Содержание

ТОМ 1. ОСНОВЫ

Предисловие

Добро пожаловать!

Несколько слов введения в эту книгу,
как получить файлы и рисунки,
и благодарности тем, кто помогал мне

Что вы получите от этой книги

Привет!

Если вы интересуетесь глубоким обучением (Deep Learning, DL) и машинным обучением (Mashing Learning, ML), то эта книга для вас.

Моя цель в этой книге – дать вам прочные навыки эффективного практического применения машинного обучения и глубокого обучения.

После прочтения этой книги вы будете уметь:

- разрабатывать и обучать собственные нейронные сети;
- использовать нейронные сети для понимания данных и создания новых данных;
- присваивать описательные категории текстам, изображениям и другим типам данных;
- предсказывать последующие значения последовательности данных;
- исследовать структуру ваших данных;
- обрабатывать ваши данные с максимальной эффективностью;
- использовать языки программирования и библиотеку DL по своему желанию;
- воспринимать новые знания и идеи и применять их на практике;
- получать удовольствие от обсуждения глубокого обучения с другими специалистами.

Мы используем серьезный, но дружелюбный подход, сопровождаемый большим количеством иллюстраций. Мы делаем это без каких-либо кодов и без всякой математики, за исключением умножения.

Если это звучит привлекательно для вас, добро пожаловать!

Для кого эта книга

Эта книга создана для тех, кто хочет использовать машинное обучение и глубокое обучение в своей работе. Это программисты, инженеры, ученые, руководители, музыканты, врачи и все, кто хочет работать с большими объемами данных, извлекая из них полезную информацию или формируя новые данные.

Многие инструменты машинного обучения и особенно глубокого обучения имеются в многочисленных библиотеках со свободным доступом, которые любой при желании может немедленно загрузить.

Но хотя эти инструменты легко доступны и легко устанавливаемы, они все же требуют значительных технических знаний для правильного их применения. Совсем не трудно попросить компьютер выполнить что-то бессмысленное, и он радостно выполнит это, выдав на выходе бессмыслицу.

Такого рода вещи происходят все время. Хотя машинное обучение и глубокое обучение – мощный инструмент, он вовсе не слишком дружелюбен к пользователю.

Выбор правильных алгоритмов и затем применение их надлежащим образом требуют в дополнение последовательных, технически грамотных решений. Когда, как часто бывает, события развиваются не так, как планировалось, необходимы знания, чтобы понять, что происходит внутри системы, с тем чтобы исправить положение дел.

Существует множество подходов к освоению этих существенных знаний в зависимости от того, как вы хотите ими овладеть.

Некоторые любят жесткий детальный алгоритмический анализ, сопровождаемый обширной математикой. Если это тот способ, с помощью которого вы хотите овладеть знаниями, то существуют солидные труды, которые предлагают этот стиль представления знаний [Bishop06], [Goodfellow17]. Он требует серьезных усилий, но дает глубокое понимание того, как и почему этот механизм работает. Если вы выберете данный способ, то вам придется проделать значительную работу, чтобы применить теоретические знания на практике.

Другая крайность, когда человек просто хочет знать, как решить некую конкретную задачу. Серьезных книг, которые содержат библиотеки с рецептами машинного обучения, достаточно много [Chollet17], [Müller-Guido16], [Raschka15], [VanderPlus6]. Эти методы проще, чем математический подход, но вы можете ощущать недостаток структурной информации, объясняющей, как это работает. Без этой информации и соответствующей терминологии трудно разобраться, почему что-то, что, вы полагали, должно работать, не работает или почему что-то не работает так хорошо, как вы полагали. Это может вызвать желание почитать литературу, описывающую новые идеи и результаты, потому что обсуждения обычно предполагают общий уровень знаний о предмете, используемой библиотеке и языке.

В этой книге принят промежуточный подход. Я намерен дать вам инструменты для уверенного практического применения глубокого обучения. Я хочу, чтобы вы могли в вашей работе сделать разумный выбор и были способны следовать в общем потоке новейших идей, появляющихся почти каждый день.

Моя цель здесь – осветить фундаментальные основы достаточно глубоко, чтобы у вас была надежная база поддержки в вашей работе. Я хочу, чтобы вы имели достаточную базу не только для понимания материалов этой книги, но и материалов, которые, возможно, понадобятся вам для консультаций и изучения в процессе работы с глубоким обучением.

Это не книга о программировании. Программирование – важный аспект, но оно неизбежно вовлекает во все детали, которые необязательны для нашего основного предмета изучения. Примеры программирования ограничат нас одной библиотекой или одним языком программирования. И хотя детали необходимы для создания законченных систем, они могут отвлечь нас, когда будет необходимо сосредоточить внимание на основополагающих идеях. Вместо того чтобы вдаваться в дискуссии о циклах, индексах и структуре данных, мы будем все это

обсуждать здесь независимо от какого-либо языка или библиотеки. Если вы усвоите главную идею, то прочтение документации для любой библиотеки будет несложным делом.

Мы спустимся на землю в главах 15, 23 и 24, когда будем обсуждать научную библиотеку машинного обучения и библиотеку глубокого обучения Keras[1]. Обе эти библиотеки базируются на языке Python. В этих главах мы погружаемся в детали библиотеки языка программирования Python и имеем в них много примеров с кодами.

Даже если вы не знаете данного языка, эти программы послужат вам примером технологий и программных структур, что поможет справляться с новыми проблемами. Коды в этих главах с программированием доступны в качестве файлов на языке Python. Они могут быть использованы с помощью программного окружения, базирующейся на браузере графической веб-оболочки Jupyter [Jupyter16] или с помощью классического, разработанного для Python окружения, такого как PyCharm [JetBrain17].

Большинство других глав также может быть поддержано выборочными файлами (notebooks) на Python. В них приводится код для каждой компьютерной графики в книге, часто используя методы, обсуждаемые в этой главе. Поскольку в книге отсутствует ориентация на Python и программирование (за исключением упомянутых выше глав), эти записи присутствуют в книге как бы «за сценой» и только слегка комментируются.

Машинное обучение, глубокое обучение и большие данные оказывают неожиданно быстрое и глубокое влияние на общество во всем мире. Что это означает для людей и культуры – сложный и важный предмет. Ряд интересных книг и статей, обсуждающих эту тему, часто приходит к тонкой смеси положительных и отрицательных выводов [Agüera y Arcas17] [Barrat15] Domingos15] [Kaplan16].

Почти без математики

Многие, отнюдь неглупые люди не являются поклонниками сложных математических уравнений, и если это вы, то здесь вы дома.

В этой книге почти нет математики. Если вы справляетесь с умножением, то этого достаточно, поскольку это вся математика, которую мы используем.

Большинство обсуждаемых нами алгоритмов базируется на солидных теоретических источниках и является результатом аккуратного анализа и проработки. Важно знать это обстоятельство, когда вы модифицируете алгоритм для какой-то новой задачи или другой реализации. Но на практике почти все используют хорошо оптимизированные реализации с открытым кодом, написанные экспертами и доступные в бесплатных библиотеках.

Наша цель – понять принципы этих технологий, понять, как применять их надлежащим образом и как интерпретировать результаты. Ничто из этого не требует от нас вдаваться в математические подробности.

Если вы любите математику или хотите ознакомиться с теорией, следуйте ссылкам к каждой главе. Многие из этих материалов превосходны, интеллекту-

---

[1] См.: *Джулли А., Пал С.* Библиотека Keras – инструмент глубокого обучения. М.: ДМК Пресс, 2017. – *Прим. перев.*

ально полновесны и описывают детали, которые я сознательно опустил в этой книге. Но если математика не ваш конек, то и нет необходимости в математических деталях.

Много рисунков

Некоторые идеи более отчетливо проявляются в рисунках, чем в словах. И даже если слова выполняют свою работу, рисунки цементируют идеи. Поэтому эта книга обильно проиллюстрирована рисунками.

Все рисунки данной книги доступны для свободного считывания (см. далее).

Загрузки

Вы можете загрузить Jupiter/Python для этой книги, все рисунки и другие файлы, относящиеся к этой книге, совершенно свободно.

Все файлы (notebooks)

Все файлы оболочки Jupiter/Python в данной книге доступны на GitHub.

Файлы для главы 15 (scikit-learn – Python-модуль для машинного обучения) и глав 23 и 24 (Keras) содержат все коды, которые представлены в этих главах.

Другие файлы доступны в качестве «за сценой» для просмотра того, как получены рисунки. Они слегка документированы и служат скорее справкой, чем учебным пособием.

Файлы опубликованы по лицензии Массачусетского технологического института (MIT), что, по существу, означает, что вы свободны использовать их для любых целей. Никаких гарантий, что в них отсутствуют дефекты, что они будут надежно работать, что они не приведут к сбоям и т. п., не дается. Чувствуйте себя свободными извлекать коды и адаптировать их так, как вам удобно, хотя в лицензии говорится об использовании только в личных целях (это для файлов с пометкой LICENSE).

https://github.com/blueberrymusic/DeepLearningBookCode-Volume1
https://github.com/blueberrymusic/DeepLearningBookCode-Volume2

Все рисунки

Все рисунки в этой книге доступны на GitHub в формате PNG с высоким разрешением. Вы можете свободно использовать их в классах, обсуждениях, лекциях, докладах, статьях и даже других книгах.

Так же, как код, рисунки опубликованы по лицензии Массачусетского технологического института (MIT), и вы можете использовать их по своему усмотрению, сохраняя уведомление об авторских правах. Используя рисунки, вы не обязаны кредитовать меня как их создателя, но я буду признателен вам, если вы сделаете это.

Наименования файлов соответствуют номерам рисунков в книге, поэтому их нетрудно найти. Когда вы будете искать что-либо визуально, то будет полезно по-

смотреть страницы в уменьшенном формате. Каждая такая страница содержит 20 изображений:

https://github.com/blueberrymusic/DeepLearningBookFigures-Thumbnails

Сами рисунки сгруппированы в двух томах:

https://github.com/blueberrymusic/DeepLearningBookFigures-Volume1
https://github.com/blueberrymusic/DeepLearningBookFigures-Volume2

Источники

Перечень литературы и ресурсов содержит другие файлы, такие как шаблоны для иконок глубокого обучения, которые используются нами в этой книге.

https://github.com/blueberrymusic/DeepLearningBook-Resources

Опечатки

Хотя мной приняты все возможные меры, в книге такого объема трудно избежать ошибок. Если вы обнаружите что-то, что покажется вам неверным, пожалуйста, дайте мне знать по адресу andrew@dlbasics.com.

Два тома

Книга оказалась очень большой, поэтому я сделал ее двухтомником с примерно одинаковым размером.

Поскольку двухтомник является, по существу, одной книгой, второй том начинается там, где заканчивается первый. Если вы в данный момент читаете второй том, то, следовательно, вы или уже прочитали первый, или чувствуете себя достаточно уверенно для понимания содержания второго тома.

Благодарности

Авторы любят говорить, что никто не пишет книги в одиночку. Мы говорим так, потому что так оно и есть.

Я чрезвычайно благодарен Эрику Брауну (Eric Braun), Эрику Хейнесу (Eric Haines), Стиву Друкеру (Steven Drucker) и Тому Рейке (Tom Reike) за последовательную и воодушевляющую помощь в этом проекте, помогавшую мне уверенно чувствовать себя на протяжении всего времени осуществления проекта. Спасибо за ваше дружелюбие и поддержку.

Большое спасибо моим рецензентам за щедрые и глубокие комментарии, значительно улучшившие книгу: Адаму Финкельштейну (Adam Finkelstein), Алексу Колберну (Alex Colburn), Александру Келлеру (Alexander Keller), Алин Рокфорд (Alyn Rockwood), Анджело Песке (Angelo Pesce), Барбаре Монес, (Barbara Mones), Брайану Уайвиллу (Brian Wyvill) , Крейгу Каплану (Craig Kaplan), Дагу Робле, (Doug Roble), Эрику Брауну (Eric Braun), Эрику Хейнесу (Eric Haines), Грегу Тэрку (Greg Turk), Джеффу Халтвисту (Jeff Hultquist), Джессике Ходжинс, (Jessica Hodgins) Кри-

сти Мортон (Kristi Morton), Лезли Истед (Lesley Istead), Луис Авардо (Luis Avarado), Матт Фарр (Matt Pharr), Майку Тика (Mike Tyka), Морган МакГвире (Morgan McGuire), Паулю Бэдсли (Paul Beardsley), Паулю Страуссу (Paul Strauss), Петеру Шэрли (Peter Shirley), Филиппу Слузаллеку (Philipp Slusallek), Сербан Порумбеску (Serban Porumbescu), Стефанусу Ду Тойту (Stefanus Du Toit), Стивен Друкер (Steven Drucker), Венхао Ю (Wenhao Yu) и Закори Эриксон (Zackory Erickson).

Особая благодарность ответственным рецензентам Александру Келлеру (Alexander Keller), Эрику Хейнесу (Eric Haines), Джессике Ходжинс, (Jessica Hodgins) и Луис Авардо (Luis Avarado), которые прочитали всю книгу или большую часть рукописи и дали существенные предложения как по представлению содержания, так и по структуре книги.

Благодарю Морган МакГвире (Morgan McGuire) за применение технологии Markdeep, что позволило мне в большей степени сосредоточиться на том, что я должен сказать, чем на том, каким должен быть формат.

Спасибо Тодду Жиманскому (Todd Szymanski) за вдумчивые советы по оформлению и компоновке содержания и титула книги и выявление ошибок в их размещении.

Спасибо первым читателям, которые выявили опечатки и другие проблемы:

Кристиану Форгангу (Christian Forfang), Дэвиду Полу (David Pol), Эрику Хейнесу (Eric Haines), Гопи Меенакшисундараму (Gopi Meenakshisundaram), Косте Смоленскому (Kostya Smolenskiy), Мурисио Вивес, (Mauricio Vives), Майку Вонгу (Mike Wong) и Мринал Мохит (Mrinal Mohit).

Все эти люди улучшили книгу, но окончательные решения были за мной. И проблемы, которые остались, являются моей ответственностью.

Справочные материалы

Этот раздел появляется в каждой главе. Он содержит справки по всем документам, которые относятся к содержанию данной главы. Здесь также могут присутствовать другие полезные материалы: статьи, веб-страницы, документация, блоги и иные источники.

Где только возможно, я предпочитал те источники, которые доступны онлайн, поэтому вы можете немедленно получить их, используя Сеть. Исключение составляют обычно книги, но иногда я привожу важные онлайн-справки, даже если за ними стоят требования оплаты.

[Agüera y Arcas17] *Blaise Agüera y Arcas, Margaret Mitchell, and Alexander Todorov*. Physiognomy's New Clothes. Medium, 2017. https://medium.com/@blaisea/physiognomys-new-clothes-f2d4b59fdd6a.

[Barrat15] *James Barrat*. Our Final Invention: Artificial Intelligence and the End of the Human Era. St. Martin's Griffin, 2015.

[Bishop06] *Christopher M. Bishop*. Pattern Recognition and Machine Learning. Springer-Verlag, 2006. С. 149–152.

[Chollet17] *François Chollet*. Deep Learning with Python. Manning Publications, 2017.

[Domingos15] *Pedro Domingos*. The Master Algorithm. Basic Books, 2015.

[Goodfellow17] *Ian Goodfellow, Yoshua Bengio, Aaron Courville*. Deep Learning. MIT Press, 2017. http://www.deeplearningbook.org/[1].

[JetBrains17] *Jet Brains*. Pycharm Community Edition IDE. 2017. https://www.jetbrains.com/pycharm/.

[Jupyter16] The Jupyter team. 2016. http://jupyter.org/.

[Kaplan16] *Jerry Kaplan*. Artificial Intelligence: What Everyone Needs to Know. Oxford University Press, 2016.

[Müller-Guido16] *Andreas C. Müller and Sarah Guido*. Introduction to Machine Learning with Python. O'Reilly Press, 2016[2].

[Raschka15] *Sebastian Raschka*. Python Machine Learning. Packt Publishing, 2015[3].

[VanderPlas16] *Jake VanderPlas*. Python Data Science Handbook. O'Reilly Media, 2016.

[1] *Гудфеллоу Я., Бенджио И., Курвилль А.* Глубокое обучение. М.: ДМК Пресс, 2017. ISBN: 978-5-97060-618-6.

[2] *Мюллер А., Гвидо С.* Введение в машинное обучение с помощью Python. М.: Вильямс, 2017. ISBN: 978-5-9908910-8-1.

[3] *Рашка С.* Python и машинное обучение. М.: ДМК Пресс, 2017. ISBN: 978-5-97060-409-0.

Глава 1

Введение в машинное обучение и глубокое обучение

Краткий обзор идей, языка, технологий, которые мы будем использовать в этой книге

1.1. Зачем здесь эта глава

Эта глава позволит нам поближе познакомиться с главными идеями и основной терминологией машинного обучения.

Термин **машинное обучение** охватывает большой круг технологий, которые преследуют одну цель: извлечь имеющую смысл информацию из данных.

Под данными здесь подразумевается все, что может быть записано и измерено (курс акций в удачные дни, или масса других планет, или рост людей, посещающих местную ярмарку, но это могут быть и звуки (слова, произносимые кем-нибудь в его мобильный телефон), картинки (фотографии цветов или кошек), слова (текст газетной статьи или романа), или все, что мы хотели бы изучить или исследовать.

«Имеющая смысл информация» – это все, что мы можем извлечь из данных и что будет полезным для нас в некотором смысле. Мы решаем, что имеет смысл для нас, и затем создаем алгоритм, чтобы найти столь много из наших данных, насколько это возможно.

Выражением «машинное обучение» описывают множество различных алгоритмов и технологий. Было бы неплохо дать строгое определение для этого термина, но им пользуется так много людей, в столь многих различных направлениях, что лучше рассматривать его как имя большого расширенного собрания алгоритмов и принципов, которые позволяют анализировать огромное количество опытных данных для извлечения из них смысловой информации.

Совсем недавно выражение **глубокое обучение** относилось к методам машинного обучения, которые использовали специализированные слои вычислений, следующие один за другим. Это создавало «глубокую» структуру, подобную стопке блинов. Поскольку термин «глубокое обучение» относится к природе создаваемых нами систем, а не какому-либо конкретному алгоритму, он действительно

соответствует такому особому стилю или методу компьютерного обучения. Этот метод в последние годы был значительно усовершенствован.

Рассмотрим некоторые примеры приложений, которые используют машинное обучение для извлечения значащей информации из данных.

1.1.1. Извлечение значащей информации из данных

Почтовому отделению требуется ежедневно сортировать большое количество писем и посылок с написанными от руки почтовыми индексами. Компьютер научили читать эти индексы и автоматически отправлять почту по адресу, как показано на рис. 1.1.

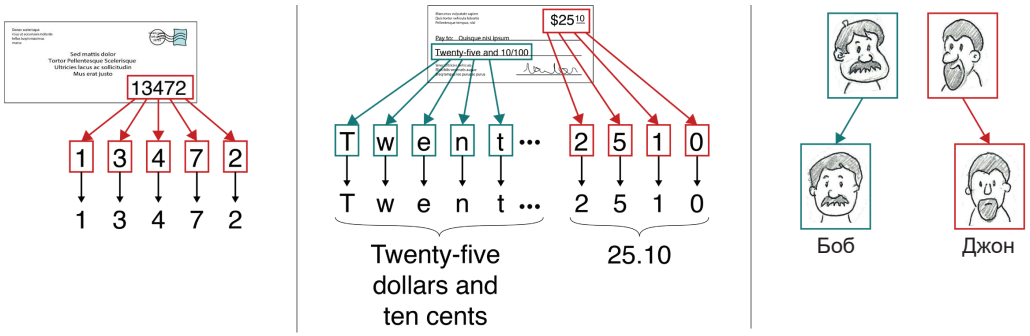

Рис. 1.1 ❖ Извлечение смысловой информации из множества данных. Слева: получение почтового индекса с конверта. В середине: считывание чисел и букв с чека. Справа: распознавание лиц по фотографии

Банки обрабатывают огромное количество рукописных чеков. Надежная проверка требует, чтобы числовая сумма, записанная вручную в соответствующее поле (например, «25.10»), соответствовала выписанной сумме в текстовой строке, например «Twenty-five dollars and ten cents» («Двадцать пять долларов и десять центов»). Компьютер может прочитать оба числа и слова и подтвердить соответствие, как это показано в середине рис. 1.1.

Сайты социальных сетей хотят идентифицировать людей по фотографиям участников сети. Это означает не только обнаружение лиц на данной фотографии, но и определение того, где расположены эти лица, и затем сопоставление каждого лица с ранее наблюдавшимися лицами. Это становится еще более трудным при понимании того, что фактически каждая фотография человека уникальна: освещение, угол зрения, выражение, одежда и много других характеристик будет различаться от любого предыдущего снимка. Они хотели бы быть способными, взяв любое фото любого человека, корректно идентифицировать его, как показано на рис. 1.1.

Производители интеллектуальных цифровых помощников исследуют то, что говорят люди при использовании своих гаджетов, чтобы помощники могли разумно реагировать на речь. Сигнал от микрофона представляет собой последовательность цифр, которые описывают действующее на мембрану микрофона звуковое давление. Поставщики хотят проанализировать эти числа, чтобы понять звуки, вызвавшие их, слова, в которых эти звуки были частью, предложения, в ко-

торые были включены слова, и в конечном счете смысл этих предложений, как это показано в левой части рис. 1.2.

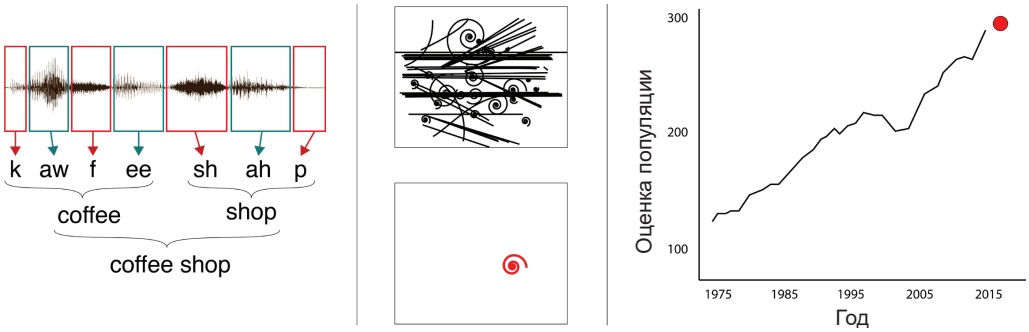

Рис. 1.2 ❖ Извлечение смысловой информации из данных. Слева: превращение записи в звуки, затем в слова и, наконец, в высказывание. В середине: нахождение характерной частицы в смазанном изображении с подобными частицами. Справа: предсказание популяции касаток, обитателей северных районов, у западных берегов Канады [Towers15]

Учеными создается огромное количество данных, поступающих от беспилотных летательных аппаратов, или как результат экспериментов по физике высоких энергий, или в качестве наблюдений за неземным пространством. Из этих переполненных потоков данных часто приходится выделять несколько отдельных событий, которые очень похожи на все остальные, но немного другие. Проанализировать все данные вручную было бы фактически невозможной задачей даже для огромного количества высококвалифицированных специалистов. Гораздо лучше автоматизировать этот процесс с помощью компьютеров, которые могут исчерпывающе разделить данные, никогда не упуская никаких подробностей, как, например, в середине рис. 1.2.

Консервисты отслеживают изменения с течением времени популяции видов животных, чтобы знать, что происходит с популяцией. Если она сокращается в течение длительного периода времени, возможно, требуется вмешательство. Если популяция стабильна или увеличивается, то нет причин для беспокойства. Прогнозирование последующих значений размеров популяции – это то, чему мы можем обучить компьютер, и он будет делать это весьма неплохо. Ежегодно регистрируемый размер популяции китов касаток у побережья Канады, а также его предсказанное возможное значение показаны справа на рис. 1.2 (адаптировано из [Towers15]).

Эти шесть примеров иллюстрируют приложения, которые многим из нас уже знакомы, но таких приложений гораздо больше. Возможность с их помощью быстро извлекать значимую информацию способствует тому, что алгоритмы машинного обучения находят свой путь во все расширяющемся диапазоне областей применения.

Общее место здесь – это объем работы и кропотливое рассмотрение деталей. У нас могут быть миллионы данных для исследования, и мы должны попытаться извлечь какую-то значимую информацию из них. Люди устают, скучают и отвлекаются, но компьютеры будут просто пахать неуклонно и надежно.

1.1.2. Экспертные системы

Первоначальный, ставший популярным подход к поиску смысловой информации, скрытой внутри данных, связан с созданием экспертных систем. Суть идеи заключалась в изучении того, что знают специалисты-эксперты, что они делают, как они это делают, а затем автоматизации этого. По сути, создается компьютерная система, основанная на знаниях экспертов, которая способна имитировать их действия.

Это часто означает разработку **системы, основанной на правилах**, в которой сосредоточено большое количество правил, следуя которым, компьютер имитирует специалиста-эксперта. Например, если мы пытаемся распознать цифры почтового индекса, мы можем создать правило, в котором говорится, что семерка имеет форму преимущественно горизонтальной линии вверху изображения, затем преимущественно диагональную линию, которая начинается от правого края горизонтальной линии и продолжается влево и вниз, как это показано на рис. 1.3.

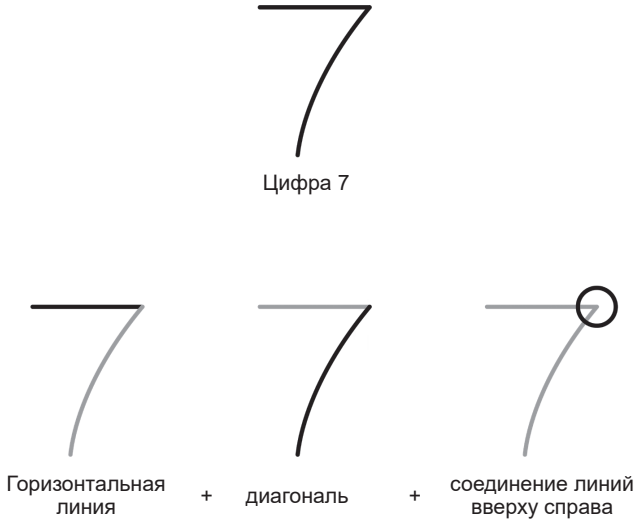

Рис. 1.3 ❖ Составление множества правил для распознавания написанной от руки цифры 7. Вверху: типичная цифра 7, которую мы хотим распознать. Внизу: три правила, создающих цифру 7. Изображение классифицируется как цифра 7, если все три правила соблюдены

Мы можем составить подобные правила для любой цифры. Это могло бы нас удовлетворить, если бы мы не имели цифры 7, изображенной на рис. 1.4. Некоторые люди ставят горизонтальную черту в середине цифры 7. Поэтому мы добавляем еще одно правило для такого специального случая.

Этот процесс ручного составления правил для распознавания иногда называют **конструированием признаков** (термин также применяется при использовании компьютера для поиска этих признаков [VanderPlas16]).

Данный термин характеризует наше желание выявить (спроектировать) все признаки (или качества), которые объединяются и используются специалистом-экспертом. Вообще-то, это трудная работа. Как мы видели, нетрудно найти одно

правило и даже много таковых. Но представьте попытку нахождения множества правил, по которым рентгенолог определяет, является ли смазанное пятно на рентгеновском снимке доброкачественной опухолью или нет, или правила, которые позволяют авиационному диспетчеру справляться с напряженным графиком полетов, или по которым водитель безопасно ведет машину в сложных погодных условиях.

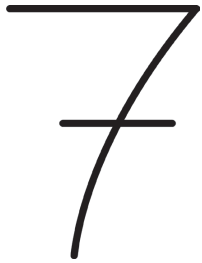

Рис. 1.4 ❖ Эта цифра 7 тоже может существовать, но не будет распознана с правилами на рис. 1.3 из-за дополнительной линии

Экспертные системы, основанные на правилах, способны справляться с некоторыми видами работ, но трудности составления качественного множества правил и необходимость обеспечения надежности их работы при наличии большого разнообразия данных определили невозможность их восприятия как общего решения проблемы. Согласование каждого шага в сложном процессе чрезвычайно затруднительно, а при необходимости также учитывать человеческий фактор при суждениях об опыте и интуиции это становится практически невозможным, за исключением самых простых сценариев.

Привлекательность систем машинного обучения состоит в том, что (на концептуальном уровне) они изучают релевантные характеристики множества данных *автоматически*. Поэтому нам не надо говорить им, как отличить цифру 2 от цифры 7, потому что система разбирается с этим сама. Но, чтобы сделать это хорошо, системе зачастую требуется большое количество данных. Огромное количество данных.

Все это является серьезной причиной того, что машинное обучение пользуется популярностью и реализуется на практике в последние годы. Поток необработанных данных, который обеспечивает интернет, позволяет извлечь множество полезной информации из множества данных. Онлайновые компании способны делать полезным каждое взаимодействие с каждым потребителем, собирая много данных, которые они затем могут обобщить и использовать как входную информацию для алгоритмов машинного обучения, получая тем самым все больше информации о своих потребителях и их пристрастиях.

1.2. Изучение маркированных данных

Существует много алгоритмов машинного обучения, и мы будем рассматривать многие из них в этой книге. Многие концептуально прямолинейны (хотя их математика и программирование могут быть сложными). Например, предположим,

что мы хотим найти наилучшую прямую линию, проходящую через группу данных точек, как на рис. 1.5.

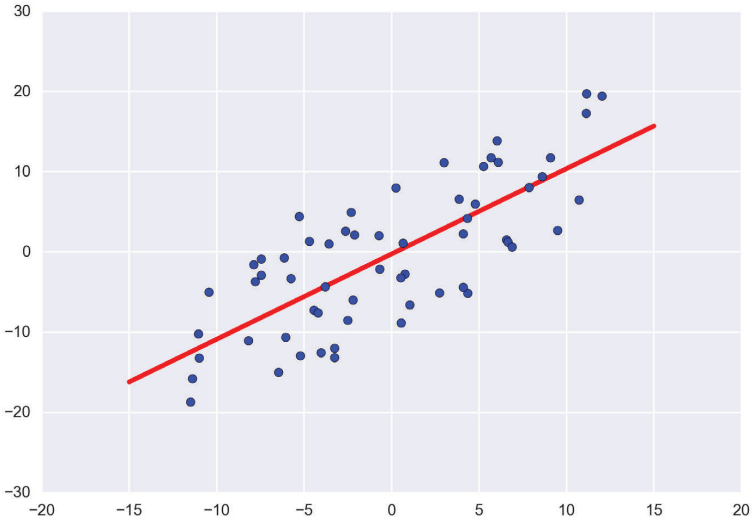

Рис. 1.5 ❖ Имея заданную группу точек (синего цвета), мы можем очевидным образом представить алгоритм, который вычислит наилучшую прямую линию (красного цвета), проходящую через эту группу точек

Концептуально мы можем представить себе алгоритм, который найдет прямую линию, пересекающую только небольшое число точек. Он будет работать в соответствии с некой формулой, чтобы подсчитать то количество точек, заданное число которых будет представлено на входе. Это знакомый алгоритм, который использует тщательно продуманный анализ, чтобы найти лучший способ решить проблему, и затем реализуется в программе, которая выполняет этот анализ. Это стратегия, используемая многими алгоритмами машинного обучения.

Напротив, стратегия, используемая многими алгоритмами глубокого обучения, менее знакома. Она предполагает медленное обучение на примерах, перебирая их снова и снова. Каждый раз, когда программа обнаруживает новые данные для обучения, она улучшает собственные параметры, найдя, в конце концов, набор значений, который обеспечит вычисления с желаемым нам результатом. В процессе выполнения алгоритма его работа значительно более непредсказуема, чем та, при которой ищется прямая линия. Идея здесь в том, что мы не знаем, как очевидным образом получить правильный ответ, и поэтому строим систему, которая сама делает это. Наш анализ и программирование создают алгоритм, который может выработать свои собственные ответы, вместо того чтобы реализовывать *известный* процесс, ведущий прямо к ответу.

Это действительно звучит немного дико, но это так. Программы, которые могут находить собственные ответы подобным путем, лежат в основе недавнего большого успеха алгоритмов глубокого обучения.

В нескольких последующих разделах мы рассмотрим более подробно эту технику, чтобы лучше понять ее, поскольку она, возможно, менее знакома, чем тради-

ционные алгоритмы машинного обучения. В пределе нам хотелось бы использовать эту технику для таких задач, как получение в качестве ответа системы имен всех, кто изображен на представленной системе фотографии.

Это серьезная задача, поэтому давайте начнем с чего-то значительно более простого и посмотрим на некоторые факторы обучения.

1.2.1. Стратегия обучения

Давайте рассмотрим один ужасный способ обучения детей новому предмету, обобщенный на рис. 1.6. Это совсем не так, как учат большинство детей на самом деле, но один из способов обучения компьютеров.

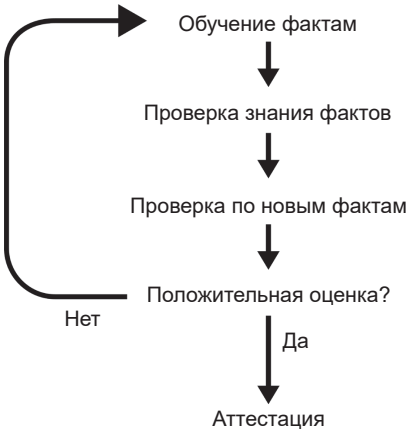

Рис. 1.6 ❖ Действительно ужасный способ пытаться так учить человека. Сначала составляется перечень фактов. Затем каждый студент проверяется на знание этих фактов и других фактов, которые отсутствовали в перечне, но которые, мы полагаем, могут быть извлечены из предъявленного перечня, если перечень был достаточно хорошо усвоен. Если студент получает хорошую оценку по этим тестам (особенно по второму), он аттестуется. В противном случае он возвращается к повторному изучению того же перечня фактов

В этом сценарии (надеюсь, воображаемом) преподаватель стоит перед классом и диктует перечень фактов, который учащиеся, предполагается, запоминают. Каждую пятницу в полдень они сдают два теста. Первый тест проверяет запоминание ими фактов из перечня. Второй тест проводится сразу после первого, при котором задаются вопросы, о которых учащийся заранее не знал, с тем чтобы проверить общее понимание им материала. Конечно, маловероятно, что кто-то будет «понимать» что-либо, если ему будет дан только перечень фактов, что является одной из причин, по которой такой подход ужасен.

Если учащийся получает положительную оценку по второму тесту, то он аттестуется.

Если данный учащийся не сдает второй тест, он на следующей неделе повторяет весь процесс: преподаватель повторяет факты из того же перечня, затем вновь проверяет таких учащихся по первому перечню на запоминание и потом по второму тесту на понимание или способность обобщать. Снова и снова каж-

дый учащийся повторяет этот процесс, пока не пройдет второй тест, чтобы получить аттестацию.

Это ужасный способ обучения для детей, но он оказывается отличным способом обучать компьютеры.

В этой книге мы увидим много других подходов к обучению компьютеров, но ограничимся пока этим и посмотрим на него более внимательно.

Мы увидим, что не в пример многим людям каждый раз, когда мы будем предъявлять компьютеру одну и ту же информацию, он будет обучаться немного больше.

1.2.2. Стратегия компьютерного обучения

Мы начнем с накопления фактов, которым собираемся обучить. Мы будем набирать так много данных, сколько только сможем. Каждую группу наблюдаемых данных (скажем, погоды в данный момент) называют **выборкой**, а имена измерений, которые осуществлялись (температура, скорость ветра, влажность и т. д.), называют **признаками**, или **характеристиками** [Bishop06]. Каждое имя измерения или признак ассоциируется с величиной, обычно запоминаемой как число. Чтобы подготовить наши данные для компьютера, мы предъявляем каждую выборку (то есть группу данных с величинами всех признаков) эксперту, который изучает эти признаки и осуществляет **маркировку** этой выборки. Например, если наша выборка является фотографией, маркировкой может быть имя человека или тип животного, показанные на ней, или, скажем, присутствует или нет движение транспорта на фотографии или транспорт на ней стоит в пробке.

Давайте используем в нашем примере изменения погоды в горах. Мнение эксперта, применяя шкалу от 0 до 100, говорит о его уверенности, что погода благоприятствует прогулке. Идея показана на рис. 1.7.

Мы будем обычно использовать эти маркированные выборки, но сейчас на время отложим их.

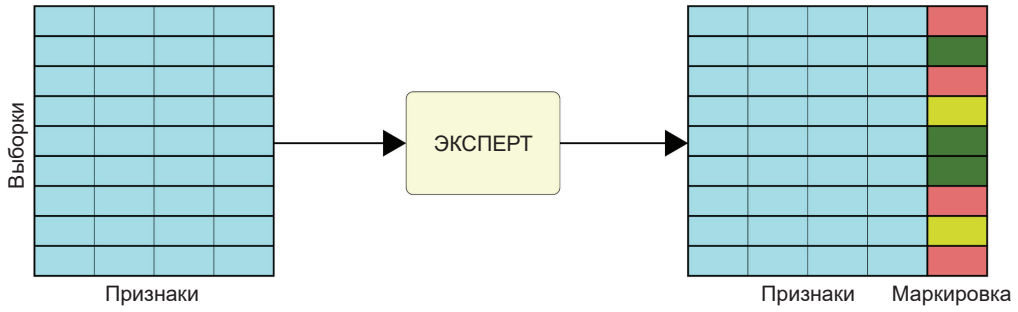

Рис. 1.7 ❖ Чтобы маркировать множество данных, мы начинаем с перечня выборок или элементов данных. Каждая выборка имеет перечень признаков, которые описывают ее. Мы даем это множество данных эксперту, который исследует признаки каждой выборки одной за другой и маркирует каждую выборку

Когда мы получим маркированные выборки, мы вводим их в компьютер и говорим ему, что надо найти способ уточнить правильность маркировки для каждого входа. Мы *не говорим* ему, как это делать. Вместо этого мы даем алгоритм

с большим количеством параметров (возможно, даже миллион), которые он может уточнять.

Различные способы обучения будут использовать различные алгоритмы, и многое в этой книге будет посвящено их рассмотрению и тому, как их успешно использовать. Когда мы выбрали алгоритм, мы можем загрузить данные на его вход, чтобы получить результат на выходе. Этот результат будет являться компьютерным **предсказанием** того, что компьютер думает об экспертной маркировке для этой выборки.

Если предсказание компьютера согласуется с экспертной маркировкой, мы не будем ничего менять. Но когда компьютер ошибется, мы попросим его модифицировать внутренние, используемые им параметры алгоритма, так чтобы при представлении ему этих данных вновь он лучше предсказал правильные ответы.

В основе такого процесса лежит метод проб и ошибок. Компьютер делает все возможное, чтобы дать нам правильный ответ, и если снова ошибается, то существует процедура, которой он может следовать и которая помогает ему изменять и улучшать ответ.

Мы только проверяем предсказания компьютера относительно маркировки эксперта, после того как оно будет получено. Если оно не согласуется, подсчитываем **ошибку**, называемую также **ценой**, или **потерями**. Это величина, которая говорит алгоритму, насколько он хорош. Система использует текущие значения своих внутренних параметров, экспертное предсказание (которое мы знаем) и свое собственное неверное предсказание, чтобы уточнить параметры алгоритма таким образом, чтобы предсказать более правильную маркировку при новом предъявлении ему той же выборки. Позже мы посмотрим более подробно, как осуществляются такие шаги. На рис. 1.8 проиллюстрирована эта идея.

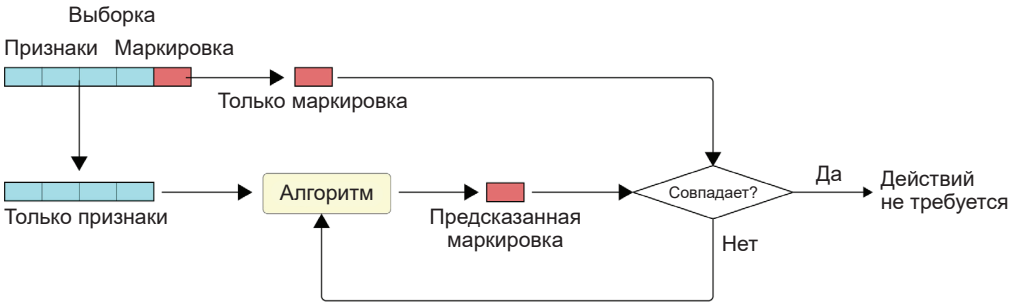

Рис. 1.8 ❖ Один шаг тренинга, или обучения. Мы разделяем признаки и их маркировку. По признакам алгоритм предсказывает маркировку. Мы сравниваем предсказание с реальной маркировкой. Если предсказанная маркировка согласуется с той, которую мы хотим, то ничего не предпринимаем. В противном случае мы говорим алгоритму модифицироваться или обновиться таким образом, чтобы в меньшей степени повторить эту ошибку вновь

Мы говорим, что **тренируем** систему, чтобы **обучить**, как **предсказывать** маркированные данные посредством анализа выборок в результате **последовательных тренировок** и **обновления** алгоритма при неправильном предсказании.

Мы посмотрим на другие возможные алгоритмы и шаги обновления в деталях в последующих главах. Сейчас важно понять, что каждый алгоритм обучает по-

средством изменения своих внутренних параметров, чтобы сделать предсказание. Он может значительно изменить их после неверного предсказания, но есть опасность, что он изменит их так сильно, что сделает предсказание еще хуже. Он может менять их в небольших пределах, но будет работать при этом медленнее, чем мог бы. Правильное соотношение между этими крайностями – это то, что мы должны найти с помощью проб и ошибок для каждого типа алгоритма и каждого множества данных в результате тренинга. Мы назовем количество обновлений **скоростью обучения**. Маленькая скорость обучения медленная, но осторожная, в то время как большая скорость обучения хотя и ускоряет процесс, но может привести к ошибкам.

В качестве аналогии предположим, что мы находимся в пустыне и используем металлоискатель для поиска закопанного ящика с провиантом. Мы водим датчиком металлоискателя над поверхностью, получаем отклик в каком-то направлении и продвигаемся в этом направлении. Если мы будем осторожны, мы будем двигаться малыми шажками, так, чтобы не пройти мимо ящика и не потерять сигнал. Если мы будем излишне активны, то будем двигаться большими шагами, чтобы найти ящик как можно быстрее. И только если мы, начав двигаться большими шагами, будем делать их все меньше и меньше при приближении к ящику, мы сможем найти его. Таким же образом обычно регулируется скорость обучения: последовательность изменений большая вначале, затем в процессе обучения уменьшается.

Есть занятный способ получения компьютером отличной оценки без всякого обучения – ничего не делать, а просто запомнить входы. Чтобы получить идеальный результат, все, что нужно алгоритму, – это просто запомнить маркировку эксперта для каждой выборки, а затем выдать нам эту маркировку. Другими словами, нет необходимости обучать как-то вычислить маркировку для данной выборки. Просто надо посмотреть правильный ответ в таблице. В нашем воображаемом сценарии обучения учащихся это эквивалентно запоминанию учащимся ответов на все вопросы в тестах.

Иногда это хорошая стратегия, и мы увидим далее, что существуют полезные алгоритмы, следующие именно такой стратегии. Но если наша цель – научить компьютер узнать что-либо о данных, что позволило бы обобщить это для получения новых данных, такой способ, как правило, неприемлем. Проблема в том, что мы уже имеем маркировки, запомненные системой, так что наша работа по ее тренировке не даст нам ничего нового, поскольку компьютер ничего не узнает из этих данных, а просто возьмет ответы, заглянув в таблицу, и у него не будет идеи, как выработать предсказание для новых данных, которые ему были еще неизвестны, и запомнить их. Задача в том, чтобы получить систему, способную прогнозировать маркировки для новых данных, которые ей не были известны раньше, чтобы мы могли затем уверенно использовать их в системе, на вход которой новые данные для предсказания будут поступать постоянно.

Если алгоритм хорошо работает с обучающим множеством, но плохо работает с новыми данными, мы будем говорить, что он плохо **обобщает**.

1.2.3. Обобщение

Теперь вернемся к маркировке данных, о которой мы не стали рассуждать в предыдущем разделе.

Мы дадим оценку того, как хорошо система может обобщать то, чему она научилась, показывая ей выборки, которых она еще не видела. Этот тестовый набор скажет нам, как хорошо система работает с новыми данными.

Давайте посмотрим на **классификатор**. Такая система присваивает маркировку каждой выборке, которая описывает, к какой из нескольких категорий, или классов, эта выборка принадлежит. Если выборка на входе представляет собой песню, то маркировка может быть жанром (например, рок или классика). Если это фотография животного, то маркировка может говорить, какое это животное (например, тигр или слон). В нашем примере мы можем маркировать каждый день как подходящий для прогулки на три категории: плохой, хороший, превосходный.

Мы попросим компьютер предсказать маркировку для каждой выборки из тестового множества (выборок, которые ему не были представлены ранее), тестового набора данных и затем сравним предсказания компьютера и маркировку эксперта, как это показано на рис. 1.9.

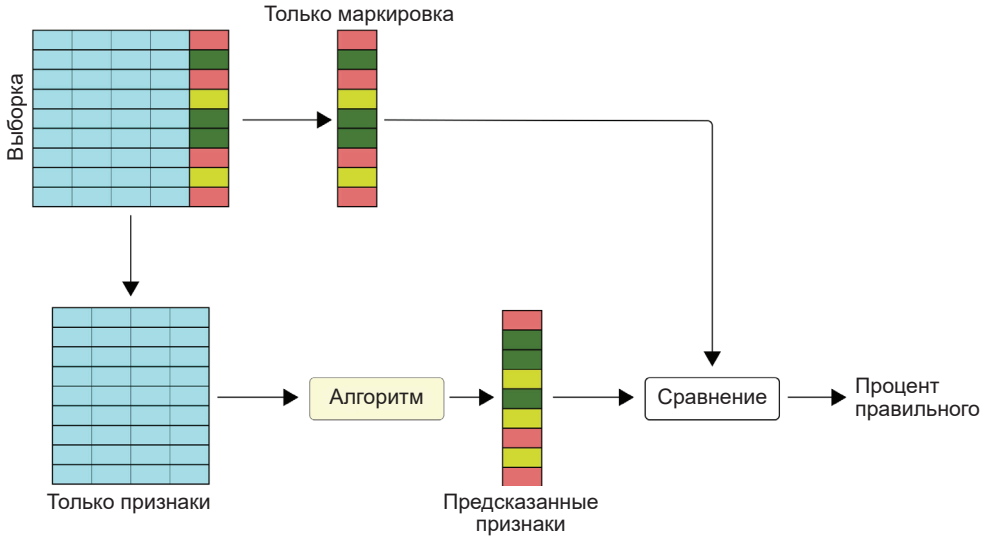

Рис. 1.9 ❖ Полный процесс оценки классификатора

На рис. 1.9 мы разделили тестовые данные на признаки и маркировку. Алгоритм присваивает или предсказывает маркировку для каждого множества признаков, затем мы сравниваем предсказание с реальной маркировкой, чтобы оценить правильность. Если она достаточно хороша, мы можем внедрять систему. Если результат не слишком хорош, мы можем вернуться и продолжить тренинг. Заметьте, что, в отличие от процесса тренинга, в процессе на рис. 1.9 отсутствуют обратная связь и обучение. Пока мы не вернемся к улучшающему тренингу, алгоритм не изменит своих параметров независимо от качества предсказания.

Если предсказания компьютера с этой новой выборкой (точнее, новой для компьютера) недостаточно хорошо согласуются с маркировкой эксперта, то мы возвращаемся к процессу, изображенному на рис. 1.8. С каждым шагом мы

предъявляем компьютеру каждую выборку обучающего множества снова, давая ему возможность опять изучить ее таким же образом, как и раньше. Подчеркнем, что это те же выборки, то есть мы просим компьютер изучить снова и снова те же самые данные. Обычно их сначала **перемешивают** так, что выборки поступают каждый раз в различной последовательности, но алгоритм при этом новой информации не получает. Затем мы просим алгоритм снова предсказать маркировку для тестового множества. Если она и на этот раз недостаточно хорошая, то мы проводим обучение с оригинальным выборочным множеством снова и затем снова проводим тестирование. Мы делаем это снова и снова, повторяя процесс и предъявляя компьютеру снова и снова те же данные, просто позволяя ему каждый раз обучаться дольше и дольше. Как было уже сказано, это ужасный путь обучения учащихся, но компьютер от этого не заскучает и не возмутится, а только каждый раз будет изучать возможность стать немного лучше, извлекая новую порцию информации из данных.

1.2.4. Более внимательный взгляд на обучение

Обычно мы верим, что в наших данных присутствует взаимосвязь. В конце концов, если данные просто чисто случайны, мы не будем пытаться извлечь из них информацию. Процесс в предыдущем разделе заключается в том, что, заставляя компьютер снова и снова обращаться к обучающей выборке, мы надеемся, что, обучаясь каждый раз понемногу от каждой выборки, алгоритм в конечном итоге найдет эту взаимосвязь между признаками в каждой выборке и осуществит маркировку, соответствующую маркировке эксперта. Затем он использует найденную взаимосвязь в новых данных при тестировании. Если он получит в большинстве своем правильные ответы, то можно говорить о высокой верности, или малой **ошибке обобщения**.

Но если компьютер постоянно не улучшает свое предсказание маркировки при предъявлении тестовой выборки, мы останавливаем обучение, поскольку прогресс в обучении отсутствует. Обычно в таком случае мы модифицируем алгоритм в надежде, что получим более удовлетворительный результат, и затем снова начинаем процесс обучения. Но это только наша надежда. Нет гарантии, что это будет для каждого множества хороший обучающий алгоритм, как и отсутствует гарантия, что мы вообще найдем таковой. Хорошей новостью является то, что мы без какой-либо математической гарантии часто можем на практике найти решения, которые иногда работают даже лучше, чем эксперты.

Одной из причин, по которой алгоритм может потерпеть неудачу при обучении, является отсутствие у него достаточных компьютерных ресурсов, чтобы найти взаимосвязь между выборками и маркировками. Мы иногда думаем о создании своего рода компьютерной **модели**, лежащей в основе данных. Например, если температура ежедневно растет в первые три часа, когда мы измеряем ее, компьютер может создать «модель», сообщающую, что по утрам температура растет. Это версия данных. Таким же образом маленькая пластиковая версия спортивного автомобиля или самолета является «моделью» этих средств транспорта. Классификатор, который мы рассматривали раньше, является одним из примеров модели.

В компьютере модель формируется структурой программного обеспечения и величинами параметров, которые она использует. Большая программа и боль-

шие множества параметров могут привести к моделям, которые способны узнать больше из данных. Мы говорим, что они имеют большую **емкость**, или **представительскую мощность**. Большая емкость позволяет в большей степени раскрыть смысловую информацию из имеющихся у нас данных. В качестве аналогии представим, что мы работаем с автомобильным дилером и нам предложили написать аннотацию на продаваемые нами автомобили. Предположим, что отдел продаж прислал нам перечень «слов одобрения», описывающих продаваемые нами автомобили. Через некоторое время после выполнения этой работы мы, возможно, поймем, как наилучшим образом представить каждый автомобиль этой модели.

Предположим, этот дилер затем купил наш первый мотоцикл. Слова, предложенные нам, или наша модель, не имеют достаточной емкости, чтобы представить это транспортное средство в дополнение ко всем другим транспортным средствам, которые мы уже описали. Нам просто недостаточно имеющегося словаря, чтобы отнести 2-колесное транспортное средство к 4-колесному. Если мы можем использовать более мощную модель с большей емкостью (то есть более обширный словарь для описания транспортных средств), мы сможем сделать такую работу лучше.

Большая модель означает и больший объем работ. И позже мы увидим в этой книге, что большая модель может часто дать лучший результат, чем маленькая, хотя потребует больше компьютерного времени и компьютерной памяти.

Величины, которые компьютер модифицирует с течением времени сам, называются его **параметрами**. Но наш обучающий алгоритм также управляется величинами, которые установили мы (такими, например, как скорость обучения, о которой говорилось ранее). Они именуются более неуклюжим словом **гиперпараметры**.

Разница между параметрами и гиперпараметрами состоит в том, что компьютер уточняет свои величины параметров в процессе обучения, в то время как гиперпараметры устанавливаются нами при написании и выполнении программ.

Если алгоритм достаточно хорошо обучен и удовлетворяет нас, хорошо справляясь с тестовым множеством, значит, он готов к **внедрению** или **реализации** во внешнем мире. Наши заказчики введут данные, и наша система выдаст правильную маркировку с предсказаниями. И таким образом изображения лиц скажут об именах, звуки – о словах, а измерения параметров среды превратятся в прогноз погоды.

Обзор литературных источников обширной и растущей области машинного обучения включает книги, целиком посвященные ей [Bishop06], [Goodfellow17].

В этой главе будет дан обзор основных категорий современных инструментов в данной области. Мы затем будем возвращаться к этим типам алгоритмов многократно на протяжении всей книги.

1.3. Обучение с учителем

Когда наши выборки поступают с предварительно обозначенными маркировками, как в примерах, приведенных ранее, мы говорим, что мы осуществляем **обучение с учителем**. Учителем являются наши маркировки. Они поступают для сравнения с предсказанной маркировкой, как это показано на рис. 1.8, и алгоритм

решает, является предсказанная маркировка правильной или нет. Существует два основных типа обучения с учителем, называемых **классификацией** и **регрессией**. Классификация относится к наблюдению за данным набором категорий, с тем чтобы найти одну, которая наилучшим образом описывает специфический вход. Регрессия имеет дело со множеством измерений и предсказанием некой другой величины (часто той, которая поступает следующей, но может также и предшествовать началу множества или быть где-то в середине его).

Рассмотрим их по очереди.

1.3.1. Классификация

Предположим, у нас есть набор фотографий повседневных предметов, и мы хотим сортировать их по изображению: яблочный чиллер, саламандра, пианино и т. д. Мы говорим, что хотим классифицировать эти фотографии. Процесс называется классификацией.

В этом методе мы начинаем тренинг, предъявляя компьютеру перечень всех маркировок (или классов, категорий), которым мы хотим его научить. Этот перечень – обычно просто комбинация всех маркировок для всех выборок в обучающем множестве, из которого изъяты дубликаты.

Затем мы обучаем систему с помощью набора фотографий и их маркировок, пока не определим, что система хорошо справляется с предсказанием правильной маркировки для каждой фотографии.

Теперь мы можем проверить систему на новых фотографиях, которые раньше ей не предъявлялись. Мы ожидаем, что она будет правильно маркировать изображения тех объектов, которые она видела во время обучения. Но если предмет не распознается или принадлежит к категории, которая не была частью обучающего набора, то система попытается выбрать лучшую категорию из тех, о которых она знает. На рис. 1.10 показана идея.

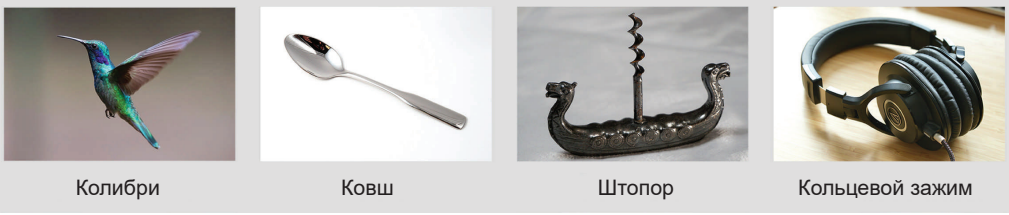

Рис. 1.10 ❖ Когда мы осуществляем квалификацию, то обучаем классификатор набором изображений, каждое из которых ассоциируется с маркировкой. По окончании обучения мы можем предъявить новые изображения, и компьютер будет пытаться выбрать лучшую маркировку для каждого изображения. Классификатор не обучался на металлическую ложку или наушники, поэтому показал наилучшее приближение, которое он смог найти

На рис. 1.10 показаны четыре предъявленные обученному классификатору для идентификации изображения, которые он никогда не видел прежде [Simonyan14]. Замечательно, что компьютер идентифицировал штопор, даже при том, что он стилизован по форме под лодку. Система не была тренирована на изображения металлических ложек или наушников, но в обоих случаях нашла наилучший ва-

риант изображения. Чтобы корректно отобразить эти предметы, система должна была видеть во время обучения множество примеров подобных предметов.

1.3.2. Регрессия

Предположим, что мы имеем неполный состав измерений и хотим оценить недостающие значения. Скажем, мы подсчитывали количество посещений серии концертов на местной сцене. Выступавший ансамбль каждый вечер получал плату за выступление, составлявшую часть средств, вырученных от продажи билетов на этот концерт.

К сожалению, был потерян отчет о присутствии зрителей на одном из вечерних концертов, и, для того чтобы спланировать наш бюджет, мы хотели бы знать, какая посещаемость концерта ожидается завтра. Подсчеты, которые были сделаны, показаны в левой части рис. 1.11. Наши оценки отсутствующих значений даны в правой части рис. 1.11.

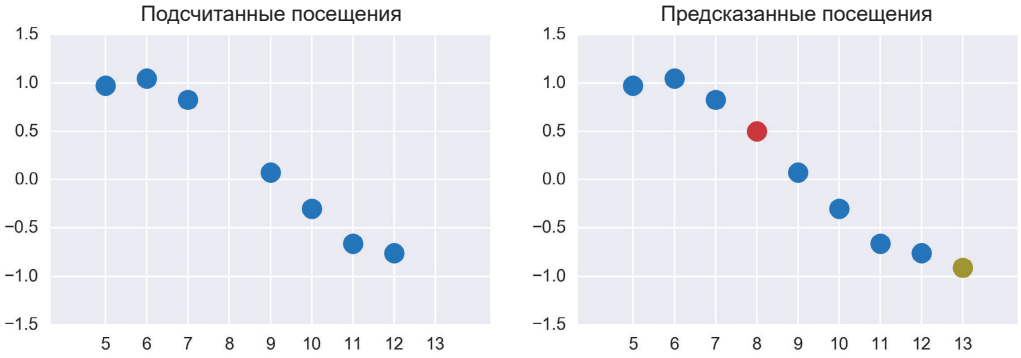

Рис. 1.11 ❖ При регрессии мы имеем множество входных и выходных значений. Здесь входами являются даты конвертов с 5 по 13 мая, а выходами – их посещения зрителями. Слева: измеренные данные с отсутствующими количествами посещений 8 мая. Справа: точки красного цвета являются оценкой отсутствующих данных 8 мая, а желтого цвета – предсказание посещения концерта зрителями 13 мая

Мы называем этот процесс восполнения и предсказания данных задачей **регрессии**. Это название, возможно, несколько не подходящее, поскольку «регрессия» означает возврат к более ранним условиям, а здесь, кажется, не происходит никакого возврата к прошлому.

Несоответствующее употребление этого слова появилось в статье, опубликованной в 1886 году, в которой ученый сообщал об исследовании роста детей [Galton86]. Он нашел, что, несмотря на то что некоторые имеют высокий рост, а другие – низкий, существует тенденция к тому, что рост людей со временем выравнивается. Он описал это явление как «регрессия к усреднению», что означало, что измерения имеют тенденцию двигаться от экстремальных значений к средним значениям.

Хотя это определение приписывают Галтону (Galton), очень похожее замечание высказано в обзоре Дарвина (Darwin) «О происхождении видов» (*On the Origin of Species*») [Darwin59]. Обозреватель Флеминг Дженкин (Fleeming Jenkin) высказывал мысль, что вариации видов будут преданы забвению, поскольку существует «общая тенденция возврата всего к стабильной посредственности» [Bryson04].

Сегодня слово «посредственность» имеет негативный оттенок, поэтому теперь чаще говорят «регрессия к среднему». Слово «регрессия» продолжает применяться для воплощения идеи использования статистических свойств данных для оценки отсутствующих или будущих значений.

Проблема «регрессии» – это просто одна из тех, при которых мы имеем некоторые значения, зависящие от входных значений (подобно тому, как посещения являются функцией дня месяца), и мы предсказываем новое значение для новых данных на входе.

Наиболее употребительным видом регрессии является **линейная регрессия**. Определение «линейная» относится к попытке согласовать наши входные данные с прямой линией, как это показано на рис. 1.12.

Прямая линия предпочтительна из-за ее простоты, но в этом примере она не слишком хорошо описывает данные. Наши данные поднимаются и опускаются относительно нее, и прямая линия не охватывает их. Это не самое плохое согласование, но оно и не самое лучшее.

В качестве альтернативы мы можем использовать более сложную форму регрессии, применив более сложный тип кривой, как показано справа на рис. 1.12. Это может обеспечить лучшее согласование с данными, но требует большего компьютерного времени. Поскольку наши кривые стали более сложными, они обычно требуют и больше данных при работе с ними.

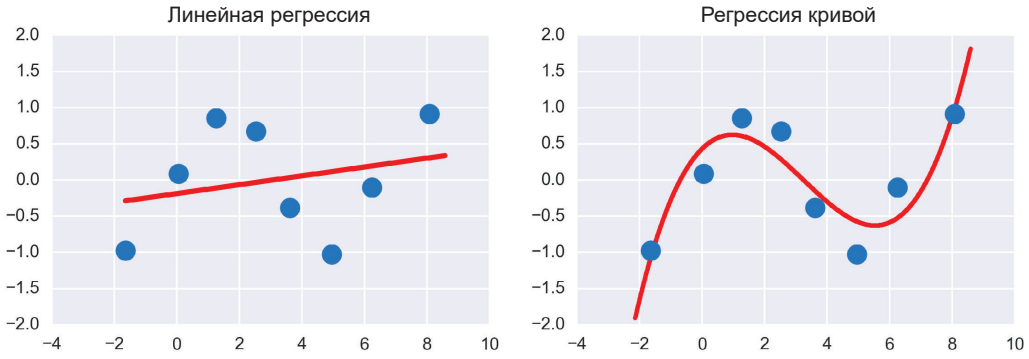

Рис. 1.12 ❖ Представление точек данных в математической формуле. Слева: линейная регрессия согласовывает прямую линию (красного цвета) и данные (синего цвета). Линия не очень хорошо согласуется с данными, но ее достоинством является простота. Справа: более сложная регрессия согласования кривой тех же данных. Она лучше согласуется с данными, но имеет более сложную форму и требует большей работы (и больше времени) при вычислениях

1.4. Обучение без учителя

Когда наши данные поступают на вход немаркированными, мы говорим, что любой обучающий алгоритм с данными на входе относится к категории **обучения без учителя**. Это просто означает, что мы не «контролируем» процесс обучения посредством маркировок. Система должна сама осуществить вычисления без нашей помощи.

Обучение без учителя часто используют при решении проблем, которые называются **кластеризация**, **подавление шумов** и **понижение размерности**. Рассмотрим эти проблемы по очереди.

1.4.1. Кластеризация

Предположим, что мы выкапываем яму под фундамент дома и, к удивлению, находим, что грунт заполнен древними глиняными горшками и вазами. Мы звоним археологу, которая устанавливает, что мы нашли перемешанную коллекцию древних сосудов, видимо, из многих разных мест, может быть, и разных времен. Археолог не может определить маркировку и символику и потому не уверена в происхождении каждого сосуда. На каких-то сосудах маркировка выглядит как вариации на одну и ту же тему, в то время как на других это совершенно разные символы. Чтобы решить эту проблему, она сняла копии маркировок и затем попыталась рассортировать их по группам. Но их было слишком много для этого. Поскольку ее студенты старших курсов занимались другими проектами, она решила использовать для этих целей алгоритм машинного обучения, чтобы автоматически разумным образом сгруппировать маркировки.

На рис. 1.13 показаны скопированные ею маркировки и группы, которые могли быть найдены алгоритмом. Мы называем это проблемой кластеризации и говорим, что это **алгоритм кластеризации**. Существует много такого рода алгоритмов, есть из чего выбрать, и позже мы увидим некоторые из них. Используя терминологию, которую мы только что ввели, археолог проводит кластеризацию, применяя ненаблюдаемый алгоритм обучения.

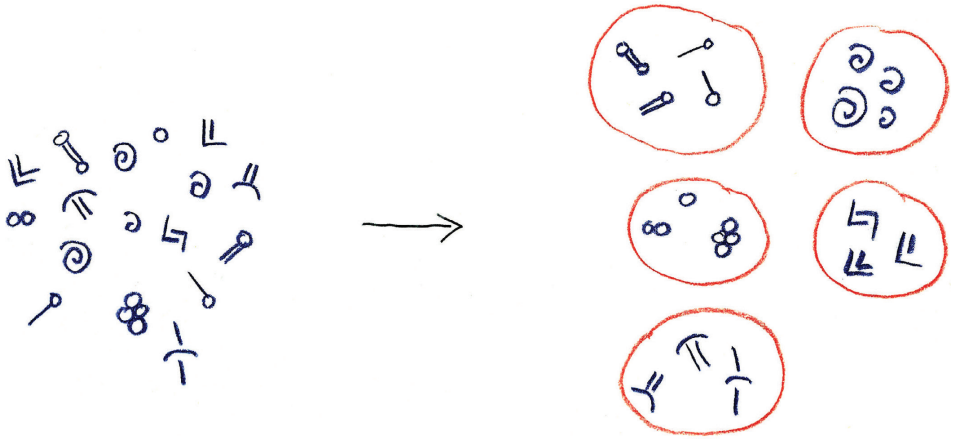

Рис. 1.13 ❖ Применение алгоритма кластеризации для маркировки глиняных сосудов. Слева: копии маркировки сосудов. Справа: маркировки, сгруппированные в кластеры

1.4.2. Подавление шума

Иногда выборки **искажены шумом**. Если выборка является аудиоклипом кого-то, говорящего по своему телефону, шумом может быть фоновый шум улицы, делающий неразборчивыми его слова.

Если выборка является электронным сигналом – скажем, радио или телевизионной программой, – шум может наводиться статической помехой, разрушающей звук в радиошоу, или мелькающими точками в телевизионном изображении. В синтетическом или создаваемом компьютером изображении можно сэкономить время, удаляя некоторые пиксели, так что они становятся просто черными точками. Для компьютера это своего рода «шум», хотя в действительности это скорее пропавшие данные, чем искаженные данные.

Наши глаза чувствительны к такому шуму и исчезнувшим пикселям, но данные часто несут в себе больше доступной информации, чем мы можем предположить, глядя на них. На рис. 1.14 показан пример зашумленного изображения и насколько алгоритм, подавляющий шум, может его очистить.

Рис. 1.14 ❖ Подавление или снижение уровня шума уменьшает эффекты, связанные со случайным искажением выборки, и может дать заполнить в некоторых местах исчезнувшие пиксели. Слева: изображение коровы искажено множеством шумовых пикселей и некоторыми исчезнувшими пикселями. Справа: оригинальное изображение. Подавляющие алгоритмы могут восстановить изображение с разным успехом

Это применимо не только к изображениям. При использовании данных, получаемых почти от любых источников, выборки содержат неточности или другие искажения, которые могут скрывать информацию, которую мы хотели бы извлечь.

Если в выборке отсутствуют некоторые величины, мы можем обратиться к алгоритму регрессии и заполнить результатом их место. Или мы можем трактовать их как форму шума и применить подавляющий шум алгоритм, чтобы попытаться заполнить их.

Поскольку мы не имеем маркировки для наших данных (например, в зашумленном фото мы имеем только пиксели), алгоритм, подавляющий шум, является формой обучения без учителя. Путем изучения статистики выборок алгоритм оценивает, какая часть каждой выборки является шумом, и удаляет ее, оставляя нам очищенные данные, что облегчает обучение и интерпретацию.

Мы часто применяем алгоритм подавления шума к данным перед использованием их для обучения. При удалении искаженных и отсутствующих величин на входе процесс обучения обычно протекает быстрее и более гладко.

1.4.3. Понижение размерности

Иногда наши выборки имеют больше признаков, чем необходимо. Для примера мы можем собирать выборки погоды в пустыне в разгар лета. Каждый день

мы записываем скорость ветра, его направление и наличие дождя. Для этого времени года и местоположения признак наличия или отсутствия дождя будет присутствовать в каждой выборке. Если мы используем эти выборки в системе машинного обучения, компьютеру будет необходимо обрабатывать и интерпретировать эту бесполезную, постоянную часть информации в каждой выборке. В лучшем случае мы замедлим анализ. В худшем – это может повлиять на точность системы, потому что компьютер потратит некоторые конечные временные ресурсы и ресурсы памяти, пытаясь получить информацию от этих неменяющихся признаков.

Иногда признаки содержат избыточную информацию. Например, в некоторых клиниках вес пациента измеряется при проходе через входную дверь в килограммах. Затем медсестра приглашает его во врачебный кабинет, где снова измеряют вес пациента, но на этот раз в фунтах, и также заносят в медицинскую карту. Теперь мы имеем одну и ту же информацию, присутствующую дважды, и ее будет трудно распознать, поскольку числовые значения разные. Так же, как и бесполезность признака отсутствия дождя в предыдущем примере, избыточность не приносит пользы при машинном обучении. Более абстрактный пример: привлекаются данные, более сложные, чем это необходимо. Предположим, что деревья в национальном парке заражены неизвестной инфекцией, которая повреждает их. Природоохранный сервис пытается определить степень распространения инфекции в глубину ствола упавших деревьев. На машине секция ствола проворачивается, и при этом снимается слой за слоем с проверкой плотности древесины в каждом слое. Тем самым получают трехмерные данные (с размерностью 3D), как показано на рис. 1.15, где каждая точка расположена в 3D-пространстве и имеет величину, сообщающую нам о степени проникновения инфекции в глубину ствола. Чтобы представить эту величину, нам необходимо 4 числа: 3 – расположение точки в пространстве и 1 – для самой величины.

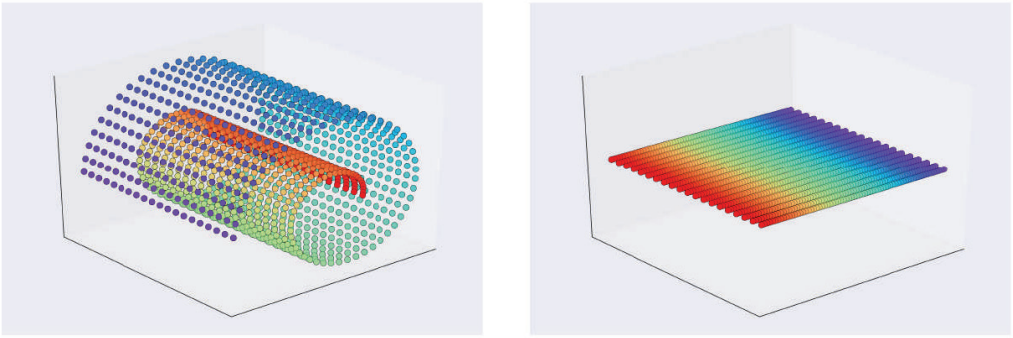

Рис. 1.15 ❖ Измерение уровня инфекции в стволе дерева при его вращении со снятием слоя за слоем. Слева: первоначальные данные с размерностью 3D. Справа: мы можем упростить задачу, уменьшив размерность до 2D

Здесь важна пространственная информация, но если мы хотим иметь информацию только вдоль спирали слоя, мы можем облегчить себе жизнь, превратив спираль в плоскость, как показано на рис. 1.15 справа. Теперь нам необходимы только 3 числа для каждой точки: 2 – для указания местоположения точки и 1 – для

самой величины. Это делает нашу обработку быстрее, не отказываясь от чего-либо важного для этого специфического анализа.

Иногда мы можем облегчить себе жизнь за счет упрощения данных при их измерении. Когда это невозможно, мы можем упростить пакет данных после их получения, убирая информацию, не релевантную вопросам, которые нас интересуют. Тогда компьютеру не нужно будет обрабатывать лишнюю информацию, экономя время и, возможно, повышая качество получаемых результатов.

Для примера давайте предположим, что мы проводим мониторинг трафика на вновь построенной горной дороге, с изгибами, односторонним движением и одной полосой движения, и мы хотим быть уверенными, что трафик на этом специфическом участке трассы будет безопасным. Для этого мы установили некое устройство для мониторинга и периодически получаем снимки, подобные показанным на рис. 1.16 и дающие нам информацию о том, где расположен каждый автомобиль, в какую сторону он движется, с какой скоростью на участке дороги, который нас интересует.

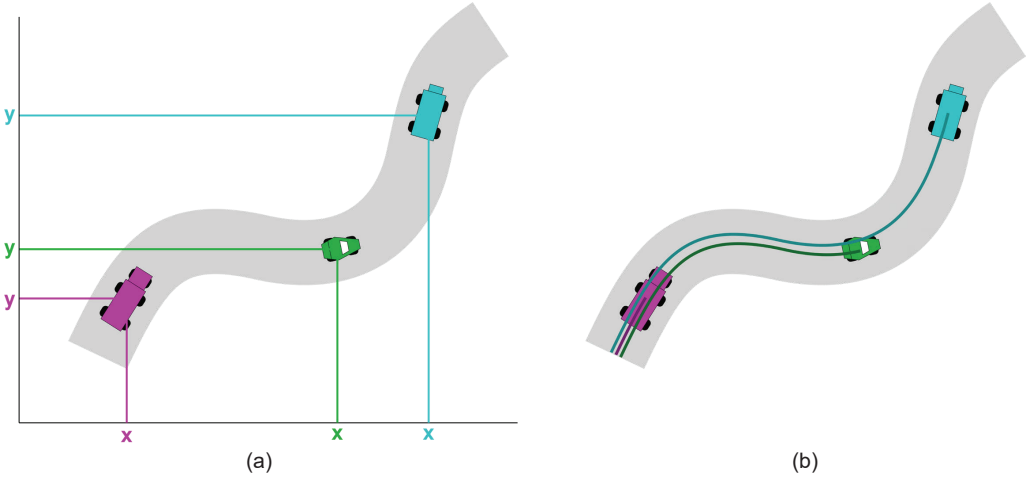

Рис. 1.16 ❖ Где транспортные средства на этой дороге? (a) Мы можем измерять положение каждого автомобиля, используя широту и долготу, требующие двух чисел (здесь обозначены как x и y); (b) как альтернатива мы можем измерять расстояние, которое каждый автомобиль проходит по этому участку дороги

Мы можем описать местоположение каждого автомобиля, используя широту и долготу. Но, поскольку дорога однополосная, есть более простой путь для описания местоположения каждого автомобиля одним числом – расстоянием от начала участка дороги, как показано на рис. 1.16 (b).

Во всех подобных случаях мы можем использовать ненаблюдаемое обучение для анализа наших выборок, удаляя признаки, которые не вносят вклада в понимание имеющейся в них информации. Делая данные проще, мы позволяем системе меньше работать по извлечению информации, что делает обучение быстрее и точнее.

Иногда мы можем сделать наши данные проще, удаляя всего несколько битов информации. В некоторых ситуациях, когда нам надо просто провести грубую

проверку, чтобы убедиться, что что-то работает надлежащим образом, стоит отказаться от большой точности, если это позволит упростить обучение и повысить скорость.

Давайте посмотрим, как это сделать в нашем примере с новой дорогой. Мы нашли на рис. 1.16, как определить местоположение транспортных средств с помощью одного числа вместо двух. В этом случае мы не потеряли информацию. Но в другой ситуации мы можем объединить измерения так, чтобы получить *наибольшее* количество информации, хотя часть ее все же потеряем. Например, в модифицированной версии нашей дороги мы можем иметь двухполосную дорогу с движением в обоих направлениях. В этом случае нам действительно, для того чтобы точно знать местоположение автомобилей, требуется два числа – широта и долгота, как на рис. 1.16 (а).

Но при желании мы можем использовать все же одно число. Как показывает нам рис. 1.17, это дает нам все же хорошую аппроксимацию местоположения автомобилей.

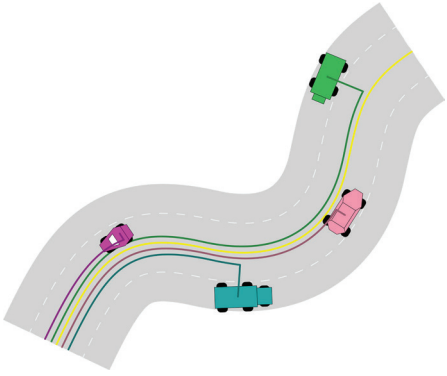

Рис. 1.17 ❖ Мы иногда можем получить упрощение, если можем отказаться от некоторой точности. Мы можем представить положение этих транспортных средств различными способами в зависимости от наших потребностей

Чтобы найти местоположения транспортных средств на рис. 1.17, мы можем измерять их удаление от начала участка дороги внизу слева, как мы это делали на рис. 1.16. Но это предполагает, что каждое транспортное средство уже расположено в центре дороги, что теперь неверно. Мы можем переключиться на более точное и более сложное представление. Компромисс состоит в том, что более простая версия будет обычно давать нам систему, которая быстрее обучается и работает, но содержит неточности. Большинство точных систем будет медленнее, но с меньшим количеством ошибок. Оба варианта приемлемы в зависимости от того, насколько высокую точность и скорость работы мы хотим получить.

В некоторых ситуациях аппроксимация местоположения, которую мы получаем, применяя вариант, изображенный на рис. 1.17, очень неплохая.

Во всех этих примерах наша цель состояла в упрощении данных или путем удаления неинформативных признаков, объединяя избыточные признаки, или в обмене точности на простоту путем объединения, или как-то иначе манипулируя признаками.