Содержание

От издательства

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 на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и 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 обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Вступительное слово

Хочу открыть вам один секрет: когда говорят «машинное обучение», складывается впечатление, будто речь идет только об одной дисциплине. А вот и нет! На самом деле есть два вида машинного обучения, и они различаются так же, как придумывание новых блюд и изобретение кухонных принадлежностей. То и другое занятия достойны уважения, но путать их не надо: не станете же вы поручать шеф-повару разработку кухонной плиты, а инженеру-электрику – печь хлеб!

Увы, почти все путают эти два вида машинного обучения. Поэтому неудивительно, что так много компаний терпят крах на этапе машинного обучения. Начинающим, похоже, никто не объяснил, что все то, что преподают на курсах машинного обучения и пишут в учебниках, – это теоретическое машинное обучение: как конструировать кухонные плиты (а также микроволновки, блендеры, тостеры, чайники … и мойку, в которой все смешивается!) с нуля, а не как придумывать новые блюда и готовить на огромную компанию. Иными словами, если ваша цель – создавать инновационные решения конкретных задач на базе МО, то вам нужно прикладное, а не теоретическое машинное обучение. Поэтому большинство книг – не для вас.

А теперь хорошая новость! Перед вами первая книга, посвященная именно прикладному машинному обучению. Да, вы ее нашли! Настоящая прикладная иголка в стоге теоретического сена. Поздравляю, любезный читатель… если, конечно, вы не искали книгу, которая помогла бы отточить навыки проектирования алгоритмов общего назначения. Но тогда я надеюсь, что автор не будет на меня в обиде за то, что я посоветую вам купить чуть ли не любую другую книгу по МО. А эта от них отличается.

В 2016 году, разрабатывая курс Google по прикладному МО «Как подружиться с машинным обучением», полюбившийся нашим инженерам и руководителям групп – а их больше десяти тысяч, – я придала ему примерно такую же структуру, как в этой книге. А все потому, что действовать в правильном порядке очень важно в прикладных дисциплинах. Попытка выполнить определенные шаги, прежде чем будут завершены другие, может иметь печальные последствия: от зря потраченного времени до феерического краха проекта. На самом деле сходство оглавления этой книги и моего курса как раз и побудило меня прочитать ее. И в полном согласии с теорией параллельной эволюции я узрела в авторе единомышленника, терзаемого по ночам отсутствием ресурсов по прикладному машинному обучению, одной из потенциально самых полезных и вместе с тем самых недопонятых отраслей инженерной практики, – терзаемого с такой силой, что он захотел как-то исправить ситуацию. Так что если вы вознамерились захлопнуть книгу, то, пожалуйста, сделайте мне одолжение – задумайтесь на минутку, почему главы следуют именно в таком порядке. Обещаю, уже это принесет полезные плоды.

Так что же будет в книге дальше? Машинное обучение можно сравнить с руководством по приготовлению еды в массовом масштабе. Поскольку вы

еще не прочитали книгу, опишу содержание книги в кулинарных терминах. Вам нужно определить, что имеет смысл приготовить и каковы цели (*принятие решений и управление продуктом*), понять, кто является поставщиками и клиентами (*знакомство с предметной областью и деловое чутье*), как обрабатывать ингредиенты в большом количестве (*инженерия и анализ данных*), как быстро оценивать много разных сочетаний ингредиентов, пригодных для создания потенциальных рецептов блюд (*инженерная разработка прототипа*), как проверить качество рецепта (*статистика*), как эффективно превратить потенциальный рецепт в миллионы порций (*организация производства*) и как гарантировать, что блюда останутся высококачественными, даже если грузовик привез тонну картошки вместо заказанного риса (*обеспечение надежности*). Эта книга – одна из немногих, где рассматриваются все этапы технологического процесса.

Теперь самое время подпустить немного дегтя в бочку меда. Это хорошая книга. Правда. Но не идеальная. Иногда автор срезает углы – как часто делают инженеры, профессионально занимающиеся машинным обучением, – хотя в целом все изложено правильно. А поскольку предмет книги быстро эволюционирует, автор и не пытается всегда быть на переднем крае. Но даже в отсутствие совершенства книгу все равно стоит прочитать. Учитывая, как мало на рынке подробных руководств по прикладному машинному обучению, ясное и последовательное введение в эту тему дорогого стоит. Я очень рада, что такая книга появилась!

Что мне в ней очень нравится, так это полнота, с которой излагается самая важная вещь, которую нужно знать о машинном обучении: ошибки возможны… и иногда болезненные ошибки. Как любят говорить мои коллеги, занимающиеся надежностью: «Надежда на лучшее – это не стратегия». Надеяться, что ошибок не будет, – худший из возможных подходов. Эта книга устроена иначе – и лучше. Она быстро заставляет расстаться с ложным чувством безопасности, которое вы могли лелеять, надеясь построить систему ИИ, более «умную», чем вы сами. (Забудьте об этом, ничего не получится.) Затем автор становится вашим заботливым проводником по пути, на котором можно наделать самых разнообразных ошибок, и объясняет, как их предотвратить, обнаружить и исправить. В книге прекрасно подчеркнута важность мониторинга, описаны подходы к сопровождению модели, рассказано, что делать, если что-то поломается, как спланировать резервные стратегии на случай разного рода отказов и как управлять ожиданиями пользователей (есть также раздел о том, что делать, если пользователями являются машины). Эти вопросы очень важны при практическом применении машинного обучения, но в других книгах для них часто не находится места. Впрочем, эта книга не из их числа.

Если вы собираетесь использовать машинное обучение для решения крупномасштабных практических задач, могу только порадоваться, что вы наткнулись на эту книгу. Читайте с удовольствием!

Кэсси Козырков,
главный специалист по теории принятия решений в Google,
автор курса «MakingFriends with Machine Learning»
на облачной платформе Google

Сентябрь 2020

Предисловие

За последние несколько лет машинное обучение (МО) для многих стало синонимом искусственного интеллекта. И хотя машинное обучение как научная дисциплина существует уже несколько десятков лет, в мире найдется лишь горстка организаций, в полной мере осознавших его потенциал.

Несмотря на доступность современных библиотек, пакетов и каркасов машинного обучения с открытым исходным кодом, которые поддерживаются ведущими организациями и широким сообществом ученых и программистов, большинство компаний все еще испытывают трудности с применением машинного обучения к решению практических деловых задач.

Одна из проблем – нехватка кадров. Но, даже располагая талантливыми специалистами по машинному обучению и анализу данных, в 2020 году большинство организаций все еще тратят от 31 до 90 дней на развертывание всего одной модели, а 18 % – более 90 дней, у некоторых на доведение идеи до продукта уходит даже больше года. Основные проблемы, с которыми приходится сталкиваться при освоении потенциала МО, например управление версиями модели, воспроизводимость результатов и масштабирование, имеют скорее инженерную, чем научную природу.

Существует много книг по машинному обучению – как теоретических, так и практических. Из типичного учебника вы можете узнать о разных типах машинного обучения, об основных семействах алгоритмов, о том, как они работают и как с их помощью создавать модели из данных.

В типичном учебнике меньше внимания уделяется инженерным аспектам реализации проектов машинного обучения. Такие вопросы, как сбор, хранение и предобработка данных, конструирование признаков, а также тестирование и отладка моделей, развертывание и вывод из эксплуатации, сопровождение на этапе выполнения и в процессе эксплуатации, зачастую остаются за кадром.

Эта книга призвана заполнить пробел.

На кого рассчитана эта книга

Я предполагаю, что читатель знаком с основами машинного обучения и способен построить модель при наличии подходящим образом отформатированного набора данных, применяя свой любимый язык программирования или библиотеку. Если вы неуверенно владеете применением алгоритмов машинного обучения к данным и не видите разницы между логистической регрессией, методом опорных векторов и случайным лесом, то я рекомендую предварительно прочитать мою книгу «The Hundred-Page Machine Learning Book»[1].

---

[1] *Бурков А.* Машинное обучение без лишних слов. СПб.: Питер, 2020.

Целевая аудитория этой книги – аналитики данных, стремящиеся стать инженерами по машинному обучению, инженеры по машинному обучению, стремящиеся привнести больше порядка в свою работу, студенты, изучающие машинное обучение, а также архитекторы программных систем, которым приходится иметь дело с моделями, разработанными аналитиками данных и инженерами по машинному обучению.

Как использовать эту книгу

Эта книга представляет собой подробный обзор передовых практик и паттернов проектирования в области прикладного машинного обучения. Я рекомендую читать ее с начала до конца. Но можете читать главы в любом порядке, поскольку в них рассматриваются различные аспекты жизненного цикла проекта с использованием машинного обучения и прямых зависимостей между главами нет.

Андрей Бурков

Глава 1

Введение

Хотя предполагается, что читатель этой книги знаком с основами машинного обучения, все же важно начать с определений, чтобы у нас было общее понимание используемых в книге терминов.

Ниже я повторяю некоторые определения из главы 2 книги «Машинное обучение без лишних слов», а также даю несколько новых определений. Если вы читали мою первую книгу, то некоторые части этой главы могут показаться вам знакомыми.

После этой главы мы будем одинаково понимать такие понятия, как обучение с учителем и без учителя. У нас будет общий взгляд на прямо и косвенно используемые данные, первичные и аккуратные данные, обучающие и зарезервированные данные.

Мы будем знать о том, когда следует использовать машинное обучение, а когда этого делать не стоит, а также о различных формах машинного обучения, например: на основе модели и на основе экземпляров, глубокое и поверхностное, классификация и регрессия и т. д.

Наконец, мы определим предмет инженерии машинного обучения и представим жизненный цикл проекта машинного обучения.

1.1. Обозначения и определения

Начнем с базовых математических обозначений и определим термины и понятия, к которым часто будем обращаться в этой книге.

1.1.1. Структуры данных

Скаляром[1] называется простое числовое значение, например 15 или –3.25. Переменные или константы, принимающие скалярные значения, обозначаются курсивом, например *x* или *a*.

Вектором называется упорядоченный список скалярных значений, именуемых атрибутами. Мы обозначаем вектор полужирным шрифтом, например **x** или **w**. Векторы изображаются в виде направленных стрелок, а также

[1] Если термин выделен полужирным шрифтом, значит, он присутствует в алфавитном указателе в конце книги.

точек в многомерном пространстве. Иллюстрации трех двумерных векторов **a** = [2, 3], **b** = [−2, 5] и **c** = [1, 0] приведены на рис. 1.1. Атрибут вектора обозначается курсивной буквой с верхним индексом, например: $w^{(j)}$ или $x^{(j)}$. Индекс j обозначает конкретное **измерение** вектора, т. е. позицию атрибута в списке. Например, в векторе **a**, показанном красным цветом на рис. 1.1, $a^{(1)} = 2$ и $a^{(2)} = 3$.

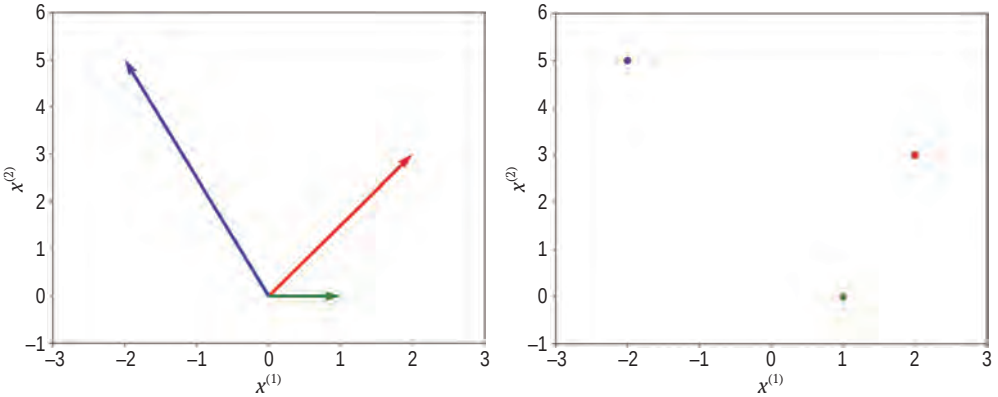

Рис. 1.1 ❖ Три вектора, представленных направленными стрелками и точками

Обозначение $x^{(j)}$ не следует путать с оператором возведения в степень, например 2 в x^2 (возведение в квадрат) или 3 в x^3 (возведение в куб). Если мы хотим применить оператор возведения в степень, скажем, в квадрат, к индексированному атрибуту вектора, то пишем: $(x^{(j)})^2$.

Переменная может иметь два и более индексов, например: $x_i^{(j)}$ или $x_{i,j}^{(k)}$. Так, в нейронных сетях $x_{l,u}^{(j)}$ обозначает j-й входной признак u-го блока в l-м слое.

Матрица – это прямоугольный массив чисел, организованный по строкам и столбцам. Ниже приведен пример матрицы с двумя строками и тремя столбцами:

$$\mathbf{A} = \begin{bmatrix} 2 & -2 & 1 \\ 3 & 5 & 0 \end{bmatrix}.$$

Матрицы обозначаются полужирными заглавными буквами, например **A** или **W**. Из примера матрицы **A** выше видно, что матрицы можно трактовать как регулярные структуры, состоящие из векторов. И действительно, столбцами приведенной выше матрицы **A** являются векторы **a**, **b** и **c**, показанные на рис. 1.1.

Множеством называется неупорядоченная коллекция неповторяющихся элементов. Множество обозначается каллиграфической заглавной буквой, например $\mathcal{S}$. Множество чисел может быть конечным (содержать фиксированное количество значений). В этом случае его элементы перечисляются в фигурных скобках, например {1, 3, 18, 23, 235} или $\{x_1, x_2, x_3, x_4, \ldots, x_n\}$. Множество также может быть бесконечным и включать все значения в некотором

интервале. Множество, содержащее все значения между a и b включительно, обозначается $[a, b]$. Если же множество не включает граничные значения a и b, то оно обозначается (a, b). Например, множество $[0, 1]$ включает среди прочего значения 0, 0.0001, 0.25, 0.784, 0.9995 и 1.0. Буквой $\mathbb{R}$ обозначается множество всех чисел от минус бесконечности до плюс бесконечности.

Если элемент x принадлежит множеству $\mathcal{S}$, то мы пишем $x \in \mathcal{S}$. Новое множество $\mathcal{S}_3$ можно получить как **пересечение** двух множеств $\mathcal{S}_1$ и $\mathcal{S}_2$. В этом случае мы пишем $\mathcal{S}_3 \leftarrow \mathcal{S}_1 \cap \mathcal{S}_2$. Например, $\{1, 3, 5, 8\} \cap \{1, 8, 4\}$ дает новое множество $\{1, 8\}$.

Новое множество $\mathcal{S}_3$ можно получить как **объединение** двух множеств $\mathcal{S}_1$ и $\mathcal{S}_2$. В этом случае мы пишем $\mathcal{S}_3 \leftarrow \mathcal{S}_1 \cup \mathcal{S}_2$. Например, $\{1, 3, 5, 8\} \cup \{1, 8, 4\}$ дает новое множество $\{1, 3, 5, 8, 4\}$.

$|\mathcal{S}|$ обозначает размер множества $\mathcal{S}$, то есть число элементов в нем.

1.1.2. Заглавная сигма

Сумма множества чисел $\mathcal{X} = \{x_1, x_2, \ldots, x_{n-1}, x_n\}$ или атрибутов вектора $\mathbf{x} = [x^{(1)}, x^{(2)}, \ldots, x^{(m-1)}, x^{(m)}]$ обозначается следующим образом:

$$\sum_{i=1}^{n} x_i \stackrel{\text{def}}{=} x_1 + x_2 + \ldots + x_{n-1} + x_n,$$

или

$$\sum_{i=1}^{n} x^{(j)} \stackrel{\text{def}}{=} x^{(1)} + x^{(2)} + \ldots + x^{(m-1)} + x^{(m)}.$$

Здесь $\stackrel{\text{def}}{=}$ означает «по определению равно».

Евклидова норма вектора $\mathbf{x}$, обозначаемая $||\mathbf{x}||$, характеризует «размер», или «длину», вектора. Она определяется как $\sqrt{\sum_{j=1}^{D}(x^{(j)})^2}$.

В качестве расстояния между векторами $\mathbf{a}$ и $\mathbf{b}$ берется **евклидово расстояние**:

$$||a - b|| \stackrel{\text{def}}{=} \sqrt{\sum_{i=1}^{N}(a^{(i)} - b^{(i)})^2}.$$

1.2. Что такое машинное обучение

Машинное обучение – это раздел информатики, посвященный построению алгоритмов, которые работают с набором примеров, описывающих какое-то явление. Примеры могут поступать из природы, создаваться людьми или генерироваться другим алгоритмом.

Машинное обучение также можно определить как процесс решения практической задачи путем

1) сбора набора данных и
2) алгоритмического обучения **статистической модели** на этом наборе.

Предполагается, что эта статистическая модель каким-то образом используется для решения практической задачи. Для краткости я буду использовать термины «обучение» и «машинное обучение» как синонимы. По той же причине я буду часто говорить «модель», имея в виду статистическую модель.

Обучение бывает с учителем, без учителя, с частичным привлечением учителя и с подкреплением.

1.2.1. Обучение с учителем

В случае **обучения с учителем** аналитик работает с набором **помеченных примеров** $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$. Каждый элемент x_i называется **вектором признаков**. В информатике вектор – это одномерный массив. Одномерный массив, в свою очередь, является упорядоченной и индексированной последовательностью значений. Длина этой последовательности значений, D, называется **размерностью** вектора.

Вектор признаков – это вектор, в котором каждое измерение $j = 1 \ldots D$ содержит значение, описывающее пример. Каждое такое значение называется **признаком** и обозначается $x^{(j)}$. Например, если каждый пример x представляет человека, то первый признак, $x^{(1)}$, может содержать рост в сантиметрах, второй признак, $x^{(2)}$, – вес в килограммах, $x^{(3)}$ – пол и т. д. Для всех примеров в наборе данных значение в позиции j вектора признаков всегда имеет один и тот же тип. Это означает, что если $x_i^{(2)}$ содержит вес в килограммах для некоторого i, то и для любого k от 1 до N $x_k^{(2)}$ будет содержать вес в килограммах. **Метка** y_i может быть либо элементом конечного множества классов $\{1, 2, \ldots, C\}$, либо вещественным числом, либо более сложной структурой, такой как вектор, матрица, дерево или граф. Если явно не оговорено противное, то в этой книге y_i является либо элементом конечного множества классов, либо вещественным числом[1]. Можно считать, что класс – это категория, к которой относится пример.

Скажем, если примерами являются сообщения электронной почты, а наша задача заключается в обнаружении спама, то есть два класса: спам и не спам. В случае обучения с учителем задача предсказания класса называется **классификацией**, а задача предсказания вещественного числа называется **регрессией**. Значение, которое должно быть предсказано моделью, обученной с учителем, называется **целевым показателем**, или целью. Примером регрессии является задача предсказания заработной платы сотрудника с учетом его опыта работы и знаний. Примером классификации является ситуация, когда врач вводит характеристики пациента в приложение, а то возвращает диагноз.

Различие между классификацией и регрессией показано на рис. 1.2. В случае классификации алгоритм обучения ищет линию (или, в более общем слу-

[1] Вещественное число – это величина, представляющая расстояние вдоль прямой от некоторой начальной точки на ней. Примеры: 0, –256.34, 1000, 1000.2.

чае, гиперповерхность), которая разделяет примеры разных классов. С другой стороны, в случае регрессии алгоритм обучения стремится отыскать линию или гиперповерхность, которая хорошо соответствует обучающим примерам.

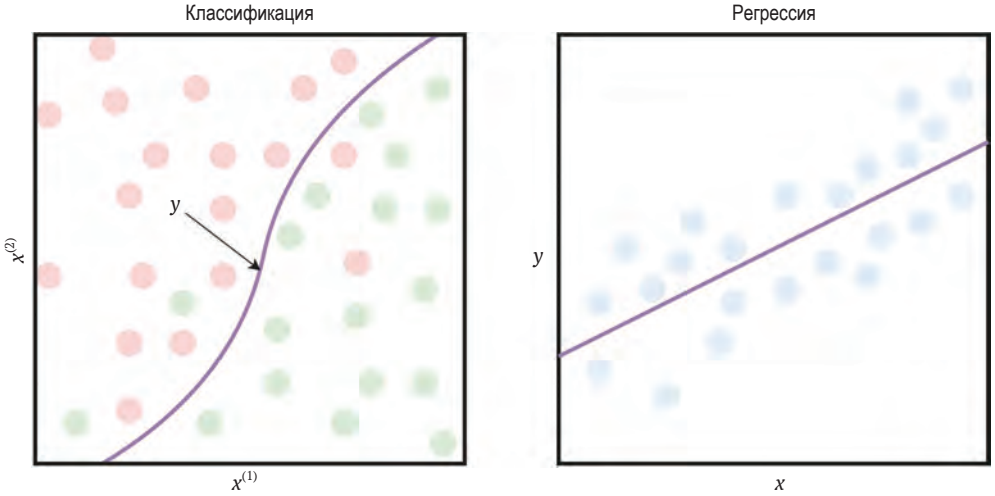

Рис. 1.2 ❖ Разница между классификацией и регрессией

Цель **алгоритма обучения с учителем** – использовать набор данных для порождения модели, которая на входе принимает вектор признаков **x**, а на выходе выдает информацию, позволяющую вывести метку для этого вектора признаков. Например, модель, созданная с использованием набора данных о пациентах, может на входе принимать вектор признаков, описывающий пациента, и на выходе выдавать вероятность наличия у пациента рака.

Даже если модель традиционно представляет собой математическую функцию, размышляя о действиях модели с входными данными, удобно думать, что модель «смотрит» на значения некоторых признаков на входе и, основываясь на опыте работы с аналогичными примерами, выводит значение. Это выходное значение представляет собой число или класс, «наиболее похожий» на метки, которые мы видели в прошлом в примерах с похожими значениями признаков. Такое описание выглядит упрощенно, но именно так работает модель решающего дерева и алгоритм *k* ближайших соседей.

1.2.2. Обучение без учителя

В случае **обучения без учителя** набор данных содержит **непомеченные примеры** $\{x_1, x_2, \ldots, x_N\}$. Как и раньше, **x** – вектор признаков, а цель **алгоритма обучения без учителя** заключается в порождении модели, которая на входе принимает вектор признаков **x** и преобразовывает его либо в другой вектор, либо в значение, используемое для решения практической задачи. Например, в случае **кластеризации** для каждого вектора признаков из на-

бора данных модель возвращает ИД кластера. Кластеризация применяется для отыскания групп похожих объектов в большом наборе объектов, например изображений или текстовых документов. Используя кластеризацию, аналитик может, например, выбрать достаточно репрезентативное, но малое подмножество непомеченных примеров из большого набора, чтобы потом пометить их вручную: из каждого кластера выбирается всего несколько примеров, вместо того чтобы отбирать непосредственно из большого набора с риском выбрать очень похожие примеры.

В задаче **понижения размерности** выходом модели является вектор признаков с меньшим числом измерений, чем на входе. Например, у исследователя имеется вектор признаков, слишком сложный для визуализации (поскольку число измерений больше трех). Модель понижения размерности может преобразовать этот вектор в другой (сохраняя часть информации) – двумерный или трехмерный. Этот новый вектор признаков можно изобразить на графике.

В задаче **обнаружения выбросов** выходом является действительное число, показывающее, насколько входной вектор признаков отличается от «типичного» примера в наборе данных. Обнаружение выбросов применяется для решения задачи о проникновении в сеть (путем обнаружения аномальных сетевых пакетов, которые отличаются от типичного пакета в «нормальном» трафике) или обнаружения новизны (например, документа, отличающегося от других документов в наборе).

1.2.3. Обучение с частичным привлечением учителя

В случае **обучения с частичным привлечением учителя** набор данных содержит как помеченные, так и непомеченные примеры. Обычно число непомеченных примеров намного превышает число помеченных. Цель **алгоритма с частичным привлечением учителя** такая же, как у алгоритма обучения с учителем. Мы надеемся, что, располагая большим числом непомеченных примеров, алгоритм обучения сможет отыскать (можно было бы сказать «породить» или «вычислить») лучшую модель.

1.2.4. Обучение с подкреплением

Обучение с подкреплением – это раздел машинного обучения, в котором рассматривается ситуация, когда машина (именуемая агентом) «обитает» в окружающей среде и способна воспринимать состояние этой среды как вектор признаков. Машина может выполнять действия в незаключительных состояниях. Разные действия приносят разные вознаграждения, а также могут переводить машину в другое состояние окружающей среды. Типичная цель алгоритма обучения с подкреплением – обучиться оптимальной **политике**.

Оптимальная политика – это функция (аналогичная модели в обучении с учителем), которая на входе принимает вектор признаков состояния, а на

выходе выдает оптимальное действие, которое следует выполнить в этом состоянии. Действие является оптимальным, если оно максимизирует ожидаемое среднее долгосрочное вознаграждение.

Обучение с подкреплением решает конкретную задачу, в которой принятие решений является последовательным, а цель – долгосрочной. Такие задачи возникают в играх, робототехнике, управлении ресурсами или логистике.

В этой книге мы для простоты ограничиваемся обучением с учителем (в большинстве случаев). Однако весь представленный в книге материал применим и к другим видам машинного обучения.

1.3. Терминология машинного обучения

Теперь познакомимся с общепринятой терминологией, относящейся к самим данным (например, данные, используемые прямо и косвенно, первичные и аккуратные данные, обучающие и зарезервированные данные) и к машинному обучению (например, ориентир, гиперпараметр, конвейер и др.).

1.3.1. Данные, используемые прямо и косвенно

Данные в проекте машинного обучения могут использоваться для формирования примеров **x прямо** или **косвенно**.

Допустим, что мы строим систему распознавания именованных сущностей. На вход модели поступает последовательность слов, на выходе выдается последовательность меток[1] той же длины, что вход. Для того чтобы алгоритм машинного обучения мог прочитать данные, мы должны преобразовать каждое слово естественного языка в машиночитаемый массив атрибутов, называемый вектором признаков[2]. Некоторые признаки могут содержать информацию, которая отличает это конкретное слово от других слов в словаре. Другие признаки могут содержать дополнительные атрибуты слова в этой конкретной последовательности, такие как его форма (в нижнем регистре, в верхнем регистре, с заглавной буквы и т. д.). Либо это могут быть бинарные атрибуты, указывающие на то, является ли слово первым словом имени человека или последним словом названия какого-либо места или организации. Для создания таких бинарных признаков мы можем прибегнуть к словарям, справочным таблицам, географическим справочникам или другим моделям машинного обучения, делающим предсказания относительно слов.

Вы, вероятно, уже поняли, что набор последовательностей слов – это данные, используемые для формирования обучающих примеров непосредствен-

---

[1] Метки могут быть, например, значениями из множества {«Местоположение», «Организация», «Лицо», «Прочее»}.

[2] Термины «атрибут» и «признак» часто используются как синонимы. В этой книге я применяю термин «атрибут» для описания конкретного свойства примера, тогда как термин «признак» относится к значению $x^{(j)}$ в позиции j вектора признаков **x**, используемого алгоритмом машинного обучения.

но, тогда как данные, содержащиеся в словарях, справочных таблицах и географических справочниках, используются косвенно: они могут служить для пополнения векторов признаков дополнительными признаками, но не для создания новых векторов.

1.3.2. Первичные и аккуратные данные

Как мы только что обсудили, непосредственно используемые данные – это набор сущностей, составляющих основу набора данных. Каждая сущность в этом наборе может быть преобразована в обучающий пример. **Первичные данные** – это набор сущностей в их естественной форме; их не всегда можно задействовать для машинного обучения непосредственно. Например, документ Word или JPEG-файл являются первичными данными; алгоритм машинного обучения не может использовать их напрямую[1].

Необходимым (но не достаточным) условием использования данных в машинном обучении является их аккуратность. **Аккуратные данные** можно рассматривать как электронную таблицу, в которой строки представляют примеры, а столбцы – **атрибуты** примеров, как показано на рис. 1.3. Иногда сами первичные данные могут быть аккуратными, например если они уже представлены в виде электронной таблицы. Однако на практике, чтобы получить аккуратные данные из первичных, аналитики нередко прибегают к процедуре **конструирования признаков**, которая применяется к прямым и, факультативно, к косвенным данным с целью преобразования каждого первичного примера в вектор признаков **x**. Глава 4 полностью посвящена конструированию признаков.

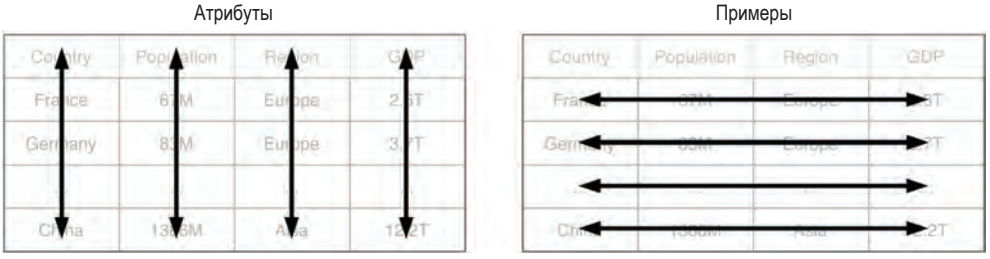

Рис. 1.3 ❖ Аккуратные данные:
строки содержат примеры, а столбцы – атрибуты

[1] Термин «неструктурированные данные» часто используется для обозначения элемента данных, содержащего информацию, тип которой формально не определен. Примерами неструктурированных данных являются фотографии, изображения, видео, текстовые сообщения, сообщения в социальных сетях, PDF-файлы, текстовые документы и электронные письма. Термин «полуструктурированные данные» относится к элементам данных, структура которых помогает выводить типы некоторой закодированной в них информации. К полуструктурированным данным можно отнести файлы журналов, текстовые файлы с запятыми или табуляторами в качестве разделителей, а также документы в форматах JSON и XML.

Здесь важно отметить, что в некоторых задачах пример, используемый алгоритмом обучения, может быть последовательностью векторов, матрицей или последовательностью матриц. Понятие аккуратности данных для таких алгоритмов определяется в схожем ключе: нужно лишь заменить «строку фиксированной ширины в электронной таблице» матрицей фиксированной ширины и высоты либо обобщением матрицы на большее число измерений – **тензором**.

Термин «аккуратные данные» был введен Хэдли Уикхемом в одноименной статье[1].

Как было сказано в начале этого раздела, данные могут быть аккуратными, но все равно непригодными для использования конкретным алгоритмом машинного обучения. На самом деле большинство алгоритмов машинного обучения принимают обучающие данные только в виде набора векторов числовых признаков. Возьмем данные, показанные на рис. 1.3. Атрибут «Регион» является категориальным, а не числовым. Алгоритм обучения на основе решающего дерева может работать со значениями категориальных атрибутов, но большинство алгоритмов обучения на это не способны. В разделе 4.2 главы 4 мы увидим, как преобразовать категориальный атрибут в числовой.

Обратите внимание, что в академической литературе по машинному обучению слово «пример» обычно относится к примеру аккуратных данных, возможно, с соответственной меткой. Однако на этапе сбора данных и их разметки, который мы рассмотрим в следующей главе, примеры могут быть еще представлены в первичной форме: изображения, тексты или строки с категориальными атрибутами в электронной таблице. В этой книге, когда важно подчеркнуть разницу, я буду говорить «**первичный пример**», чтобы указать, что часть данных еще не была преобразована в вектор признаков. В противном случае мы будем предполагать, что примеры имеют форму векторов признаков.

1.3.3. Обучающие и зарезервированные наборы

На практике аналитики работают с тремя разными наборами примеров:

1) обучающий набор;
2) контрольный набор[2];
3) тестовый набор.

Получив данные в виде набора примеров, первое, что нужно сделать в проекте, – перемешать примеры и разделить данные на три отдельных набора: **обучающий**, **контрольный** и **тестовый**. Обучающий набор обычно является самым крупным; он используется в алгоритме обучения для порождения модели. Контрольный и тестовый наборы имеют примерно одинаковый, гораздо меньший, размер. Алгоритму обучения не разрешается использовать

---

[1] *Wickham Hadley.* Tidy data // Journal of Statistical Software 59.10 (2014): 1–23.

[2] Иногда, если помеченных примеров недостаточно, аналитик может обойтись без контрольного набора, как мы увидим в разделе главы 5, посвященном **перекрестной проверке**.

примеры из контрольного или тестового набора для обучения модели. Поэтому оба этих набора иногда называют **зарезервированными**.

Причина наличия трех наборов, а не одного, проста: в процессе обучения мы не хотим, чтобы модель хорошо предсказывала метки только тех примеров, которые алгоритм уже встречал. Тривиальный алгоритм, который просто запоминает все обучающие примеры, а затем использует их для «предсказания» меток, не будет делать никаких ошибок, когда его попросят предсказать метки обучающих примеров. Однако на практике такой алгоритм будет бесполезен. Действительно, нам нужна модель, хорошо предсказывающая примеры, которые не предъявлялись алгоритму во время обучения. Иными словами, мы хотим иметь хорошее качество на зарезервированном наборе[1].

Два зарезервированных набора, а не один, нужно, потому что контрольный набор используется, чтобы 1) выбирать алгоритм обучения и 2) отыскивать оптимальные конфигурационные параметры для этого алгоритма (именуемые **гиперпараметрами**). Тестовый набор применяется для оценивания модели перед ее передачей заказчику или развертыванием в производственной среде. Вот почему так важно, чтобы никакая информация из контрольного или тестового набора не была предъявлена алгоритму обучения. В противном случае результаты контроля и тестирования, скорее всего, будут слишком оптимистичными. Такая неприятность иногда случается из-за **просачивания данных**, явления, которое мы рассмотрим в разделе 3.2.8 главы 3 и последующих главах.

1.3.4. Ориентир

В машинном обучении **ориентиром** называется простой алгоритм решения задачи, обычно основанный на эвристике, простой сводной статистике, рандомизации или очень простом алгоритме машинного обучения. Например, если задача связана с классификацией, то можно выбрать классификатор, служащий ориентиром, и измерить его качество. С этим ориентиром затем будет сравниваться любая будущая модель (обычно построенная с применением более изощренного подхода).

1.3.5. Конвейер машинного обучения

Конвейер машинного обучения – это последовательность операций с набором данных, ведущая из начального состояния к модели.

Конвейер может включать, среди прочего, такие этапы, как разбиение данных, подстановка пропущенных данных, выделение признаков, приращение

[1] Если быть точным, мы хотим, чтобы модель хорошо работала на большинстве случайных выборок из статистического распределения, которому принадлежат наши данные. Мы исходим из того, что если модель хорошо работает на зарезервированном наборе, случайно выбранном из неизвестного распределения данных, то высоки шансы, что она будет хорошо работать и на других случайно выбранных примерах.

данных, уменьшение несбалансированности классов, понижение размерности и обучение модели.

На практике, развертывая модель в производственной среде, мы обычно развертываем весь конвейер. Кроме того, при настройке гиперпараметров обычно оптимизируется весь конвейер целиком.

1.3.6. Параметры и гиперпараметры

Гиперпараметры – это входные данные алгоритма машинного обучения или конвейера, которые влияют на качество модели. Они не относятся к обучающим данным и не могут быть обучены на их основе. Например, максимальная глубина дерева в алгоритме решающего дерева, штраф за неправильную классификацию в методе опорных векторов, величина k в алгоритме k ближайших соседей, целевая размерность в алгоритме понижения размерности и выбор способа подстановки пропущенных данных – все это примеры гиперпараметров.

Параметры, с другой стороны, являются переменными, которые определяют обучаемую модель. Параметры напрямую модифицируются алгоритмом обучения на основе обучающих данных. Цель обучения состоит в том, чтобы отыскать такие значения параметров, которые делают модель в определенном смысле оптимальной. Примерами параметров являются w и b в уравнении линейной регрессии $y = wx + b$, где x является входом модели, а y – ее выходом (предсказанием).

1.3.7. Классификация и регрессия

Классификация – это задача автоматического назначения метки **непомеченному примеру**. Распознавание спама – широко известный пример классификации.

В машинном обучении задача классификации решается **алгоритмом обучения классификатора**, который на входе принимает набор **помеченных примеров**, а на выходе порождает **модель**, которая может получать непомеченный пример и возвращать либо саму метку, либо число, которое аналитик может использовать для нахождения метки. Примером такого числа является вероятность того, что элемент входных данных имеет определенную метку.

В задаче классификации метка является элементом конечного множества **классов**. Если размер этого множества равен двум (больной/здоровый, спам/не спам), то мы говорим о **бинарной классификации** (в некоторых источниках она называется **биномиальной**). **Многоклассовая** (или **мультиномиальная**) **классификация** – это задача классификации с тремя или более классами[1].

В то время как некоторые алгоритмы обучения естественным образом допускают наличие более двух классов, другие по природе своей являются

[1] Тем не менее каждому примеру по-прежнему соответствует одна метка.

алгоритмами бинарной классификации. Существуют стратегии преобразования алгоритма бинарной классификации в многоклассовый. Об одном из них, «один против остальных», я расскажу в разделе 6.5 главы 6.

Регрессия – это задача предсказания вещественной величины по непомеченному примеру. Широко известный пример регрессии – оценка стоимости дома на основе таких его характеристик, как площадь, число спален, местоположение и т. д.

Задача регрессии решается с помощью **алгоритма обучения регрессии**; на входе он принимает набор помеченных примеров и порождает модель, которая принимает непомеченный пример и выводит целевой показатель.

1.3.8. Обучение на основе модели и обучение на основе экземпляров

Большинство алгоритмов обучения с учителем **основаны на моделях**. Типичной **моделью** является **метод опорных векторов** (**Support Vector Machine** – **SVM**). В алгоритмах обучения на основе моделей обучающие данные используются для создания модели с обученными **параметрами**. В SVM таких параметров два: **w** (вектор) и b (вещественное число). Обученную модель можно сохранить на диске, а обучающие данные удалить.

В **алгоритмах обучения на основе экземпляров** весь набор данных целиком используется в качестве модели. Одним из часто используемых на практике алгоритмов такого рода является метод **k ближайших соседей** (kNN). Для предсказания метки входного примера алгоритм kNN рассматривает его окрестность в пространстве векторов признаков и выводит метку, которая встречается в этой окрестности чаще других.

1.3.9. Поверхностное и глубокое обучение

Алгоритм **поверхностного обучения** обучает параметры непосредственно на признаках обучающих примеров. Большинство алгоритмов машинного обучения поверхностные. Широко известными исключениями являются **нейросетевые** алгоритмы обучения, а именно такие, которые строят нейронные сети с несколькими **слоями** между входом и выходом. Такие сети называются **глубокими нейронными сетями**. В глубоком нейросетевом обучении (или просто **глубоком обучении**), в отличие от поверхностного, большинство параметров модели обучаются не на самих признаках обучающих примеров, а на выходах предыдущих слоев.

1.3.10. Обучение и оценивание

Когда алгоритм машинного обучения применяется к набору данных с целью получения модели, говорят об **обучении модели** или просто обучении.

Когда обученная модель применятся к входному примеру (иногда к последовательности примеров) с целью получить предсказание (или несколько

предсказаний) либо каким-то образом преобразовать входные данные, говорят об **оценивании**.

1.4. Когда следует использовать машинное обучение

Машинное обучение – мощный инструмент для решения практических задач. Однако, как и любой инструмент, его следует использовать в правильном контексте. Пытаться решать все задачи с помощью машинного обучения было бы ошибкой.

Возможность использования машинного обучения следует рассматривать в одной из следующих ниже ситуаций.

1.4.1. Когда задача слишком сложна для кодирования

В ситуации, когда задача настолько сложна или велика, что попытка написать все правила для ее решения попросту безнадежна, а частичное решение допустимо и представляет интерес, можно попытаться решить задачу с помощью машинного обучения.

Один из примеров – обнаружение спама: невозможно написать исходный код, реализующий алгоритм, который будет эффективно обнаруживать спамные сообщения и помещать в папку входящих только хорошие сообщения. Придется учитывать слишком много факторов. Например, если запрограммировать спам-фильтр на отклонение всех сообщений от лиц, отсутствующих в контактах, то имеется риск потерять сообщения от человека, получившего вашу визитную карточку на конференции. Сделав исключение для сообщений, содержащих определенные ключевые слова, связанные с работой, вы, скорее всего, пропустите сообщение от учителя своего ребенка и т. д.

Если вы все же решите запрограммировать решение этой сложной задачи в лоб, то со временем в исходном коде окажется так много правил и исключений из них, что сопровождать этот код станет невозможно. В этой ситуации обучение классификатора на примерах спама и не спама кажется логичным и единственно жизнеспособным вариантом.

Еще одна трудность при написании исходного кода для решения задачи – тот факт, что людям трудно решать задачи прогнозирования, основываясь на входных данных, содержащих слишком много параметров, особенно если параметры **коррелируют** неизвестным образом. Возьмем, к примеру, предсказание возможности погашения кредита заемщиком. Каждый заемщик представлен сотнями показателей: возраст, зарплата, остаток на счете, частота прошлых платежей, семейное положение, число детей, марка и год выпуска автомобиля, остаток по ипотеке и т. д. Одни из них могут быть важны для принятия решения, другие сами по себе менее важны, но становятся более важными, если рассматривать их в сочетании.

Писать исходный код, который будет принимать такие решения, трудно, потому что даже эксперту не ясно, как оптимально объединять в предсказание все описывающие человека атрибуты.

1.4.2. Когда задача постоянно меняется

Некоторые задачи со временем постоянно изменяются, поэтому исходный код необходимо регулярно обновлять. Это «бесит» программистов, работающих над задачей, растет вероятность ошибок, становится трудно совмещать «старую» логику с «новой», и значительно возрастают накладные расходы на тестирование и развертывание обновленных решений.

Например, рассмотрим задачу выделения представляющих интерес элементов данных из коллекции веб-страниц. Предположим, что для каждой веб-страницы создается фиксированный набор правил извлечения данных в следующей форме: «выбрать третий элемент <p> из <body>, а затем выбрать данные из второго внутри этого <p>». Если владелец сайта изменит дизайн страницы, то интересующие вас данные могут оказаться во втором или четвертом элементе <p>, и правило извлечения станет недействительным. Если коллекция обрабатываемых страниц велика (тысячи URL-адресов), то недействительные правила будут возникать каждый день, так что процесс их настройки никогда не закончится. Излишне говорить, что очень немногие программисты хотели бы выполнять такую работу ежедневно.

1.4.3. Когда речь идет о задаче восприятия

Сегодня трудно представить себе, что кто-то попытается решать **задачи восприятия**, такие как распознавание речи, изображений и видео, без использования машинного обучения. Возьмем, к примеру, изображение. Оно представлено миллионами пикселей. Каждый пиксель обозначается тремя числами: интенсивностью красного, зеленого и синего каналов. В прошлом инженеры пытались решать задачу распознавания изображений (определения того, что изображено на картинке) путем применения самодельных «фильтров» к квадратным участкам пикселей – патчам. Если, например, фильтр, предназначенный для «обнаружения» травы, порождает большое значение при применении ко многим патчам, а другой фильтр, предназначенный для обнаружения коричневого подшерстка, тоже возвращает большие значения для многих патчей, то высоки шансы, что изображена корова на лугу (я немного упрощаю).

Сегодня задачи восприятия эффективно решаются с помощью моделей машинного обучения, таких как нейронные сети. Мы рассмотрим задачу обучения нейронных сетей в главе 6.

1.4.4. Когда это неизученное явление

Если требуется предсказывать явление, которое недостаточно изучено с научной точки зрения, но наблюдаемо, то машинное обучение, возможно, будет

подходящим (а в некоторых случаях единственно доступным) вариантом. Например, машинное обучение можно использовать для разработки персонализированных способов лечения психических расстройств на основе генетических и сенсорных данных. Врачи не всегда могут интерпретировать такие данные и дать оптимальную рекомендацию, тогда как машина способна обнаружить в данных закономерности благодаря анализу тысяч пациентов и предсказать молекулу, которая имеет наибольшие шансы помочь данному пациенту.

Еще один пример наблюдаемых, но неизученных явлений – журналы сложной вычислительной системы или сети. Такие журналы создаются несколькими независимыми или взаимозависимыми процессами. Человеку трудно делать предсказания о будущем состоянии системы, располагая только журналами, но не имея модели каждого процесса и их взаимозависимостей. Если количество исторических журнальных записей достаточно велико (как часто и бывает), то машина сможет выявлять скрытые закономерности и делать предсказания, ничего не зная о каждом процессе.

Наконец, трудно делать прогнозы о людях, основываясь на их наблюдаемом поведении. В этой задаче у нас, очевидно, отсутствует модель мозга человека, но имеются легкодоступные примеры выражения его мыслей (в виде онлайновых сообщений, комментариев и других действий). Основываясь только на этих выражениях, развернутая в социальной сети модель машинного обучения сможет рекомендовать контент или других людей для общения.

1.4.5. Когда задача имеет простую целевую функцию

Машинное обучение особенно хорошо подходит в случаях, когда задачу можно сформулировать как оптимизацию простой целевой функции: например, когда требуется найти бинарное решение типа да/нет или одно-единственное число. С другой стороны, машинное обучение не подойдет для построения модели, работающей как типичная видеоигра, например Mario, или как текстовый процессор типа Word. Дело в том, что число принимаемых решений слишком велико: что, где и когда выводить на экран, как система должна реагировать на введенные пользователем данные, что записывать или читать с жесткого диска и т. д.; получить примеры, иллюстрирующие все (или даже большинство) из этих решений, практически невозможно.

1.4.6. Когда это экономически выгодно

Три основных источника затрат в машинном обучении таковы:

- сбор, подготовка и очистка данных;
- обучение модели;
- создание и эксплуатация инфраструктуры для выполнения и мониторинга модели, а также трудовые ресурсы для ее сопровождения.

Стоимость обучения модели включает в себя человеческий труд и в некоторых случаях дорогостоящее оборудование, необходимое для обучения глубоких моделей. Сопровождение модели включает постоянный мониторинг ее работы и сбор дополнительных данных для поддержания модели в актуальном состоянии.

1.5. Когда не следует использовать машинное обучение

Существует много задач, которые невозможно решить с помощью машинного обучения; трудно охарактеризовать их все. Здесь мы дадим лишь несколько рекомендаций.

Пожалуй, не стоит использовать машинное обучение, когда:

- любое действие системы или принятое ею решение должно быть объяснимым;
- любое изменение в поведении системы по сравнению с ее прошлым поведением в аналогичной ситуации должно быть объяснимым;
- стоимость допущенной системой ошибки слишком высока;
- требуется вывести продукт на рынок как можно быстрее;
- получить правильные данные слишком трудно или невозможно;
- задачу можно решить с помощью традиционных методов разработки программ с меньшими затратами;
- простая эвристика работает достаточно хорошо;
- у явления слишком много исходов, а получить примеры в количестве, достаточном для их представления, невозможно (например, в видеоиграх или текстовых процессорах);
- построенную систему со временем не придется часто улучшать;
- можно вручную подготовить исчерпывающую справочную таблицу, указав ожидаемый результат для любого входного значения (то есть число возможных входных значений не слишком велико или получать выходные данные можно быстро и дешево).

1.6. Что такое инженерия машинного обучения

Инженерия машинного обучения (machine learning engineering – MLE) – это использование научных принципов, инструментов и методов машинного обучения и традиционной программной инженерии для проектирования и создания сложных компьютерных систем. MLE охватывает все этапы от сбора данных до обучения модели и использования ее продуктом или клиентами.

Традиционно аналитик данных[1] занимается осмыслением деловой задачи, построением модели для ее решения и оцениванием модели в ограниченной среде разработки. В обязанности инженера по машинному обучению входят получение данных из различных систем и мест, их предобработка, конструирование признаков, обучение эффективной модели, которая будет работать в производственной среде, сосуществовать с другими производственными процессами, будет стабильной, удобной для сопровождения и легкодоступной для разных типов пользователей, по-разному применяющих ее.

Иными словами, MLE включает в себя любую деятельность, которая позволяет реализовать алгоритмы машинного обучения как часть эффективной производственной системы.

На практике инженеры по машинному обучению могут быть задействованы в таких видах деятельности, как переписывание созданного аналитиками данных кода с таких относительно медленных языков, как R и Python[2], на более эффективных языках типа Java или C++, масштабирование этого исходного кода и повышение его надежности, его упаковка в легко развертываемый пакет, хранящийся в системе управления версиями, оптимизация алгоритма машинного обучения, так чтобы порождаемая им модель была совместима с производственной средой организации и работала в ней правильно.

Во многих организациях аналитики данных выполняют такие задачи MLE, как сбор данных, их преобразование и конструирование признаков. С другой стороны, инженеры по машинному обучению нередко выполняют некоторые задачи анализа данных, включая выбор алгоритма обучения, настройку гиперпараметров и оценивание модели.

Работа над проектом машинного обучения отличается от работы над типичным проектом в области программной инженерии. В отличие от традиционной программной инженерии, где поведение программы обычно детерминировано, приложения машинного обучения включают модели, поведение которых со временем может по естественным причинам ухудшаться или становиться аномальным. Такое аномальное поведение модели может иметь различные корни, в т. ч. фундаментальное изменение входных данных или внедрение нового экстрактора признаков, возвращающего другое распределение значений или значения иного типа. Нередко можно услышать, что системы машинного обучения «отказывают молча». Инженер по машинному обучению должен быть способен предотвращать такие отказы либо, если полное предотвращение невозможно, знать, как их обнаруживать и устранять.

---

[1] Должность исследователя данных (data scientist) приобрела популярность примерно с 2013 года. К сожалению, компании и эксперты не пришли к единому мнению относительно определения этого термина. Вместо этого я использую термин «аналитик данных», имея в виду человека, способного применять численный или статистический анализ к имеющимся данным.

[2] Многие научные модули в Python на самом деле написаны на быстром C/C++; однако собственный код аналитика данных на Python все равно может быть медленным.

1.7. Жизненный цикл проекта машинного обучения

Проект машинного обучения начинается с осмысления целевого критерия с точки зрения бизнеса. Обычно бизнес-аналитик работает с заказчиком[1] и аналитиком данных, чтобы трансформировать деловую задачу в технический проект. Технический проект может содержать или не содержать часть, связанную с машинным обучением. В этой книге мы, разумеется, будем рассматривать проекты, в которых машинное обучение используется.

После определения технического проекта как раз и начинается область инженерии машинного обучения. Машинное обучение в рамках более широкого технического проекта прежде всего должно иметь четко определенную **цель**. Целью машинного обучения является описание того, что статистическая модель получает на входе, что она генерирует на выходе, и критериев приемлемого (или неприемлемого) поведения модели.

Цель машинного обучения не обязательно совпадает с бизнес-целью. Бизнес-цель – это то, чего хочет достичь организация. Например, бизнес-цель компании Google в случае Gmail может заключаться в том, чтобы сделать Gmail наиболее популярной почтовой службой в мире. Для достижения этой цели Google может создавать многочисленные технические проекты, включающие машинное обучение. Цель одного из таких проектов может состоять в том, чтобы отличать содержательные электронные письма от рекламных акций с точностью выше 90 %.

В целом жизненный цикл проекта машинного обучения, показанный на рис. 1.4, состоит из следующих этапов: 1) определение цели, 2) сбор и подготовка данных, 3) конструирование признаков, 4) обучение модели, 5) оценивание модели, 6) развертывание модели, 7) выполнение модели, 8) мониторинг модели и 9) сопровождение модели.

На рис. 1.4 область применения машинного обучения (и этой книги) ограничена синей зоной. Сплошные стрелки показывают типичный технологический поток. Пунктирные стрелки означают, что на некоторых этапах может быть принято решение вернуться назад и либо собрать больше данных, либо собрать другие данные и пересмотреть признаки (исключив одни и сконструировав другие).

Каждый из упомянутых этапов будет рассмотрен в одной из глав книги. Но сначала давайте обсудим вопрос о том, как расставить приоритеты в проектах машинного обучения, определить цель проекта и организовать коллектив разработчиков. Этим трем вопросам посвящена следующая глава.

---

[1] Если проект машинного обучения используется для поддержки продукта, разработанного и продаваемого организацией, то бизнес-аналитик работает с владельцем продукта.

Рис. 1.4 ❖ Жизненный цикл проекта машинного обучения

1.8. Резюме

Алгоритм машинного обучения на основе модели принимает на входе набор обучающих примеров и выдает на выходе модель. Алгоритм машинного обучения на основе экземпляров использует весь обучающий набор данных в качестве модели. Обучающие данные предъявляются алгоритму машинного обучения, а зарезервированные данные – нет.

Алгоритм обучения с учителем строит модель, которая принимает вектор признаков и выводит для него предсказание. Алгоритм обучения без учителя строит модель, которая принимает вектор признаков и преобразовывает его во что-то полезное.

Классификация – это задача предсказания для входного примера одного из конечного множества классов.

В задачу регрессии входит предсказание числового целевого показателя.

Данные могут использоваться прямо либо косвенно. Прямо используемые данные являются основой для формирования набора примеров. Косвенно используемые данные используются для обогащения этих примеров.

Данные для машинного обучения должны быть аккуратными. Аккуратный набор данных можно трактовать как электронную таблицу, в которой строки представляют примеры, а столбцы – свойства примеров. Помимо аккуратности, в большинстве алгоритмов машинного обучения данные должны быть числовыми, а не категориальными. Конструирование признаков – это процесс преобразования данных в форму, которая может использоваться алгоритмами машинного обучения.

Ориентир необходим для того, чтобы модель могла работать лучше, чем простая эвристика.

На практике машинное обучение реализуется в виде конвейера, содержащего последовательные этапы преобразования данных, от разбиения и подстановки пропущенных данных до устранения несбалансированности классов, понижения размерности и обучения модели. Гиперпараметры всего конвейера обычно оптимизируются; конвейер можно развернуть и использовать для предсказаний.

Параметры модели оптимизируются алгоритмом обучения на основе тренировочных данных. Значения гиперпараметров не обучаются алгоритмически, а настраиваются с помощью контрольного набора данных. Тестовый набор используется только для оценивания качества модели и предоставления отчета о ней клиенту или владельцу продукта.

Алгоритм поверхностного обучения обучает модель, которая делает предсказания непосредственно по входным данным. Алгоритм глубокого обучения обучает многослойную модель, в которой каждый слой генерирует выходные данные, принимая на входе результаты предыдущего слоя.

Рассматривать возможность использования машинного обучения для решения деловой задачи следует, когда задачу слишком сложно закодировать, задача постоянно меняется, задача подразумевает восприятие, представляет неизученное явление, имеет простую целевую функцию и является экономически эффективной.

Во многих ситуациях машинное обучение использовать не стоит: когда требуется объяснимость, когда ошибки недопустимы, когда традиционная программная инженерия менее затратна, когда все входные и выходные данные могут быть перечислены и сохранены в базе данных, а также когда данные трудно или слишком дорого собрать.

Инженерия машинного обучения (MLE) – это использование научных принципов, инструментов и методов машинного обучения и традиционной программной инженерии для проектирования и построения сложных компьютерных систем. MLE охватывает все этапы от сбора данных до обучения модели и использования ее продуктом или потребителями.

Жизненный цикл проекта машинного обучения состоит из следующих этапов: 1) определение цели, 2) сбор и подготовка данных, 3) конструирование признаков, 4) обучение модели, 5) оценивание модели, 6) развертывание модели, 7) выполнение модели, 8) мониторинг модели и 9) сопровождение модели.

Каждый этап будет рассмотрен в одной из глав книги.