Содержание

Предисловие

О структуре книги

В первых четырех главах излагаются теория и фундаментальные факты в объеме, достаточном для того, чтобы специалист-практик мог двигаться дальше. А в последних пяти главах на этой основе исследуются различные области глубокого обучения с помощью библиотеки DL4J:

- построение глубоких сетей;
- методы настройки сетей;
- векторизация для различных типов данных;
- реализация процессов глубокого обучения на платформе Spark.

DL4J – сокращение для Deeplearning4j
В этой книге названия DL4J и Deeplearning4j считаются синонимами. Оба они относятся к набору инструментов в библиотеке Deeplearning4j.

Мы организовали материал именно таким образом, потому что ощущали потребность в книге, которая содержала бы «достаточно теории» и вместе с тем была бы достаточно практичной для построения процессов глубокого обучения производственного уровня. Мы полагаем, что выбранный гибридный подход хорошо отвечает поставленной цели.

Глава 1 содержит обзор концепций машинного обучения вообще и глубокого обучения в частности с целью ввести читателя в курс дела и сообщить достаточно сведений для понимания остальной части книги. Мы включили эту главу, чтобы начинающие могли освежить подзабытое или познакомиться с новыми для себя идеями, и тем самым хотели сделать книгу доступной максимально широкой аудитории.

В главе 2, основанной на материале главы 1, излагаются основы нейронных сетей. По сути своей она посвящена теории нейронных сетей, но мы старались представить информацию в доступной форме. В главе 3 описывается, как из нейронных сетей вырастают глубокие сети. В главе 4 представлены четыре основные архитектуры глубоких сетей, это фундамент для понимания последующих глав.

Глава 5 содержит ряд примеров Java-кода, в которых используются методы из первой части книги. В главах 6 и 7 рассматриваются фундаментальные основы настройки нейронных сетей общего вида, а затем эти знания применяются к настройке специальных архитектур глубоких сетей. Эти главы платформенно-независимы и в равной мере относятся к любой библиотеке глубокого обучения. В главе 8 приводятся обзор методов векторизации и основы работы с библиотекой DataVec (средством ETL и векторизации, входящим в состав DL4J). Глава 9 завершает основной текст книги обзором применения DL4J в системах Spark и Hadoop – здесь вы найдете реальные примеры, которые сможете выполнить в собственном кластере Spark.

В книге много приложений, в которых рассматриваются важные темы, не нашедшие отражения в основных главах, а именно:

- искусственный интеллект;
- использование Maven в проектах на основе DL4J;
- работа с GPU;
- использование ND4J API
- и прочее.

Кто такой «практик»?

В настоящее время термин «наука о данных» (data science) не имеет четкого определения и зачастую означает разные вещи. Мир науки о данных и искусственного интеллекта (ИИ) широк и расплывчат, как и многие термины в современной информатике. Связано это с тем, что машинное обучение проникло почти во все дисциплины.

У этого широкого проникновения есть исторические параллели с тем, как в 1990-х годах Всемирная паутина проникла во все дисциплины и привела много новых людей в область технологии. Так и теперь самые разные специалисты – инженеры, статистики, аналитики, люди творческих профессий – каждодневно пополняют ряды поборников машинного обучения. Эта книга призвана сделать глубокое (и машинное) обучение доступным самой широкой аудитории.

Если эта тематика вам интересна и вы читаете это предисловие – значит, *вы практик, и эта книга для вас*.

Кому стоит прочитать эту книгу?

Мы решили начать книгу не с игрушечных примеров, которые затем постепенно обрастают деталями, а с фундаментальных основ, которые позволят в полной мере погрузиться в глубокое обучение.

Нам кажется, что во многих книгах опущены базовые вопросы, с которыми каждый практик должен хотя бы бегло познакомиться. Опираясь на свой опыт работы в области машинного обучения, мы решили включить материал, который необходим начинающему практику для успешной работы над своими проектами в сфере глубокого обучения.

Возможно, вы захотите пропустить первые две главы и сразу перейти к основам глубокого обучения. Но мы полагаем, что многие оценят наличие вводных сведений, поскольку это делает гладким переход к основанным на них более трудным темам. Ниже мы предлагаем различные подходы к чтению книги в зависимости от подготовки читателя.

Практический специалист по машинному обучению в коммерческой компании

В этой категории мы выделяем две подгруппы:

- специалист по анализу данных (data scientist);
- Java-программист.

Специалист по анализу данных

Эти люди уже строили модели раньше и свободно ориентируются в науке о данных. Если вы из их числа, то главу 1 можете пропустить, а главу 2 бегло просмот-

реть. Мы рекомендуем сразу перейти к главе 3, поскольку вы, скорее всего, достаточно подготовлены к знакомству с основаниями глубоких сетей.

Java-программист

Java-программистам обычно поручают интегрировать код машинного обучения в производственные системы. Если это как раз к вам и относится, то вас, наверное, заинтересует глава 1, поскольку она позволит познакомиться с терминологией, применяемой в науке о данных. Вам также будет интересно приложение E, поскольку код интеграции модельных оценок обычно не обходится без ND4J API.

Руководитель предприятия

Среди рецензентов нашей книги были и руководители компаний из списка Fortune 500, им материал понравился, т. к. позволил лучше понять, что вообще происходит в области глубокого обучения. Один из них отметил, что после окончания колледжа уже «прошло какое-то время», так что глава 1 пришлась весьма кстати для освежения памяти. Если вы руководитель, то рекомендуем начать с беглого прочтения главы 1, чтобы вспомнить кое-какую терминологию. Но, вероятно, вам стоит пропустить главы, посвященные API и конкретным примерам.

Ученый

Если вы работаете в академическом учреждении, то главы 1 и 2 вряд ли будут вам интересны, т. к. в высшей школе вы это уже проходили. Зато вас наверняка заинтересуют главы о настройке нейронных сетей вообще и для конкретной архитектуры, поскольку эта информация основана на результатах исследований и применима ко всем реализациям глубокого обучения. Тем, кто предпочитает высокопроизводительную линейную алгебру на виртуальной машине Java (JVM), будет также интересно обсуждение библиотеки ND4J.

Графические выделения

В книге применяются следующие графические выделения.

Курсив
: Новые термины, URL-адреса, адреса электронной почты, имена и расширения имен файлов.

`Моноширинный`
: Листинги программ, а также элементы кода в основном тексте: имена переменных и функций, базы данных, типы данных, переменные окружения, предложения и ключевые слова языка. Также применяется для набора имен модулей и пакетов, команд или иного текста, которые следует вводить буквально, и результатов работы команд.

`Моноширинный курсив`
: Текст, вместо которого следует подставить значения, заданные пользователем или определяемые контекстом.

Так обозначается примечание общего характера.

Так обозначается предупреждение или предостережение.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф на странице с описанием соответствующей книги.

Мы высоко ценим, хотя и не требуем, ссылки на наши издания. В ссылке обычно указываются название книги, имя автора, издательство и ISBN, например: «Программирование на языке Rust» Джима Блэнди, Джейсона Орендорф (O'Reilly, ДМК Пресс). Copyright © 2018 Jim Blandy and Jason Orendorff, 978-1-491-92728-1 (англ.), 978-5-97060-236-2 (рус.).

Если вы полагаете, что планируемое использование кода выходит за рамки изложенной выше лицензии, пожалуйста, обратитесь к нам по адресу dmkpress@gmail.com.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Благодарности

От Джоша

В отборе материала для этой книги и ее рецензировании я опирался на помощь людей куда умнее меня. Проект такого размера, как DL4J, не может существовать в вакууме, поэтому в формулировании многих идей и рекомендаций я полагался на мнение экспертов из сообщества и инженеров компании Skymind.

Я никогда не думал, что работа над тем, что впоследствии превратилось в DL4J, вместе с Адамом (после случайной встречи на конференции MLConf) когда-ни-

будь воплотится в книгу. Честно говоря, хотя я стоял у истоков DL4J, Адам сделал куда больше записей в репозиторий, чем я. Поэтому я чрезвычайно признателен Адаму за его преданность проекту, самой идее глубокого обучения на JVM и за то, что он не отступился на начальном этапе, когда все было зыбко и неясно. И да, ты был прав: ND4J оказалась отличной идеей.

Написание книги может стать долгим путешествием в одиночестве, поэтому я особенно благодарен Алексу Блэку, который не только потратил массу времени на рецензирование, но и предложил материал для приложений. Энциклопедические познания Алекса касательно опубликованной литературы по нейронным сетям были чрезвычайно важны при шлифовке многих мелких деталей и стали гарантией правильности изложения в целом и в частности. Без Алекса главы 6 и 7 и наполовину не были бы тем, чем являются теперь.

Сьюзан Эрали (Susan Eraly) оказала ценнейшую помощь в написании раздела о функциях потерь, предоставила материал для приложений (многие уравнения, приведенные в этой книге, были выправлены Сьюзан) и сделала много замечаний в процессе рецензирования. Мелани Уоррик (Melanie Warrick) сыграла основную роль в рецензировании ранних вариантов книги, делилась своими замечаниями и рассказала немало интересного о внутренних механизмах работы сверточных нейронных сетей (CNN).

Дэвид Кэйл (David Kale) частенько оставлял замечания, не пропуская небрежностей в отношении деталей работы сетей и библиографических ссылок. Дэйв всегда представлял академический взгляд на необходимый уровень строгости с учетом того, для какой аудитории мы пишем.

Джеймс Лонг (James Long) критически внимал моим тирадам на тему того, что включать в книгу, а что опустить, и представлял точку зрения статистика-практика. Нередко, когда было не ясно, как лучше изложить какой-то сложный вопрос, Джеймс выступал в роли слушателя, на котором я опробовал обсуждение с разных сторон. Если Дэвид Кэйл и Алекс Блэк зачастую напоминали мне о необходимости математической строгости, то Джеймс играл роль рационального адвоката дьявола, не допускающего, чтобы читатель «утонул в математических деталях».

Вячеслав Кокорин (Raver) привнес свои знания в разработку примеров из области обработки естественного языка и технологии Word2Vec.

Хотел бы отметить поддержку со стороны Криса Николсона (Chris Nicholson), генерального директора компании Skymind. Крис оказывал поддержку этой книге на каждом этапе, и в значительной степени благодаря ему у нас были время и ресурсы для завершения работы над ней.

Я выражаю благодарность всем, кто предлагал материалы для приложений: Алексу Блэку (обратное распространение, DataVec), Вячеславу Кокорину (GPU), Сьюзан Эрали (GPU) и Рубену Фижелю (Ruben Fiszel) (обучение с подкреплением). На различных этапах в рецензировании книги принимали также участие Гран Ингерсол (Grant Ingersol), Дин Уэмплер (Dean Wampler), Роберт Чонг (Robert Chong), Тэд Маласка (Ted Malaska), Райан Джено (Ryan Geno), Ларс Джордж (Lars George), Сунеель Мартхи (Suneel Marthi), Франсу Гарилло (Francois Garillot) и Дон Браун (Don Brown). Все ошибки, которые могли остаться в книге, – целиком и полностью моя вина.

Я благодарен нашему уважаемому редактору Тиму Макговерну (Tim McGovern) за отзывы, замечания и просто терпение – ведь проект растянулся на годы и стал

больше на три главы. Мы высоко ценим, что он позволил нам сорвать сроки, но сделать все как надо.

Ниже перечислены другие люди, которые оказали влияние на мою карьеру, приведшую к появлению этой книги: мои родители (Льюис и Конни), д-р Энди Новобилиски (Andy Novobiliski) (аспирантура), д-р Мина Сартипи (Dr. Mina Sartipi) (научный руководитель), д-р Билли Харрис (Billy Harris) (курс алгоритмов для аспирантов), д-р Джо Дьюмас (Joe Dumas) (аспирантура), Ричи Кэрролл (Ritchie Carroll) (создатель openPDC), Пол Трачан (Paul Trachian), Кристоф Бисиглиа (Christophe Bisciglia) и Майк Олсон (Mike Olson) (затащившие меня в компанию Cloudera), Малком Рэйми (Malcom Ramey) (за мою первую работу в качестве программиста), университет Теннесси в Чаттануге и пиццерию Лупи (где я питался во время учебы в аспирантуре).

И последними по порядку, но отнюдь не по значимости я хочу поблагодарить свою жену Лесли и сыновей Этана, Гриффина и Дэйна за терпение, с которым они переносили мою работу по ночам, а иногда и во время отпуска.

От Адама

Я благодарен своей команде в компании Skymind за ту огромную работу, которую они проделали, рецензируя бесчисленные варианты книги. Особое спасибо Крису, который мирился с моей безумной идеей писать книгу, одновременно пытаясь основать стартап.

Библиотека родилась в 2013 году со случайной встречи с Джошем на конференции MLConf и выросла в проект, который ныне используется во всем мире. Благодаря DL4J я поездил по миру и открыл для себя целый мир новых впечатлений.

Прежде всего я хочу поблагодарить своего соавтора Джоша Паттерсона, который взял на себя львиную долю работы над книгой и заслуживает всяческой признательности. Он тратил вечера и выходные, торопясь выпустить книгу в свет, пока я продолжал работать над кодом и включал в него все новые возможности.

Вслед за Джошем отдаю должное многим нашим коллегам и соавторам – тем, кто был с нами с самого начала, как Алекс, Вячеслав Кокорин (Raver), и тем, кто, как Дэйв, присоединился позже и зорко следил за правильностью математики.

Тим Макговерн оказался прекрасным слушателем, на котором я опробовал некоторые свои завиральные идеи относительно книг для издательства O'Reilly, а также любезно позволил мне самому выбрать название книги.

Глава 1

Обзор машинного обучения

Сконденсировать факт из туманности нюансов.
– Нил Стивенсон «Лавина»

Обучающиеся машины

Интерес к машинному обучению резко возрос в последние десять лет. Машинное обучение включают в программы факультетов информатики, по нему проводят конференции, а в заголовках «Wall Street Journal» оно встречается чуть ли не ежедневно. Говоря о машинном обучении, многие путают два вопроса: что оно действительно может и чего от него хотели бы. По большому счету, *машинное обучение* – это применение алгоритмов, которые извлекают информацию из исходных данных и представляют ее в виде той или иной *модели*. Эта модель используется, чтобы вывести другие данные, которые в модели отсутствуют.

Нейронные сети – один из типов моделей машинного обучения, они существуют уже по меньшей мере 50 лет. Фундаментальной единицей нейронной сети является *блок*, или *узел*, – приблизительный аналог биологического нейрона в мозге млекопитающих. Модель связей между нейронами также основана на работе биологического мозга, равно как и эволюция связей с течением времени (в результате «обучения»). В следующих двух главах мы будем подробно рассматривать функционирование таких моделей.

В середине 1980-х и в начале 1990-х годов произошли важные подвижки в области архитектуры нейронных сетей. Но для получения хороших результатов требовалось очень много данных и времени, что замедляло их практическое внедрение, так что постепенно интерес угас. В начале 2000-х годов наблюдался экспоненциальный рост вычислительной мощности, и промышленность стала свидетелем «кембрийского взрыва» методов вычислений – ничего подобного раньше не было и в помине. Глубокое обучение, появившееся в результате взрывного роста вычислительной мощности в этом десятилетии, стало серьезным игроком и победителем многих важных состязаний по машинному обучению. В 2017 году интерес не спадает, сегодня глубокое обучение можно встретить в любом закоулке машинного обучения.

Ниже мы еще вернемся к нашему определению глубокого обучения. Эта книга построена так, чтобы практик, например вы, мог снять ее с полки и сделать следующее:

- получить общие сведения об основах линейной алгебры и машинного обучения;

- получить общие сведения об основах нейронных сетей;
- изучить четыре основные архитектуры глубоких сетей;
- воспользоваться примерами кода для исследования вариантов глубоких сетей на практике.

Мы надеемся, что материал покажется вам практичным и не слишком сложным. И давайте начнем с вопроса о том, что же такое машинное обучение, а также с базовых понятий, которые помогут лучше понять, о чем говорится в книге.

Как машины могут обучаться?

Чтобы определить, как машины могут обучаться, нужно определить, что мы понимаем под «обучением». В быту мы имеем в виду «приобретение знаний посредством самостоятельного изучения, на опыте или в процессе общения с преподавателем». Немного сместив фокус, можно считать, что машинное обучение – это применение алгоритмов для получения структурных описаний из примеров данных. Компьютер обучается чему-то, относящемуся к информационным структурам, содержащимся в исходных данных. Структурные описания – другое название моделей, которые служат для представления информации, извлеченной из данных. Эти структуры, или модели, можно использовать для предсказания неизвестных данных. Структурные описания (или модели) могут принимать разную форму, в том числе:

- решающие деревья;
- линейная регрессия;
- веса связей в нейронных сетях.

Модель каждого типа определяет свой способ применения правил к известным данным для предсказания неизвестных. В случае решающих деревьев создается древовидный набор правил, а в случае линейных моделей – набор параметров, представляющих входные данные.

В нейронных сетях имеется так называемый *вектор параметров*, представляющий веса связей между узлами сети. Ниже в этой главе мы опишем этот тип моделей подробнее.

Машинное обучение и добыча данных

Термин *добыча данных*[1] существует уже много десятилетий и, как многие термины машинного обучения, понимается или используется неправильно. В этой книге под «добычей данных» мы будем понимать «извлечение информации из данных». Отличие машинного обучения состоит в том, что оно относится к алгоритмам, которые применяются в процессе добычи данных для получения структурных описаний из исходных данных. Вот простая интерпретация добычи данных:

- для обучения концепциям нам нужны примеры исходных данных;
- примеры состоят из строк или экземпляров данных, из которых видны определенные закономерности в данных;

[1] В отечественной литературе устоялся термин «добыча данных», хотя он совершенно не отражает смысла понятия. Английское слово «mining» означает «добыча полезных ископаемых» с ударением на слове «полезных», т. е. «извлечение руды из породы». Правильнее было бы говорить о «добыче информации» или прибегнуть к описательному переводу типа «глубокий анализ данных», но мы будем придерживаться сложившейся терминологии. – *Прим. перев.*

- машина обучается концепциям на этих закономерностях посредством алгоритмов машинного обучения.

В целом этот процесс можно считать «добычей данных».

Артур Сэмюэл (Arthur Samuel), пионер в области искусственного интеллекта (ИИ), работавший в компании IBM и в Стэнфордском университете, определял машинное обучение следующим образом:

> Область исследований, посвященная наделению компьютеров способностью обучаться без явного программирования.

Сэмюэл разработал программу, которая могла играть в шашки и адаптировала свою стратегию по мере того, как обучалась ассоциировать вероятность выигрыша или проигрыша с определенными позициями на доске. Эта фундаментальная схема поиска паттернов, ведущих к победе или к поражению, с последующим распознаванием и подкреплением успешных паттернов и по сей день лежит в основе машинного обучения и ИИ.

Концепция машин, способных самостоятельно обучаться для достижения целей, поражала наше воображение в течение многих десятилетий. Наверное, наилучшим образом она была выражена дедушками современного ИИ, Стюартом Расселом (Stuart Russell[1]) и Питером Норвигом (Peter Norvig[2]), в книге «Artificial Intelligence: A Modern Approach»[3]:

> Как может медленный, крохотный мозг, не важно, биологический или электронный, воспринимать, понимать, предсказывать и манипулировать миром, который гораздо больше и сложнее его самого?

Эта цитата отсылает нас к идее о том, что концепции обучения были заимствованы у процессов и алгоритмов, подсмотренных в природе. На рис. 1.1 наглядно изображено наше представление о связи между ИИ, машинным обучением и глубоким обучением.

Рис. 1.1 ❖ Связь между ИИ и глубоким обучением

[1] https://www2.eecs.berkeley.edu/Faculty/Homepages/russell.html.

[2] http://norvig.com/.

[3] *Стюарт Р., Норвиг П.* Искусственный интеллект. Современный подход. М.: Вильямс, 2017.

Область ИИ широка и давно уже является предметом исследований. Глубокое обучение – это часть машинного обучения, которая, в свою очередь, является частью ИИ. Теперь кратко рассмотрим другой источник глубокого обучения: связь искусственных нейронных сетей с биологией.

Биологические корни

Биологическая нейронная сеть (мозг) содержит приблизительно 86 миллиардов нейронов, каждый из которых соединен с большим числом других.

Общее число связей в мозге человека
По осторожной оценке исследователей, между нейронами в мозге человека существуют более 500 триллионов связей. Самые крупные современные искусственные нейронные сети даже отдаленно не приближаются к этой величине.

С точки зрения обработки информации, биологический нейрон представляет собой возбуждаемый блок, который может обрабатывать и передавать информацию с помощью электрических и химических сигналов. Нейрон считается основным компонентом головного мозга, спинного мозга и ганглий периферийной нервной системы. Как мы увидим ниже, искусственные нейронные сети структурно гораздо проще.

Сравнение биологической и искусственной нейронных сетей
Биологические нейронные сети гораздо сложнее (на несколько порядков) искусственных!

У искусственных нейронных сетей есть два свойства, отражающих общие представления о работе мозга. Во-первых, самой мелкой единицей нейронной сети является *искусственный нейрон* (или, для краткости, *блок*). Образцом для искусственного нейрона служит биологический нейрон мозга: как и биологический нейрон, он возбуждается в результате поступления входных сигналов. Искусственные нейроны передают часть полученной информации (но не всю) другим искусственным нейронам, часто в преобразованном виде. Далее в этой главе мы детально рассмотрим эти преобразования в контексте нейронных сетей.

Во-вторых, как биологические нейроны можно научить передавать дальше только сигналы, полезные для достижения более общих целей мозга, так и искусственные нейроны можно обучить передавать только полезные сигналы. Далее мы увидим, как эти идеи позволяют искусственной нейронной сети моделировать биологическую посредством битов и функций.

Биологические идеи в информатике

Применение биологических идей в информатике не ограничивается искусственными нейронными сетями. За последние 50 лет предметом исследований, нашедших отражение в информатике, стали и другие природные объекты, например:

- муравьи:
- термиты[1];
- пчелы;
- генетические алгоритмы.

[1] Patterson, 2008. TinyTermite: A Secure Routing Algorithm; Sartipi and Patterson, 2009. TinyTermite: A Secure Routing Algorithm on Intel Mote 2 Sensor Network Platform.

Например, муравейник рассматривался учеными как мощный децентрализованный компьютер, в котором ни один отдельный муравей не является точкой общего отказа[1]. Муравьи постоянно переключаются с одной задачи на другую в поисках близких к оптимальному решений задачи балансировки нагрузки с помощью таких метаэвристик, как количественная стигмергия. Колонии муравьев способны убирать территорию, обороняться, строить гнездо и добывать пищу, выделяя для каждой задачи почти оптимальное число работников, основанное на относительной потребности, при этом не существует какого-то одного муравья, координирующего работу.

Что такое глубокое обучение?

Дать определение глубокому обучению не так-то просто, потому что его формы медленно изменялись на протяжении последнего десятилетия. Одно из полезных определений гласит, что глубокое обучение имеет дело с «нейронными сетями, имеющими более двух слоев». Проблема в том, что такие сети считались глубокими где-то в 1980-х годах. Но нам кажется, что для демонстрации впечатляющих результатов, наблюдаемых в последние годы, нейронные сети архитектурно должны были перерасти ранние формы (что стало возможно благодаря значительному увеличению вычислительной мощности). Перечислим некоторые аспекты эволюции нейронных сетей:

- больше нейронов, чем в предшествующих сетях;
- более сложные способы соединения нейронов и слоев;
- резкий рост вычислительной мощности, доступной для обучения;
- автоматическое выделение признаков.

В этой книге мы будем определять глубокую сеть как нейронную сеть с большим числом параметров и слоев, имеющую одну из четырех фундаментальных архитектур:

- сети, предобученные без учителя;
- сверточные нейронные сети;
- рекуррентные нейронные сети;
- рекурсивные нейронные сети.

Существует ряд вариаций этих архитектур, например гибридная сверточно-рекуррентная нейронная сеть. Но в этой книге мы сосредоточимся только на четырех вышеперечисленных архитектурах.

Автоматическое выделение признаков – еще одно крупное преимущество глубокого обучения над традиционными алгоритмами машинного обучения. Под этим мы понимаем процесс определения сетью тех характеристик набора данных, которые можно надежно использовать для пометки данных. Исторически специалисты по машинному обучению тратили месяцы, годы, а иногда и десятки лет жизни на то, чтобы вручную создать исчерпывающий набор признаков для классификации данных. Со времени Большого взрыва глубокого обучения, начавшегося в 2006 году, необходимость тратить годы ручного труда исчезла, поскольку новейшие алгоритмы машинного обучения способны сами находить признаки, по которым можно классифицировать входные данные. По точности глубокое обучение превзошло традиционные алгоритмы почти для всех типов данных, требуя

[1] https://mitpress.mit.edu/books/ant-colony-optimization.

при этом минимальной настройки и человеческого труда. Глубокие сети помогают коллективам ученых расходовать кровь, пот и слезы на более важные дела.

Вниз по кроличьей норе

Глубокое обучение переплелось со всеми отраслями информатики теснее, чем все прочие методы в недавней истории. С одной стороны, это объясняется высочайшей точностью моделирования, а с другой – порождающими механизмами, очаровывающими даже непрофессионалов. Например, глубокая сеть, обученная на картинах знаменитого художника, оказалась способна генерировать новые картины в его стиле – смотрите рис. 1.2.

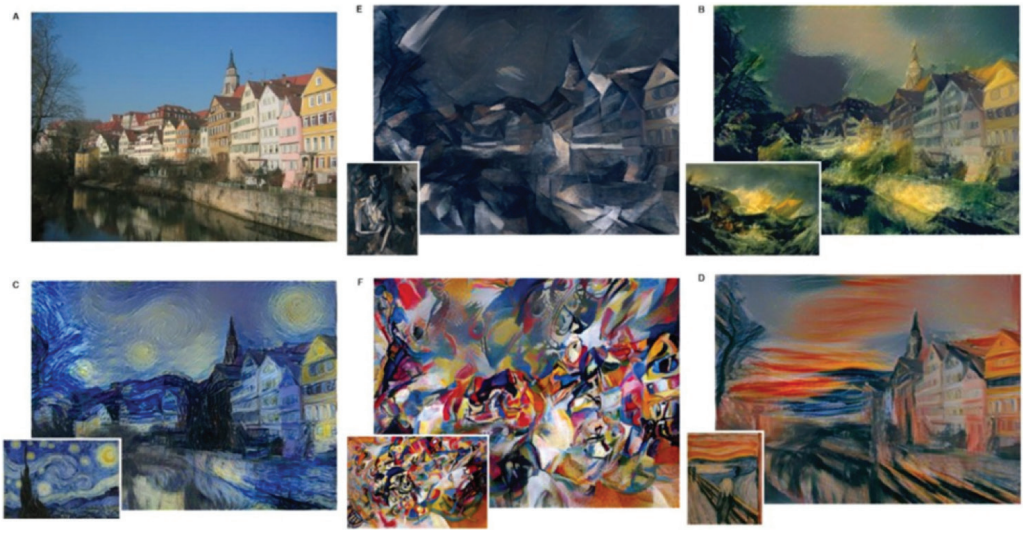

Рис. 1.2 ❖ Стилизованные изображения, взятые из работы Gatys et al., 2015[1]

Это открывает простор для разного рода философских дискуссий, например: «Способна ли машина к творчеству?» и тогда уж «Что такое творчество?». Оставляем эти вопросы вам – поразмыслите на досуге. Машинное обучение развивалось медленно, как смена времен года: потихоньку-полегоньку – и вдруг в один прекрасный день машина становится чемпионом в игре *Jeopardy* или обыгрывает гроссмейстера в го.

Могут ли машины быть разумными или обрести интеллект человеческого уровня? Что такое ИИ и насколько мощным он мог бы стать? На эти вопросы еще предстоит ответить, но не в этой книге. Мы просто хотим проиллюстрировать некоторые фрагменты машинного интеллекта, которые уже сегодня пропитывают наше окружение благодаря практическому применению глубокого обучения.

Более полное обсуждение ИИ
Если вы хотите еще почитать об ИИ, обратитесь к приложению А.

[1] Gatys et. al, 2015. A Neural Algorithm of Artistic Style // https://arxiv.org/pdf/1508.06576v1.pdf.

Формулировка вопросов

Понять, как применяется машинное обучение, проще всего, задавая правильные вопросы. Вот что нам нужно определить:

- что представляют собой исходные данные, из которых мы хотим извлечь информацию (модель)?
- модель какого вида лучше всего отвечает этим данным?
- какого рода ответ мы хотели бы получить от новых данных, основываясь на этой модели?

Если мы сумеем ответить на эти три вопроса, то сможем организовать процесс машинного обучения, в ходе которого будет построена модель и получены желаемые ответы. А для этого вспомним базовые понятия, необходимые для практического применения машинного обучения. Позже мы увидим, как эти понятия соединяются вместе в машинном обучении, и воспользуемся этой информацией, чтобы подвести фундамент под наше понимание нейронных сетей и глубокого обучения.

Математические основания машинного обучения: линейная алгебра

Линейная алгебра – краеугольный камень машинного и глубокого обучения. Она дает математический аппарат для решения уравнений, используемых при построении моделей.

Прекрасным учебником по линейной алгебре является книга James E. Gentle «Matrix Algebra: Theory, Computations, and Applications in Statistics» (http://mason.gmu.edu/~jgentle/books/matbk.htm).

Прежде чем двигаться дальше, рассмотрим некоторые базовые понятия из этой области и начнем со *скаляра*.

Скаляры

В математике термин «скаляр» встречается в основном в контексте элементов вектора. Скаляр – это вещественное число или, в более общем случае, элемент поля, над которым определено векторное пространство.

В информатике скаляр – синоним «переменной», т. е. область памяти вместе с сопоставленным ей символическим именем. В этой области хранится неизвестная величина, называемая *значением*.

Векторы

Нас устроит следующее определение вектора:

> Если n – целое положительное число, то вектором называется n-кортеж, т. е. упорядоченное мультимножество или массив n чисел, называемых элементами, или скалярами.

Сказанное означает, что мы хотим создавать структуру данных, именуемую вектором, посредством процесса *векторизации*. Количество элементов вектора

называется его «порядком», или «длиной». Векторы могут также представлять точки в n-мерном пространстве. При такой интерпретации длина вектора равна евклидову расстоянию между представленной им точкой и началом координат.

В математических текстах векторы часто записываются в виде:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{bmatrix}$$

или

$$x = [x_1, x_2, x_3, \dots, x_n].$$

Существует много способов векторизации. Кроме того, можно применять различные шаги предварительной обработки, получая на выходе модели разной эффективности. Вопрос о преобразовании исходных данных в векторы мы рассмотрим ниже в этой главе, а затем более полно в главе 5.

Матрицы

Будем называть матрицей группу векторов одинаковой размерности (с одинаковым числом столбцов). Таким образом, матрица – это двумерный массив, состоящий из строк и столбцов.

Матрицей размера $n \times m$ называют матрицу, имеющую n строк и m столбцов. На рис. 1.3 изображена матрица 3×3. Матрицы – основная структура в линейной алгебре и машинном обучении, в чем мы убедимся далее в этой главе.

1.0	0.0	0.0
0.0	1.0	0.0
0.0	0.0	1.0

Рис. 1.3 ❖ Матрица 3×3

Тензоры

Тензором называется многомерный массив. Вектор можно считать частным случаем тензора.

В тензорах строки расположены вдоль оси y, а столбцы – вдоль оси x. Каждая ось является измерением, и у тензоров имеются дополнительные измерения, помимо

x и y. Тензор также характеризуется рангом. Так, скаляр имеет ранг 0, вектор – ранг 1, а матрица – ранг 2. Структуры ранга 3 и выше называются тензорами.

Гиперплоскости

Еще один полезный объект из линейной алгебры – *гиперплоскость*. В геометрии гиперплоскостью называется подпространство, размерность которого на единицу меньше размерности объемлющего пространства. В трехмерном случае размерность гиперплоскости равна 2, а в двумерном гиперплоскостью считается одномерная прямая.

Гиперплоскость делит *n*-мерное пространство на две части и потому имеет полезные применения в приложениях классификации. Оптимизация параметров гиперплоскости – ключевая идея линейного моделирования, как мы увидим в этой главе ниже.

Математические операции

В этом разделе мы кратко рассмотрим наиболее распространенные в линейной алгебре операции.

Скалярное произведение

В машинном обучении часто встречается операция *скалярного произведения* (иногда ее еще называют «внутренним произведением»). Скалярное произведение применяется к двум векторам одинаковой длины и возвращает одно число. Для этого соответственные элементы векторов перемножаются, а произведения суммируются. Не вдаваясь пока в математические детали, отметим, что в этом числе закодировано много информации.

Прежде всего скалярное произведение показывает, насколько велики отдельные элементы обоих векторов. Произведение двух векторов с большими элементами будет большим, а с малыми – малым. Если применить к значениям элементов математическую процедуру *нормировки*, т. е. принимать во внимание не абсолютные, а относительные значения, то скалярное произведение векторов будет мерой их похожести. Скалярное произведение нормированных векторов называется *косинусным коэффициентом*, или *коэффициентом Отиаи*.

Поэлементное произведение

На практике также часто встречается операция *поэлементного произведения*, или *произведения Адамара*. Она применяется к двум векторам одинаковой длины и возвращает вектор той же длины, каждый элемент которого равен произведению соответственных элементов векторов-сомножителей.

Внешнее произведение

Оно называется еще *тензорным произведением* двух векторов. Каждый элемент вектора-столбца умножается на все элементы вектора-строки, в результате чего создается новая строка в результирующей матрице.

Преобразование данных в векторы

В машинном обучении и науке о данных требуется анализировать данные самых разных типов. Ключевое требование – возможность представить тип дан-

ных в виде вектора. В машинном обучении рассматривается много типов данных (текст, временные ряды, звук, изображения, видео и т. п.).

Так почему бы не подать на вход алгоритма обучения исходные данные – и пусть обрабатывает? Проблема в том, что машинное обучение основано на линейной алгебре и решении систем уравнений. Решатель уравнений ожидает получить на входе числа с плавающей точкой, поэтому нужно каким-то образом преобразовать исходные данные во множество чисел с плавающей точкой. Мы соединим эти две концепции в следующем разделе, посвященном решению систем уравнений. Примером исходных данных является канонический набор данных Iris об ирисах (http://archive.ics.uci.edu/ml/datasets/Iris):

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
```

Еще один пример – текстовый документ:

```
Go, Dogs. Go!
Go on skates
or go by bike.
```

В этих двух случаях мы имеем данные разного типа, но для каждого необходима та или иная векторизация, чтобы их можно было использовать в машинном обучении. На каком-то этапе входные данные должны быть представлены в виде матрицы, но можно сначала преобразовать их в некий промежуточный формат (например, в формат «svmlight», показанный в примере кода ниже). Мы хотим, чтобы входные данные для алгоритма машинного обучения выглядели как сериализованный разреженный вектор в следующем формате svmlight:

```
1.0 1:0.7500000000000001 2:0.41666666666666663 3:0.702127659574468 4:0.5652173913043479
2.0 1:0.6666666666666666 2:0.5 3:0.9148936170212765 4:0.6956521739130436
2.0 1:0.45833333333333326 2:0.3333333333333336 3:0.8085106382978723 4:0.7391304347826088
0.0 1:0.1666666666666665 2:1.0 3:0.021276595744680823
2.0 1:1.0 2:0.5833333333333334 3:0.9787234042553192 4:0.8260869565217392
1.0 1:0.3333333333333333 3:0.574468085106383 4:0.47826086956521746
1.0 1:0.7083333333333336 2:0.7500000000000002 3:0.6808510638297872 4:0.5652173913043479
1.0 1:0.916666666666667 2:0.6666666666666667 3:0.7659574468085107 4:0.5652173913043479
0.0 1:0.08333333333333343 2:0.5833333333333334 3:0.021276595744680823
2.0 1:0.6666666666666666 2:0.8333333333333333 3:1.0 4:1.0
1.0 1:0.9583333333333335 2:0.7500000000000002 3:0.723404255319149 4:0.5217391304347826
0.0 2:0.7500000000000002
```

Из данных в таком формате легко получить матрицу и вектор-столбец меток (первое число в каждой строке). После метки идут пары «номер столбца:значение», которые загружаются в соответствующие позиции матрицы и интерпретируются

как «признаки». Полученная матрица готова к различным операциям линейной алгебры в алгоритме машинного обучения. Более подробно процесс векторизации обсуждается в главе 8.

Зададимся стандартным вопросом: «Почему алгоритмам машинного обучения (обычно) требуются данные, представленные в виде (разреженной) матрицы?» Для ответа на него сделаем небольшой экскурс в область решения систем уравнений.

Решение систем уравнений

В линейной алгебре нас интересует решение систем линейных уравнений вида:

$$Ax = b,$$

где A – матрица, образованная входными векторами-строками, а b – вектор-столбец меток, соответствующих каждой строке матрицы A. Если взять первые три строки сериализованного разреженного представления из предыдущего примера и представить их в линейно-алгебраической форме, то получим:

```
Столбец 1           Столбец 2           Столбец 3           Столбец 4
0.7500000000000001  0.41666666666666663 0.702127659574468   0.5652173913043479
0.6666666666666666  0.5                 0.9148936170212765  0.6956521739130436
0.45833333333333326 0.3333333333333336  0.8085106382978723  0.7391304347826088
```

Эта числовая матрица и есть переменная A в нашем уравнении, а независимые значения в каждой строке считаются признаками входных данных.

Что такое признак?

В машинном обучении признаком называется любой столбец матрицы A, рассматриваемый как независимая переменная. В качестве признаков можно брать непосредственно исходные данные, но обычно производится некоторое преобразование исходных данных в форму, более подходящую для моделирования.

Примером может служить столбец входных данных, который изначально содержал четыре разные текстовые метки. Мы должны просмотреть все исходные данные и присвоить каждой метке числовой индекс. Затем четыре значения (0, 1, 2, 3) следует нормировать, приведя к интервалу от 0.0 до 1.0, учитывая, сколько раз каждый индекс встречается в строках. Такого рода преобразования очень способствуют нахождению хороших решений задач моделирования. В главе 5 мы подробнее поговорим о методах векторизации и преобразованиях.

Мы хотим найти для каждого столбца такие коэффициенты, которые после умножения и суммирования дадут вектор-столбец меток b. В сериализованном разреженном представлении были такие метки:

```
Метки
1.0
2.0
2.0
```

Вышеупомянутые коэффициенты образуют вектор-столбец x (называемый также *вектором-параметром*), показанный на рис. 1.4.

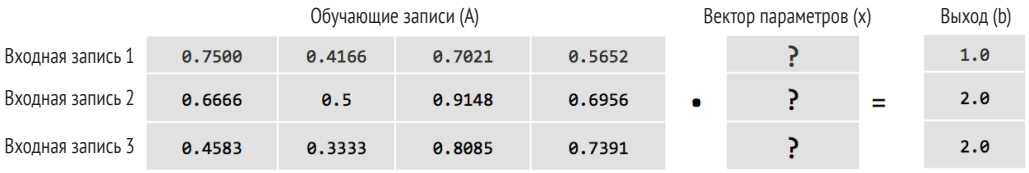

Рис. 1.4 ❖ Визуализация уравнения Ax = b

Говорят, что эта система уравнений *совместна*, если существует вектор параметров x – такой, что решение можно записать в виде:

$x = A^{-1}b$.

Важно отличать выражение $x = A^{-1}b$ от метода вычисления решения. Это выражение всего лишь представляет решение. Переменная A^{-1} – это матрица, обратная A, которая вычисляется с помощью процедуры *обращения матрицы*. Учитывая, что не все матрицы обратимы, нам нужен такой метод решения уравнения, который не прибегает к обращению матрицы. Один из таких методов называется *разложением* (или *декомпозицией*) *матрицы*. Примером может служить LU-разложение матрицы A. Рассмотрим также общие методы решения систем линейных уравнений, помимо разложения матриц.

Методы решения систем линейных уравнений

Существуют два подхода к решению систем линейных уравнений. Первый – прямые методы – заключается в применении фиксированных вычислений, объем которых известен заранее. Другой подход – так называемые *итеративные методы* – заключается в нахождении последовательных приближений к вектору параметров x, при этом процесс прекращается, как только будут выполнены заранее заданные условия завершения. Прямые методы особенно эффективны, если все обучающие данные (A и b) помещаются в памяти одного компьютера. Хорошо известными примерами прямых методов являются *метод исключения Гаусса* и *метод нормальных уравнений*.

Итеративные методы

Итеративные методы особенно эффективны, когда данные не помещаются в оперативную память одного компьютера, а загрузка записей с диска в цикле позволяет включить в модель гораздо больший объем данных. Канонический пример итеративного метода, повсеместно встречающийся в современном машинном обучении, – стохастический градиентный спуск (СГС), мы обсудим его ниже в этой главе. Упомянем также *метод сопряженных градиентов* и *метод чередующихся наименьших квадратов* (обсуждаются в главе 3). Итеративные методы также доказали свою эффективность в горизонтально масштабируемых конфигурациях, когда весь набор данных распределен между машинами из одного кластера и вектор параметров периодически усредняется по всем агентам, а затем обновляется на каждом локальном агенте моделирования (подробное объяснение отложим до главы 9).

Итеративные методы и линейная алгебра

Мы хотели бы применить эти алгоритмы к операциям с входным набором данных. Это означает, что исходные данные следует преобразовать во входную матрицу

A. Краткий обзор линейной алгебры отвечает на вопрос, *зачем* нужно подвергать данные векторизации. В этой книге мы на примерах кода продемонстрируем преобразование исходных данных в матрицу *A* и тем ответим на вопрос – *как*. Механизм векторизации данных также оказывает влияние на результаты процесса обучения. Ниже мы увидим, что этап предварительной обработки данных еще до векторизации может привести к созданию более точных моделей.

Математические основания машинного обучения: статистика

Мы приведем ровно столько сведений из области статистики, сколько необходимо, чтобы двигаться дальше. Необходимо вспомнить некоторые базовые понятия:

- вероятность;
- распределения;
- правдоподобие.

Существуют и другие важные понятия, относящиеся к описательной статистике и индуктивной статистике. К описательной статистике относятся:

- гистограммы;
- коробчатые диаграммы;
- диаграммы рассеяния;
- среднее;
- стандартное отклонение;
- коэффициент корреляции.

С другой стороны, предмет индуктивной статистики – обобщение с выборки на всю генеральную совокупность. Вот несколько примеров индуктивной статистики:

- p-значения;
- интервалы правдоподобности (credibility interval).

Сформулируем соотношение между вероятностью и индуктивной статистикой:

- вероятность подразумевает переход от генеральной совокупности к выборке (дедуктивное рассуждение);
- индуктивная статистика подразумевает переход от выборки к генеральной совокупности.

Чтобы понять, что конкретная выборка говорит нам об исходной генеральной совокупности, нужно сначала оценить недостоверность, связанную с извлечением выборки из заданной генеральной совокупности.

Говоря об общей статистике, мы не будем задерживаться на весьма обширной проблематике, глубоко раскрытой в других книгах. Этот раздел ни в коей мере не является сколько-нибудь полным обзором статистики, мы лишь хотели обозначить темы, которые можно более подробно изучить по другим источникам. Теперь, оградив себя от критики, начнем с определения вероятности в статистике.

Вероятность

По определению, вероятность события *E* – число от 0 до 1. Вероятность 0 означает, что событие *E* вообще не может произойти, а вероятность 1 – что оно произойдет наверняка. Часто вероятность выражается в виде числа с плавающей точкой, но

иногда также в виде процентной величины от 0 до 100%: вероятность не может быть меньше 0 и больше 100 %. Например, вероятность 0.35 можно выразить как 35% (0.35 × 100 == 35%).

Канонический пример измерения вероятности – наблюдение за тем, сколько раз выпали орел и решка при подбрасывании правильной монеты (такой, что каждая сторона выпадает с вероятностью 0.5). Вероятность выборочного пространства всегда равна 1, потому что оно представляет все возможные исходы испытания. Как мы видим на примере двух исходов (выпадение орла и решки), 0.5 + 0.5 = 1.0, потому что полная вероятность выборочного пространства должна быть равна 1. Вероятность события записывается в виде:

$P(E) = 0.5$.

Эта формула читается так:

Вероятность события *E* равна 0.5.

Вероятность и шанс

Начинающие изучать статистику и машинное обучение часто путают смысл слов «вероятность» и «шанс». Давайте разберемся.
Вероятность события *E* определяется следующим образом:

$P(E)$ = (Число благоприятных для *E* исходов) / (Общее число исходов).

Так, вероятность вытянуть туза (4 штуки) из колоды карт (52 штуки) равна:

4/52 = 0.077.

С другой стороны, шанс (odds) определяется следующим образом:

(Число благоприятных для *E* исходов) : (Число неблагоприятных для *E* исходов).

В примере с картами «шанс вытянуть туза» равен:

4 : (52 – 4) = 1/12 = 0.0833333...

Отличие – в знаменателе: «общее число исходов» и «число неблагоприятных для *E* исходов».

Понятие вероятности является центральным для нейронных сетей и глубокого обучения в силу своей роли в выделении признаков и классификации – двух основных функций глубоких нейронных сетей. Более полное изложение математической статистики см. в книге Boslaugh and Watters «Statistics in a Nutshell: A Desktop Quick Reference», выпущенной издательством O'Reilly (http://shop.oreilly.com/product/0636920023074.do).

Еще об определении вероятности: частотный и байесовский подходы

В статистике имеются два разных подхода к определению вероятности: *байесовский* и *частотный*.
Сторонники частотного подхода считают, что говорить о вероятности имеет смысл только в контексте повторяющихся измерений. Измеряя некую величину, мы наблюдаем небольшие вариации из-за физических особенностей оборудования, исполь-

зуемого для сбора данных. При многократном повторении измерения частота конкретного значения определяет его вероятность.
При байесовском подходе вероятность связывают с достоверностью утверждений. Вероятность характеризует степень нашей уверенности в том, каким будет результат измерения. Сторонники байесовского подхода считают, что наши знания о событии по сути своей связаны с его вероятностью.
Сторонники частотного подхода полагаются на большое число слепых испытаний и только после этого готовы дать оценку случайной величины. А сторонники байесовского подхода имеют дело с «верой» (математическим «распределением») относительно случайной величины и обновляют эту веру по мере поступления новой информации.

Условные вероятности

Если мы хотим знать, какова вероятность данного события при условии, что произошло некоторое другое событие, то говорим об *условной вероятности*. В литературе условная вероятность записывается в виде:

$P(E|F)$,

где E – событие, вероятность которого нас интересует; F – событие, которое уже произошло.

В качестве примера выразим тот факт, что для человека с нормальной частотой сердечных сокращений вероятность умереть в палате интенсивной терапии ниже:

P(смерть в ПИТ | плохой пульс) > P(смерть в ПИТ | хороший пульс).

Иногда второе событие F называют «условием». Условная вероятность представляет интерес для машинного и глубокого обучений, потому что часто нам важно знать, как взаимодействуют различные факторы. В контексте машинного обучения мы обучаем классификатор путем определения условной вероятности

$P(E|F)$,

где E – метка, а F – ряд атрибутов сущности, для которой мы хотим предсказать E. Примером может служить предсказание вероятности смерти (E), если известны результаты измерений, сделанных в палате интенсивной терапии для каждого пациента (F).

Теорема Байеса

Одно из основных применений условных вероятностей дает теорема (или формула) Байеса. В медицине она применяется для вычисления вероятности того, что пациент, сдавший положительный анализ на некоторое заболевание, действительно болен им. Формула Байеса для двух событий A и B имеет вид:

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

Апостериорная вероятность

В байесовской статистике апостериорной вероятностью случайного события называется условная вероятность после наблюдения факта. Апостериорное распре-

деление вероятности определяется как распределение вероятности неизвестной величины, обусловленное фактическими данными, собранными в результате эксперимента и рассматриваемыми как случайная величина. Как эта идея применяется на практике, мы увидим ниже на примере функции активации softmax, которая преобразует входные значения в апостериорные вероятности.

Распределения вероятности

Распределение вероятности – это описание стохастической структуры случайных величин. В статистике мы высказываем предположения о распределении данных, чтобы делать о них какие-то выводы. Нам нужна формула, описывающая частоту наблюдаемых значений величины с данным распределением. Часто встречается *нормальное распределение* (называемое также *гауссовым*, или *колоколообразной кривой*). Мы хотим аппроксимировать набор данных распределением, поскольку если набор данных действительно близок к распределению, то о данных можно высказывать гипотезы, исходя из теоретических свойств распределения.

Выделяют *непрерывные* и *дискретные* распределения. Дискретная случайная величина может принимать значения из дискретного множества, а непрерывная – значения из некоторого диапазона. Нормальное распределение – пример непрерывного распределения. Примером дискретного может служить биномиальное распределение.

Нормальное распределение (см. рис. 5.1) было открыто Карлом Гауссом, математиком и физиком, жившим в XVIII веке. Оно полностью определяется своим средним и стандартным отклонением и имеет одинаковую характерную форму для любых параметров.

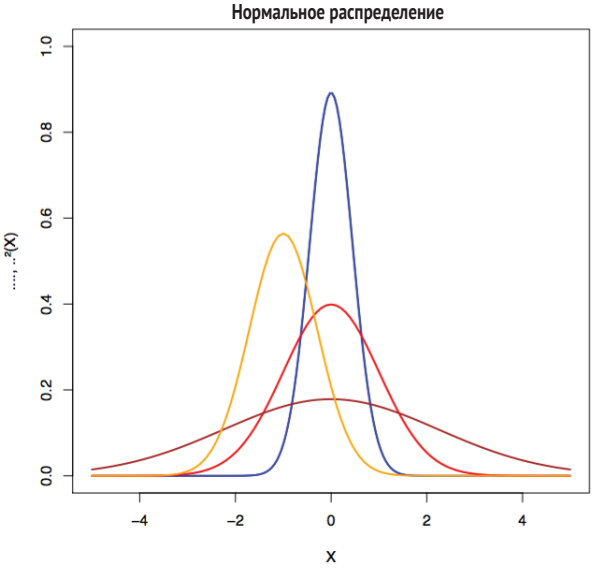

Рис. 1.5 ❖ Примеры нормальных распределений

Из других распределений, встречающих в машинном обучении, отметим:

- биномиальное распределение;

- обратное нормальное распределение;
- логнормальное распределение.

Понимать, как распределены обучающие данные, важно, чтобы правильно векторизовать данные для моделирования.

Центральная предельная теорема

Если размер выборки достаточно велик, то выборочное распределение выборочных средних аппроксимируется нормальным распределением. Это справедливо независимо от распределения генеральной совокупности, из которой производилась выборка.

Это позволяет делать статистические выводы, применяя критерии, основанные на приближенной нормальности среднего. И выводы остаются справедливыми, даже если выборка произведена не из нормально распределенной генеральной совокупности.

В информатике мы столкнемся с этим в алгоритмах, которые многократно производят выборку заданного размера из ненормальной генеральной совокупности. Построив гистограмму выборочной совокупности выборок из нормального распределения, мы увидим этот эффект в действии.

Распределения с длинным хвостом (например, Zipf, степенные законы и распределение Парето) – ситуация, когда за областью значений с высокой частотой следует асимптотически убывающая область значений с низкой частотой. Такие распределения изучались Бенуа Мандельбротом в 1950-х годах, а затем популяризированы в книге Криса Андерсона «The Long Tail: Why the Future of Business is Selling Less of More»[1].

Примером может служить ранжирование товаров в розничном магазине: несколько товаров пользуется очень большой популярностью, но наряду с ними имеется много уникальных товаров, которые продаются в небольших объемах. Такое распределение ранга по частоте (в основном популярности или «количеству проданных») часто подчиняется степенному закону. Поэтому мы можем рассматривать его как распределение с длинным хвостом.

Распределения с длинным хвостом встречаются в следующих случаях:

- *ущерб от землетрясения* – ущерб тем выше, чем сильнее землетрясение. Ущерб в худшем случае сдвигается в область длинного хвоста;
- *урожайность* – иногда наблюдаются события, выходящие за пределы исторических данных. Но в целом модель сосредоточена вблизи средних значений;
- *прогнозирование смертности после пребывания в палате интенсивной терапии* – на смертность могут оказывать влияние события, слабо связанные с тем, что происходило в палате интенсивной терапии.

Эти примеры имеют отношение к рассматриваемым в книге задачам классификации, потому что большинство статистических моделей опирается на выводы, сделанные на основе очень больших объемов данных. Если наиболее интересные

[1] *Крис А.* Длинный хвост. Эффективная модель Интернета в бизнесе. М.: Манн, Иванов и Фербер, 2002.

события случаются в хвосте распределения, а он не представлен в обучающей выборке, то модель может работать непредсказуемо. Этот эффект усиливается в нелинейных моделях, к каковым относятся и нейронные сети. Мы рассматриваем эту ситуацию как частный случай проблемы «входит в выборку/не входит в выборку». Даже профессионал в области машинного обучения может поразиться тому, что модель, прекрасно работающая на асимметричной обучающей выборке, не обобщается на большую совокупность данных.

Распределения с длинным хвостом описывают реальную возможность событий, отстоящих от среднего более чем на пять стандартных отклонений. Мы должны следить за тем, чтобы в обучающих данных были представлены и такие события, чтобы предотвратить переобучение. О том, как этого добиться, мы поговорим подробнее ниже, когда перейдем к переобучению, а также в главе 4, посвященной настройке модели.

Выборка и генеральная совокупность

Под генеральной совокупностью понимается все множество данных, которые мы собираемся изучать или моделировать в эксперименте. Примером может служить «множество всех Java-программистов в штате Теннесси».

Выборкой называется подмножество генеральной совокупности, которое предположительно достаточно точно представляет распределение данных, не внося смещения (обусловленного способом извлечения выборки).

Методы с перевыборкой

Бутстрэппинг и *перекрестная проверка* – два типичных статистических метода с перевыборкой, полезных в машинном обучении. В случае бутстрэппинга мы случайным образом выбираем примеры из другой выборки с целью сгенерировать новую выборку, в которой представители каждого класса были бы сбалансированы. Это полезно при моделировании набора данных с сильно разбалансированными классами.

Перекрестная проверка позволяет оценить, насколько хорошо обобщается обученная модель. Весь набор данных разбивается на N порций, а затем каждая порция разделяется на обучающий и тестовый наборы. Сначала модель обучается на обучающих наборах, а затем проверяется на тестовых. Порции меняются местами, пока не будут исчерпаны все варианты. На количество порций не налагается никаких ограничений, но на практике обычно берут 10 порций, это дает хорошие результаты. Нередко часть данных резервируют под контрольный набор, используемый во время обучения.

Смещение выборки

Смещение возникает, когда метод выборки не обеспечивает надлежащую рандомизацию, вследствие чего выборка оказывается нерепрезентативной для моделируемой генеральной совокупности. Мы должны помнить о смещении, производя перевыборку из набора данных, чтобы не внести искажений в модель, что приведет к снижению ее верности на данных из большей генеральной совокупности.

Правдоподобие

Обсуждая возможность события, мы не указываем его числовую вероятность, а употребляем менее формальный термин – *правдоподобие*. Обычно мы имеем в виду, что событие может произойти с довольно высокой вероятностью, но все же не наверняка. На событие могут повлиять пока еще не наблюдавшиеся факторы. Но в целом слово «правдоподобие» используется как неформальный синоним «вероятности».

Как работает машинное обучение?

В предыдущем разделе, говоря о решении систем линейных уравнений, мы ввели обозначение $Ax = b$. По существу, машинное обучение сводится к алгоритмам, которые минимизируют погрешность решения этого уравнения посредством *оптимизации*.

Оптимизация означает, что мы изменяем элементы вектора параметров x, пока не найдем такой набор значений, при котором получается наилучшее приближение к значениям b. Весовая матрица корректируется после вычисления функции потерь (основанной на расхождении с вектором-столбцом b), т. е. ошибки, порожденной сетью. Матрица ошибок, приписывающая некоторую часть потери каждому весу, умножается на сами веса.

Ниже в этой главе мы обсудим метод стохастического градиентного спуска – один из основных методов оптимизации в машинном обучении, а далее в книге рассмотрим и другие алгоритмы оптимизации. Мы также поговорим о гиперпараметрах и, в частности, о регуляризации и скорости обучения.

Регрессия

Термин «регрессия» относится к функциям, пытающимся предсказать вещественное значение. Функции такого типа оценивают зависимую величину, зная независимую. Наиболее распространена *линейная регрессия*, в основе которой лежат идеи, описанные выше в контексте моделирования систем линейных уравнений. Задача линейной регрессии заключается в отыскании функции, описывающей связь между x и y, так чтобы для известных значений x предсказанные ей значения y были верны.

Структура модели

Предсказание, порождаемое линейной регрессионной моделью, является линейной комбинацией коэффициентов (вектора параметров x) и входных величин (входного вектора). Это можно выразить в виде следующего уравнения:

$$y = a + Bx,$$

где a – свободный член, B – входные признаки, x – вектор параметров.

В скалярном виде уравнение записывается так:

$$y = a + b_0 * x_0 + b_1 * x_1 + \ldots + b_n * x_n.$$

Простой пример задачи линейной регрессии – предсказание месячных расходов на бензин на основании длины маршрута. В данном случае сумма, уплаченная на заправке, является функцией расстояния, которое вы проехали. Стоимость

бензина – зависимая величина, а длина маршрута – независимая. Мы можем связать эти величины функциональной зависимостью:

стоимость = *f* (*расстояние*).

Это позволяет дать разумный прогноз расходов на бензин в зависимости от длины поездки.

Вот еще несколько примеров линейных регрессионных моделей:

- предсказание веса как функции роста;
- предсказание продажной цены дома как функции его площади.

Визуализация линейной регрессии

Визуально линейную регрессию можно представить как отыскание прямой линии, которая проходит максимально близко к максимально возможному числу экспериментальных точек, как показано на рис. 1.6.

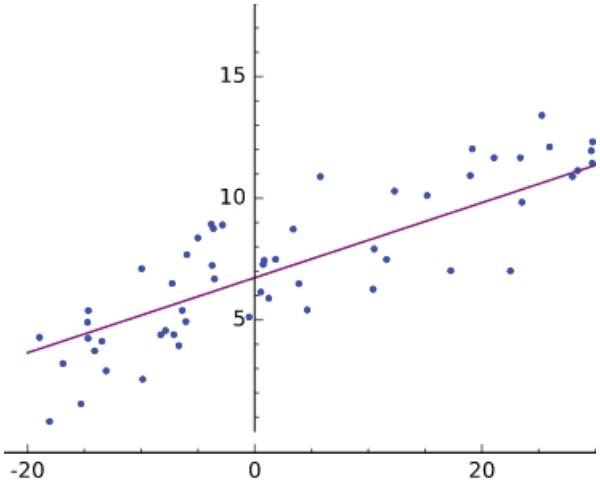

Рис. 1.6 ❖ График линейной регрессии

Подгонкой (fitting), или аппроксимацией, называется нахождение функции $f(x)$, значения которой близки к измеренным значениям y.

Линейная регрессионная модель и система уравнений

Мы можем связать эту функцию с приведенным выше уравнением $Ax = b$, где A – признаки (например, «вес» или «площадь дома») для всех входных примеров модели. Каждая входная запись – строка матрицы A. Вектор-столбец b – известные метки, соответствующие входным записям. Применяя некоторую функцию ошибок и метод оптимизации (например, СГС), мы можем найти такой набор параметров x, который минимизирует отклонение от истинных меток на всем множестве входных данных.

Для применения СГС нам понадобятся три вещи:

1) гипотеза о данных – произведение вектора параметров x и матрицы входных признаков;
2) функция стоимости – квадрат ошибки (предсказание – факт);

3) функция обновления – производная квадратичной функции потерь (функции стоимости).

Если линейная регрессия имеет дело с прямыми линиями, то нелинейная аппроксимация – со всем остальным, чаще всего с полиномами от *x* степени, большей 1. (Потому-то машинное обучение иногда описывают как «подбор аппроксимирующей кривой».) Точная аппроксимация означает, что кривая проходит через все экспериментальные точки. Но точная аппроксимация – обычно очень плохой результат, поскольку это означает, что модель идеально подогнана к обучающему набору и за его пределами не имеет почти никакой предсказательной силы (т. е. не обобщается).

Классификация

Под классификацией понимают моделирование, цель которого – разделить классы данных на основе некоторого множества входных признаков. Если регрессия отвечает на вопрос «сколько», то классификация – на вопрос «какого вида». Зависимая величина *y* не числовая, а категориальная.

Самая простая форма классификации – бинарный классификатор, порождающий на выходе одну из двух меток (всего два класса: 0 и 1). Результатом классификации может быть также число с плавающей точкой от 0.0 до 1.0, означающее степень уверенности. В этом случае требуется задать пороговую величину (обычно 0.5), определяющую границу между двумя классами. В литературе классы обычно называют положительным (например, 1.0) и отрицательным (например, 0.0). Мы еще вернемся к этому вопросу в разделе «Оценивание моделей» ниже.

Вот несколько примеров бинарной классификации:

- имеется у пациента некоторое заболевание или нет;
- является почтовое сообщение спамом или нет;
- является банковская транзакция мошеннической или законной.

Существуют модели классификации не только с двумя, но и с *N* метками, когда мы вычисляем для каждой метки оценку, а затем в качестве результирующей выбираем метку с наивысшей оценкой. Мы вернемся к этой теме, когда будем говорить о нейронных сетях с несколькими выходами. А в этой главе мы еще обсудим классификацию, когда перейдем к логистической регрессии и архитектурам нейронных сетей в целом.

Рекомендование

Рекомендованием называется процесс предложения товаров пользователю системы в зависимости от того, какие товары он просматривал прежде или какие товары покупают похожие на него пользователи. Один из самых известных рекомендательных алгоритмов – *коллаборативная фильтрация* – стал знаменитым благодаря сайту Amazon.com.

Кластеризация

Кластеризация – это метод обучения без учителя, в котором измеряется расстояние между объектами и похожие объекты собираются вместе. В конце процесса объекты, концентрирующиеся вокруг *n* центроидов, считаются принадлежащими одной группе – кластеру. Метод *K* средних – один из самых известных алгоритмов кластеризации в машинном обучении.