронной почты в списке поддельных имен. Только настоящий владелец будет знать, кто они".

"Как вы собираетесь получить поддельные имена? — спросил я. — Может быть, использовав имена других людей, которые не являются контактами владельца?"

"Ни за что, — сказал Ю Фан. — Клиенты расстроятся, если мы покажем их имена плохому парню. Но мы можем просто придумывать имена, выбирая имена и фамилии и склеивая их вместе".

"Но Тор Рабинович никого не обманет", — возразил я, объясняя необходимость культурной совместимости.

Нам нужен был способ представлять имена так, чтобы охватить тонкие культурные связи. Он предположил, что нечто вроде встраивания слов может справиться с этой работой, но нам нужен обучающий текст, который бы кодировал эту информацию.

Ю Фан оказался на высоте. Он получил набор данных, состоящий из имен наиболее важных контактов электронной почты для более чем 2 миллионов человек. Списки контактов для репрезентативных лиц[5] может быть таким.

- *Брэд Питт*. Анджелина Джоли, Дженнифер Энистон, Джордж Клуни, Кейт Бланшетт, Джулия Робертс.
- *Дональд Трамп*. Майк Пенс, Иванка Трамп, Пол Райан, Владимир Путин, Митч Макконнелл.
- *Си Цзиньпин*. Ху Цзиньтао, Цзян Цзэминь, Пэн Лиюань, Си Минцзе, Кэ Линлин.

Мы могли бы рассматривать каждый список контактов электронной почты как строку имен, а затем объединять эти строки так, чтобы они были предложениями в документе с 2 миллионами строк. Подача этого в *word2vec* будет обучать векторные представления для каждого токена имени/фамилии, появляющегося в корпусе. Поскольку некоторые токены имен, такие как *John*, могут отображаться как имена или фамилии, мы создали отдельные символы, чтобы отличать случаи *John/1* от *John/2*.

Word2vec выполнил короткую работу по этой задаче, создав стомерный вектор для каждого токена имени с чудесными свойствами локальности. Имена, связанные с одним и тем же полом, сгруппированы рядом друг с другом. Зачем? Мужчины обычно имеют больше друзей-мужчин в своем списке контактов, чем женщины, и наоборот. Эти совместные места сблизили полы. Внутри каждого пола мы видим группы имен по этническим группам: китайские имена рядом с китайскими именами и турецкие имена рядом с другими турецкими именами. Принцип гомофильности "рыбак рыбака видит издалека" (birds of a feather flock together) здесь также выполняется.

Имена регулярно входят и выходят из моды. Мы даже видим кластеризацию популярности имен по возрасту. У всех друзей моей дочери, похоже, есть имена, такие как *Брианна*, *Британи*, *Джессика* и *Саманта*. Конечно же, эти встраивания имен тесно связаны между собой в пространстве, потому что они сделаны в одно и то же вовремя: эти дети, как правило, чаще общаются со сверстниками того же возраста.

Мы видим аналогичные явления с токенами. На рис. 11.16 представлена карта из 5000 наиболее часто встречающихся фамилий, нарисованная в результате проецирования наших векторных представлений в одномерные имена до

[5] В рамках этого проекта были предприняты большие усилия для сохранения конфиденциальности пользователей. Имена владельцев учетной записи электронной почты никогда не включались в данные, и все необычные имена были отфильтрованы. Чтобы предотвратить возможную неправильную интерпретацию, показанные здесь примеры *не являются* списками контактов для Брэда Питта, Дональда Трампа и Си Цзиньпина.

двух измерений. Имена были помечены цветом в соответствии с их доминирующей расовой классификацией по данным переписи населения США. Вырезы на рис. 11.16 подчеркивают однородность регионов по культурным группам. В целом векторное представление четко помещает имена белых, черных, латиноамериканцев и азиатов в большие смежные регионы. На карте есть два разных азиатских региона. На рис. 11.17 представлены вставки для этих двух регионов, показывающие, что одна группа состоит из китайских имен, а другая — из индийских.

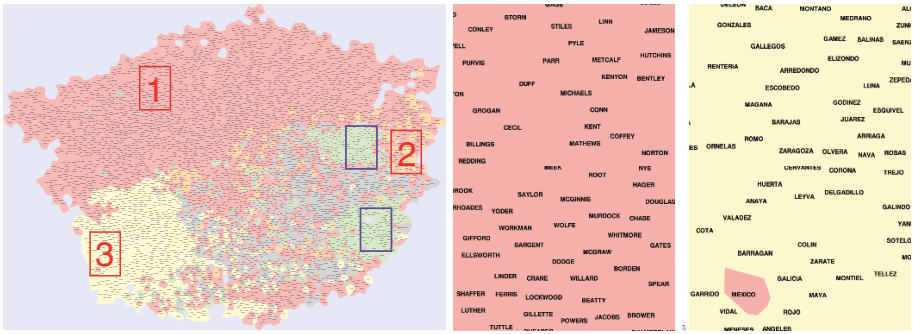


Рис. 11.16. Визуализация встраивания имени для 5000 наиболее популярных фамилий из контактных данных электронной почты, демонстрирующая двумерный проекционный вид векторного представления (слева). Врезки слева направо выделяют британские (в центре) и испанские (справа) имена

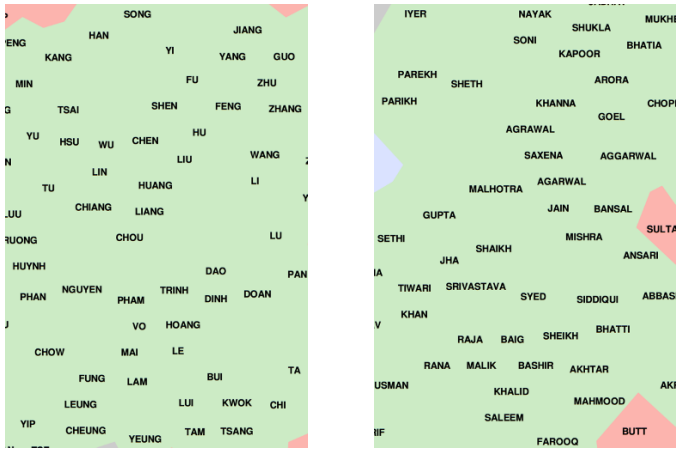


Рис. 11.17. Два отдельных азиатских кластера в пространстве имен отражают разные культурные группы. Слева вставка с китайскими/южноазиатскими именами. Справа вставка из группы индийских фамилий

При очень небольшим количестве *Торов*, соответствующих очень немногим *Рабиновичам*, этим токенам имен суждено лежать далеко друг от друга в пространстве векторного представления. Но токены имен, популярные в данной демографической группе, вероятно, лежат рядом с фамилиями из той же демографической группы, поскольку в отдельных списках контактов присутствуют одинаковые тесные связи. Таким образом, ближайший токен *y* фамилии к определенному токену *x* имени, вероятно, будет культурно совместимым, что делает *xy* хорошим кандидатом на разумно звучащее имя.

Мораль этой истории — сила векторного представления слов для легкого понимания структуры, скрытой в любой длинной последовательности символов, где порядок имеет значение. С такими программами, как word2vec, играть очень весело и удивительно просто. Поэкспериментируйте с любым интересным набором данных, который у вас есть, и вы будете удивлены свойствами, которые он раскрывает.

11.8. Дополнительная информация

К хорошим введениям в машинное обучение относятся книги Бишопа [91], а также Фридмана и др. [92]. Глубокое обучение в настоящее время является наиболее захватывающей областью машинного обучения, а книга Гудфеллоу, Бенджио и Курвилля [93] служит наиболее всеобъемлющим пособием.

С векторным представлением слов знакомит книга Миколов и др. [94], а также их мощная реализация word2vec. Голдберг и Леви [95] показали, что word2vec неявно учитывает точечную взаимную информационную матрицу словосочетаний. На самом деле модель нейронной сети не является фундаментальной для того, что она делает. Наш подход *DeepWalk* к встраиванию графов описан в книге Перозци и др. [96].

Примеры выживания на "Титанике" получены из конкурса Kaggle `https://www.kaggle.com/c/titanic`. Случай из жизни по созданию поддельных имен — результат работы с Шу-чу Ханя, Ю Фан Ху, Бориса Кошкина и Мэйчжу Лю в лабораториях Yahoo.

11.9. Упражнения

Классификация

11.1. *[3]* Используя наивный байесовский классификатор на рис. 11.2, решите, являются ли дни (Облачно, Высокая, Нормальная) и (Солнечно, Низкая, Высокая) пляжными.

11.2. *[8]* Примените метод наивного байесовского классификатора для классификации мультиклассовых текстов. В частности, используйте *The New York Times Developer API* для получения последних статей из нескольких разделов газеты. Затем, используя простую модель Бернулли для присутствия слова, внедрите классификатор, который, учитывая текст статьи из *The New York Times*, предсказывает, к какому разделу принадлежит статья.

11.3. *[3]* Что такое регуляризация и какие проблемы с машинным обучением она решает?

Деревья решений

11.4. *[3]* Примените деревья решений для представления следующих булевых функций:
(a) A И $\overline{B}$;
(b) A ИЛИ (B И C);
(c) (A И B) ИЛИ (C И D).

11.5. *[3]* Предположим, дана матрица $n \times d$ классифицированных помеченных данных, где каждый элемент имеет связанный класс меток A или B. Приведите доказательство или контрпример по каждому из приведенных ниже утверждений:
(a) Всегда ли существует классификатор дерева решений, который отлично отделяет A от B?
(b) Всегда ли существует классификатор дерева решений, который отлично отделяет A от B, если все n векторов признаков различны?
(c) Всегда ли существует классификатор логистической регрессии, который отлично отделяет A от B?
(d) Всегда ли существует классификатор логистической регрессии, который отлично отделяет A от B, если все n векторов признаков различны?

11.6. *[3]* Рассмотрим набор из n помеченных точек в двух измерениях. Можно ли построить классификатор дерева решений конечного размера с помощью тестов вида "*справедливо ли, что* $x > c$*?*", "*справедливо ли, что* $x < c$*?*", "*справедливо ли, что* $y > c$*?*" и "*справедливо ли, что* $y < c$*?*", которые классифицируют каждый возможный запрос точно так же, как классификатор ближайшего соседа?

Метод опорных векторов

11.7. *[3]* Предоставьте алгоритм линейного времени для нахождения разделительной линии максимальной ширины в одном измерении.

11.8. *[8]* Предоставьте алгоритм $O(n^{k+1})$ для нахождения разделительной линии максимальной ширины в k измерениях.

11.9. *[3]* Предположим, что мы используем метод опорных векторов для нахождения идеальной разделительной линии между заданным набором из n красных и синих точек. Теперь предположим, что мы удаляем все точки, которые не являются опорными векторами, и используем SVM, чтобы найти лучший разделитель того, что осталось. Может ли эта разделительная линия отличаться от предыдущей?

Нейронные сети

11.10. *[5]* Укажите структуру сети и функции активизации узла, чтобы позволить модели нейронной сети реализовать линейную регрессию.

11.11. *[5]* Укажите структуру сети и функции активизации узла, чтобы позволить модели нейронной сети реализовать логистическую регрессию.

Реализация проектов

11.12. *[5]* Найдите набор данных, включающий интересующую последовательность символов: возможно, текст, цветовые последовательности на изображениях или журналы событий с какого-либо устройства. Используйте word2vec, чтобы построить из них векторные представления символов, и проведите анализ ближайших соседей. Какие интересные структуры выявят векторные представления?

11.13. *[5]* Поэкспериментируйте с различными методами дисконтирования, оценивая частоту слов в английском языке. В частности, оцените степень, с которой частоты коротких текстовых файлов (1000 слов, 10 000 слов, 100 000 слов и 1 000 000 слов) отражают частоты в большом текстовом корпусе, скажем, 10 000 000 слов.

Вопросы на интервью

11.14. *[5]* Что такое глубокое обучение? Каковы некоторые из характеристик, которые отличают его от традиционного машинного обучения?

11.15. *[5]* Когда бы вы использовали случайные леса вместо SVM и почему?

11.16. *[5]* Как вы думаете, пятьдесят небольших деревьев решений лучше, чем одно большое? Почему?

11.17. *[8]* Как бы вы разработали программу для выявления плагиата в документах?

Конкурсы Kaggle

11.18. Насколько актуальны результаты поиска для пользователя?

```
https://www.kaggle.com/c/crowdflower-search-relevance
```

11.19. Понравился фильм рецензенту или нет?

```
https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews
```

11.20. По данным датчика определите, какой бытовой прибор используется в настоящее время.

```
https://www.kaggle.com/c/belkin-energy-disaggregation-
competition
```

Глава 12

Большие данные: достижение крупного масштаба

> Количественные изменения ведут к качественным.
>
> — Фридрих Энгельс (Friedrich Engels)

Однажды я давал интервью на телевидении и меня спросили, в чем разница между *данными* (data) и *большими данными* (big data). Подумав немного, я дал ответ, которого и придерживаюсь до сих пор: "в размере".

Бапкис (Bupkis, или крохи) — это чудесное слово на идише, означающее "слишком мало, чтобы иметь значение". Оно используется в предложении типа "Он получил за это крохи", т.е. жалоба на ничтожную сумму денег. Возможно, самой близкой аналогией в английском языке будет слово "арахис" (peanuts) ("Работать за арахисовые орехи").

По правде говоря, объемы данных, которые мы рассмотрели в этой книге до сих пор, составляют крохи. Учебные наборы с аннотациями, написанными людьми, приводятся в сотнях и тысячах примеров, но все, что вы должны заплатить людям за их создание, едва дотягивает до миллионов. Журнал всех поездок такси в Нью-Йорке за несколько лет, о котором говорилось в разделе 1.6, составил 80 миллионов записей. Неплохо, но все-таки крохи: вы легко можете сохранить это на своем ноутбуке и просканировать файл, чтобы собрать статистику, за считанные минуты.

Громкие слова *большие данные*, возможно, немного устарели, но они предполагают анализ действительно массивных наборов данных. Смысл слова *большие* со временем увеличивается, но сейчас я рисую начальную линию на уровне порядка 1 терабайта.

Это не так впечатляет, как может показаться. В конце концов, на момент написания книги терабайтный диск обойдется вам менее 100 долл., что является *крохами*. Но приобретение значимого набора данных для его заполнения потребует некоторой инициативы, возможно, привилегированного доступа к

данным крупной интернет-компании или большим объемам видео. Есть много организаций, оперирующих петабайтами и даже эксабайтами данных на регулярной основе.

Большие данные требуют более крупномасштабной инфраструктуры, чем проекты, которые мы рассматривали до настоящего времени. Перемещение огромных объемов данных между машинами требует быстрых сетей и терпения. Нам нужно отказаться от последовательной обработки, даже выйти за пределы нескольких ядер, и перейти к большому количеству машин, плавающих в облаках. Эти вычисления масштабируются до такой степени, что мы должны учитывать надежность, поскольку почти наверняка произойдет сбой какого-либо аппаратного компонента, прежде чем мы получим наш ответ.

По мере увеличения размера, работа с данными, как правило, становится сложнее. В этом разделе я постараюсь проинформировать вас об общих проблемах, связанных с массивными наборами данных. Важно понимать, почему размер имеет значение, ведь вы сможете участвовать в проектах, которые работают в таком масштабе.

12.1. Что такое большие данные?

Насколько велики большие данные? Любое число, которое я вам дам, успеет устареть к тому времени, когда я его наберу, но вот некоторые статистические данные за 2016 год, которые я нашел, в основном по адресу `http://www.internetlivestats.com/`.

- Twitter: 600 миллионов твитов в день.
- Facebook: 600 терабайт поступающих данных каждый день от 1,6 миллиарда активных пользователей.
- Google: 3,5 миллиарда поисковых запросов в день.
- Instagram: 52 миллиона новых фотографий в день.
- Apple: всего 130 миллиардов загрузок приложений.
- Netflix: ежедневно транслируется 125 миллионов часов телевизионных шоу и фильмов.
- Электронная почта: 205 миллиардов сообщений в день.

Размер имеет значение: мы можем делать удивительные вещи с этим материалом. Но и другие вещи также имеют значение. В этом разделе будут рассмотрены некоторые технические и концептуальные сложности работы с большими данными.

На заметку. Большие данные обычно состоят из огромного количества строк (записей) в относительно небольшом количестве столбцов (объектов). Таким образом, большие данные зачастую избыточны, чтобы точно соответствовать одной модели для данной проблемы. Ценность обычно заключается в подборе *множества* различных моделей, например при обучении отдельной модели, индивидуализированной для каждого отдельного пользователя.

12.1.1. Большие данные — плохие данные

Массивные наборы данных обычно являются результатом возможности, а не замысла. В традиционной науке, основанной на гипотезах, мы разрабатываем эксперимент, чтобы собрать именно те данные, которые нам нужны для ответа на наш конкретный вопрос. Но большие данные чаще всего являются продуктом какого-то процесса регистрации, записывающего отдельные события, или, возможно, распределенного вклада миллионов людей через социальные сети. Ученый, занимающийся данными, как правило, мало или совсем не контролирует процесс сбора, а лишь использует расплывчатые подходы, чтобы превратить все эти кусочки в деньги.

Рассмотрим задачу измерения общественного мнения по публикациям в социальной сети или на сайте онлайн-опроса. Большие данные могут быть прекрасным ресурсом. Однако они подвержены ошибкам и ограничениям, из-за которых сложно сделать точные выводы.

- *Необъективное представительство.* Любому внешнему источнику данных присуща необъективность выборки. Данные с некоего конкретного сайта социальной сети не отражают людей, которые ими не пользуются, поэтому с обобщениями следует быть осторожным.

  Пользователи Amazon покупают гораздо больше книг, чем покупатели на Walmart. Их политическая принадлежность и экономический статус также различаются. Вы получаете одинаково предвзятые, но очень разные взгляды на мир, если анализируете данные из Instagram (слишком молодые), *The New York Times* (слишком либеральные), *Fox News* (слишком консервативные) или *The Wall Street Journal* (слишком богатые).

- *Спам и содержимое, созданное машиной.* Большие источники данных еще хуже, чем необъективные. Нередко они были спроектированы так, чтобы намеренно вводить в заблуждение.

  Любая сетевая платформа, достаточно большая, чтобы создавать огромные объемы данных, достаточно велика, чтобы существовали экономические стимулы для их искажения. Армии платных рецензентов каждый

день пишут фальшивые и вводящие в заблуждение обзоры продуктов. Боты выпускают механически написанные твиты и еще более длинные тексты в огромном объеме, и даже являются их основными источниками: значительная часть сообщений на любом веб-сайте производится механическими ботами, а не людьми. Почти 90% всей электронной почты, отправляемой по сетям, является спамом: эффективность фильтров спама на нескольких этапах передачи является единственной причиной, по которой вы его не видите.

Фильтрация спама является неотъемлемой частью процесса очистки данных при любом анализе социальных сетей. Если вы не удалите спам, он будет намеренно обманывать вас, а не просто вводить в заблуждение.

- *Избыточность*. Многие виды человеческой деятельности подчиняются распределению по степенному закону, а значит, очень маленький процент предметов составляет большой процент от общей активности. Новости и социальные сети в значительной степени концентрируются на последних оплошностях Кардашьян и подобных знаменитостей, посвящая им тысячи статей. Многие из них будут почти точными копиями других статей. Насколько больше полный набор из них говорит вам о чем-либо в мире?

  Этот закон неравномерного охвата подразумевает, что бо́льшая часть данных, которые мы видим в окружающих источниках, — это то, что мы уже видели раньше. Устранение такого дублирования является важным этапом очистки для многих приложений. Любой сайт обмена фотографиями будет содержать тысячи изображений Эмпайр-стейт-билдинг, но ни одного из зданий, в которых я работаю. Обучение классификатора с такими изображениями создаст невероятные функции для ориентиров, которые могут или не могут быть полезны для более общих задач.

- *Восприимчивость к временному смещению*. Продукты меняются в ответ на конкуренцию и изменения потребительского спроса. Зачастую данные улучшения меняют способ использования этих продуктов. Временной ряд, полученный в результате сбора данных, может хорошо кодировать несколько изменений продукта/интерфейса, что затрудняет различие между артефактом и сигналом.

  Общеизвестный пример вращается вокруг Google Flu Trends, который на протяжении нескольких лет успешно прогнозирует вспышки заболеваний на основе запросов поисковых систем. Но потом модель начала работать плохо. Одним из факторов было то, что Google добавил механизм автозаполнения в поле, где он предлагает ввести поисковый запрос. Это изменило распределение поисковых запросов в достаточной степени,

чтобы данные временного ряда до изменения были несопоставимы с новыми данными.

Некоторые из этих эффектов могут быть смягчены за счет тщательной нормализации, но зачастую они так плотно заполнены данными, что не допускают значимого динамического анализа.

На заметку. Большие данные — это данные, которые у нас есть. Хорошие данные — это данные, соответствующие поставленной задаче. Большие данные — это плохие данные, если они не могут действительно ответить на интересующие нас вопросы.

12.1.2. Три V

Типы управленческого консалтинга (management consulting types) привязаны к понятию *трех V больших данных* (three Vs of big data) в качестве средства объяснения таких их свойств, как *объем* (volume), *разнообразие* (variety) и *скорость* (velocity). Они дают основание говорить о том, что отличает большие данные. Эти V таковы.

- *Объем.* Само собой разумеется, что большие данные больше, чем маленькие. Различие является одним из классов. Мы покидаем мир, где можем представить наши данные в электронной таблице или обработать их на одном компьютере. Это потребует разработки более сложной вычислительной инфраструктуры и ограничения эффективности нашего анализа алгоритмами с линейным временем.
- *Разнообразие.* Сбор плотных данных обычно выходит за рамки матрицы. Накопление разнородных данных зачастую требует специальных методов интеграции.

  Рассмотрим социальные сети. Сообщения могут включать текст, ссылки, фотографии и видео. В зависимости от нашей задачи, все это может быть актуально, но обработка текста требует совершенно других методов, чем сетевые данные и мультимедиа. Даже изображения и видео — это совершенно разные звери, которых нельзя обрабатывать на одном и том же конвейере. Осмысленная интеграция этих материалов в единый набор данных для анализа требует значительных усилий.
- *Скорость.* Сбор данных из внешних источников подразумевает, что система *живая* (live), т.е. она всегда включена, всегда собирает данные. Наборы данных, которые мы изучали до настоящего времени, напротив, как правило, были *мертвыми* (dead), т.е. собирались один раз и помещались в файл для последующего анализа.

Живые данные означают, что должны быть созданы инфраструктуры для сбора, индексации, доступа и визуализации результатов, как правило, через систему панели мониторинга. Живые данные означают, что потребители хотят в режиме реального времени получать доступ к последним результатам через графики, диаграммы и API.

В зависимости от отрасли доступ в режиме реального времени может включать обновление состояния базы данных в течение нескольких секунд или даже миллисекунд после фактических событий. В частности, финансовые системы, связанные с высокочастотной торговлей (high-frequency trading), требуют немедленного доступа к самой последней информации. Вы соревнуетесь с другим парнем и получаете прибыль, только если выиграете.

Скорость передачи данных, возможно, является тем местом, где наука о данных наиболее существенно отличается от классической статистики. Это то, что стимулирует спрос на передовые системные архитектуры, которые требуют инженеров, способных обеспечить масштабирование с использованием новейших технологий.

Управленческий набор иногда определяет четвертую V: *достоверность* (veracity), меру того, насколько мы доверяем базовым данным. Здесь мы сталкиваемся с проблемой устранения спама и других артефактов, возникающих в результате сбора данных, за пределами уровня обычной очистки.

12.2. Случай из жизни: вопросы инфраструктуры

Я должен был понять глубину страданий Михаила, как только он нахмурил брови.

Мой аспирант Михаил Баутин, пожалуй, лучший программист, которого я когда-либо видел. Вы, возможно, слышали о нем. Он занял *1-е место* на 12-й Международной олимпиаде по информатике с отличным счетом, позволившим отметить его как лучшего программиста среди учеников средних школ в мире в том году.

На этом этапе наш проект анализа новостей Lydia уже имел существенную инфраструктуру, работающую на нескольких машинах. Текст, извлеченный из источников новостей по всему миру, был нормализован и прошел через конвейер обработки естественного языка (Natural Language Processing — NLP), который мы написали для английского языка, а извлеченные сути и их настроения были идентифицированы по тексту и сохранены в большой базе данных.

С помощью ряда команд SQL эти данные могут быть извлечены в том формате, в котором вы можете отобразить их на веб-странице или использовать в электронной таблице.

Я хотел, чтобы мы изучили степень, в которой машинный перевод сохранил обнаруживаемые настроения. Если бы это получилось, то обеспечило бы простой способ обобщить наш анализ настроений на языках помимо английского. Было бы довольно затруднительно задействовать стороннего переводчика в наш конвейер и посмотреть, что получится.

Я думал, что это было своевременное и важное исследование, и действительно, наша последующая статья [97] цитировалась 155 раз на момент написания этой книги. Поэтому я передал проект моему лучшему аспиранту, и даже предложил ему услуги очень способного студента-старшекурсника, чтобы помочь с некоторыми техническими вопросами. Задание он принял тихо и послушно. Но он нахмурил брови.

Три недели спустя он вошел в мой кабинет. Инфраструктура, разработанная моей лабораторией для поддержки анализа новостей в нашей базе данных, была жесткой и устаревшей. Она не допускала масштабирования. Это оскорбляло его чувство бытия. Если я не позволю ему переписать все с нуля с использованием современных технологий, он сразу же покинет аспирантуру. В течение этих трех недель он использовал свое свободное время, чтобы получить очень выгодное предложение о работе от хедж-фонда мирового уровня, и зашел попрощаться.

Я могу быть очень разумным человеком, если все изложено достаточно ясно. Да, его диссертация может быть и о такой инфраструктуре. Он сразу же приступил к работе.

Первое, что нужно было сделать, — это центральная база данных MYSQL, где хранились бы все наши новости и настроения. Это было узким местом. Она не может быть распределена по кластеру машин. Он собрался хранить все в распределенной файловой системе (HDFS), чтобы не было единого узкого места: чтение и запись могли происходить по всему нашему кластеру.

Второе, что нужно было сделать, — это импровизированный подход к координации машин в нашем кластере для выполнения им различных задач. Он был ненадежен. Не было механизма восстановления после ошибок. Он собирался переписать всю нашу внутреннюю обработку как задания MapReduce с использованием Hadoop.

Третье, что нужно было сделать, — это специальный формат файла, который мы использовали для представления новостных статей и их аннотаций. Он имел ошибки. Исключения были везде. Наши анализаторы часто ломались по глупым причинам. Вот почему был изобретен язык XML, чтобы обеспечить способ точного выражения структурированных данных и готовые

эффективные инструменты для их анализа. Любой текст, который проходил через его код, сначала должен был пройти агент проверки XML. Он отказался прикасаться к болезненным сценариям Perl, которые выполняли наш анализ NLP, но полностью изолировал этот код, чтобы сдерживать инфекцию.

С таким количеством подвижных частей даже Михаил потратил некоторое время на то, чтобы наладить инфраструктуру. Замена нашей инфраструктуры означала, что мы не могли продвигаться ни по какому другому проекту, пока не был завершен этот. Всякий раз, когда я беспокоился, что мы не сможем провести экспериментальный анализ, пока он не будет готов, он тихо напоминал мне о постоянном предложении, которое он получил от хедж-фонда, и продолжал с тем, что он делал.

И, конечно, Михаил был прав. Масштаб того, что мы могли сделать в лаборатории, увеличился в десять раз с новой инфраструктурой. Было гораздо меньше времени простоя, и борьба за восстановление базы данных после сбоя питания стала делом прошлого. Разработанные им интерфейсы API для удобного и логичного регулирования всех наших приложений легли в основу наших приложений анализа. Его инфраструктура точно выдержала портирование в среду Amazon Cloud, которая работала каждую ночь, чтобы не отставать от мировых новостей.

Урок, который нужно извлечь, заключается в том, что инфраструктура имеет значение. Большая часть этой книги рассказывает о концепциях более высокого уровня: статистике, машинном обучении и визуализации, поэтому легко понять, что такое наука, а что такое сантехника. Но цивилизация не работает без эффективного водоснабжения. Чистые, эффективные, масштабируемые и обслуживаемые инфраструктуры, построенные с использованием современных программных технологий, имеют важное значение для эффективной науки о данных. Операции, которые сокращают технические затраты, такие как рефакторинг и обновление библиотек/инструментов до поддерживаемых в настоящее время версий, не являются бесполезными или несрочными, а являются ключом к упрощению выполнения того, что вы действительно хотите сделать.

12.3. Алгоритмы для больших данных

Для работы большие данные требуют эффективных алгоритмов. В этом разделе мы кратко рассмотрим основные алгоритмические проблемы, связанные с большими данными: асимптотическая сложность, хеширование и потоковые модели для оптимизации производительности ввода-вывода в больших файлах данных.

У меня здесь нет ни времени, ни места, чтобы дать исчерпывающее введение в разработку и анализ комбинаторных алгоритмов. Тем не менее я могу с уверенностью рекомендовать руководство *The Algorithm Design Manual* [1] как отличную книгу по этим вопросам, если вы случайно ее ищете.

12.3.1. Анализ большого O

Традиционный алгоритм анализа основан на таком гипотетическом компьютере, как *машина с произвольным доступом* (Random Access Machine — RAM). В такой модели:

- каждая простая операция занимает ровно один шаг;
- каждая операция с памятью занимает ровно один шаг.

Следовательно, подсчет операций, выполненных на протяжении алгоритма, дает время его работы.

По правде говоря, количество операций, выполняемых любым алгоритмом, является функцией от размера ввода n: матрицы из n строк, текста с n словами, набора из n точек. Анализ алгоритма — это процесс оценки или определения количества шагов, которые алгоритм выполняет в зависимости от n.

Для алгоритмов, определенных циклом `for`, такой анализ довольно прост. Глубина вложения этих циклов определяет сложность алгоритма. Один цикл от 1 до n определяет алгоритм с *линейным временем* (linear-time) или $O(n)$, а два вложенных цикла определяют алгоритм с *квадратичным временем* (quadratic-time) или $O(n^2)$. Два последовательных цикла `for`, которые не являются вложенными, по-прежнему линейны, поскольку вместо $n \times n = n^2$ таких операций используются $n + n = 2n$ шагов.

Вот несколько примеров алгоритмов, включающих цикл `for`.

- *Поиск ближайшего соседа точки p.* Нам нужно сравнить p со всеми n точками в данном массиве a. Вычисление расстояния между точками p и $a[i]$ требует вычитания и возведения в квадрат d членов, где d — это размерность p. Перебор всех n точек и отслеживание ближайшей точки занимает $O(dn)$ времени. Поскольку d обычно достаточно мало, чтобы считаться константой, это считается алгоритмом с линейным временем.
- *Поиск ближайшей пары точек в наборе.* Нам нужно сравнить каждую точку $a[i]$ с любой другой точкой $a[j]$, где $1 \le i \ne j \le n$. По рассмотренным выше причинам это занимает $O(dn^2)$ времени и будет рассматриваться как алгоритм с квадратичным временем.
- *Матричное умножение.* Умножение матрицы $x \times y$ на матрицу $y \times z$ приводит к матрице $x \times z$, где каждый из членов $x \cdot z$ является скалярным произведением двух векторов длиной y:

```
C = numpy.zeros((x, z))

for i in range(0,x-1):
    for j in range(0, z-1):
        for k in range(0, y-1):
            C[i][j]  += A[i][k] * B[k][j]
```

Этот алгоритм выполняет $x \cdot y \cdot z$ шагов. Если $n = \max(x, y, z)$, то для этого требуется не более $O(n^3)$ шагов, и он будет рассматриваться как алгоритм с кубическим временем.

Для алгоритмов, которые определяются условными циклами `while` или рекурсией, анализ зачастую куда сложней. Вот несколько примеров с очень краткими пояснениями.

- *Сложение двух чисел.* Очень простые операции могут не иметь условий, как сложение двух чисел. Здесь нет реального значения n, только два, поэтому для этого требуется постоянное время, или $O(1)$.
- *Бинарный поиск.* Мы стремимся найти заданный ключ k в отсортированном массиве A, содержащем n элементов. Подумайте о поиске имени в телефонной книге. Мы сравниваем k со средним элементом $A[n/2]$ и решаем, находится искомое в верхней или нижней половине. Число делений пополам до 1 равно $\log_2 n$, как мы обсуждали в разделе 2.4. Таким образом, бинарный поиск выполняется за $O(\log n)$ шагов.
- *Сортировка слиянием.* Два отсортированных списка с общим количеством n элементов могут быть объединены в один отсортированный список за линейное время: извлечь меньший из двух ведущих элементов, как первый в отсортированной последовательности и повторять так далее. Сортировка слиянием разделяет n элементов на две половины, сортирует каждую, а затем объединяет их. Количество делений пополам до 1 снова равно $\log_2 n$ (см. раздел 2.4), а объединение всех элементов на всех уровнях дает алгоритм сортировки $O(n \log n)$.

Это был очень быстрый обзор алгоритмов, возможно, слишком быстрый для понимания, но он смог познакомить с шестью представителями алгоритмов различных классов сложности. Эти функции сложности определяют спектр от самого быстрого до самого медленного алгоритма, определяемого следующим порядком:

$$O(1) \ll O(\log n) \ll O(n) \ll O(n \log n) \ll O(n^2) \ll O(n^3).$$

На заметку. Алгоритмы, работающие с большими наборами данных, должны быть линейными или почти линейными, возможно, $O(n \log n)$. Квадратичные алгоритмы становятся невозможны для рассмотрения при $n > 10\,000$.

12.3.2. Хеширование

Хеширование (hashing) — это метод, который зачастую превращает квадратичные алгоритмы в алгоритмы с линейным временем, делая их пригодными для работы с таким масштабом данных, с которыми мы надеемся работать.

Ранее в разделе 10.2.4 мы обсудили хеш-функции в контексте *локально-чувствительного хеширования* (Locality-Sensitive Hashing — LSH). *Хеш-функция* (hash function) h получает объект x и преобразует его в определенное целое число $h(x)$. Основная идея заключается в том, что всякий раз, когда $x = y$, $h(x) = h(y)$. Таким образом, мы можем использовать $h(x)$ как целое число для индексации массива и собирать все похожие объекты в одном месте. Различные элементы *обычно* преобразуются в разных местах, предполагая наличие хорошо разработанной хеш-функции, но никаких гарантий нет.

Объекты, которые мы стремимся хешировать, зачастую представляют собой последовательности более простых элементов. Например, файлы или текстовые строки являются просто последовательностями элементарных символов. Эти элементарные компоненты обычно имеют естественное соотнесение с числом: например, коды таких символов, как Unicode, по определению соотносят символы с числами. Первый шаг к созданию хеша x заключается в его представлении как последовательности таких чисел без потери информации. Предположим, что каждое из $n = |S|$ чисел символов x являются целыми числами от 0 до $\alpha - 1$.

Превращение вектора чисел в одно репрезентативное число является задачей хеш-функции $h(x)$. Хороший способ сделать это — представить вектор как число по основанию α, так что

$$h(x) = \sum_{i=0}^{n-1} \alpha^{n-(i+1)} x_i \pmod{m}.$$

Функция mod (x mod m) возвращает остаток от x, деленный на m, а следовательно, возвращает число от 0 до $m - 1$. Это n-разрядное число по основанию α обречено быть огромным, поэтому взятие остатка позволяет нам получить представительный код скромного размера. Принцип здесь тот же, что и у колеса рулетки для азартных игр: длинный путь шарика вокруг колеса в конечном итоге заканчивается в одном из $m = 38$ слотов, что определяется оставшейся частью длины пути, деленной на длину окружности колеса.

Такие хеш-функции — удивительно полезные вещи. Их основные области применения таковы.

- *Обслуживание словаря*. Хеш-таблица — это структура данных на основе массива, использующая $h(x)$ для определения положения объекта x в сочетании с соответствующим методом разрешения конфликтов. При

правильном применении такие хеш-таблицы приводят на практике к постоянному (или $O(1)$) времени поиска.

Это намного лучше, чем бинарный поиск, а следовательно, хеш-таблицы широко используются на практике. Действительно, язык Python внутренне использует хеширование, чтобы связать имена переменных со значениями, которые они хранят. Хеширование является также фундаментальной идеей распределенных вычислительных систем, таких как MapReduce, которая будет обсуждаться в разделе 12.6.

- *Подсчет частот*. Обычной задачей при анализе журналов является подсчет частот заданных событий, таких как количество слов или посещений страниц. Самый быстрый и простой способ — создать хеш-таблицу с типами событий в качестве ключа и увеличивать счетчик для каждого нового события. При правильной реализации этот алгоритм является линейным по общему количеству анализируемых событий.
- *Удаление дубликатов*. Важной задачей очистки данных является выявление дублированных записей в потоке данных и их удаление. Возможно, это все адреса электронной почты наших клиентов, и мы хотим убедиться, что рассылаем спам только по разу каждому из них. С другой стороны, мы можем попытаться построить полный словарь некого языка из больших объемов текста.

  Основной алгоритм прост. Для каждого элемента в потоке проверить, не находится ли он уже в хеш-таблице. Если нет, вставить его, если находится, то игнорировать его. При правильной реализации этот алгоритм занимает линейное время в общем количестве анализируемых записей.
- *Канонизация*. Зачастую на один и тот же объект могут ссылаться несколько разных имен. Словарные слова, как правило, нечувствительны к регистру, а значит «The» эквивалентно «the». Определение словарного запаса языка требует объединения альтернативных форм и сопоставления их с одним ключом.

  Этот процесс построения *канонического представления* (canonical representation) можно интерпретировать как хеширование. По правде говоря, для этого требуется специфическая для данной области функция упрощения, выполняющая такие действия, как приведение к нижнему регистру, удаление пробелов, удаление стоп-слов и расширение аббревиатур. Эти канонические ключи затем можно хешировать, используя обычные хеш-функции.
- *Криптографическое хеширование*. Благодаря построению кратких и *необратимых* (uninvertible) представлений хеширование можно использовать для мониторинга и ограничения поведения человека. Как вы можете

доказать, что входной файл остается неизменным с момента его последнего анализа? Создайте для файла хеш-код или *контрольную сумму*, когда вы работали с ним, и сохраните этот код для сравнения с хешем файла впоследствии. Если файл не изменен, они будут одинаковыми, и почти наверняка будут отличаться, если произошли какие-либо изменения.

Предположим, что вы хотите зафиксировать ставку для определенного элемента, но не раскрываете фактическую цену, которую вы будете платить, пока не будут приняты все ставки. Хешируйте вашу ставку, используя заданную криптографическую хеш-функцию, и отправьте полученный хеш-код. По истечении указанного срока снова отправьте заявку, на этот раз без шифрования. Любой подозревающий может хешировать вашу теперь открытую ставку и подтвердить, что значение соответствует ранее введенному хеш-коду. Ключевым моментом является то, что создать коллизии с данной хеш-функцией будет трудно, а значит, вы не можете легко создать другое сообщение, которое даст тот же хеш-код. В противном случае вы сможете отправить второе сообщение вместо первого, изменив ставку после установленного срока.

12.3.3. Использование иерархии хранилищ

Алгоритмы больших данных зачастую привязаны к *хранилищу* (storage-bound) или *пропускной способности* (bandwidth-bound), а не к *скорости вычислений* (compute-bound). Это значит, что стоимость ожидания при получении данных там, где это необходимо, превышает стоимость алгоритмического манипулирования ими для получения желаемых результатов. Чтобы прочитать 1 терабайт данных даже с современного диска, все равно потребуется полчаса. Достижение хорошей производительности может опираться скорее на интеллектуальное управление данными, чем на сложные алгоритмы.

Чтобы стать доступными для анализа, данные должны храниться где-то в компьютерной системе. Существует несколько возможных типов устройств хранения, которые сильно различаются по скорости, емкости и задержке. Различия в производительности между разными уровнями *иерархии хранения* (storage hierarchy) настолько огромны, что мы не можем игнорировать это в нашей абстракции машины RAM. Действительно, отношение скоростей доступа к диску и к кеш-памяти составляет примерно 10^6, как скорость черепахи к первой космической скорости!!

Основные уровни иерархии хранения таковы.

- *Кеш-память*. Современные компьютерные архитектуры имеют сложную систему регистров и кешей для хранения рабочих копий активно используемых данных. Частично они используются для предварительной

выборки: захват больших блоков данных в областях памяти, к которым недавно обращались, в ожидании их необходимости в будущем. Размеры кеша обычно измеряются в мегабайтах, а доступ к нему в пять-сто раз быстрее, чем к основной памяти. Такая производительность делает его очень выгодным для вычислений с использованием *локальности* (locality), чтобы интенсивно использовать отдельные элементы данных в концентрированных пакетах, а не периодически в течение длительного вычисления.

- *Основная память*. Это то, что содержит общее состояние вычислений и где размещаются и поддерживаются большие структуры данных. Основная память обычно измеряется в гигабайтах и работает в сотни тысяч раз быстрее, чем дисковое хранилище. Нам нужны структуры данных, которые в максимально возможной степени вписываются в основную память и избегают постраничного сохранения в виртуальную память.
- *Основная память на другой машине*. Время задержки в локальной сети составляет минимум миллисекунды, что делает ее, как правило, быстрее, чем вторичные устройства хранения, такие как диски. Это означает, что распределенные структуры данных, такие как хеш-таблицы, *могут* содержательно поддерживаться в компьютерных сетях, но с доступом, который может быть в сотни раз медленнее, чем основная память.
- *Дисковое хранилище*. Объем вторичных устройств хранения может измеряться в терабайтах, обеспечивая емкость, которая позволяет большим данным становиться на самом деле большими. Физическим устройствам, таким как вращающиеся диски, требуется значительное время для перемещения считывающей головки в положение, в котором находятся данные. Оказавшись там, она сравнительно быстро читает большой блок данных. Это мотивирует предварительную выборку, копирующую в память большие куски файлов, предполагая, что они понадобятся позже.

Проблемы с задержкой, как правило, действуют как оптовая скидка: мы платим много за первый товар, к которому у нас есть доступ, но затем получаем кучу товара очень дешево. Нам нужно организовать наши вычисления так, чтобы воспользоваться этим, используя следующие методы.

- *Обрабатывать файлы и структуры данных в потоках*. Важно применять последовательный доступ к файлам и структурам данных, чтобы использовать предварительную выборку. Это означает, что массивы лучше, чем связанные структуры, поскольку логически соседние элементы располагаются на устройстве хранения рядом друг с другом. Это означает, что необходимо осуществлять полные передачи файлов данных, которые читают каждый элемент один раз, а затем выполняют все необходимые

вычисления с ним прежде, чем двигаться дальше. Большим преимуществом сортировки данных является то, что мы можем перейти к тому местоположению, о котором идет речь. Поймите, что такой произвольный доступ стоит дорого: лучше думать, а не искать.

- *Думать о больших файлах вместо каталогов.* Можно организовать совокупность документов таким образом, чтобы каждый находился в своем собственном файле. Это логично для людей, но медленно для машин, когда существуют миллионы крошечных файлов. Гораздо лучше организовать их в один большой файл для эффективного просмотра всех примеров, а не требовать отдельного доступа к диску для каждого из них.
- *Плотно упаковывать данные.* Стоимость распаковки данных, хранящихся в основной памяти, как правило, намного меньше дополнительных расходов на передачу больших файлов. Это аргумент в пользу того, чтобы представлять большие файлы данных плотно, когда это возможно. Это может означать явные схемы сжатия файлов до достаточно маленьких размеров, чтобы их можно было развернуть в памяти.

  Это означает разработку форматов файлов и структур данных для плотного кодирования. Подумайте о представлении последовательностей ДНК, которые представляют собой длинные строки в четырехбуквенном алфавите. Каждая буква/основание может быть представлена в 2 битах, т.е. четыре основания могут быть представлены одним 8-разрядным байтом, а тридцать два основания — в 64-разрядном слове. Такое уменьшение размера данных может значительно сократить время передачи и стоит вычислительных усилий для упаковки и распаковки.

  Ранее в разделе 3.1.2 мы уже говорили о важности читаемости в форматах файлов и придерживаемся этого мнения здесь. Незначительное уменьшение размера, скорее всего, не стоит потери читабельности или простоты распознавания. Но сокращение размера файла в два раза эквивалентно удвоению скорости передачи, что может иметь значение в среде больших данных.

12.3.4. Потоковые и однопроходные алгоритмы

Данные не обязательно хранятся вечно. Или даже не хранятся вообще. В приложениях с очень большим объемом обновлений и активности может потребоваться вычислять статистику по мере появления данных, чтобы мы могли затем отбросить оригинал.

В *потоковом* (streaming) или *однопроходном* (single-pass) алгоритме мы получаем только один шанс просмотреть каждый элемент ввода. Мы можем подразумевать некоторую память, недостаточную для хранения основной массы

отдельных записей. Нам нужно решить, что делать с каждым элементом, когда мы его встретим, а затем он исчезнет.

Предположим, например, что нам нужно вычислить среднее значение потока чисел при его прохождении. Это не сложная задача: мы можем хранить две переменные: s, представляющую текущую сумму, и n — количество элементов, которые мы получили на данный момент. Для каждого нового наблюдения a_i мы добавляем его к s и увеличиваем n. Всякий раз, когда кому-то нужно узнать текущее среднее значение $\mu = \bar{A}$ потока A, мы сообщаем значение s/n.

Как насчет вычисления дисперсии или стандартного отклонения потока? Это кажется сложнее. Напомним, что

$$V(A) = \sigma^2 = \frac{\sum_{i=1}^{n}(a_i - \bar{A})^2}{n-1}.$$

Проблема в том, что среднее значение последовательности $\bar{A}$ не может быть известно до тех пор, пока мы не достигнем конца потока, к тому времени мы потеряем исходные элементы, чтобы вычесть их из среднего значения.

Но еще не все потеряно. Напомним, что существует альтернативная формула для дисперсии, среднее квадратов минус квадрат среднего:

$$V(a) = \frac{1}{n}\sum_{i=1}^{n}(a_i)^2 - \bar{A}^2.$$

Таким образом, отслеживая текущую сумму квадратов элементов, в дополнение к n и s мы получаем весь материал, необходимый для вычисления отклонения по требованию.

В потоковой модели многие величины не могут быть вычислены точно. Примером может быть нахождение *медианного* элемента длинной последовательности. Предположим, у нас недостаточно памяти для хранения половины элементов полного потока. Первый элемент, который мы решили удалить, что бы это ни было, можно сделать срединным в еще не поступившем тщательно продуманном потоке элементов. Для решения определенных задач нам нужно иметь все данные одновременно.

Но даже если мы не можем что-то вычислить точно, мы зачастую можем предложить оценку, которая достаточно хороша для работы. Важные задачи этого типа включают в себя определение наиболее частых элементов в потоке, количество отдельных элементов или даже оценку частоты элементов, когда у нас недостаточно памяти для точного подсчета.

Скетчинг (sketching) предполагает использование имеющегося хранилища для необходимого отслеживания частичного представления последовательности. Возможно, это частотная гистограмма элементов, сгруппированных по значению, или небольшая хеш-таблица значений, которую мы видели ранее.

Качество нашей оценки увеличивается с количеством доступной памяти, в которой мы должны хранить наш скетч. *Случайная выборка* (random sampling) — чрезвычайно полезный инструмент для построения скетчей, и она находится в центре внимания раздела 12.4.

12.4. Фильтрация и выборка

Одним из важнейших преимуществ больших данных является то, что при достаточном объеме вы можете позволить себе отбросить бо́льшую часть своих данных. И это может быть весьма полезным, чтобы упростить ваш анализ.

Я различаю два разных способа отбрасывания данных: фильтрация и выборка. *Фильтрация* (filtering) означает выбор соответствующего подмножества данных на основе определенных критериев. Предположим, например, что мы хотели создать языковую модель для приложения в Соединенных Штатах и обучить ее на данных из Twitter. На английский язык приходится только около трети всех сообщений в Twitter, поэтому отфильтровывание всех остальных языков оставляет достаточно данных для содержательного анализа.

Мы можем рассматривать фильтрацию как особую форму очистки, когда мы удаляем данные не потому, что они ошибочны, а потому, что они отвлекают внимание от рассматриваемого вопроса. Фильтрация неподходящих или трудно интерпретируемых данных требует специальных знаний области применения. Английский язык действительно является основным языком, используемым в Соединенных Штатах, что делает решение о фильтрации данных таким способом совершенно разумным.

Однако фильтрация вносит предубеждения. Более 10% населения США говорит по-испански. Разве они не должны быть представлены в языковой модели, *амиго*? Важно выбрать правильные критерии фильтрации для достижения желаемого результата. Возможно, мы могли бы лучше фильтровать твиты по месту происхождения, а не по языку.

Выборка (sampling), напротив, означает выбор подмножества подходящего размера произвольным образом без критериев, специфичных для предметной области. Есть несколько причин, по которым мы можем захотеть отобрать хорошие, релевантные данные.

- *Правильный подбор учебных данных.* Простые, надежные модели обычно имеют мало параметров, что делает большие данные ненужными для их подгонки. Подвыборка ваших данных беспристрастным способом приводит к эффективной подгонке модели, но все еще является представителем всего набора данных.
- *Разделение данных.* Гигиена построения модели требует четкого разделения учебных данных, данных проверки и оценки, как правило, в

соотношении 60, 20 и 20%. Непредвзятое построение таких разделов необходимо для достоверности этого процесса.

- *Исследовательский анализ и визуализация данных*. Наборы данных, размером с электронную таблицу, быстры и просты в изучении. Беспристрастная выборка представляет целое, оставаясь понятной.

Выборка *n* записей эффективным и беспристрастным способом — это более тонкая задача, чем может показаться на первый взгляд. Существует два основных подхода: детерминированный и случайный, которые подробно описаны в следующих разделах.

12.4.1. Детерминированные алгоритмы выборки

Наш алгоритм выборки будет осуществлять *выборку с усечением* (sampling by truncation), которая в качестве желаемой выборки просто берет первые *n* записей из файла. Это просто и легко *воспроизводимо*, а значит, некто другой с полным файлом данных может легко воссоздать выборку.

Однако порядок записей в файле зачастую кодирует семантическую информацию, а это значит, что усеченные выборки нередко содержат тонкие эффекты от следующих факторов.

- *Временное смещение*. Файлы журнала обычно создаются в ходе добавления новых записей в конец файла. Таким образом, первые *n* записей будут самыми старыми из доступных и не будут отражать недавние изменения режима.
- *Лексикографическое смещение*. Многие файлы сортируются в соответствии с первичным ключом, т.е. первые *n* записей смещены к определенной совокупности. Представьте себе список персонала, отсортированный по именам. Первые *n* записей могут начинаться только на *A*, а значит мы, вероятно, будем перебирать арабские и китайские имена из общей популяции.
- *Численное смещение*. Зачастую файлы сортируются по идентификационным номерам, которые могут быть определены произвольно. Но идентификационные номера могут кодировать значение. Рассмотрим возможность сортировки данных о кадрах по номерам социального страхования в США. Первые пять цифр номеров социального страхования, как правило, являются функцией года и места рождения. Таким образом, усечение приводит к географической и возрастной выборке.

  Файлы данных нередко создаются в результате объединения небольших файлов, некоторые из которых могут быть гораздо более полезными в

положительных примерах, чем другие. В особенно патологических случаях номер записи может полностью кодировать переменную класса, а это означает, что точный, но совершенно бесполезный классификатор может проистекать из использования идентификатора класса в качестве параметра.

Так что обычно усечение — плохая идея. Несколько лучший подход — *равномерная выборка* (uniform sampling). Предположим, что мы пытаемся выбрать n/m записей из n записей данного файла. Простой подход заключается в том, чтобы начать с i-й записи, где i — это некоторое значение от 1 до m, а затем произвести выборку каждой m-й записи, начиная с i. Другой способ сделать это — выводить j-ю запись, если $j \pmod m = i$. Такая равномерная выборка позволяет уравновесить многие проблемы.

- Мы получаем точно необходимое количество записей для нашей выборки.
- Это быстро и воспроизводимо кем угодно, учитывая файл и значения i и m.
- Легко построить несколько непересекающихся выборок. Если мы повторим процесс с другим смещением i, мы получим независимую выборку.

Twitter использует этот метод для управления службами API, которые предоставляют доступ к сообщениям. Свободный уровень доступа (spritzer hose) выдает 1% потока, передавая каждое сотое сообщение. Профессиональные уровни доступа дают каждое десятое сообщение или даже больше, в зависимости от того, за что вы готовы платить.

Обычно это лучше, чем усечение, но все еще существуют потенциальные периодические временны*е* смещения. Если вы выберете каждую m-ю запись в журнале, возможно, каждый элемент, который вы увидите, будет связан с событием со вторника или в 23:00 каждую ночь. В файлах, отсортированных по номерам, вы рискуете получить элементы с одинаковыми цифрами младших разрядов. Телефонные номера, оканчивающиеся на “000”, или повторяющиеся цифры, например “8888”, нередко зарезервированы для служебного пользования, а не для домашнего, что приводит к смещению выборки. Вы можете минимизировать вероятность такого явления, если выбрать m достаточно большим простым числом, но единственный определенный способ избежать смещения выборки — это использовать случайность (рандомизацию).

12.4.2. Случайная и потоковая выборка

Случайная выборка записей с вероятностью p приводит к выбору ожидаемых pn элементов без каких-либо явных отклонений. Типичные генераторы случайных чисел возвращают значение от 0 до 1, полученное из равномерного

распределения. Мы можем использовать вероятность выборки p в качестве порога. Когда мы сканируем каждую новую запись, создается новое случайное число r. Когда $r \leq p$, мы принимаем эту запись в нашу выборку, но когда $r > p$, мы игнорируем ее.

Случайная выборка — это, как правило, надежная методология, но она сопровождается определенными техническими особенностями. Статистические расхождения гарантируют, что некоторые региональные или демографические данные будут подвергнуты чрезмерной выборке по отношению к населению, но непредвзято и в предсказуемой степени. Несколько случайных выборок не будут непересекающимися, и случайная выборка не воспроизводится без начальной позиции и случайного генератора.

Поскольку окончательное количество выбранных записей зависит от случайности, мы можем получить слишком много или слишком мало элементов. Если нам нужно *ровно* k элементов, мы можем построить случайную перестановку элементов и обрезать ее после первых k. Алгоритмы построения случайных перестановок обсуждались в разделе 5.5.1. Они просты, но требуют большого количества нерегулярных перемещений данных, что делает их потенциально плохой новостью для больших файлов. Более простой подход — добавить новое поле случайных чисел к каждой записи и отсортировать его в качестве ключа. Взятие первых k записей из этого отсортированного файла эквивалентно случайной выборке из ровно k записей.

Получение случайной выборки фиксированного размера из потока является более сложной задачей, поскольку мы не можем сохранить все элементы до конца. На самом деле мы даже не знаем, насколько большим будет n.

Чтобы решить эту проблему, мы будем поддерживать равномерную выборку в массиве размером k, обновляемом по мере поступления каждого нового элемента из потока. Вероятность того, что n-й элемент потока принадлежит выборке, равна k/n, поэтому мы вставим его в наш массив, если случайное число $r \leq k/n$. Это должно удалить текущий элемент из таблицы, а выбор того, какой текущий элемент массива будет жертвой, может быть сделан с помощью другого вызова генератора случайных чисел.

12.5. Параллелизм

Одна голова хорошо, две — лучше, а сто голов лучше, чем две. Вычислительные технологии развивались таким образом, что для вашего приложения становится все более целесообразным объединять несколько элементов обработки по требованию. Микропроцессоры обычно имеют 4 и более ядер, поэтому стоит задуматься о параллельности даже на отдельных машинах. Появление центров обработки данных и облачных вычислений облегчило аренду

большого количества машин по требованию, позволяя даже небольшим операторам воспользоваться преимуществами крупных распределенных инфраструктур.

Существует два разных подхода к одновременным вычислениям с несколькими компьютерами, а именно параллельные и распределенные вычисления. Различие здесь в том, насколько тесно связаны машины, и связаны задачи с процессором или с вводом-выводом в память.

- *Параллельная обработка* осуществляется на одном компьютере с участием нескольких ядер и/или процессоров, которые взаимодействуют через потоки и ресурсы операционной системы. Такие тесно связанные вычисления зачастую связаны с процессором и ограничены в большей степени количеством циклов, чем перемещением данных между машинами. Акцент делается на решении конкретной вычислительной проблемы быстрее, чем можно было бы сделать последовательно.
- *Распределенная обработка* осуществляется на многих машинах с использованием сетевой связи. Потенциальный масштаб здесь огромен, но наиболее это подходит для слабо связанных задач, которые мало общаются. Зачастую целью распределенной обработки является совместное использование несколькими компьютерами скорее таких ресурсов, как память и вторичное хранилище, чем нескольких процессоров. Всякий раз, когда скорость чтения данных с диска становится узким местом, нам лучше иметь много машин, читающих как можно больше разных дисков одновременно.

В этом разделе мы представим основные принципы параллельных вычислений и два относительно простых способа их использования: параллелизм данных и сеточный поиск. MapReduce является основной парадигмой для распределенных вычислений для больших данных и будет темой раздела 12.6.

12.5.1. Один, два, много

Первобытные культуры были не очень подкованы в числовом отношении, и предположительно считали только с использованием слов "*один*", "*два*" и "*много*". На самом деле это очень хороший способ думать о параллельных и распределенных вычислениях, потому что сложность очень быстро увеличивается с количеством машин.

- *Один*. Стараясь занять все ядра своего компьютера, вы все еще работаете на одном компьютере. Это нераспределенные вычисления.
- *Два*. Возможно, вы попытаетесь вручную разделить работу между несколькими машинами в вашей локальной сети. Это едва-едва распределенные вычисления, и обычно ими управляют с помощью специальных методов.

- *Много*. Чтобы воспользоваться преимуществами десятков или даже сотен машин, возможно, в облаке, у нас нет другого выбора, кроме как использовать такую систему, как MapReduce, которая может эффективно управлять этими ресурсами.

Сложность увеличивается параллельно с количеством агентов, координирующих выполнение задачи. Подумайте, что происходит по мере увеличения количества собравшихся людей. Существует постоянная тенденция обходиться все более слабой координацией по мере увеличения размера и роста вероятности возникновения неожиданных и катастрофических событий, пока они не станут настолько вероятными, что вполне можно ожидать неожиданного.

- 1 человек. Встречу легко организовать с помощью личного общения.
- > 2 человека. Ужин среди друзей требует активной координации.
- > 10 человек. Групповое собрание требует, чтобы был ответственный лидер.
- > 100 человек. Свадебный ужин требует фиксированного меню, поскольку кухня не может управиться с разнообразием возможных заказов.
- > 1000 человек. На любом фестивале или параде никто не знаком с большинством участников.
- > 10000 человек. После любой крупной политической демонстрации кто-то собирается провести ночь в больнице, даже если марш проходит мирно.
- > 100 000 человек. На любом крупном спортивном мероприятии один из зрителей, по-видимому, умрет в тот день либо от сердечного приступа [98], либо от несчастного случая по дороге домой.

Если некоторые из этих примеров кажутся вам нереальными, вспомните, что продолжительность типичной человеческой жизни составляет 80 *лет* × × 365 *дней в году* = 29 200 *дней*. Но, возможно, это проливает свет на некоторые проблемы параллельных и распределенных вычислений.

- *Координация*. Как мы назначаем рабочие единицы процессорам, особенно когда у нас больше рабочих единиц, чем рабочих? Как мы агрегируем или объединяем усилия каждого работника в единый результат?
- *Связь*. В какой степени работники могут поделиться частичными результатами? Как мы можем узнать, когда все рабочие закончат свои задачи?
- *Отказоустойчивость*. Как мы переназначаем задачи, если рабочие уходят или умирают? Должны ли мы защищаться от злонамеренных и систематических атак или просто случайных сбоев?

На заметку. Параллельные вычисления работают, когда мы можем минимизировать сложность связи и координации и выполнить задачу с низкой вероятностью отказа.

12.5.2. Параллелизм данных

Параллелизм данных (data parallelism) включает в себя разбиение и репликацию данных между несколькими процессорами и дисками, запуск одного и того же алгоритма на каждом фрагменте, а также последующий сбор результатов вместе для получения окончательных результатов. Мы предполагаем, что *главная* машина распределяет задачи среди нескольких подчиненных и собирает результаты.

Типичная задача — собрать статистику из большого набора файлов, скажем, подсчитать, как часто слова встречаются в массивном текстовом корпусе. Подсчет для каждого файла может быть осуществлен независимо как частичный результат для целого, и задача объединения этих результирующих файлов подсчетов легко осуществляется в конце одной машиной. Основным преимуществом этого является простота, поскольку все процессы подсчета выполняются одной и той же программой. Связь между процессорами проста: переместить файлы на соответствующий компьютер, запустить задание, а затем сообщить результаты обратно на главный компьютер.

Самый простой подход к многоядерным вычислениям предполагает параллелизм данных. Данные естественным образом образуют разделы, установленные временем, алгоритмами кластеризации или естественными категориями. Для большинства задач агрегации записи могут быть произвольно разделены при условии, что все подзадачи будут объединены вместе в конце, как показано на рис. 12.1.

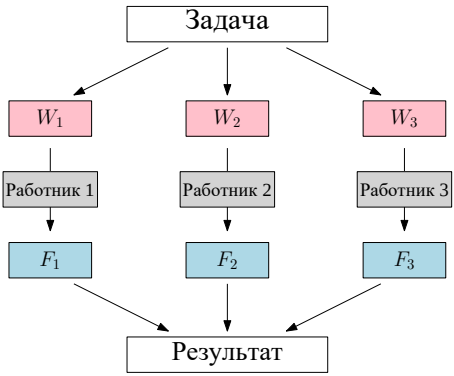


Рис. 12.1. Разделяй и властвуй — это алгоритмическая парадигма распределенных вычислений

Для более сложных задач требуется дополнительная работа, чтобы объединить впоследствии результаты этих запусков. Вспомним алгоритм кластеризации k-средних (раздел 10.5.1), который состоит из двух этапов.

1. Для каждой точки указывает, какой текущий центр кластера находится ближе всего к ней.
2. Вычисляет новый центр тяжести точек, связанных с ним.

Предполагая, что точки распределены по нескольким машинам, первый шаг требует, чтобы главная машина связывала все текущие центры с каждой машиной, в то время как второй шаг требует, чтобы каждый подчиненный докладывал главному о новых центроидах точек в его разделе. Затем главный соответствующим образом вычисляет средние значения этих центроидов, чтобы завершить итерацию.

12.5.3. Сеточный поиск

Второй подход к использованию параллелизма предусматривает несколько независимых запусков для одних и тех же данных. Мы видели, что многие методы машинного обучения включают параметры, которые влияют на качество конечного результата, такие как выбор правильного количества кластеров k для кластеризации k-средних. Выбор лучшего означает опробование их всех, и каждый из этих запусков может проводиться одновременно на разных машинах.

Сеточный поиск (grid search) — это поиск правильных метапараметров при обучении. Довольно трудно точно предсказать, как изменение скорости обучения или размера партии при стохастическом градиентном спуске влияет на качество окончательной модели. Несколько независимых подборов могут выполняться параллельно, и, в конце концов, мы выбираем лучший в соответствии с нашей оценкой.

Эффективный поиск пространства по k различным параметрам затруднен из-за взаимодействий: определение наилучшего значения, отдельного для каждого параметра, не обязательно дает лучший набор параметров при объединении. Как правило, разумные минимальные и максимальные значения для каждого параметра p_i, а также количество значений t_i для этого параметра устанавливаются пользователем. Каждый интервал разбивается на равные интервалы значений, регулируемых этим t_i. Затем мы опробуем все наборы параметров, которые можно сформировать, выбирая по одному значению на интервал, устанавливая сетку в сеточном поиске.

Насколько мы можем доверять утверждению, что лучшая модель в сеточном поиске *действительно* лучше других? Зачастую есть простая разница, которая объясняет небольшие различия в производительности на данном наборе тестов, превращая сеточный поиск в *избирательный подход* (cherry-picking),

выбирающий число, которое делает нашу производительность лучше. Если у вас есть вычислительные ресурсы, доступные для проведения сеточного поиска в вашей модели, не стесняйтесь идти вперед, но осознайте пределы того, что можно сделать методом проб и ошибок.

12.5.4. Службы облачных вычислений

Такие платформы, как Amazon AWS, Google Cloud и Microsoft Azure, позволяют легко арендовать большое (или небольшое) количество компьютеров для краткосрочных (или долгосрочных) заданий. Они предоставляют вам возможность получать доступ к нужному количеству вычислительных ресурсов при условии, что вы, конечно, можете за них заплатить.

Однако модели оплаты у этих поставщиков услуг несколько сложны. Обычно они составляют почасовую оплату для каждой виртуальной машины в зависимости от типа процессора, количества ядер и задействованной основной памяти. Аренда машин будут стоить от 10 до 50 центов в час. Вы будете платить за объем долговременного хранилища в зависимости от количества гигабайтов в месяц с разными уровнями стоимости в зависимости от моделей доступа. Кроме того, вы платите за пропускную способность, обеспечивающую необходимый объем передачи данных между компьютерами и через Интернет.

Спотовые цены и зарезервированные экземпляры виртуальных машин могут привести к снижению почасовой оплаты за особые шаблоны использования, но с дополнительными оговорками. При *спотовых ценах* (spot pricing) машины переходят на самые высокие цены, поэтому ваша работа может быть прервана, если кому-то это нужно больше, чем вам. При *зарезервированных экземплярах* (reserved instance) виртуальных машин вы платите определенную сумму заранее, чтобы получить более низкую почасовую цену. Это имеет смысл, если вам понадобится один компьютер 24/7 в течение года, но не тогда, когда вам нужно сто компьютеров на один конкретный день.

К счастью, экспериментировать можно и бесплатно. Все крупные поставщики облачных услуг предоставляют бесплатное время новым пользователям, поэтому вы можете поиграть с настройками и решить, насколько это вам подходит.

12.6. MapReduce

Парадигма MapReduce от Google для распределенных вычислений широко распространилась в реализациях с открытым исходным кодом, таких как Hadoop и Spark. Она предлагает простую модель программирования с несколькими преимуществами, включая простое масштабирование до сотен или даже тысяч машин, а также отказоустойчивость за счет избыточности.

Уровень абстракции программных моделей со временем постоянно увеличивается, что выражается в более мощных инструментах и системах, которые скрывают детали реализации от пользователя. Если вы занимаетесь наукой о данных в масштабе нескольких компьютеров, вычисления MapReduce, вероятно, будут скрыты, даже если вы не программируете их явно.

Важный класс крупномасштабных задач в науке о данных имеет следующую базовую структуру.

- Перебор большого количества элементов, будь то записи данных, текстовые строки или каталоги файлов.
- Извлечение чего-нибудь интересного из каждого элемента, будь то значение определенного поля, подсчет частоты каждого слова или наличие/отсутствие определенных шаблонов в каждом файле.
- Объединение промежуточных результатов по всем элементам и получение соответствующего комбинированного результата.

Представители этого класса задач включают подсчет частоты слов, кластеризацию методом k-средних и вычисления PageRank. Все они разрешимы с помощью простых итеративных алгоритмов, чье время работы линейно зависит от размера входных данных. Но этого может быть недостаточно для входных данных большого размера, когда файлы не помещаются в памяти одной машины. Подумайте о задачах масштаба веб-сайтов, таких как подсчет частоты слов в миллиардах сообщений, кластеризации методом k-средних в сотнях миллионов профилей Facebook и PageRank на всех веб-сайтах в Интернете.

Типичное решение здесь — *разделяй и властвуй*. Распределите входные файлы между m различными машинами, выполните вычисления параллельно на каждой из них, а затем объедините результаты на соответствующей машине. Такое решение, в принципе, работает для подсчета слов, потому что даже огромные текстовые корпуса в конечном итоге сократятся до сравнительно небольших файлов отдельных словарных слов с соответствующими частотными отсчетами, которые затем могут быть легко суммированы вместе для получения полных результатов подсчетов.

Но давайте рассмотрим вычисление PageRank, где для каждого узла v нам нужно суммировать PageRank от всех узлов x, где x ссылаются на v. Нет никакого способа, которым мы можем разделить график на отдельные части так, чтобы все эти вершины x находились на тех же машинах, что и v. Получение необходимого в нужном месте для работы как раз и лежит в основе MapReduce.

12.6.1. Программирование MapReduce

Ключом к распределению таких вычислений является настройка распределенной хеш-таблицы сегментов, в которой все элементы с одинаковым ключом соотносятся с одним и тем же сегментом.

- *Подсчет слов.* Для подсчета общей частоты конкретного слова w в наборе файлов нам нужно собрать значения частоты для всех файлов в одном сегменте, связанном с w. Там их можно будет сложить, чтобы получить итоговую сумму.
- *Кластеризация методом k-средних.* Критическим шагом в кластеризации методом k-средних является обновление нового центроида c' точек, ближайших к текущему центроиду c. После хеширования всех точек p, ближайших к c, в одном сегменте, связанном с c, мы можем вычислить c' за один проход через этот сегмент.
- *PageRank.* Новый PageRank вершины v является суммой старого PageRank для всех соседних вершин x, где (x, v) — это направленное ребро в графе. Хеширование PageRank x в сегмент для всех смежных вершин v собирает всю необходимую информацию в нужном месте, поэтому мы можем обновить PageRank за один проход через него.

Эти алгоритмы могут быть определены с помощью двух написанных программистом функций, `map` и `reduce`.

- *Функция* `map`. Осуществляет трансформацию каждого входного файла, хеширует или *испускает* (emitting) пары ключевых значений, в зависимости от ситуации. Рассмотрим следующий псевдокод для трансформации количества слов:

```
Map(String docid, String text):
    for each word w in text:
        Emit(w, 1);
```

- *Функция* `reduce`. Осуществляет просмотр набора значений v, связанных с конкретным ключом k, агрегируя и обрабатывая их соответственно. Псевдокод этой функции для количества слов:

```
Reduce(String term, Iterator<Int> values):
    int sum = 0;
    for each v in values:
        sum + = v;
```

Эффективность программы MapReduce зависит от многих вещей, но одна из важнейших задач заключается в том, чтобы свести к минимуму количество выбросов. Создание счетчика для каждого слова запускает сообщение по

разным машинам, и это общение и связанные с ним записи в пакет оказываются дорогостоящими. Чем больше материалов трансформируется (mapped), тем больше должно быть редуцировано (reduced).

Идеальным вариантом является *объединение* подсчетов из отдельных входных потоков и последующее испускание только суммы для каждого отдельного слова в файле. Это можно сделать, добавив дополнительную логику/структуру данных в функцию `map`. Альтернативный подход — запускать мини-редукторы в памяти после фазы трансформации, но до межпроцессорной связи, в качестве оптимизации для снижения сетевого трафика. Заметим, что оптимизация для вычислений в памяти является одним из основных преимуществ производительности в Spark по сравнению с Hadoop для программирования в стиле MapReduce.

На рис. 12.2 показан поток задачи MapReduce для подсчета слов с использованием трех *трансформаторов* (mapper) и двух *редукторов* (reducer). Объединение было осуществлено локально, поэтому счетчики для каждого слова, использованного более одного раза во входном файле (здесь *doc* и *be*)[1], были сведены в таблицу до их передачи редукторам.

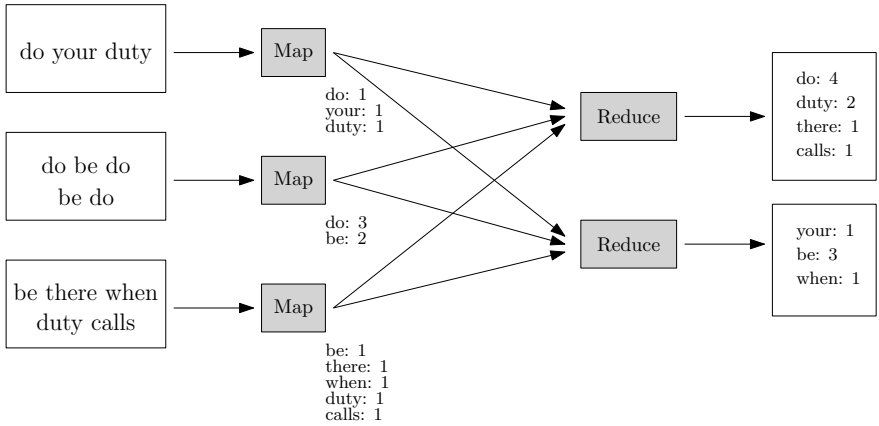


Рис. 12.2. Подсчет количества слов в действии. Перед трансформацией было осуществлено объединение подсчетов для уменьшения размера файлов трансформации

Одной из проблем, приведенных на рис. 12.2, является проблема *перекоса* (skew) — естественного дисбаланса в объеме работы, назначенной для каждой задачи редукции. В данном примере верхнему редуктору были назначены файлы трансформации, содержащие на 33% и на 60% больше слов больше, чем у его партнера. Для задачи с последовательным временем выполнения T идеальная параллелизация с n процессорами даст время выполнения T/n. Но время

[1] Вероятно, имелось в виду *do*, *duty* и *be*. — *Примеч. ред.*

выполнения задачи MapReduce определяется самой большой и самой медленной частью. Перекос трансформатора обрекает нас на самый большой фрагмент, который зачастую существенно больше среднего размера.

Одним из источников перекоса трансформатора является удача розыгрыша, поскольку очень редко выпадает ровно столько же орлов, сколько и решек при n бросках монеты. Но куда более серьезная проблема заключается в том, что ключевая частота зачастую распределяется по степенному закону, поэтому наиболее частый ключ будет доминировать в подсчетах. Рассмотрим проблему подсчета слов и предположим, что частота слов соответствует закону Ципфа из раздела 5.1.5. Тогда частота самого популярного слова (*the*) должна быть больше, чем сумма тысяч слов, ранжированных от 1000 до 2000. В зависимости от того, в какой интервал попадет *the*, это может оказаться самым трудным для усвоения.[2]

12.6.2. MapReduce под капотом

Все это хорошо. Но как такая реализация MapReduce, как Hadoop, гарантирует, что все трансформируемые элементы будут перемещены в нужное место? И как он назначает работу процессорам и синхронизирует операции MapReduce, причем все отказоустойчивым способом?

Существует два основных компонента: распределенная хеш-таблица (или файловая система) и система времени выполнения, занимающаяся координацией и управлением ресурсами. Оба этих компонента подробно описаны ниже.

Распределенные файловые системы

Большие коллекции компьютеров могут использовать для выполнения задачи свое пространство памяти (RAM) и локальное дисковое хранилище, а не только свои процессоры. Распределенная файловая система, такая как у Hadoop (Hadoop Distributed File System — HDFS), может быть реализована в виде распределенной хеш-таблицы. После того как набор машин зарегистрирует свою доступную память в координирующей системе времени выполнения, каждому из них может быть назначен определенный диапазон хеш-таблиц, за который он будет отвечать. Каждый процесс, выполняющий трансформацию, может затем гарантировать, что переданные элементы будут отправлены в соответствующее хранилище на соответствующем компьютере.

Поскольку большое количество элементов может быть сопоставлено с одним сегментом, мы можем выбирать сегменты в виде дисковых файлов с

[2] Это одна из причин, по которой наиболее часто встречающиеся слова в языке объявляются *стоп-словами* (stop word) и зачастую не указываются в качестве параметров в задачах анализа текста.

новыми элементами, добавляемыми в конец. Доступ к диску медленный, но пропускная способность диска приемлемая, поэтому линейное сканирование файлов обычно управляемо.

Одна из проблем с такой распределенной хеш-таблицей кроется в отказоустойчивости: отказ одной машины может привести к потере достаточного количества значений, чтобы сделать недействительными все вычисления. Решение заключается в репликации всего для надежности на стандартном оборудовании. В частности, во время выполнения система будет реплицировать каждый элемент на трех разных компьютерах, чтобы минимизировать вероятность потери данных при сбое оборудования. Когда система во время выполнения обнаруживает, что компьютер или диск не работает, она начинает реплицировать потерянные данные из этих копий, чтобы восстановить работоспособность файловой системы.

Система времени выполнения MapReduce

Другим важным компонентом среды MapReduce для Hadoop или Spark является их *система времени выполнения* (runtime system) — уровень программного обеспечения, который регулирует следующие задачи.

- *Планирование процессора.* Какие ядра назначены для выполнения, какие задачи трансформации и редуцирования и для каких входных файлов? Программист может помочь, предложив, сколько трансформаторов и редукторов должно быть активно одновременно, но назначение заданий ядрам зависит от системы времени выполнения.
- *Распределение данных.* Это может подразумевать перемещение данных на доступный процессор, который может с ними справиться, но следует помнить, что типичные операции трансформации и редуцирования требуют простой линейной развертки через потенциально большие файлы. Таким образом, перемещение файла может быть более дорогим, чем просто локальное выполнение желаемых вычислений.

  Таким образом, лучше перенести процессы в данные. Система времени выполнения должна иметь конфигурацию доступных ресурсов, а также общую схему сети. Она может принять соответствующее решение о том, какие процессы должны выполняться и где.
- *Синхронизация.* Редукторы не могут работать, пока что-то не было сопоставлено с ними, и не могут завершить работу, пока трансформация не будет выполнена. Spark позволяет выполнять более сложные рабочие процессы, помимо синхронизированных циклов трансформации и редукции. Эту синхронизацию осуществляет система времени выполнения.

- *Ошибки и отказоустойчивость*. Надежность MapReduce требует надежного восстановления после отказов оборудования и связи. Когда система времени выполнения обнаруживает отказ рабочего, она пытается перезапустить вычисление. Когда это не удается, она передает незавершенные задачи другим работникам. То, что все это происходит незаметно, без участия программиста, позволяет нам масштабировать вычисления для больших сетей компьютеров в таких масштабах, в которых отказы вполне вероятные события, а не редкие.

Слои поверх слоев

Такие системы, как HDFS и Hadoop, являются просто слоями программного обеспечения, на которых могут основываться другие системы. Хотя Spark можно считать конкурентом Hadoop, на самом деле он может использовать распределенную файловую систему Hadoop и зачастую наиболее эффективен при этом. В наши дни мои ученики, кажется, тратят меньше времени на написание низкоуровневых заданий MapReduce, поскольку вместо этого они используют программные слои, работающие на более высоких уровнях абстракции.

Полная экосистема больших данных состоит из множества различных видов. Важным классом являются базы данных *NoSQL*, которые способны распределять структурированные данные по распределенной сети машин, позволяя объединять RAM и диски нескольких машин. Кроме того, эти системы обычно разрабатываются так, чтобы вы могли добавлять дополнительные машины и ресурсы по мере необходимости. Цена такой гибкости заключается в том, что базы обычно поддерживают более простые языки запросов, чем полноценный SQL, но все же они достаточно богаты для многих приложений.

Программная экосистема больших данных развивается гораздо быстрее, чем основополагающие вопросы, обсуждаемые в этой книге. Поиск в Google и сканирование каталога книг O'Reilly должны выявить новейшие технологии, когда вы будете готовы приступить к делу.

12.7. Социальные и этические последствия

Наша способность попасть в серьезные неприятности увеличивается с ростом размера. Автомобиль может вызвать более серьезную аварию, чем велосипед, а самолет — более серьезную катастрофу, чем автомобиль.

Большие данные могут многое сделать для мира, но они также могут нанести вред людям и обществу в целом. Поведение, которое безвредно в небольших масштабах, например плагиат, в больших масштабах становится воровством интеллектуальной собственности. Описание точности вашей модели в чрезмерно благоприятном свете является обычным явлением для презентаций

PowerPoint, но имеет реальные последствия, когда ваша модель регулирует разрешение на получение кредита или доступ к медицинскому обслуживанию. Потеря доступа к вашей учетной записи электронной почты является глупой неудачей, но неправильная защита личных данных 100 миллионов клиентов становится потенциально преступной.

Я заканчиваю эту книгу кратким обзором общих этических проблем в мире больших данных, чтобы помочь вам понять, о каких вещах беспокоится общественность.

- *Честность в общении и моделировании.* Ученый, работающий с данными, служит проводником между их анализом и работодателем или широкой публикой. Существует большой соблазн заставить наши результаты казаться лучше, чем они есть на самом деле, используя различные проверенные временем методы.
  - Мы можем сообщить об уровне корреляции или точности, не сравнивая его с базовой линией или не сообщая значение *p*.
  - Мы можем выбирать только лучшие результаты из множества экспериментов и представлять только их, вместо того, чтобы представить более точную картину.
  - Мы можем использовать методы визуализации так, чтобы скрыть информацию, а не предоставить ее.
- В каждую модель заложены предположения и недостатки. Хороший разработчик знает, каковы ограничения его модели: они верят в то, что она может сделать, и где они начинают чувствовать себя менее уверенно. Честный разработчик сообщает полную картину своей работы: что они знают и в чем они не так уверены.
- Конфликты интересов представляют собой серьезную проблему в науке о данных. Зачастую кто-то знает, каков должен быть "правильный ответ" еще перед началом исследования, особенно тот результат, который босс хочет больше всего услышать. Возможно, ваши результаты будут использованы для влияния на общественное мнение или появятся в качестве свидетельских показаний перед юридическими или государственными органами. Точная отчетность и распространение результатов являются важным элементом поведения для специалистов по этике.
- *Прозрачность и собственность.* Обычно компании и исследовательские организации публикуют условия использования и хранения данных, чтобы продемонстрировать, что им можно доверять данные своих клиентов. Такая прозрачность важна, но, как оказалось, она может измениться, как только станет очевидной коммерческая ценность данных. Зачастую легче получить прощение, чем разрешение.

В какой степени пользователи владеют созданными ими данными? Право собственности означает, что у них должно быть право видеть, какая информация была получена от них, и возможность предотвратить использование этого материала в будущем. Эти проблемы могут стать весьма сложными, как с технической, так и с этической точек зрения. Должен ли преступник быть в состоянии потребовать, чтобы все ссылки на его преступление были удалены из такой поисковой системы, как Google? Может ли моя дочь потребовать удалить ее изображения, опубликованные другими без ее разрешения?

Ошибки в данных могут распространяться и наносить вред отдельным лицам, не позволяя людям получить доступ к механизму и понять, какая информация о них была собрана. Неправильная или неполная финансовая информация может испортить чей-либо кредитный рейтинг, но по закону кредитные агентства вынуждены предоставлять записи каждого человека и предоставлять механизм для исправления ошибок. Однако происхождение данных, как правило, теряется при объединении файлов, поэтому эти обновления не обязательно возвращаются ко всем производным продуктам, созданным на основе дефектных данных. Как без этого ваши клиенты могут обнаружить и исправить неверную информацию о вас?

- *Неисправимые решения и петли обратной связи.* Использование моделей в качестве жестких критериев отбора может быть опасным, особенно в тех областях, где модель является просто приближением для того, что вы действительно хотите измерить. Корреляция — это не причинно-следственная связь. Однако рассмотрим модель, предполагающую, что нанимать конкретного кандидата на работу опасно, поскольку таких людей, как он, из нижних слоев населения, арестовывают с большей вероятностью. Если все работодатели будут использовать такие модели, эти люди просто не будут приняты на работу нигде и будут ввергнуты в бедность не по своей вине.

  Эти проблемы особенно коварны, поскольку они, как правило, неисправимы. У жертвы модели, как правило, нет средств для апелляции. Владелец модели также не имеет возможности узнать, чего ему не хватает, т.е. сколько хороших кандидатов было отброшено без дальнейшего рассмотрения.

- *Искажения и фильтры, присущие модели.* Большие данные позволяют настраивать продукты так, чтобы они наилучшим образом подходили для каждого отдельного пользователя. Google, Facebook и другие системы анализируют ваши данные так, чтобы показать вам те результаты, которые, по мнению их алгоритмов, вы больше всего хотите увидеть.

Но эти алгоритмы могут содержать непреднамеренные искажения, полученные от алгоритмов машинного обучения, обученных на сомнительных обучающих наборах. Возможно, поисковая система покажет хорошие возможности трудоустройства мужчинам гораздо чаще, чем женщинам, или дискриминирует их по другим критериям.

Точное отображение того, что вы хотите увидеть, может помешать вам увидеть информацию, которая вам действительно нужна. Такие фильтры могут нести определенную ответственность за политическую поляризацию в нашем обществе: увидите ли вы противоположные точки зрения или просто эхо для ваших собственных мыслей?

- *Обеспечение безопасности больших наборов данных.* Большие данные представляют собой бо́льшую цель для хакеров, чем электронная таблица на вашем жестком диске. Мы объявили, что файлы со 100 миллионами записей являются *крохами*, но они могут представлять личные данные 30% населения США. Взломы данных такого масштаба происходят с частотой стихийных бедствий.

  Заставить 100 миллионов человек сменить пароль стоит 190 человеко-лет бесполезных усилий, даже если каждое исправление занимает всего одну минуту. Но большая часть информации не может быть изменена с такой быстротой: адреса, идентификационные номера и данные учетной записи сохраняются годами, если не всю жизнь, что делает невозможным полное возмещение ущерба за обозримое время от пакетных потерь данных.

  Специалисты по обработке данных обязаны полностью придерживаться методов обеспечения безопасности своих организаций и выявлять потенциальные недостатки. Они также несут ответственность за минимизацию угроз нарушения безопасности в результате шифрования и обезличивания. Но, пожалуй, самое важное — это избегать запроса полей и записей, которые вам не нужны, и (это, безусловно, самое сложное) удалять данные, когда потребность вашего проекта в них истекла.

- *Обеспечение конфиденциальности в агрегированных данных.* Недостаточно удалять имена, адреса и идентификационные номера, чтобы сохранить конфиденциальность в наборе данных. Даже анонимные данные могут быть эффективно деанонимизированы с помощью ортогональных источников данных. Рассмотрим набор данных такси, который мы ввели в разделе 1.6. Он никогда не содержал никакой информации об идентификаторе пассажира. Тем не менее он обеспечивает получение GPS-координат точки отправления, которые могут точно указать конкретный дом в качестве точки отправления или пункта назначения. Теперь у нас

есть довольно хорошая идея о том, кто совершил эту поездку, и такая же хорошая идея о том, кто может быть заинтересован в этой информации, если парень женат.

Соответствующий эксперимент выявил особые поездки на такси, предпринятые знаменитостями, чтобы выяснить их пункт назначения и то, насколько хорошие они дали чаевые [99]. Используя Google, можно найти сделанные папарацци фотографии знаменитостей, садящихся в такси, извлечь время и место, где они были сняты, и легко идентифицировать запись, соответствующую именно этой поездке, чтобы узнать ее цель.

Этические проблемы в науке о данных настолько серьезны, что профессиональные организации взвесили лучшие практики и опубликовали *Data Science Code of Professional Conduct* (Кодекс профессионального поведения в области науки о данных) (`http://www.datascienceassn.org/code-of-conduct.html`) Ассоциации по науке о данных, а также *Ethical Guidelines for Statistical Practices* (Этическое руководство по статистической практике) (`http://www.amstat.org/about/ethicalguidelines.cfm`) Американской статистической ассоциации.

Я призываю вас прочитать эти документы, чтобы развить чувство этических проблем и стандартов профессионального поведения. Напомним, что люди обращаются к ученым за мудростью и советом, а не просто за кодом. Делай, что можешь, чтобы доказать, что ты достоин этого доверия.

12.8. Дополнительная информация

Нет недостатка в книгах по теме анализа больших данных. Издание Лесковца (Leskovec), Раджарамана (Rajaraman) и Ульмана (Ullman) [100] — пожалуй, наиболее полное из них, в нем куда глубже рассмотрены темы, которые мы здесь обсуждаем. Эта книга и некоторые сопутствующие видео доступны по адресу `http://www.mmds.org`.

Мои любимые практические ресурсы по программным технологиям — это, как правило, книги издательства O'Reilly Media. В контексте этой главы я рекомендую их книги по аналитике данных с использованием Hadoop [101] и Spark [102].

О'Нил (O'Neil) [103] бросает задумчивый взгляд на социальные опасности анализа больших данных, подчеркивая неправильность использования непрозрачных моделей, опирающихся на косвенные источники данных, которые создают петли обратной связи, усугубляющие проблемы, которые они пытаются решить.

Аналогия скорости диска/кеша скорости черепахи/бегуна получена благодаря Майклу Бендеру (Michael Bender).

12.9. Упражнения

Параллельная и распределенная обработка

12.1. *[3]* В чем разница между параллельной обработкой и распределенной обработкой?

12.2. *[3]* Каковы преимущества MapReduce?

12.3. *[5]* Разработайте алгоритмы MapReduce для получения больших файлов целых чисел и вычисления:

- наибольшего целого числа;
- среднего из всех целых чисел;
- количества различных целых чисел на входе;
- моды целых чисел;
- медианы целых чисел.

12.4. *[3]* Будем ли мы ожидать, что перекос трансформации станет большей проблемой, когда есть десять редукторов или сто редукторов?

12.5. *[3]* Будем ли мы ожидать, что проблема перекоса трансформации будет увеличиваться или уменьшаться, когда мы объединяем счетчики из каждого файла перед их испусканием?

12.6. *[5]* Для каждого из следующих конкурсов *The Quant Shop* придумайте самый массивный из возможных источников данных, которые могут существовать реально, у кого они могут быть, и какие предубеждения могут скрываться в его взглядах на мир.

(a) *Miss Universe.*
(b) *Movie gross.*
(c) *Baby weight.*
(d) *Art auction price.*
(e) *White Christmas.*
(f) *Football champions.*
(g) *Ghoul pool.*
(h) *Gold/oil prices.*

Этика

12.7. *[3]* Каковы пять практических способов защиты конфиденциальности больших данных?

12.8. *[3]* Какие границы вы считаете приемлемыми для Facebook, чтобы использовать имеющиеся у них данные о вас? Приведите примеры использования, которые были бы неприемлемы для вас. Запрещено ли это соглашением об использовании данных?

12.9. *[3]* Приведите примеры принятия решений, когда вы доверяете алгоритму, позволяющему принимать такие же хорошие решения, как человек, или даже лучшие. Для каких задач вы бы доверяли человеческому суждению больше, чем алгоритму? Почему?

Реализация проектов

12.10. *[5]* Действительно ли методы выборки из потока, которые мы обсуждали, дают случайные выборки из желаемого распределения? Реализуйте их, получите выборки и проверьте их соответствующим статистическим тестом.

12.11. *[5]* Настройте кластер Hadoop или Spark, который охватывает два или более компьютеров. Запустите простую задачу, такую как подсчет слов. Действительно ли он работает быстрее, чем на одной машине? Сколько машин/ядер вам нужно, чтобы получить выигрыш?

12.12. *[5]* Найдите достаточно большой источник данных, к которому у вас есть доступ и обработку которого более чем одной машиной вы можете оправдать. Сделайте что-нибудь интересное с этим набором.

Вопросы на интервью

12.13. *[3]* Каково ваше определение больших данных?

12.14. *[5]* Какой самый большой набор данных вы обработали? Что вы делали и каковы были результаты?

12.15. *[8]* Дайте пять прогнозов о том, что произойдет в мире в течение следующих двадцати лет?

12.16. *[5]* Приведите несколько примеров лучших практик в науке о данных.

12.17. *[5]* Как вы можете обнаружить поддельные обзоры или поддельные учетные записи Facebook, используемые для плохих целей?

12.18. *[5]* Что делают функция `map` и функция `reduction` в рамках парадигмы MapReduce? Что делают *сумматор* (combiner) и *разделитель* (partitioner)?

12.19. *[5]* Не думаете ли вы, что введенный логин/пароль со временем исчезнет? Как их можно заменить?

12.20. *[5]* Когда ученый по данным не может сделать какой-либо вывод из набора данных, что он должен сказать своему начальнику/клиенту?

12.21. *[3]* Что такое конфликт хеш-таблиц? Как его можно избежать? Как часто они происходят?

Конкурсы Kaggle

12.22. Какие клиенты станут постоянными покупателями?

`https://www.kaggle.com/c/acquire-valued-shoppers-challenge`

12.23. Каким клиентам стоит рассылать спам?

`https://www.kaggle.com/c/springleaf-marketing-response`

12.24. Какой отель следует порекомендовать данному путешественнику?

`https://www.kaggle.com/c/expedia-hotel-recommendations`

Глава 13

Заключение

> — Начни с начала, — торжественно произнес Король, — и продолжай, пока не дойдешь до конца. Тогда остановись!
>
> — Льюис Кэрролл (Lewis Carroll)

Надеемся, что вас, читатель, хотя бы частично просветила эта книга и вы по-прежнему восхищены силой данных. Самый распространенный способ применения этих навыков — устроиться на работу по данной специальности. Это благородное призвание, но знайте, что есть и другие возможности.

13.1. Получить работу!

Для будущих аналитиков данных есть очень радужные прогнозы по перспективам получить работу. Институт McKinsey Global Institute прогнозирует, что спрос на "серьезные аналитические таланты в Соединенных Штатах может быть на 50-60% выше запланированного в 2018 году". Сайт по трудоустройству `www.glassdoor.com` информирует меня о том, что на сегодняшний день средняя заработная плата ученого составляет 113 436 долл. Ежемесячный научно-популярный журнал *Harvard Business Review* заявил, что быть специалистом по данным — это "самая сексуальная работа 21-го века" [104]. Это похоже на то место, где я хочу быть!

Но все эти утверждения были бы гораздо более убедительными, если бы существовало общее понимание того, что именно представляет собой специалист по данным. Для меня вовсе не очевидно, что когда-либо существовало огромное количество рабочих мест с официальным названием *аналитик данных*, как, например, *инженер по программному обеспечению* или *программист*. Однако не паникуйте.

Справедливо будет сказать, что существует несколько различных видов работ, связанных с наукой о данных, которые отличаются относительной важностью знаний приложений и технической мощью. Я вижу следующие основные специальности, связанные с наукой о данных.

- *Разработка программного обеспечения для науки о данных.* Значительная доля высоких позиций в разработке программного обеспечения принадлежит компаниям, работающим с большими данными, таким как Google, Facebook и Amazon, или компаниям, ориентированным на данные, в финансовом секторе, таким как банки и хедж-фонды. Эти работы связаны с созданием крупномасштабной программной инфраструктуры для управления данными, и, как правило, для получения необходимых технических навыков и опыта требуются ученые степени в области информатики.
- *Статистик/аналитик данных.* Для квалифицированных статистиков всегда существовал широкий рынок труда, особенно в области здравоохранения, производства, бизнеса, образования и государственных/некоммерческих секторов. Этот мир будет продолжать расти и процветать, хотя я подозреваю, что он потребует более глубоких навыков в области информатики, чем в прошлом. Эти статистические аналитики, ориентированные на вычисления, будут иметь подготовку или опыт в науке о данных, опираясь на прочную основу статистики.
- *Количественные бизнес-аналитики.* Большая группа профессионалов работает в области маркетинга, продаж, рекламы и управления, обеспечивая необходимые функции в любой товарной или консалтинговой компании. Эти профессии требуют больше знаний в области бизнеса, чем две предыдущие категории, но все чаще ожидают навыков количественного анализа. Возможно, они нанимают вас для работы в области маркетинга, но требуют опыта или знаний в области анализа данных и аналитики. Или они нанимают вас для работы в отделе кадров, но ожидают, что вы сможете разрабатывать критерии для производительности труда и оплаты труда.

Описанный в этой книге материал важен для всех трех профессиональных направлений, но, очевидно, вам есть чему поучиться. Специальность, которой легче всего обучить, оказывается также наиболее быстро насыщаемой кадрами, поэтому продолжайте развивать свои навыки с использованием курсовых работ, проектов и практики.

13.2. Пойти в аспирантуру!

Если вы находите идеи и методы, представленные в этой книге, интересными, возможно, вы именно тот человек, который должен подумать о поступлении в аспирантуру.[1] Технические навыки быстро устаревают без повышения

[1] Хорошо, если вы уже там!

квалификации, а после начала трудовой деятельности может быть трудно найти время для профессиональной подготовки.

Аспирантура по науке о данных быстро появляется на факультетах информатики, статистики, бизнеса, прикладной математики и т.д. Какой тип программы является наиболее подходящим для вас, зависит от вашей подготовки студентом и жизненного опыта. В зависимости от их направленности программы по науке о данных будут сильно отличаться по ожидаемой вычислительной и статистической подготовке. По правде говоря, технически сложные программы с точки зрения программирования, машинного обучения и статистики обеспечивают наилучшую подготовку на будущее. Помните о грандиозных претензиях программ, которые минимизируют эти требования.

Большинство из этих программ находятся на уровне магистратуры, но выдающиеся студенты, способные взять на себя повышенные обязательства, должны рассмотреть возможность докторантуры. Аспирантура по информатике, машинному обучению или статистике включает в себя курсы по передовым темам, которые основаны на том, что вы узнали как студент. Но, что более важно, вы будете проводить новые и оригинальные исследования в выбранной вами области. Все соответствующие американские докторские программы будут оплачивать обучение всех принятых аспирантов, а также выплачивать достаточную стипендию, чтобы жить комфортно, если не лучше.

Если у вас большой опыт работы в области информатики и есть необходимые навыки, я бы посоветовал вам продолжить учебу, в идеале приехав к нам в Stony Brook! Моя группа проводит исследования по множеству интересных тем в науке о данных, как вы можете судить по случаям из жизни. Пожалуйста, посетите нас по адресу `http://www.data-manual.com/gradstudy`.

13.3. Профессиональные консалтинговые услуги

Algorist Technologies — это консалтинговая фирма, которая предоставляет своим клиентам быструю экспертную помощь в области данных и разработки алгоритмов. Как правило, консультант компании Algorist привлекается для интенсивных обсуждений с персоналом клиента и анализа на месте сроком от одного до трех дней. Компания Algorist добилась впечатляющих результатов в повышении производительности нескольких компаний, а также в услугах привлеченных экспертов и долгосрочных консультаций.

Для дополнительной информации об услугах, предоставляемых компанией Algorist Technologies, посетите веб-сайт `www.algorist.com/consulting`.

Глава 14

Список литературы

[1] S. Skiena. *The Algorithm Design Manual* (Алгоритмы. Руководство по разработке). Springer-Verlag, London, second edition, 2008.

[2] Michael Lewis. *Moneyball: The art of winning an unfair game* (Money-Ball. Как математика изменила самую популярную спортивную лигу в мире). WW Norton & Company, 2004.

[3] Nate Silver. *The Signal and the Noise: Why so many predictions fail-but some don't*. Penguin, 2012.

[4] Diane F Halpern and Stanley Coren. Do right-handers live longer? *Nature*, 333:213, 1988.

[5] Robert M. Bell and Yehuda Koren. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[6] J. Michel, Y. Shen A. Aiden, A. Veres, M. Gray, Google Books Team, J. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. Nowak, and E. Aiden. Quantitative analysis of culture using millions of digitized books. *Science*, 331:176–182, 2011.

[7] Oleksii Starov and Steven Skiena. GIS technology supports taxi tip prediction. Esri Map Book, 2014 User Conference, July 14-17, San Diego, 2015.

[8] S. Skiena. Calculated Bets: *Computers, Gambling, and Mathematical Modeling to Win*. Cambridge University Press, New York, 2001.

[9] Wenbin Zhang and Steven Skiena. Improving movie gross prediction through news analysis. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 301–304. IEEE Computer Society, 2009.

[10] Wenbin Zhang and Steven Skiena. Trading strategies to exploit blog and news sentiment. In *Proc. Int. Conf. Weblogs and Social Media (ICWSM)*, 2010.

[11] Yancheng Hong and Steven Skiena. The wisdom of bookies? sentiment analysis vs. the NFL point spread. In *Int. Conf. on Weblogs and Social Media*, 2010.

[12] Steven S. Skiena and Charles B.Ward. *Who's Bigger?: Where Historical Figures Really Rank*. Cambridge University Press, 2013.

[13] Diane F Halpern and Stanley Coren. Handedness and life span. *N Engl J Med*, 324(14):998–998, 1991.

[14] Chris McManus. *Right Hand, Left Hand: The origins of asymmetry in brains, bodies, atoms and cultures.* Harvard University Press, 2004.

[15] Bill James. *The New Bill James Historical Baseball Abstract.* Simon and Schuster, 2010.

[16] Paolo Santi, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven H Strogatz, and Carlo Ratti. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, 111(37):13290–13294, 2014.

[17] Namrata Godbole, Manja Srinivasaiah, and Steven Skiena. Large-scale sentiment analysis for news and blogs. *Int. Conf. Weblogs and Social Media*, 7:21, 2007.

[18] Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, pages 625–635. ACM, 2015.

[19] Henk Tijms. *Understanding Probability.* Cambridge University Press, 2012.

[20] Dimitri Bertsekas and John Tsitsklis. *Introduction to Probability.* Athena Scientific, 2nd edition, 2008.

[21] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning* (Введение в статистическое обучение с примерами на языке R). Springer-Verlag, sixth edition, 2013.

[22] Charles Wheelan. *Naked Statistics: Stripping the dread from the data* (Голая статистика. Самая интересная книга о самой скучной науке). WW Norton & Company, 2013.

[23] Warren Weaver. *Lady Luck.* Dover Publications, 1982.

[24] Burton Gordon Malkiel. *A Random Walk Down Wall Street: Including a life-cycle guide to personal investing* (Случайная прогулка по Уолл-стрит). WW Norton & Company, 1999.

[25] Ronald Bracewell. *The Fourier Transform and its Applications.* McGraw-Hill, 3rd edition, 1999.

[26] E. Oran Brigham. *The Fast Fourier Transform.* Prentice-Hall, Englewood Cliffs NJ, facimile edition, 1988.

[27] William Press, Brian Flannery, Saul Teukolsky, and William T. Vetterling. *Numerical Recipes: The art of scientific computing.* Cambridge University Press, 3rd edition, 2007.

[28] J Robert Coleman, Dimitris Papamichail, Steven Skiena, Bruce Futcher, Eckard Wimmer, and Steffen Mueller. Virus attenuation by genome-scale changes in codon pair bias. *Science*, 320(5884):1784–1787, 2008.

[29] Steffen Mueller, J Robert Coleman, Dimitris Papamichail, Charles B Ward, Anjaruwee Nimnual, Bruce Futcher, Steven Skiena, and Eckard Wimmer.

Live attenuated influenza virus vaccines by computer-aided rational design. *Nature Biotechnology*, 28(7):723–726, 2010.

[30] Steven Skiena. Redesigning viral genomes (Перепрофилирование вирусных геномов). *Computer*, 45(3):0047–53, 2012.

[31] Ed Reingold and Nachum Dershowitz. *Calendrical Calculations: The Millennium Edition*. Cambridge University Press, New York, 2001.

[32] David Goldenberg. The biggest dinosaur in history may never have existed. FiveThirtyEight, `http://fivethirtyeight.com/features/the-biggest-dinosaur-in-history-may-never-have-existed/`, January 11, 2016.

[33] James Surowiecki. *The wisdom of crowds* (Мудрость толпы). Anchor, 2005.

[34] Charles Babbage. *Passages from the Life of a Philosopher*. Cambridge University Press, 2011.

[35] Sydney Padua. *The Thrilling Adventures of Lovelace and Babbage: The (mostly) true story of the first computer* (Невероятные приключения Лавлейс и Бэббиджа: (почти) правдивая история первого компьютера). Penguin, 2015.

[36] Joel Grus. Data *Science from Scratch: First principles with Python* (Наука о данных с нуля). O'Reilly Media, Inc., 2015.

[37] Wes McKinney. *Python for Data Analysis: Data wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc., 2012.

[38] James Gleick. A bug and a crash: sometimes a bug is more than a nuisance. *The New York Times Magazine*, December 1, 1996.

[39] Arthur G Stephenson, Daniel R Mulville, Frank H Bauer, Greg A Dukeman, Peter Norvig, Lia S LaPiana, Peter J Rutledge, David Folta, and Robert Sackheim. Mars climate orbiter mishap investigation board phase i report. *NASA, Washington, DC*, page 44, 1999.

[40] Matthew Faulkner, Robert Clayton, Thomas Heaton, K Mani Chandy, Monica Kohler, Julian Bunn, Richard Guy, Annie Liu, Michael Olson, MingHei Cheng, et al. Community sense and response systems: Your phone as quake detector. *Communications of the ACM*, 57(7):66–75, 2014.

[41] Slava Kisilevich, Milos Krstajic, Daniel Keim, Natalia Andrienko, and Gennady Andrienko. Event-based analysis of people's activities and behavior using flicker and panoramio geotagged photo collections. In *2010 14th International Conference Information Visualisation*, pages 289–296. IEEE, 2010.

[42] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 453–456. ACM, 2008.

[43] Yanqing Chen, Bryan Perozzi, and Steven Skiena. Vector-based similarity measurements for historical figures. In *International Conference on Similarity Search and Applications*, pages 179–190. Springer, 2015.

[44] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning future media environments*, pages 9–15. ACM, 2011.

[45] Karl M Kapp. *The Gamification of Learning and Instruction: Game-based methods and strategies for training and education.* Wiley, 2012.

[46] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[47] Michal Kosinski, David Stillwell, and Thore Graepel. Private traits and attributes are predictable from digital records of human behavior. *Proc. National Academy of Sciences*, 110(15):5802–5805, 2013.

[48] Amy Langville and Carl Meyer. *Who's #1? The Science of Rating and Ranking*. Princeton Univ. Press, 2012.

[49] Thorsten Joachims. Optimizing search engines using clickthrough data (Оптимизация поисковых систем посредством использования данных кликабельности). In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142. ACM, 2002.

[50] Peter Eades, X. Lin, and William F. Smyth. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, 47:319–323, 1993.

[51] Thayer Watkins. Arrow's impossibility theorem for aggregating individual preferences into social preferences. `http://www.sjsu.edu/faculty/watkins/arrow.htm`, 2016.

[52] Tyler Vigen. *Spurious Correlations* (Ложные корреляции). Hatchette Books, 2015.

[53] Gail M. Sullivan and Richard Feinn. Using effect size: or why the p value is not enough. *J. Graduate Medical Education*, 4:279282, 2012.

[54] David Freedman, Robert Pisani, and Roger Purves. *Statistics*. WW Norton & Co, New York, 2007.

[55] Darrell Huff. *How to Lie with Statistics* (Как лгать при помощи статистики). WW Norton & Company, 2010.

[56] David Donoho. 50 years of data science. Tukey Centennial Workshop, Princeton NJ, 2015.

[57] Vic Jennings, Bill Lloyd-Smith, and Duncan Ironmonger. Household size and the Poisson distribution. *J. Australian Population Association*, 16:65–84, 1999.

[58] Edward R Tufte. *The Visual Display of Quantitative Information* (Визуальное отображение количественной информации). Graphics Press, Cheshire, CT, 1983.

[59] Edward R Tufte. *Envisioning Information* (Представление информации). Graphics Press, Cheshire, CT, 1990.

[60] Edward R Tufte. *Visual Explanations*. Graphics Press, Cheshire, CT, 1997.

[61] Andrew Abela. *Advanced Presentations by Design: Creating Communication that Drives Action*. Pfeiffer, 2nd edition, 2013.

[62] Stephen Few. *Now You See It: simple visualization techniques for quantitative analysis*. Analytics Press, Oakland CA, 2009.

[63] Steven Johnson. *The Ghost Map: The story of London's most terrifying epidemic — and how it changed science, cities, and the modern world*. Riverhead Books, 2007.

[64] Francis J Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973.

[65] Levon Lloyd, Dimitrios Kechagias, and Steven Skiena. Lydia: A system for large-scale news analysis. In *SPIRE*, pages 161–166, 2005.

[66] Andrew Mehler, Yunfan Bao, Xin Li, Yue Wang, and Steven Skiena. Spatial Analysis of News Sources. In *IEEE Trans. Vis. Comput. Graph.*, volume 12, pages 765–772, 2006.

[67] Mikhail Bautin, Charles B Ward, Akshay Patil, and Steven S Skiena. Access: news and blog analysis for the social sciences. In *Proceedings of the 19th International Conference on World Wide Web*, pages 1229–1232. ACM, 2010.

[68] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 427–436. IEEE, 2015.

[69] Edward A Bender. *An Introduction to Mathematical Modeling*. Courier Corporation, 2012.

[70] Frank R. Giordano, William P. Fox, and Steven B. Horton. *A First Course in Mathematical Modeling*. Nelson Education, 2013.

[71] Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2009.

[72] David Lazer, Ryan Kennedy, Gary King, and Alessandro Vespignani. The parable of Google flu: traps in big data analysis. *Science*, 343(6176):1203–1205, 2014.

[73] George N Sazaklis, Esther M Arkin, Joseph SB Mitchell, and Steven S Skiena. Geometric decision trees for optical character recognition. In *Proceedings*

of the 13th Annual Symposium on Computational Geometry, pages 394–396. ACM, 1997.

[74] Vivek Kulkarni, Yingtao Tian, Parth Dandiwala, and Steven Skiena. Dating documents: A domain independent approach to predict year of authorship. Submitted for publication, 2017.

[75] David Lay, Steven Lay, and Judi McDonald. *Linear Algebra and its Applications*. Pearson, 5th edition, 2015.

[76] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2011.

[77] Alan Tucker. *A Unified Introduction to Linear Algebra: Models, methods, and theory*. Macmillan, 1988.

[78] Phillip Klein. *Coding the Matrix: Linear Algebra through Computer Science Applications*. Newtonian Press, 2013.

[79] Sanford Weisberg. *Applied linear regression*, volume 528. Wiley, 2005.

[80] David J. Leinweber. Stupid data miner tricks: overfitting the S&P 500. *The Journal of Investing*, 16(1):15–22, 2007.

[81] Serge Brin and Larry Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine (Анатомия системы крупномасштабного гипертекстового интернет-поиска). In *Proc. 7th Int. Conf. on World Wide Web (WWW)*, pages 107–117, 1998.

[82] Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, New York, 2nd edition, 2001

[83] Mark de Berg, Mark van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications* (Вычислительная геометрия. Алгоритмы и приложения). Springer, 2nd edition, 2000.

[84] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.

[85] H. Samet. Multidimensional spatial data structures. In D. Mehta and S. Sahni, editors, *Handbook of Data Structures and Applications*, pages 16:1–16:29. Chapman and Hall/CRC, 2005.

[86] Piotr Indyk. Nearest neighbors in high-dimensional spaces. In J. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 877–892. CRC Press, 2004.

[87] Doug West. *Introduction to Graph Theory*. Prentice-Hall, Englewood Cliffs NJ, second edition, 2000.

[88] David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning about a highly connected world* (Сети, толпы и рынок: рассуждения о взаимосвязанном мире). Cambridge University Press, 2010.

[89] Brian Everitt, Sabine Landau, Mmorven Leese, and Daniel Stahl. *Cluster Analysis*. Wiley, 5th edition, 2011.

[90] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[91] Christopher Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, New York, 2007.

[92] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning*. Springer, 2001.

[93] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning* (Глубокое обучение). MIT Press, 2016.

[94] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[95] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

[96] Bryan Perozzi, Rami al Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710. ACM, 2014.

[97] Mikhail Bautin, Lohit Vijayarenu, and Steven Skiena. International Sentiment Analysis for News and Blogs. In *Proceedings of the International Conference on Weblogs and Social Media*, Seattle, WA, April 2008.

[98] M. Borjesson, L. Serratosa, F. Carre, D. Corrado, J. Drezner, D. Dugmore, H. Heidbuchel, K. Mellwig, N. Panhuyzen-Goedkoop, M. Papadakis, H. Rasmusen, S. Sharma, E. Solberg, F. van Buuren, and A. Pelliccia. Consensus document regarding cardiovascular safety at sports arenas. *European Heart Journal*, 32:2119–2124, 2011.

[99] C. Gayomali. NYC taxi data blunder reveals which celebs don't tip and who frequents strip clubs. `http://www.fastcompany.com/3036573/`, October 2. 2014.

[100] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets* (Анализ больших наборов данных). Cambridge University Press, 2014.

[101] Benjamin Bengfort and Jenny Kim. *Data Analytics with Hadoop: An Introduction for Data Scientists*. O'Reilly Media, Inc., 2016.

[102] Sandy Ryza, Uri Laserson, Sean Owen, and Josh Wills. *Advanced Analytics with Spark: Patterns for Learning from Data at Scale*. O'Reilly Media, Inc., 2015.

[103] Cathy O'Neil. *Weapons of Math Destruction: How big data increases inequality and threatens democracy* (Убийственные большие данные. Как

математика превратилась в оружие массового поражения). Crown Publishing Group, 2016.

[104] Thomas H Davenport and DJ Patil. Data scientist. *Harvard Business Review*, 90(5):70–76, 2012.

[105] James Barron. Apple's new device looks like a winner. From 1988. *The New York Times*, January 28, 2010.

[106] Bartlett W Mel, Stephen M Omohundro, Arch D Robison, Steven S Skiena, Kurt H. Thearling, Luke T. Young, and Stephen Wolfram. Tablet: personal computer of the year 2000. *Communications of the ACM*, 31(6):638–648, 1988.

Предметный указатель

СЕРИЯ КНИГ О КОМПЬЮТЕРНЫХ НАУКАХ

Для того чтобы понять мир, необходимо собрать и проанализировать данные о нем. Объединение последних технологических тенденций предоставляет новые возможности для применения анализа данных к более сложным задачам, чем когда-либо прежде.

Емкость компьютерных хранилищ увеличивается экспоненциально; хранение данных сейчас стало настолько дешевым, что компьютерным системам почти невозможно ничего забыть. Сенсорные устройства все шире и шире контролируют все, за чем только можно наблюдать: потоки видео, действия в социальных сетях и местоположение всего, что перемещается. Сетевая вычислительная среда позволяет использовать огромные количества машин для манипулирования этими данными. Каждый раз, когда вы осуществляете поиск в Google, задействуются сотни компьютеров, тщательно исследующие все ваши предыдущие действия, только для того, чтобы решить, какая реклама является наилучшей для демонстрации именно вам.

Результатом всего этого стало рождение *науки о данных* — новой области, посвященной максимизации значения обширных коллекций информации. Как дисциплина, наука о данных находится где-то на пересечении статистики, информатики и машинного обучения, но стоит она отдельно, как самостоятельный персонаж. Эта книга служит введением в науку о данных, сосредоточиваясь на навыках и принципах, необходимых для построения систем, предназначенных для анализа и интерпретации данных.

ОБ АВТОРЕ

Стивен С. Скиена — ученый в области компьютерных наук и заслуженный профессор информатики в университете Стони Брукс. Он также является директором Института искусственного интеллекта в Стони Брук. В сферу его научного интереса входит анализ и разработка компьютерных алгоритмов и их применение в области биологии. Скиена написал несколько популярных книг в области алгоритмов, программирования и математики.

ДИАЛЕКТИКА
www.dialektika.com

Springer

ISBN 978-5-907144-74-3
19106
9 785907 144743