Педро Домингос

ВЕРХОВНЫЙ АЛГОРИТМ

КАК МАШИННОЕ ОБУЧЕНИЕ ИЗМЕНИТ НАШ МИР

Эту книгу хорошо дополняют:

Искусственный интеллект
Ник Бостром

Красота в квадрате
Алекс Беллос

Теория игр
Авинаш Диксит и Барри Нейлбафф

Pedro Domingos

The Master Algorithm

How the Quest for the Ultimate Learning Machine Will Remake Our World

BASIC BOOKS
A Member of the Perseus Books Group
New York

Педро Домингос

Верховный алгоритм

Как машинное обучение изменит наш мир

Москва
«МАНН, ИВАНОВ И ФЕРБЕР»
2016

Информация от издательства

Научные редакторы Александр Сбоев, Алексей Серенко

Издано с разрешения Pedro Domingos c/o Levine Greenberg Rostan Literary Agency и литературного агентства Synopsis

На русском языке публикуется впервые

Домингос, Педро
Верховный алгоритм: как машинное обучение изменит наш мир / Педро Домингос ; пер. с англ. В. Горохова ; [науч. ред. А. Сбоев, А. Серенко]. — М. : Манн, Иванов и Фербер, 2016.

ISBN 978-5-00100-172-0

Машинное обучение преображает науку, технологию, бизнес и позволяет глубже узнать природу и человеческое поведение. Программирующие сами себя компьютеры — одна из самых важных современных технологий, и она же — одна из самых таинственных.

Ученый-практик Педро Домингос приоткрывает завесу и впервые доступно рассказывает о машинном обучении и о поиске универсального обучающегося алгоритма, который сможет выуживать любые знания из данных и решать любые задачи. Чтобы заглянуть в будущее и узнать, как машинное обучение изменит наш мир, не нужно специального технического образования — достаточно прочитать эту книгу.

Светлой памяти моей сестры Риты, которая проиграла битву с раком, когда я писал эту книгу

Величайшая задача науки — объяснить как можно больше экспериментальных фактов логической дедукцией, исходящей из как можно меньшего количества гипотез и аксиом.

Альберт Эйнштейн

Прогресс цивилизации заключается в увеличении количества важных действий, которые мы выполняем не думая.

Альфред Норт Уайтхед

ПРОЛОГ

Машинное обучение окружает вас повсюду, хотя, может быть, вы об этом и не подозреваете. Именно благодаря машинному обучению поисковая машина понимает, какие результаты (и рекламу) показывать в ответ на ваш запрос. Когда вы просматриваете почту, большая часть спама проходит мимо вас, потому что он был отфильтрован с помощью машинного обучения. Если вы решили что-нибудь купить на Amazon.com или заглянули на Netflix посмотреть фильм, система машинного обучения услужливо предложит варианты, которые могут прийтись вам по вкусу. С помощью машинного обучения Facebook решает, какие новости вам показывать, а Twitter подбирает подходящие твиты. Когда бы вы ни пользовались компьютером, очень вероятно, что где-то задействовано машинное обучение.

Единственным способом заставить компьютер что-то делать — от сложения двух чисел до управления самолетом — было составление некоего алгоритма, скрупулезно объясняющего машине, что именно от нее требуется. Однако алгоритмы машинного обучения — совсем другое дело: они угадывают все сами, делая выводы на основе данных, и чем больше данных, тем лучше у них получается. Это значит, что компьютеры не надо программировать: они программируют себя сами.

Это верно не только в киберпространстве: машинным обучением пронизана вся наша жизнь, начиная с пробуждения и заканчивая отходом ко сну.

Семь утра. Будильник включает радио. Играет незнакомая, но очень приятная музыка: благодаря сервису Pandora[1] радио познакомилось с вашими вкусами и превратилось в «персонального диджея». Не исключено, что сама песня тоже появилась на свет с помощью машинного обучения. За завтраком вы листаете утреннюю газету. Несколькими часами ранее она сошла с печатного станка, а тот был тщательно настроен с помощью обучающегося алгоритма, позволяющего устранить типографские дефекты. В комнате исключительно комфортная температура, а счета

за электричество не кусаются, потому что вы поставили умный термостат Nest.

По дороге на работу автомобиль постоянно корректирует впрыск топлива и рециркуляцию выхлопных газов, чтобы свести к минимуму расходы на бензин. В часы пик Inrix, система прогнозирования трафика, экономит время, не говоря уже о нервах. На работе машинное обучение помогает вам бороться с информационной перегрузкой: вы используете куб данных[2], чтобы суммировать большой объем информации, смотрите на него под разными углами и подробно изучаете все самое важное. Если надо принять решение, какой макет сайта — *A* или *B* — привлечет больше перспективных клиентов, обученная система протестирует оба варианта и предоставит вам отчет. Надо заглянуть на сайт потенциального поставщика, а он на иностранном языке? Никаких проблем: Google автоматически его для вас переведет. Электронные письма удобно рассортированы по папкам, а во «Входящих» осталось только самое важное. Текстовый процессор проверяет за вас грамматику и орфографию. Вы нашли авиарейс для предстоящей командировки, но билет пока не покупаете, потому что, по прогнозу Bing Travel, цены вскоре станут ниже. Сами того не осознавая, вы ежечасно делаете намного больше работы, чем могли бы без помощи машинного обучения.

В свободную минуту вы проверяете, как там ваши вклады в фонде взаимных инвестиций. Большинство таких фондов используют обучающиеся алгоритмы для выбора перспективных акций, а одним из них вообще полностью управляет система на основе машинного обучения. Во время обеда вы выходите на улицу и думаете, где бы перекусить. Обучающаяся система Yelp в смартфоне вам поможет. Мобильные телефоны вообще под завязку наполнены обучающимися алгоритмами, которые без устали исправляют опечатки, узнают голосовые команды, корректируют ошибки передачи данных, считывают штрихкоды и делают много других полезных дел. Смартфон даже научился угадывать ваше следующее действие и давать полезные советы. Например, он подскажет, что встреча начнется позже, потому что самолет, на котором должен прилететь ваш гость, задерживается.

Если вы закончите работать поздно вечером, машинное обучение поможет без приключений дойти до машины на парковке: алгоритмы отслеживают видео с камеры наблюдения и дистанционно предупреждают охрану, когда замечают что-то подозрительное. Предположим, по дороге домой вы тормозите у супермаркета. Товары на полках расположены согласно указаниям алгоритмов с обучением: именно они решают, какие товары лучше заказать, что поставить в конце ряда и где место сальсе — в отделе соусов или рядом с чипсами тортильяс. На кассе вы расплачиваетесь кредитной карточкой. В свое время обученный алгоритм решил, что вам надо отправить предложение ее оформить, а затем одобрил вашу заявку. Другой алгоритм постоянно выискивает подозрительные операции и непременно предупредит вас, если ему покажется, что номер карточки украден. Третий алгоритм пытается понять, насколько вы удовлетворены. Если вы хороший клиент, но выглядите недовольным, вам отправят «подслащенное» предложение еще до того, как вы уйдете к конкурентам.

Вернувшись домой, вы подходите к почтовому ящику и находите там письмо от друга. Оно было доставлено благодаря алгоритму, который научился читать написанные от руки адреса. Кроме письма в ящике лежит обычная макулатура, тоже отобранная для вас алгоритмами с обучением (ничего не поделаешь). Вы на минутку останавливаетесь, чтобы подышать свежим вечерним воздухом. Преступников в городе сильно поубавилось, с тех пор как полиция начала использовать статистическое обучение для прогнозирования вероятности правонарушений и направлять в проблемные районы патрульных. Вы ужинаете в кругу семьи и смотрите телевизор. В новостях показывают мэра. Вы за него проголосовали, потому что в день выборов он лично вам позвонил. Ему на вас указал обучающийся алгоритм, увидевший в вас ключевого неопределившегося избирателя. После ужина можно посмотреть футбол. Обе команды подбирали игроков с помощью статистического обучения. Или лучше поиграть с детьми в Xbox? В таком случае обучающийся алгоритм в приставке Kinect будет отслеживать положение и движения вашего тела. Прежде чем отойти ко сну, вы принимаете лекарство, разработанное и протестированное с помощью алгоритмов с обучением.

Не исключено, что даже ваш доктор пользовался машинным обучением при постановке диагноза, начиная с интерпретации рентгенограммы и заканчивая выводом на основе необычного набора симптомов.

Машинное обучение делает свое дело на всех этапах жизни человека. Если вы готовились к экзаменам в колледж с помощью интернета, специальный алгоритм оценивал ваши пробные сочинения. А если вы недавно поступали в бизнес-школу и сдавали GMAT[3], обучающаяся система была одним из экзаменаторов, оценивающих эссе. Возможно, когда вы устраивались на работу, обученный алгоритм выудил ваше резюме из «виртуальной кучи» и сказал потенциальному работодателю: «Взгляни, вот сильная кандидатура». Вполне вероятно, что недавним повышением зарплаты вы тоже обязаны какому-то обученному алгоритму. Если вы собираетесь купить дом, Zillow.com оценит, чего стоит каждый заинтересовавший вас вариант. Когда вы определитесь и решите взять ипотеку, алгоритм на основе машинного обучения рассмотрит заявку и порекомендует ее одобрить (или отклонить). И наверное, самое важное: если вас интересуют интернет-знакомства, машинное обучение поможет найти настоящую любовь.

Общество меняется с каждым новым алгоритмом. Машинное обучение преображает науку, технологию, бизнес, политику, военное искусство. Спутники и ускорители частиц зондируют природу все более тщательно, а обучающиеся алгоритмы превращают реки данных в новое научное знание. Компании знают своих клиентов, как никогда раньше. На выборах побеждают кандидаты, умеющие лучше моделировать поведение избирателей (пример — Обама против Ромни[4]). Беспилотные транспортные средства завоевывают сушу, воду и воздушное пространство. В систему рекомендаций Amazon никто не вводит информацию о наших вкусах: обучающийся алгоритм определяет их самостоятельно, обобщая сведения о сделанных покупках. Беспилотный автомобиль Google сам научился не съезжать с дороги: никакой инженер не писал для него алгоритм, шаг за шагом объясняющий, как добраться из точки *A* в точку *B*. Никто не знает, как написать программу вождения автомобиля, да никому это

и не надо, потому что машина, оборудованная обучающимся алгоритмом, посмотрит на действия водителя и разберется сама.

Машинное обучение — технология, которая строит саму себя. Это новое явление в нашем мире. С тех пор как наши далекие предки научились заострять камни и смастерили первые орудия труда, человечество разработало артефакты самостоятельно, вручную или массово. Обучающиеся алгоритмы — артефакты, которые создают другие артефакты. «От компьютеров никакой пользы, — говорил Пикассо. — Они умеют только давать ответы». Компьютеры не предназначены для творчества: они должны делать ровно то, что им говорят. Но если приказать им заняться творчеством, получится машинное обучение. Обучающийся алгоритм как искусный ремесленник: каждое из его творений уникально, и каждое создано именно таким, каким пожелал заказчик. Просто в отличие от мастеров обучающиеся алгоритмы превращают не камень в кладку и не золото в ювелирные изделия, а данные в алгоритмы. И чем больше у них данных, тем качественнее может получиться алгоритм.

Homo sapiens научился приспосабливать мир под себя, вместо того чтобы самому приспосабливаться к существующим условиям. Машинное обучение открывает новую главу в долгой, растянувшейся на миллион лет эволюционной саге: с его помощью мир сам почувствует, чего вы хотите, и сам под вас подстроится. Не надо даже волшебной палочки: окружающий вас мир — сегодня виртуальный, а завтра физический — станет похож на волшебный лес. Если вы пойдете по тропинке в чаще, она станет дорогой. Если вы заблудитесь, из ниоткуда появятся стрелки, указывающие направление.

Эти волшебные технологии возможны потому, что глубинная суть машинного обучения — предсказание: предсказание наших желаний, результатов наших действий, путей достижения целей, изменений мира. Когда-то нам приходилось полагаться на шаманов и прорицателей, но это оказалось слишком ненадежно. Научные прогнозы более достойны доверия, однако они ограничены областями, которые мы можем систематически наблюдать и которые поддаются моделированию. Большие данные и машинное обучение заметно расширили эти границы. Иногда

человек может предсказывать и без посторонней помощи, например, когда ловит мячик или ведет разговор. Бывает, что предсказать не получится, как бы мы ни старались. Но между этими крайностями лежит широкая область, для которой пригодится машинное обучение.

Хотя обучающиеся алгоритмы позволяют глубже узнать природу и человеческое поведение, сами они, как ни странно, окутаны пеленой тайны. Не проходит и дня, чтобы в СМИ не появилась новая история, связанная с машинным обучением, будь то запуск Apple личного помощника Siri, суперкомпьютер IBM Watson, победивший чемпиона в Jeopardy! (аналог «Своей игры»)[5], торговая сеть Target, узнавшая о беременности подростка раньше родителей, или Агентство национальной безопасности, собирающее воедино разрозненные улики. Однако во всех этих случаях обучающиеся алгоритмы, сделавшие эти истории возможными, остаются для зрителей черным ящиком. Даже книги о больших данных обходят стороной вопрос, как именно компьютер, проглотив все эти терабайты, волшебным образом приходит к ценным выводам. В лучшем случае у нас остается впечатление, что обучающиеся алгоритмы просто находят корреляции между двумя событиями, например запросом «лекарство от простуды» в строке Google и самой простудой. Однако нахождение корреляций для машинного обучения — не более чем кирпичи для дома. В горе кирпичей жить не получится.

Если новая технология пронизывает нашу жизнь до такой степени, как машинное обучение, нельзя, чтобы она оставалась для нас загадкой. Неясности создают благодатную почву для ошибок и неправильного применения. Алгоритм Amazon лучше, чем любой человек, умеет определять, какие книги читают сегодня в мире. Алгоритмы Агентства национальной безопасности способны узнать в человеке потенциального террориста. Моделирование климата находит безопасный уровень углекислого газа в атмосфере, а модели подбора акций больше вкладывают в развитие экономики, чем большинство из нас. Но нельзя контролировать то, чего не понимаешь, и именно поэтому вы должны понимать машинное обучение — как гражданин, как специалист и как человек, стремящийся к счастью.

Первейшая задача этой книги — посвятить вас в секреты машинного обучения. Разбираться в автомобильном двигателе нужно только инженерам и механикам, однако любой водитель должен знать, что поворот руля меняет направление движения, а если нажать на тормоз, машина остановится. Сегодня лишь немногие имеют представление об обучающихся алгоритмах хотя бы на таком уровне, не говоря уже об умении ими пользоваться. Психолог Дональд Норман придумал термин «концептуальная модель»: это грубое знание какой-либо технологии, достаточное для того, чтобы эффективно ею пользоваться. Эта книга даст вам концептуальную модель машинного обучения.

Не все обучающиеся алгоритмы работают одинаково, и это имеет определенные последствия. Возьмем, например, системы рекомендаций Amazon и Netflix и прогуляемся с ними по обычному книжному магазину. Пытаясь найти книги, которые «точно вам понравятся», Amazon, скорее всего, подведет вас к полке, к которой вы в прошлом чаще подходили, а Netflix позовет вас в незнакомый и неочевидный на первый взгляд уголок, но то, что вы там найдете, обязательно вам понравится. Из этой книги вы узнаете, что у Amazon и Netflix просто разные типы алгоритмов. Алгоритм Netflix вникает в ваши вкусы глубже (хотя все еще довольно скромно), однако, как ни странно, это еще не значит, что Amazon выиграла бы от такого подхода. Дело в том, что для успешного развития бизнеса Netflix нужно направлять спрос к длинному шлейфу малоизвестных и поэтому недорогих фильмов и телешоу и отвлекать клиентов от блокбастеров, на оплату которых абонемента просто не хватит. У менеджеров Amazon такой проблемы нет: им тоже выгодно сбыть неходовые товары, но продавать популярные и дорогие варианты не менее приятно (к тому же это упрощает логистику). Кроме того, клиенты с большей вероятностью посмотрят что-то необычное по подписке, чем купят специально.

Каждый год в мире появляются сотни новых алгоритмов с обучением, но все они основаны на небольшом наборе фундаментальных идей. Именно этим идеям и посвящена эта книга, и их вам будет вполне достаточно, чтобы понять, как машинное обучение меняет наш мир. Не уходя в дебри и даже не очень

затрагивая применение алгоритмов в компьютерах, мы дадим ответы на важные для всех нас вопросы: «Как мы учимся?», «Можно ли учиться эффективнее?», «Что мы способны предсказать?», «Можно ли доверять полученному знанию?» Соперничающие школы машинного обучения отвечают на эти вопросы по-разному. Всего существует пять основных научных течений, каждому из которых мы посвятим отдельную главу. Символисты рассматривают обучение как процесс, обратный дедукции, и черпают идеи из философии, психологии и логики. Коннекционисты[6] воссоздают мозг путем обратной инженерии и вдохновляются нейробиологией и физикой. Эволюционисты симулируют эволюцию на компьютерах и обращаются к генетике и эволюционной биологии. Сторонники байесовского подхода[7] полагают, что обучение — это разновидность вероятностного вывода, и корни этой школы уходят в статистику. Аналогисты занимаются экстраполяцией на основе схожести суждений и находятся под влиянием психологии и математической оптимизации. Стремясь построить обучающиеся машины, мы пройдемся по истории мысли за последнюю сотню лет и увидим ее в новом свете.

У каждого из пяти «племен» машинного обучения есть собственный универсальный обучающийся — Верховный — алгоритм, который в принципе можно использовать для извлечения знания из данных в любой области. Для символистов это обратная дедукция, для коннекционистов — обратное распространение ошибки, для эволюционистов — генетическое программирование, для байесовцев — байесовский вывод, а для аналогистов — метод опорных векторов. Однако на практике каждый из этих алгоритмов хорош для одних задач, но не очень подходит для других. Хотелось бы, чтобы все их черты слились воедино в окончательном, совершенном Верховном алгоритме. Кто-то считает это несбыточной мечтой, но у многих из нас — людей, занимающихся машинным обучением, — при этих словах загораются глаза, и мечта заставляет нас работать до поздней ночи.

Верховный алгоритм сумеет извлечь из данных вообще все знание — знание прошлого, настоящего и будущего. Изобретение этого алгоритма станет одним из величайших прорывов в истории

науки. Оно ускорит прогресс буквально во всем, изменит мир так, как мы едва можем себе сегодня представить. Верховный алгоритм для машинного обучения — это нечто вроде стандартной модели в физике элементарных частиц и центральной догмы молекулярной биологии: единая теория, объясняющая все, что мы сегодня знаем, и закладывающая фундамент десятилетий или целых веков будущего прогресса. Верховный алгоритм — ключ к решению стоящих перед человечеством сложнейших проблем — от создания домашних роботов до лечения рака.

Ведь рак так сложно лечить, потому что это не одно, а целый комплекс заболеваний. Опухоли бывают вызваны невообразимо широким спектром причин, к тому же они мутируют и дают метастазы. Самый надежный способ уничтожить опухоль — секвенировать[8] ее геном, определить, какие лекарства помогут без ущерба для пациента с учетом *конкретного* генома и истории болезни, и, возможно, даже разработать новое лекарство именно для данного случая. Ни один врач не в состоянии овладеть всеми необходимыми для этого знаниями, но решение таких задач — идеальная работа для машинного обучения. В сущности, это просто более сложная и серьезная версия поиска, которым каждый день занимаются Amazon и Netflix, только ищем мы не подходящую книгу или фильм, а подходящее лекарство. К сожалению, хотя обучающиеся алгоритмы уже умеют со сверхчеловеческой точностью диагностировать многие болезни, лечение рака выходит далеко за пределы их возможностей. Если нам удастся отыскать Верховный алгоритм, ситуация изменится. Поэтому вторая цель этой книги — помочь *вам самостоятельно* изобрести его. Можно подумать, что для этого нужны глубочайшие познания в математике и серьезная теоретическая работа. Отнюдь нет. Для этого нужно как раз отвлечься от тайн математики и посмотреть на всеобъемлющие механизмы обучения, и здесь неспециалист, подходящий к лесу издалека, во многом находится в более выгодном положении, чем профессионал, увязнувший в изучении отдельных деревьев. Концептуальное решение проблемы можно дополнить математикой, но это не самое главное, и к тому же не тема этой книги. Так что, когда мы будем заходить в гости к каждому «племени», надо будет собрать кусочки мозаики

и сообразить, куда они подходят, не забывая при этом, что ни один слепец не может увидеть слона целиком. Мы увидим, какой вклад каждое из «племен» способно внести в лечение рака, чего ему не хватает, а затем шаг за шагом соберем кусочки в решение — вернее, *одно из* решений. Конечно, это не будет Верховным алгоритмом, но ближе к нему еще никто не подбирался. Будем надеяться, что результат станет удачной отправной точкой для вашего воображения. Потом мы посмотрим, как можно было бы использовать полученный алгоритм в качестве оружия в борьбе с раком. Читая эту книгу, не стесняйтесь пробегать глазами или пропускать сложные для понимания фрагменты. Важна общая картина, и, возможно, будет полезнее вернуться к этим местам уже после того, как мозаика сложится.

Я занимаюсь исследованиями машинного обучения более 20 лет. Интерес к этой теме во мне пробудила книга со странным названием, которую на последнем курсе колледжа я заприметил в книжном магазине. Она называлась «Искусственный интеллект». Машинному обучению в ней была посвящена одна короткая глава, но, прочитав ее, я немедленно пришел к убеждению, что в ней — ключ к искусственному интеллекту, что знаем мы об этой области так немного и что, может быть, я смогу внести свой вклад. Поэтому я распрощался с планами получить степень MBA и поступил в аспирантуру в Калифорнийском университете в Ирвайне. Машинное обучение было тогда второстепенной дисциплиной, а в Калифорнийском университете работала одна из немногих приличных исследовательских групп. Некоторые из моих однокурсников ушли, сочтя тему бесперспективной, но я не сдался. Для меня не было ничего важнее, чем научить компьютеры учиться — ведь если удастся это сделать, мы получим фору в решении любой другой проблемы. Прошло пять лет. Я заканчивал аспирантуру, а на дворе царила революция добычи данных. Диссертацию я посвятил объединению подхода символистов и аналогистов, бо́льшую часть последнего десятилетия соединял символизм и байесовский подход, а в последнее время — оба этих метода с коннекционизмом. Теперь пора сделать следующий шаг и попытаться свести воедино все пять парадигм.

Работая над этой книгой, я представлял себе несколько разных, но пересекающихся групп читателей.

Если вам просто любопытно, откуда столько шума вокруг больших данных и машинного обучения, и вы подозреваете, что тут все не так просто, как пишут в газетах, вы правы! Эта книга станет для вас своеобразным путеводителем.

Если вы интересуетесь прежде всего применением машинного обучения в бизнесе, она поможет вам 1) стать более разборчивым потребителем аналитики; 2) получить максимальную отдачу от своих специалистов по обработке и анализу информации; 3) избежать ловушек, которые убили столь многие проекты по добыче данных; 4) узнать, какие области можно автоматизировать без затрат на ручное кодирование программ; 5) уменьшить жесткость своих информационных систем и 6) предвидеть появление ряда новых технологий, которые уже не за горами. Я много раз наблюдал, как деньги и время уходят впустую из-за того, что проблемы решаются неправильным обучающимся алгоритмом, и как неверно интерпретируют то, что алгоритм сообщает. Чтобы избежать фиаско, достаточно лишь прочитать эту книгу.

Если вы сознательный гражданин или если вы отвечаете за решение социальных, государственных вопросов, возникших после появления больших данных и машинного обучения, эта книга станет для вас азбукой новой технологии. Не углубляясь в скучные подробности, вы узнаете, что эта технология собой представляет, к чему может привести, что она умеет, а чего нет. Вы увидите, в чем состоят реальные проблемы — от защиты частной жизни до рабочих мест в будущем и этики боевых роботов — и как к ним подступиться.

Если вы ученый или инженер, машинное обучение — мощнейший инструмент, который очень вам пригодится. Старые, проверенные временем статистические подходы не спасут вас в век больших (и даже средних) объемов данных. Для точного моделирования большинства явлений понадобятся нелинейные подходы машинного обучения, и оно несет с собой новое научное мировоззрение. В наши дни любят злоупотреблять выражением

«смена парадигмы», но я уверен, что тема моей книги именно так и звучит.

Даже если вы эксперт по машинному обучению и уже знакомы с большинством освещаемых мной тем, вы найдете в книге много свежих идей, экскурсов в историю, полезных примеров и аналогий. Я очень надеюсь, что это поможет вам по-новому взглянуть на машинное обучение и, может быть, даже по-новому направит ваши мысли. Полезно сорвать то, что висит на нижних ветках, однако не стоит терять из виду и то, что расположено чуть выше. (Кстати, прошу простить мне поэтическую вольность — эпитет «Верховный» в отношении универсального обучающегося алгоритма.)

Если вы учащийся любого возраста: старшеклассник, выбирающий, где учиться, студент старших курсов, размышляющий, идти ли в науку, или маститый ученый, планирующий изменение карьеры, моя книга, надеюсь, заронит в вас искорку интереса к этой захватывающей области знания. В мире остро не хватает специалистов по машинному обучению, и, если вы решите к нам присоединиться, можете быть уверены: вас ждут не только захватывающие мгновения и материальные блага, но и уникальный шанс послужить обществу. А если вы уже осваиваете машинное обучение, надеюсь, эта книга поможет вам лучше сориентироваться в теме. Если в своих поисках вы случайно наткнетесь на Верховный алгоритм, ради этого мне стоило браться за перо.

И последнее, но не менее важное. Если вы хотите ощутить вкус чуда, машинное обучение — настоящий пир для ума. Примите приглашение и угощайтесь!

ГЛАВА 1

РЕВОЛЮЦИЯ МАШИННОГО ОБУЧЕНИЯ

Мы живем в эпоху алгоритмов. Всего поколение-другое назад слово «алгоритм» у большинства людей вызвало бы лишь непонимание. Сегодня алгоритмы проникли во все уголки нашей цивилизации. Они вшиты в ткань повседневной жизни и нашли себе место не только в мобильных телефонах и ноутбуках, но и в автомобилях, квартирах, бытовой технике и игрушках. Так, банк — гигантское хитросплетение алгоритмов, а люди просто слегка регулируют настройки то тут, то там. Алгоритмы составляют расписание полетов, а затем ведут самолеты. Алгоритмы управляют производством, торговлей, снабжением, подсчитывают выручку и занимаются бухгалтерией. Если все алгоритмы вдруг перестанут работать, настанет конец света — такого, каким мы его знаем.

Алгоритм — определенная последовательность инструкций, диктующая компьютеру его действия. Компьютеры состоят из миллиардов крохотных переключателей — транзисторов, и алгоритмы включают и выключают эти транзисторы миллиарды раз в секунду.

Самый простой алгоритм — «нажми переключатель». Положение одного транзистора — одна единица информации: «один», если транзистор включен, и «ноль», если выключен. Единичка где-то в компьютерах банка информирует, превысили ли вы кредит. Еще одна единичка в недрах Управления социального обеспечения сообщает, живы вы или уже умерли.

Второй простейший алгоритм — «соедини два бита». Клод Шеннон, признанный отец теории информации, первым осознал, что включение и выключение транзисторов в ответ на действия других транзисторов — это, в сущности, логический вывод. (Этой теме он посвятил свою дипломную работу в Массачусетском технологическом институте — самую важную дипломную работу в истории.) «Транзистор *A* включается, только если включены транзисторы *B* и *C*» — это крохотное логическое рассуждение. «*A* включается, когда включен либо *B*, либо *C*» — еще одна крупица логики. «*A* включается всегда, когда выключен *B*, и наоборот» —

третья операция. Хотите верьте, хотите нет, любой алгоритм, как бы сложен он ни был, сводится всего к трем операциям: И, ИЛИ и НЕ. Используя для этих операций специальные символы, можно представить простые алгоритмы в виде диаграмм. Например, если у человека грипп или малярия и ему надо принять лекарство от температуры и головной боли, это можно выразить следующим образом:

Соединяя множество подобных операций, можно составлять очень сложные цепочки логических рассуждений. Люди часто думают, что вся суть компьютеров в вычислениях, но это не так. Сердце компьютеров — логика. Из логики в компьютере состоят и числа, и арифметика, и все остальное. Хотите сложить два числа? Есть комбинация транзисторов, которая это сделает. Хотите победить чемпиона в «Своей игре»? Для этого тоже найдется комбинация (естественно, она будет намного больше).

Однако строить новый компьютер для каждой новой задачи, которая нам придет в голову, было бы невероятно дорого, поэтому современный компьютер представляет собой большую совокупность транзисторов, способных решать много разных задач в зависимости от того, какие из них активны. Микеланджело говорил, что вся его работа — увидеть статую в глыбе мрамора и открыть ее миру, убрав лишнее. Аналогично алгоритмы «отсекают» избыточные транзисторы в компьютере, пока не обнажается нужная функция, будь то автопилот авиалайнера или новый мультфильм студии Pixar.

Алгоритм — не просто произвольный набор инструкций. Чтобы компьютер его выполнил, указания должны быть достаточно точными и однозначными. Например, кулинарный рецепт — это не алгоритм, потому что не задает однозначного порядка действий и не объясняет, что делать на каждом этапе. Например, сколько

именно сахара умещается в столовую ложку? Любой человек, который хоть раз пробовал готовить по незнакомому рецепту, знает, что может получиться и восхитительное блюдо, и не пойми что. А алгоритмы всегда дают идентичный результат. При этом, даже если указать в рецепте ровно 15 граммов сахара, это по-прежнему не решает проблему, потому что компьютер не знает ни что такое сахар, ни что такое грамм. Если бы мы захотели запрограммировать робота-повара для выпечки тортов, пришлось бы научить его узнавать сахар на видео, научить брать ложку и так далее (ученые все еще над этим работают). Компьютер должен знать, как выполнять алгоритм — вплоть до включения и выключения конкретных транзисторов, поэтому рецепт готовки очень далек от алгоритма.

С другой стороны, вот вам алгоритм игры в крестики-нолики.

Если вы или ваш противник поставили две отметки на одной линии, ставьте отметку в оставшейся на этой линии клетке.

Если такой ход невозможен, но есть ход, который создаст две линии по две отметки, — делайте его.

Если такой ход невозможен, но центральная клетка свободна, ставьте отметку в ней.

Если такой ход невозможен, но противник поставил отметку в углу, ставьте отметку в противоположном углу.

Если такой ход невозможен, но одна из угловых клеток свободна, ставьте отметку в ней.

Если такой ход невозможен, ставьте отметку в любой пустой клетке.

У этого алгоритма есть одно приятное свойство: он беспроигрышный! Конечно, ему не хватает многих деталей — как доска отображается в памяти компьютера и как это отображение меняется после каждого хода. Например, каждой клетке могут соответствовать два бита: 00 — если клетка пуста, 01 — если в ней поставили нолик и 10 — если крестик. Тем не менее предложенный алгоритм достаточно точен и однозначен, и любой грамотный программист сможет его дописать. Еще полезно не конкретизировать алгоритмы вплоть до отдельных

транзисторов, а пользоваться уже существующими алгоритмами как кирпичиками. Их огромное количество, так что есть из чего выбирать.

Алгоритмы предъявляют строгие требования: часто говорят, что по-настоящему понимаешь что-то только тогда, когда можешь выразить это в виде алгоритма (как заметил Ричард Фейнман[9], «я не понимаю того, чего не могу создать»). Уравнения — хлеб насущный физиков и инженеров — на самом деле всего лишь особая разновидность алгоритмов. Например, второй закон Ньютона, который считают самым важным в мире уравнением, гласит, что для вычисления действующей на тело суммарной силы надо массу этого тела умножить на его ускорение. Он также подразумевает, что ускорение — это сила, разделенная на массу, но выведение этого следствия тоже алгоритм. Если теорию в любой научной дисциплине не получается выразить в виде алгоритма, она недостаточно строгая, не говоря уже о том, что ее решение нельзя компьютеризировать, а это всерьез ограничивает сферу ее применения. Ученые строят теории, инженеры изобретают устройства, а специалисты в области информатики создают алгоритмы, которые представляют собой и теории, и устройства одновременно.

Написать алгоритм непросто: есть очень много ловушек, и ни в чем нельзя быть уверенным. Интуитивные предположения вполне могут оказаться ошибочными, и тогда придется искать другой подход. Затем алгоритм надо выразить на понятном компьютеру языке, например Java или Python, и с этого момента алгоритм начнет называться программой. Потом программу надо отладить: найти все до единой ошибки и исправить их, пока компьютер не начнет выполнять ее без запинки. Но когда у вас наконец появится программа, которая умеет делает то, что вам нужно, вы получите все козыри. Компьютер станет послушно выполнять ваши задания миллионы раз со сверхвысокой скоростью. Созданной вами программой сможет пользоваться любой человек в мире. Она даже сделает вас миллиардером, если решенная проблема достаточно важна. Программист — человек, пишущий алгоритмы и кодирующий их, — маленький бог, создающий вселенные по своему желанию. Можно даже сказать, что сам

Господь тоже был программистом, ведь в Книге Бытия он творил с помощью слов, а не руками. Речения стали мирами. Сегодня, сидя в кресле перед ноутбуком, вы тоже можете почувствовать себя богом: нарисуйте в воображении Вселенную и сделайте ее реальной. Законы физики соблюдать необязательно.

Со временем информатики начинают опираться на уже проделанную работу и придумывают алгоритмы для все новых процессов. Одни алгоритмы соединяются с другими, чтобы использовать результаты третьих, производя, в свою очередь, еще больше алгоритмов. Каждую секунду миллиарды раз переключаются миллиарды транзисторов в миллиардах компьютеров. Алгоритмы образуют экосистему нового типа — непрерывно растущую и сопоставимую по богатству лишь с самой жизнью.

Однако, как это всегда бывает, в райском саду обитает змей — Монстр Сложности. У него, как у лернейской гидры, много голов. Одна из них — пространственная: количество битов информации, которое алгоритм должен хранить в памяти компьютера. Если алгоритму требуется больше памяти, чем есть в наличии, он бесполезен, и его приходится отбрасывать. У пространственной сложности есть злая сестрица: временн*ая* сложность. Сколько будет длиться выполнение алгоритма, то есть сколько раз нужно использовать транзисторы, прежде чем алгоритм даст желаемый результат? Если мы не можем столько ждать, алгоритм снова оказывается бесполезным. Но самая пугающая голова Монстра Сложности — сложность человеческая. Когда алгоритм становится слишком запутанным и непонятным для нашего скромного разума, а взаимодействия между его элементами — слишком многочисленными и обширными, в него начинают вкрадываться ошибки. Человек не в состоянии их отыскать и исправить, поэтому алгоритм не делает то, что от него требуется. Даже если каким-то образом заставить его работать, он окажется неоправданно сложным для пользователя, будет плохо взаимодействовать с другими алгоритмами и порождать все больше проблем.

Специалисты-информатики сражаются с Монстром Сложности каждый день. Когда они проигрывают, сложность прорывается в нашу жизнь. Вы, наверное, и сами замечали, как много было

проиграно битв. Тем не менее башня алгоритмов продолжает расти, хотя строить ее все труднее: каждое новое поколение алгоритмов приходится возводить на вершине предшественников, их сложность суммируется. Башня растет и растет, алгоритмы опутывают весь мир, но конструкция становится все более шаткой — как карточный домик, который только и ждет толчка. Мизерная ошибка в алгоритме — и ракета, стоившая миллиард долларов, взрывается на старте, или миллионы людей остаются без электричества. Непредвиденное взаимодействие алгоритмов — и рушится фондовый рынок.

Если программисты — маленькие боги, то Монстр Сложности — его величество Сатана. И мало-помалу он выигрывает войну.

Должен быть способ лучше.

Познакомимся с обучающимся алгоритмом

У любого алгоритма есть вход и выход: данные поступают в компьютер, алгоритм делает с ними то, что должен, и выдает результат. Машинное обучение переворачивает все задом наперед: имея в своем распоряжении данные и желаемый результат, оно выдает алгоритм, превращающий одно в другое. Обучающиеся алгоритмы — те, что создают другие алгоритмы, обученные на основе данных. С помощью машинного обучения компьютеры пишут себе программы, и нам не надо этим заниматься.

Здорово, правда?

Компьютеры сами пишут для себя программы. Эта мысль потрясает настолько, что даже страшно: если компьютеры начнут программировать сами себя, сможем ли мы их контролировать? Оказывается — и мы в этом убедимся, — людям вполне по силам с ними совладать. Но есть и другое возражение — все это слишком хорошо, чтобы быть правдой. Разве для написания алгоритмов не нужны ум, творческая жилка, умение решать проблемы — все те качества, которых у компьютеров просто нет? Чем машинное обучение отличается от магии? Все это правда: сегодня мы умеем писать много программ, которым компьютер научиться не может. Но еще удивительнее то, что и компьютеры могут научиться программам, которые не в состоянии написать человек. Мы умеем

водить машину или читать написанный от руки текст, но эти навыки у нас подсознательные: рассказать компьютеру, как это делать, не получится. Однако если дать обучающемуся алгоритму достаточное количество примеров каждого из этих действий, он с легкостью во всем разберется и без нашей помощи, и тогда можно будет развязать ему руки. Именно так машины научились читать почтовые индексы, и именно поэтому на дорогах скоро появятся автомобили без водителей.

Мощь машинного обучения, наверное, лучше всего показать, сравнив технологию с сельским хозяйством. В индустриальном обществе товары делают на заводах, а это значит, что инженерам надо точно определить, как именно их собирать, как изготавливать все элементы и так далее, вплоть до сырья. Это требует больших усилий. Самые сложные устройства, которые человеку удалось изобрести, — компьютеры, и их разработка, производство и написание для них программ требуют колоссального труда. Но есть другой, намного более древний способ получить некоторые необходимые нам вещи: предоставить их изготовление самой природе. Посадить семечко, полить его, добавить удобрений, а потом сорвать спелый плод. Может ли технология выглядеть примерно так же? Может! Именно это сулит нам машинное обучение. Обучающиеся алгоритмы — как семена, почва — это данные, а обученные программы — это наша жатва. Эксперт по машинному обучению похож на крестьянина, сеющего, поливающего и удобряющего землю. Он присматривает за здоровьем растущего урожая, но в целом не вмешивается.

Если посмотреть на машинное обучение под этим углом, сразу бросаются в глаза два момента. Во-первых, чем больше у нас данных, тем больше мы можем узнать. Нет данных? Тогда и учиться нечему. Большой объем информации? Огромное поле для обучения. Вот почему машинное обучение заявляет о себе везде, где появляются экспоненциально растущие горы данных. Если бы в магазине продавали машинное обучение быстрого приготовления, на коробке было бы написано: «Просто добавь данных».

Второе наблюдение заключается в том, что машинное обучение — это меч-кладенец, которым можно обезглавить Монстра Сложности. Если дать обучающей программе длиной всего

пару сотен строк достаточно данных, она не только с легкостью сгенерирует программу из миллионов строк кода, но и сможет делать это вновь и вновь для разных проблем. Уменьшение сложности для программиста просто феноменальное. Конечно, как и гидра, Монстр Сложности будет отращивать все новые и новые головы, но они окажутся меньше и вырастут не сразу, так что у нас все равно будет большое преимущество.

Машинное обучение можно представить себе как вывернутое наизнанку программирование, точно так же как квадратный корень противоположен возведению во вторую степень, а интегрирование обратно дифференцированию. Если можно спросить, квадрат какого числа равен 16 или производной какой функции является $x + 1$, уместен и вопрос: «Какой алгоритм даст такой результат?» Вскоре мы увидим, как превратить оба наблюдения в конкретные обучающиеся алгоритмы.

Некоторые обучающиеся алгоритмы добывают знания, а некоторые — навыки. «Все люди смертны» — это знание. Езда на велосипеде — навык. В машинном обучении знание часто предстает в форме статистических моделей, потому что знание как таковое — это во многом статистика: смертны все люди, но только четыре процента людей американцы. Навыки зачастую представляют собой наборы процедур: если дорога сворачивает влево, поверни руль влево. Если перед тобой выскочил олень, дави на тормоз. (К сожалению, на момент написания этой книги беспилотная машина Google все еще путает оленей с полиэтиленовыми пакетами.) Часто процедура довольно проста, хотя заложенное в ней знание сложно. Спам надо отправить в корзину, однако сначала придется научиться отличать его от обычных писем. Если разобраться, какая позиция на шахматной доске удачна, станет ясно, какой сделать ход (тот, что приведет к лучшей позиции).

Машинное обучение принимает много разных форм и скрывается под разными именами: распознавание паттернов, статистическое моделирование, добыча данных, выявление знаний, предсказательная аналитика, наука о данных, адаптивные и самоорганизующиеся системы и так далее. Все они находят свое применение и имеют разные ассоциации. Некоторые живут долго,

а некоторые не очень. Все это многообразие я буду называть просто — *машинное обучение*.

Машинное обучение иногда путают с искусственным интеллектом. С формальной точки зрения это действительно подраздел науки об искусственном интеллекте, однако он очень разросся и оказался настолько успешным, что затмил гордого родителя. Цель искусственного интеллекта — научить компьютеры делать то, что люди пока делают лучше, а умение учиться — наверное, самый важный из этих навыков, без которого компьютерам никогда не угнаться за человеком. Остальное приложится.

Если представить обработку данных в виде экосистемы, обучающиеся алгоритмы будут в ней суперхищниками. Базы данных, поисковые роботы, индексаторы и так далее — это травоядные, мирно пасущиеся на бескрайних лугах данных. Статистические алгоритмы, оперативная аналитическая обработка и так далее — просто хищники. Без травоядных не обойтись, потому что без них все остальное бы умерло, однако у суперхищника жизнь интереснее. Поисковый робот, как корова, пасется в интернете — поле мирового масштаба, а каждая страница в нем — травинка. Робот пощипывает травку, копии страниц оседают на его жестком диске. Затем индексатор создает список страниц, где встречается каждое слово, во многом как предметный указатель в конце книги. Базы данных похожи на слонов: они большие, тяжелые и никогда ни о чем не забывают. Среди этих степенных животных носятся статистические и аналитические алгоритмы, которые сжимают, выбирают и превращают данные в информацию. Обучающиеся алгоритмы поглощают эту информацию, переваривают ее и дают нам знание.

Эксперты по машинному обучению — элита, каста священников среди ученых-информатиков. Многие компьютерщики, особенно старшего поколения, понимают машинное обучение не так хорошо, как им хотелось бы. Дело в том, что компьютерные науки традиционно следовали в русле детерминизма, а в машинном обучении нужно мыслить в категориях статистики. Если какое-то правило, скажем, отмечать определенные письма как спам, срабатывает в 99, а не в 100 процентах случаев, это не значит, что

в нем какая-то ошибка: может быть, это лучшее, что можно сделать, и даже такая точность очень полезна. Различия в стиле мышления во многом послужили причиной, по которой Microsoft оказалось намного сложнее нагнать Google, чем в свое время Netscape. В конце концов, браузер всего лишь стандартная программа, а вот поисковая система требует другого склада ума.

Еще одна причина, по которой эксперты по машинному обучению слывут сверхумниками, заключается в том, что в мире их намного меньше, чем надо, даже по меркам компьютерных наук. Тим О'Райли, гуру в области технологий, утверждает, что «специалист по обработке данных» — самая востребованная вакансия в Кремниевой долине. По оценке McKinsey Global Institute, в 2018 году в одних только Соединенных Штатах спрос на экспертов по машинному обучению будет превышать предложение на 140–190 тысяч человек. Кроме того, потребуется дополнительно полтора миллиона разбирающихся в данных управленцев. Поток программ, связанных с машинным обучением, оказался слишком внезапным и мощным — система образования просто не успевает за спросом, к тому же машинное обучение считается трудной специальностью, и учебники вполне могут вызвать неприятие математики. Однако сложность скорее мнимая: все важнейшие идеи машинного обучения можно выразить и без математики. Читая эту книгу, вы, может быть, даже поймаете себя на том, что изобретаете обучающиеся алгоритмы без всяких уравнений.

Промышленная революция автоматизировала ручной труд, информационная революция проделала то же с трудом умственным, а машинное обучение автоматизировало саму автоматизацию. Без него программирование стало бы узким горлом, сдерживающим прогресс. Если вы ленивый и не слишком сообразительный компьютерщик, машинное обучение для вас — идеальная специальность, потому что обучающиеся алгоритмы сделают всю работу сами, а вам достанутся только лавры. С другой стороны, обучающиеся алгоритмы могут оставить нас без работы, и поделом.

Подняв автоматизацию на невиданные высоты, революция машинного обучения вызовет огромные изменения в экономике и обществе, как в свое время интернет, персональные компьютеры,

автомобили и паровой двигатель. Одна из областей, где изменения уже очевидны, — бизнес.

Почему бизнес рад машинному обучению?

Почему Google стоит намного дороже Yahoo? Обе компании зарабатывают на показе рекламы в интернете, и у той, и у другой прекрасная посещаемость, обе проводят аукционы по продаже рекламы и используют машинное обучение, чтобы предсказать, с какой вероятностью пользователь на нее кликнет (чем выше вероятность, тем ценнее реклама). Дело, однако, в том, что обучающиеся алгоритмы у Google намного совершеннее, чем у Yahoo. Конечно, это не единственная весьма серьезная причина разницы в капитализации. Каждый предсказанный, но не сделанный клик — упущенная возможность для рекламодателя и потерянная прибыль для поисковика. Учитывая, что годовая выручка Google составляет 50 миллиардов долларов, улучшение прогнозирования всего на один процент потенциально означает еще полмиллиарда долларов в год на банковском счету. Неудивительно, что Google — большая поклонница машинного обучения, а Yahoo и другие конкуренты изо всех сил пытаются за ней угнаться.

Реклама в сети — всего лишь один из аспектов более широкого явления. На любом рынке производители и потребители перед тем, как заключить сделку, должны выйти друг на друга. До появления интернета основные препятствия между ними были физическими: книгу можно было купить только в книжном магазине поблизости, а полки там не безразмерные. Однако теперь, когда книги можно в любой момент скачать на «читалку», проблемой становится колоссальное число вариантов. Как тут искать, если на полках книжного магазина стоят миллионы томов? Это верно и для других информационных продуктов: видео, музыки, новостей, твитов, блогов, старых добрых сайтов. Это также касается продуктов и услуг, которые можно получить на расстоянии: обуви, цветов, гаджетов, гостиничных номеров, обучения, инвестиций и даже поисков работы и спутника жизни. Как найти друг друга? Это

определяющая проблема информационной эры, и машинное обучение помогает ее решить.

В процессе развития компании можно выделить три фазы. Сначала все делается вручную: владельцы семейного магазинчика знают своих клиентов лично и в соответствии с этим заказывают, выставляют и рекомендуют товары. Это мило, но не позволяет увеличить масштаб. На втором, и самом неприятном, этапе компания вырастает настолько, что возникает необходимость пользоваться компьютерами. Появляются программисты, консультанты, менеджеры баз данных, пишутся миллионы строк кода, чтобы автоматизировать все, что только можно. Компания начинает обслуживать намного больше людей, однако качество падает: решения принимаются на основе грубой демографической классификации, а компьютерные программы недостаточно эластичны, чтобы подстроиться под бесконечную изменчивость человечества.

В какой-то момент программистов и консультантов начинает просто не хватать, и компания неизбежно обращается к машинному обучению. Amazon не может изящно заложить в компьютерную программу вкусы всех своих клиентов, а Facebook не смогла бы написать программу, чтобы выбрать обновления, которые понравятся каждому из пользователей. Walmart ежедневно продает миллионы продуктов. Если бы программисты этой торговой сети попытались создать программу, способную делать миллионы выборов, они бы работали целую вечность. Вместо этого компании спускают с цепи обучающиеся алгоритмы, науськивают их на уже накопленные горы данных и дают им предсказать, чего хотят клиенты.

Алгоритмы машинного обучения пробиваются через информационные завалы и, как свахи, находят производителей и потребителей друг для друга. Если алгоритмы достаточно умны, они объединяют лучшее из двух миров: широкий выбор, низкие затраты огромной корпорации и индивидуальный подход маленькой компании. Обучающиеся алгоритмы не идеальны, и последний шаг в принятии решения все равно остается за человеком, но они разумно сужают выбор, чтобы человеку было под силу принять решение.

Сегодня очевидно, что переход от компьютеров к интернету, а затем к машинному обучению был неизбежен. Компьютеры сделали возможным интернет, тот породил поток данных и проблему безграничного выбора, а машинное обучение использует потоки данных, чтобы решить проблему безграничного выбора. Чтобы сдвинуть спрос от «одного размера на всех» до длинного, бесконечно разнообразного списка вариантов, одного интернета мало. У Netflix может быть хоть сто тысяч разных DVD-дисков, но, если клиент не знает, как найти то, что ему понравится, он будет по умолчанию выбирать хиты. И только когда Netflix обзавелся обучающимся алгоритмом, который угадывает ваши вкусы и советует музыку, длинный хвост менее популярных исполнителей «взлетел».

Когда-нибудь произойдет неизбежное: обучающиеся алгоритмы станут незаменимым посредником и в них сосредоточится власть. Алгоритмы Google во многом определяют, какую информацию вы видите, Amazon — какие продукты вы покупаете, а Match.com — с кем вы станете встречаться. Последний этап — выбрать из предложенных алгоритмом вариантов — все равно придется преодолеть вам, однако 99,9 процента отбора будет проходить без вашего участия. Успех или неудача компании станет зависеть от того, будут ли алгоритмы машинного обучения предпочитать ее продукцию. Успех экономики в целом, то есть получат ли все игроки нужные продукты по лучшей цене, будет зависеть от того, насколько хороши обучающиеся алгоритмы.

Лучший способ гарантировать, что алгоритмы машинного обучения станут отдавать предпочтение продукции вашей компании, — применять их. Победит тот, у кого лучше алгоритмы и больше данных. Здесь проявляется новый сетевой эффект: тот, у кого больше клиентов, собирает больше информации, лучше обучает модели, завоевывает новых клиентов и так далее по спирали (а с точки зрения конкурентов — по порочному кругу). Перейти с Google на Bing, может быть, даже проще, чем с Windows на Mac OS, но на практике вы этого не сделаете, потому что благодаря удачному старту и большей доле на рынке Google лучше знает, чего вы хотите, даже если непосредственно технологии у Bing не хуже. Новичкам на рынке поисковиков можно только

посочувствовать: не имея данных, они вынуждены бороться против систем, которые обучают свои алгоритмы более десятка лет.

Можно подумать, что в какой-то момент данные просто начнут повторяться, однако точки насыщения не видно, и «длинный хвост» продолжает тянуться. Вы, конечно, и сами видите: рекомендации Amazon или Netflix пока еще очень грубы, а результаты, которые выдает Google, оставляют желать много лучшего. С помощью машинного обучения можно улучшить каждое свойство продукта, каждый уголок сайта. Ссылку внизу страницы лучше сделать красной или голубой? Попробуйте оба варианта и посмотрите, какой соберет больше кликов. А еще лучше вообще не выключать обучающиеся алгоритмы и постоянно корректировать все элементы сайта.

Та же динамика наблюдается на любом рынке, где имеется много вариантов и огромный объем данных. Гонка в разгаре, и побеждает тот, кто учится быстрее. Дело не только в лучшем понимании клиента: компании могут применять машинное обучение к каждому аспекту своей деятельности при условии, что на эту тему есть данные, а источники данных — компьютеры, устройства связи и все более дешевые и вездесущие сенсоры. Сейчас любят повторять, что «данные — это новая нефть» и, как и с нефтью, переработка — большой бизнес. IBM, как и все остальные корпорации, построила свою стратегию роста на предоставлении аналитических услуг компаниям. Бизнес видит в данных стратегический ресурс: что есть у нас, но отсутствует у конкурентов? Как воспользоваться этим преимуществом? А какие данные есть у конкурентов, но нет у нас?

Как банк, не располагающий базами данных, не может тягаться с банком, их имеющим, так и компания, не применяющая машинное обучение, не сможет соперничать с теми, кто его использует. Пока в первой компании будут писать тысячи правил для прогнозирования пожеланий покупателей, алгоритмы второй компании найдут миллиарды правил, по целому набору для каждого отдельного клиента. Такая конкуренция напоминает атаку с копьями на пулеметы. Конечно, машинное обучение — крутая новая технология, но для бизнеса дело даже не в этом: ее придется применять, потому что другого выбора просто нет.

Турбоускорение для научного метода

Машинное обучение — все равно что научный метод с допингом. Оно следует той же схеме обобщения, проверки, исключения и уточнения гипотез, однако ученый может за свою жизнь придумать и протестировать несколько сотен предположений, а система машинного обучения проделает то же самое в долю секунды. Машинное обучение ставит открытия на поток, поэтому неудивительно, что в науке оно производит революцию, во многом подобную революции в бизнесе.

Чтобы развиваться, любая область науки нуждается в данных, соизмеримых по сложности с явлениями, которые она изучает. Именно поэтому физика первой пошла вперед: записей Тихо Браге о положении планет и наблюдений Галилея за маятником и наклонными плоскостями оказалось достаточно, чтобы сформулировать законы Ньютона. По той же причине молекулярная биология обогнала более старую нейробиологию: ДНК-микрочипы и высокоэффективное секвенирование дают столько данных, сколько нейробиологам и не снилось. Социальные науки находятся в этом отношении в невыгодном положении: с выборкой всего лишь в сотню человек по десятку измерений на каждого смоделировать получается лишь очень узкие явления. Но даже такие небольшие феномены не существуют в изоляции: на них влияют мириады факторов, а это значит, что ученые очень далеки от того, чтобы их понять.

Хорошая новость: сегодня даже науки, некогда оперировавшие небольшими объемами информации, получили приток данных. Вместо того чтобы платить 50 студентам, которые будут клевать носом в лаборатории психолога, можно получить сколько угодно испытуемых, дав задание краудсорсинговой площадке Amazon Mechanical Turk (к тому же выборка окажется более разнообразной). Сейчас уже не все помнят, как немногим более десятилетия назад социологи, изучавшие социальные сети, жаловались, что не могут найти такую сеть, в которой было бы больше нескольких сотен участников. Теперь в их распоряжении весь Facebook, где больше миллиарда пользователей рассказывают о своей жизни во всех подробностях — чем не прямая трансляция общественной жизни

на планете Земля? Коннектомика[10] и функциональная магнитно-резонансная томография распахнули перед нейробиологами окно, через которое прекрасно виден головной мозг. В молекулярной биологии экспоненциально растут базы данных генов и белков. Даже «старые» дисциплины, например физика и астрономия, не стоят на месте благодаря потокам данных, льющимся из ускорителей частиц и цифрового исследования неба.

Однако от больших данных нет пользы, если их нельзя превратить в знание, и в мире слишком мало ученых, чтобы справиться с этой задачей. В свое время Эдвин Хаббл[11] открывал новые галактики, скрупулезно изучая фотографические пластинки, но можно ручаться, что таким способом не получилось бы найти полмиллиарда небесных тел, которые нам подарил проект Digital Sky Survey, — это было бы подобно ручному подсчету песчинок на пляже. Конечно, можно вручную написать правила, чтобы отличить галактики от звезд и шумов (например, птиц, самолетов или пролетающего мимо Супермена), но они будут не очень точными. Поэтому в проекте SKICAT, посвященном анализу и каталогизации изображений неба, был применен обучающийся алгоритм. Получив пластинки, где объектам уже были присвоены правильные категории, он разобрался, что характеризует каждую из них, а затем применил результаты ко всем необозначенным пластинкам. Эффективность превзошла все ожидания: алгоритм сумел классифицировать объекты настолько слабые, что человек не смог бы их выявить, и таких оказалось больше всего.

Благодаря большим данным и машинному обучению можно понять намного более сложные феномены, чем до появления этих факторов. В большинстве дисциплин ученые традиционно пользовались только очень скромными моделями, например линейной регрессией, где кривая, подобранная к данным, — всегда прямая линия. К сожалению (а может, и к счастью, потому что иначе жизнь была бы очень скучной — вообще говоря, никакой жизни бы и не было), большинство феноменов в мире нелинейны, и машинное обучение открывает перед нами огромный мир нелинейных моделей: это все равно что включить свет в комнате, которую до того освещала лишь Луна.

В биологии алгоритмы машинного обучения разбираются, где в молекуле ДНК расположены гены, какие фрагменты РНК вырезают при сплайсинге[12] перед синтезом белка, как белки принимают характерную для них форму и как заболевания влияют на экспрессию разных генов. Вместо того чтобы тестировать в лаборатории тысячи новых лекарств, обучающийся алгоритм спрогнозирует, будут ли они эффективны, и допустит до этапа тестирования только самые перспективные. Алгоритмы будут отсеивать молекулы, которые, скорее всего, вызовут неприятные побочные эффекты, например рак. Это позволит избежать дорогих ошибок, к примеру, когда лекарство запрещают только после начала испытаний на человеке.

Однако самый большой вызов — это собрать всю эту информацию в единое целое. Какие факторы усугубляют риск сердечных заболеваний и как они между собой взаимодействуют? Все, что было нужно Ньютону, — это три закона движения и один гравитации, однако одиночке открыть полную модель клетки, организма и общества не под силу. По мере роста объема знаний ученые все больше специализируются на какой-то области, но никто не способен собрать все части воедино, потому что элементов просто слишком много. Они сотрудничают друг с другом, но язык — очень медленное средство общения. Ученые пытаются быть в курсе других исследований, однако объем публикаций настолько велик, что они все больше и больше отстают, и зачастую повторить эксперимент проще, чем найти статью, в которой он описан. Машинное обучение и здесь приходит на помощь: оно просеивает литературу в поисках соответствующей информации, переводит специальный язык одной дисциплины на язык другой и даже находит связи, о которых ученые и не подозревали. Машинное обучение все больше напоминает гигантский хаб[13], через который методики моделирования, изобретенные в одной области, пробиваются в другие.

Если бы не изобрели компьютеры, наука застряла бы во второй половине XX столетия. Возможно, ученые заметили бы это не сразу и работали бы над все еще возможными небольшими успехами, но потолок прогресса был бы несравнимо ниже. Аналогично без

машинного обучения многие науки в ближайшие десятилетия столкнулись бы с проблемой ослабевающей отдачи.

Чтобы увидеть будущее науки, загляните в лабораторию Манчестерского института биотехнологий, где трудится робот по имени Адам. Ему поручено определить, какие гены кодируют ферменты дрожжей. В распоряжении Адама есть модель метаболизма дрожжевой клетки и общие знания о белках и генах. Он выдвигает гипотезы, разрабатывает эксперименты для их проверки, сам проводит опыты, анализирует результаты и выдвигает новые гипотезы, пока не будет удовлетворен. Сегодня ученые все еще независимо проверяют выводы Адама, прежде чем ему поверить, но уже завтра проверкой этих гипотез займутся роботы.

Миллиард Клинтонов

На президентских выборах 2012 года судьбу Соединенных Штатов определило машинное обучение. Традиционные факторы: взгляды на экономику, харизма и так далее — у обоих кандидатов оказались очень схожи, и исход выборов должен был определиться в ключевых колеблющихся штатах. Кампания Митта Ромни шла по классической схеме: опросы, объединение избирателей в крупные категории и выбор важнейших целевых групп. Нил Ньюхауз, специалист по общественному мнению в штабе Ромни, утверждал: «Если мы сможем победить самовыдвиженцев в Огайо, то выиграем гонку». Ромни действительно победил с перевесом в семь процентов, но все равно проиграл и в штате, и на выборах.

Барак Обама назначил главным аналитиком своей кампании Раида Гани, эксперта по машинному обучению. Гани удалось провести величайшую аналитическую операцию в истории политики. Его команда свела всю информацию об избирателях в единую базу данных, дополнила ее сведениями из социальных сетей, маркетинга и других источников и приступила к прогнозированию четырех факторов для каждого отдельного избирателя: насколько вероятно, что он поддержит Обаму, придет на выборы, отзовется на напоминание это сделать и изменит мнение об этих выборах после бесед на определенные темы. На основе этих моделей каждый вечер проводилось 66 тысяч

симуляций выборов, а результаты использовались, чтобы управлять армией волонтеров: кому звонить, в какие двери стучать, что говорить.

В политике, как в бизнесе и на войне, нет ничего хуже, чем смотреть, как противник делает что-то непонятное, и не знать, как на это ответить, пока не станет слишком поздно. Именно это произошло с Ромни. В его штабе видели, что соперники покупают рекламу на конкретных каналах кабельного телевидения в конкретных городах, но почему — было неясно. «Хрустальный шар» оказался слишком мутным. В результате Обама выиграл во всех ключевых штатах за исключением Северной Каролины, причем с бо́льшим перевесом, чем предсказывали даже самые авторитетные специалисты по общественному мнению. А наиболее авторитетные специалисты (например, Нейт Сильвер[14]), в свою очередь, использовали самые сложные методики прогнозирования. Их предсказания не сбылись, потому что у них было меньше ресурсов, чем у штаба Обамы, но и они оказались намного точнее, чем традиционные эксперты, чьи предсказания были основаны на собственных знаниях и опыте.

Вы можете возразить, что выборы 2012 года были просто случайностью: в большинстве избирательных кампаний шансы кандидатов не настолько одинаковы, и машинное обучение не может быть решающим фактором. Но дело в том, что машинное обучение будет *приводить* к тому, что в будущем все больше выборов окажутся уравновешенными. В политике, как и в других областях, использование обучения похоже на гонку вооружений. В дни Карла Роува, бывшего специалиста по прямому маркетингу и добыче данных, республиканцы лидировали. К 2012 году они отстали, но теперь вновь догоняют демократов. Неизвестно, кто вырвется вперед во время следующей избирательной кампании, но обе партии станут усердно работать над победой, а значит, лучше понимать избирателей и на основе этого знания точно наносить удары и даже подбирать кандидатов. То же самое касается общей политической платформы партий во время выборов и между ними: если основанная на достоверных данных подробная модель избирателя подсказывает, что программа у партии проигрышная, ее изменят. В результате разрыв между кандидатами на выборах будет

менее значительным и устойчивым, и при прочих равных начнут побеждать кандидаты с лучшими моделями избирателей, а избиратели будут этому способствовать.

Один из величайших талантов политика — способность понимать людей, которые за него голосуют по отдельности и в небольших группах, и апеллировать к их нуждам (или делать вид). Образцовый пример из недавней истории — Билл Клинтон. Машинное обучение действует так, будто к каждому избирателю приставлен персональный, преданный ему Клинтон. Каждый из этих мини-Клинтонов и близко не сравним с настоящим, но они берут числом, ведь даже сам Билл Клинтон не может знать, о чем думает каждый американский избиратель, хотя ему бы, конечно, хотелось. Обучающиеся алгоритмы — это агитаторы высшего класса.

Конечно, политики, как и коммерческие организации, могут использовать знание, полученное благодаря машинному обучению, и во благо, и во вред, например, давать разным избирателям противоречащие друг другу обещания. Однако избиратели, средства массовой информации и организации, следящие за выборами, могут провести собственный анализ данных и указать на политиков, переходящих черту. Гонка вооружений будет происходить не только между кандидатами, но и между всеми участниками демократического процесса.

В целом это приведет к лучшему функционированию демократических институтов, потому что канал связи между избирателями и политиками очень сильно расширится. Даже в век высокоскоростного интернета объем информации, которую получают от нас наши представители, все еще ближе к XIX веку: примерно сотня бит раз в два года — столько умещается в бюллетене. К этому прибавляются опросы общественного мнения и, может быть, периодические электронные письма и встречи в городской администрации. Это практически ничто. Большие данные и машинное обучение изменят ситуацию. Учитывая, что в будущем модели избирателей станут точнее, выборные чиновники смогут хоть тысячу раз на дню узнавать, чего хотят люди, и поступать в соответствии с этими пожеланиями, не надоедая при этом настоящим, живым гражданам.

Один сигнал, если сушей, два — если по интернету

В киберпространстве алгоритмы машинного обучения крепят национальную оборону. Каждый день иностранные хакеры пытаются взломать компьютеры Пентагона, стратегических предприятий, других организаций и государственных учреждений. Их тактика постоянно меняется, поэтому меры, работавшие против вчерашних атак, сегодня уже бессильны. Вручную написанные программы для выявления и блокировки таких атак были бы очередной линией Мажино[15], и киберкоманда Пентагона это понимает. А если это атака совершенно нового типа и научиться на прошлых примерах нельзя? Для этого обучающиеся алгоритмы строят модели нормального поведения, примеров которого хватает, и отмечают аномалии. Еще они могут вызвать кавалерию — системных администраторов. Если когда-нибудь разразится кибервойна, генералами в ней будут люди, а пехотой — алгоритмы. Люди слишком медлительны, и их слишком мало, поэтому армия ботов их быстро сметет. Нам нужна собственная армия ботов, и машинное обучение для них — как Военная академия в Вест-Пойнте[16].

Кибервойна — это частный случай асимметричного конфликта, где одна из сторон не может сравниться с другой по мощи обычного вооружения, но тем не менее способна нанести противнику тяжелый урон. Небольшой отряд террористов, вооруженных канцелярскими ножами, смог обрушить башни-близнецы и убить тысячи невинных людей. Сегодня все наиболее серьезные угрозы безопасности США — асимметричные, и от них есть эффективное противоядие: информация. Если враг не сможет скрыться, он не выживет. Информации у нас предостаточно, и это хорошо, но есть и плохие новости.

Агентство национальной безопасности США печально известно своим неуемным аппетитом к данным: по некоторым оценкам, оно перехватывает более миллиарда телефонных звонков и других сообщений по всему земному шару. Не будем сейчас рассуждать об этических вопросах защиты частной жизни. Важно, что у агентства нет столько сотрудников, чтобы прослушать все эти

звонки, прочитать электронные письма и даже отследить, кто с кем разговаривает. Большинство звонков вполне безобидны, поэтому написать программу, которая выловит из этого моря несколько подозрительных, очень сложно. Когда-то для этой цели использовались ключевые слова, но этот метод легко обвести вокруг пальца: достаточно назвать теракт свадьбой, а бомбу — свадебным тортом. В XXI веке за эту работу взялось машинное обучение. Конечно, работа агентства овеяна тайной, но в выступлении перед Конгрессом его директор признал, что анализ телефонных разговоров уже предотвратил десятки террористических угроз.

Если террористы смешаются с толпой футбольных фанатов, то обучающиеся алгоритмы смогут распознать их лица. Если террористы изобретут необычные взрывные устройства, алгоритмы обнаружат их. Алгоритмы могут решать и более тонкие задачи: связывать между собой события, которые по отдельности выглядят безобидными, но вместе складываются в зловещую схему. Такой подход мог бы предотвратить теракты 11 сентября 2001 года. Есть и еще один аспект. В ответ на действия обученной программы злоумышленники будут менять поведение, чтобы обвести ее вокруг пальца, и станут выделяться на фоне обычных людей, которые ведут себя по-прежнему. Чтобы этим воспользоваться, машинное обучение нужно объединить с теорией игр. В прошлом я работал над этой темой: надо не просто уметь побеждать сегодняшнего противника, но учиться парировать действия, которые он может предпринять против твоего алгоритма. К тому же учет плюсов и минусов различных действий, который возможен благодаря теории игр, может помочь найти правильный баланс между частной жизнью и безопасностью.

Во время битвы за Британию[17] Королевские ВВС выстояли, несмотря на значительный перевес люфтваффе. Немецкие летчики недоумевали: куда бы они ни летели, их всегда поджидали британские самолеты. У Великобритании было секретное оружие: радар, который замечал самолеты противника задолго до того, как тот входил в ее воздушное пространство. Машинное обучение — как радар, который сканирует будущее. Он позволяет не просто реагировать на ходы неприятеля, а предвосхищать их и рушить его планы.

Близкий каждому пример — так называемая полицейская профилактика. Благодаря прогнозированию тенденций в преступном мире, стратегическому распределению патрулей в наиболее опасных районах города и другим мерам правоохранительные органы эффективно выполняют задачи, которые без этих технологий потребовали бы больших сил. Работа полиции — будь то выявление мошенничества, раскрытие преступных сетей или старая добрая патрульная служба — во многом схожа с асимметричными боевыми действиями, и здесь находят применение многие из соответствующих методик обучения.

Машинное обучение играет все большую роль в военном деле. Обучающиеся алгоритмы могут развеять «туман войны»: анализ изображений, полученных при рекогносцировке, обработка рапортов после боя, составление картины положения для командира. Обучение усилит интеллект боевых роботов, поможет им ориентироваться, приспосабливаться к местности, отличать вражескую технику от гражданской, правильно целиться. Робот AlphaDog, разработанный Агентством по перспективным оборонным проектам, может нести солдату снаряжение. С помощью обучающихся алгоритмов дроны смогут летать автономно. Пока они отчасти контролируются людьми, но все идет к тому, что один пилот станет управлять все бо́льшим и бо́льшим роем летательных аппаратов. В армии будущего обучающихся алгоритмов будет значительно больше, чем солдат, а это спасет множество жизней.

Куда мы идем?

Тенденции в мире технологий приходят и уходят, но в машинном обучении необычно то, что, несмотря на все трудности, оно продолжает развиваться. Первым крупным всплеском популярности стало прогнозирование взлетов и падений на рынках ценных бумаг, появившееся в конце 1980-х годов. Следующей волной стал анализ корпоративных баз данных, который начал довольно активно внедряться в середине 1990-х годов, а также такие области, как прямой маркетинг, управление работой с клиентами, оценка кредитоспособности и выявление

мошенничества. Затем пришел черед интернета и электронной коммерции, где автоматизированная персонализация быстро стала нормой. Когда лопнувший пузырь доткомов нанес удар по этому бизнесу, приобрело популярность использование машинного обучения для поиска в интернете и размещения рекламы. События 11 сентября бросили машинное обучение на передовую войны с террором. Web 2.0 принес с собой целый спектр новых применений — от анализа социальных сетей до определения, что блогеры пишут о продукции данной компании. Параллельно ученые всех мастей все чаще обращались к масштабному моделированию. В первых рядах шли молекулярные биологи и астрономы. Едва наметился кризис на рынке недвижимости, как таланты стали перетекать с Уолл-стрит в Кремниевую долину. На 2011 год пришелся пик популярности мема[18] о больших данных, и машинное обучение оказалось прямо в центре глобального экономического кризиса. Сегодня, кажется, сложно найти область приложения человеческих усилий, не затронутую машинным обучением, включая неочевидные на первый взгляд сферы, например музыку, спорт и дегустацию вин.

Это замечательный прогресс, но он лишь предвкушение того, что нас ждет в будущем. Несмотря на пользу, которую приносит нам сегодняшнее поколение обучающихся алгоритмов, их возможности довольно скромны. Когда в нашу жизнь войдут алгоритмы, пока скрытые за стенами лабораторий, замечание Билла Гейтса о том, что прорыв в машинном обучении будет стоить десяти компаний Microsoft, покажется осторожной оценкой. Если идеи, от которых у исследователей горят глаза, принесут плоды, машинное обучение станет не только новой эрой цивилизации, но и новой стадией эволюции жизни на Земле.

Почему все это возможно? Как работают обучающиеся алгоритмы? Что им пока неподвластно и как будет выглядеть следующее поколение? Как развернется революция машинного обучения? Каких возможностей и опасностей нам следует ожидать? Именно этим вопросам посвящена эта книга. Читайте дальше!

ГЛАВА 2

ВЛАСТЕЛИН АЛГОРИТМОВ

Широта применения машинного обучения поразительна, но еще больше потрясает, что *одни и те же* алгоритмы умеют делать различные вещи. Во всех других областях для решения двух разных проблем приходится писать две разные программы. Они могут частично использовать одинаковую инфраструктуру, например те же языки программирования или ту же систему баз данных, но программа, скажем, для игры в шахматы совершенно бесполезна, если задача — обработать заявления о выдаче кредитных карт. В машинном обучении одни и те же алгоритмы могут делать и то и другое при условии, что вы дадите им соответствующие данные, на которых можно учиться. По сути, за огромным большинством приложений машинного обучения стоят всего несколько алгоритмов, с которыми мы познакомимся в следующих главах.

Посмотрите, например, на наивный байесовский классификатор — обучающийся алгоритм, который можно выразить в виде короткого уравнения. Если взять базу данных из историй болезни — симптомы, результаты анализов, наличие или отсутствие сопутствующих заболеваний, — этот алгоритм может научиться диагностировать болезнь в долю секунды, и часто лучше, чем врачи, которые много лет провели в медицинском институте. Он может победить и медицинские экспертные системы, на создание которых ушли тысячи человеко-часов. При этом тот же самый алгоритм широко используется для фильтрации спама, хотя на первый взгляд у спам-фильтров нет ничего общего с медицинской диагностикой. Другой простой обучающийся алгоритм, так называемый метод ближайших соседей, используют для массы задач — от распознавания почерка до управления манипуляторами в робототехнике и отбора книг и фильмов, которые могут понравиться клиенту. А обучающиеся алгоритмы дерева решений[19] одинаково искусно определят, можно ли выдать вам кредитную

карточку, найдут границы сплайсинга в ДНК и выберут следующий ход в шахматной партии.

Одни и те же обучающиеся алгоритмы не только способны выполнять бесконечно разнообразные задачи. По сравнению с алгоритмами, на смену которым они приходят, алгоритмы машинного обучения потрясающе просты. Большинство из них можно выразить в нескольких сотнях строк кода или, может быть, нескольких тысячах, если добавить много «примочек». В то же время программы, которые они вытесняют, иногда занимают сотни тысяч или даже миллионы строк кода, а ведь один обучающийся алгоритм способен породить неограниченное количество различных программ.

Если столь малый набор обучающихся алгоритмов может так много, возникает логичный вопрос: реально ли, чтобы один такой алгоритм делал вообще все? Другими словами, сможет ли единственный алгоритм научиться всему, что можно узнать из данных? Эта проблема — очень крепкий орешек, ведь сюда входит все, что знает взрослый человек, все, что создала эволюция, весь комплекс научных знаний. По правде говоря, все важнейшие алгоритмы машинного обучения, включая метод ближайших соседей, дерево принятия решений и байесовские сети (обобщение наивного байесовского классификатора), универсальны, то есть, если дать им достаточно соответствующих данных, они смогут аппроксимировать любую функцию сколь угодно точно: на языке математики это значит «научиться чему угодно». Ловушка в том, что «достаточно данных» может означать «бесконечный объем данных». Для обучения на основе конечных данных нужны допущения, и, как мы увидим, разные обучающиеся алгоритмы делают их по-разному, поэтому хорошо подходят для решения одних задач и не очень — для других.

А если не оставлять эти допущения внутри алгоритма, а делать их явными входными данными, наряду с собственно данными, и предоставлять пользователю право выбора, какие из них подключать и, возможно, даже задавать новые? Есть ли алгоритм, который может взять любые данные и предположения и на выходе дать скрытые в них знания? Я думаю, такой алгоритм существует. Конечно, нужно как-то ограничить эти допущения, иначе можно

обмануть самого себя, дав алгоритму все искомое знание или что-то схожее в виде допущений. Однако есть много способов этого избежать — от ограничения объема вводных до требования, чтобы исходные допущения не были больше, чем допущения текущего алгоритма.

В таком случае вопрос сводится к следующему: насколько слабыми могут быть допущения, чтобы все еще позволять получать из конечных данных все полезное знание? Обратите внимание на слово «полезное»: нас интересует только знание о нашем мире, а не о несуществующих мирах, поэтому изобретение универсального обучающегося алгоритма сводится к открытию глубочайших закономерностей нашей Вселенной, общих для всех явлений, а затем — к нахождению эффективного с точки зрения вычислений способа соединить их с данными. Как мы увидим, требование вычислительной эффективности не позволяет использовать в качестве таких закономерностей законы физики, однако оно не подразумевает, что универсальный алгоритм машинного обучения должен быть столь же эффективным, как более специализированные. Как часто бывает в информатике, мы готовы пожертвовать эффективностью ради универсальности.

Это также касается количества данных, необходимого, чтобы получить искомое знание: универсальному обучающемуся алгоритму в целом потребуется больше данных, чем специализированному, однако это не беда, при условии, что эти данные есть в нашем распоряжении, а чем больше становится общий объем данных, тем больше вероятность, что их для наших целей окажется достаточно.

Итак, вот центральная гипотеза этой книги:

> Все знание — прошлое, настоящее и будущее — можно извлечь из данных с помощью одного универсального обучающегося алгоритма.

Я называю этот алгоритм Верховным. Если его создание оказалось бы возможным, это стало бы одним из величайших научных достижений за всю историю человечества. Более того, Верховный алгоритм — последнее, что нам придется изобрести,

потому что, как только мы «спустим его с цепи», он сам изобретет вообще все, что только можно придумать. Все, что нам нужно, — дать ему достаточно подходящих данных, и он откроет соответствующее знание. Дайте ему видеопоток, и он научится видеть. Дайте библиотеку — и он научится читать. Дайте результаты физических экспериментов, и он сформулирует законы физики. Дайте данные кристаллографии ДНК, и он откроет структуру этой молекулы.

Наверное, это звучит неправдоподобно. Разве может один алгоритм получить так много разных знаний, причем таких сложных? Но на самом деле на существование Верховного алгоритма указывает много свидетельств. Давайте с ними познакомимся.

Аргумент из области нейробиологии

В апреле 2000 года группа нейробиологов из Массачусетского технологического института сообщила в журнале Nature о результатах удивительного эксперимента: они изменили мозг хорька, перенаправив нервы из глаз в слуховую кору (часть мозга, отвечающую за обработку звуков), а из ушей — в зрительную. Казалось бы, в результате этих манипуляций хорек должен был стать тяжелым инвалидом, но этого не произошло: слуховая кора научилась видеть, зрительная — слышать. У нормальных млекопитающих в зрительной коре есть карта сетчатки: нейроны, соединенные с близлежащими областями сетчатки, в коре расположены по соседству. У подопытных хорьков такая же карта сетчатки сформировалась в слуховой коре. Если зрительные данные направить в соматосенсорную кору, отвечающую за осязание, научится видеть и она. Такая способность есть и у других млекопитающих.

У слепых от рождения зрительная кора может брать на себя другие функции головного мозга. У глухих то же самое делает слуховая кора. Слепые могут научиться «видеть» с помощью языка, если прикрепить к нему электроды и направить по ним зрительные образы от прикрепленной к голове камеры. Высокое напряжение будет соответствовать ярким пикселям, низкое — темным. Слепой

ребенок по имени Бен Андервуд научился ориентироваться в пространстве с помощью эхолокации, как летучие мыши. Щелкая языком и слушая эхо, он мог ходить, не натыкаясь на препятствия, ездить на скейтборде и даже играть в баскетбол. Все это доказывает, что головной мозг везде использует один и тот же алгоритм обучения и области, выделенные для различных чувств, отличаются лишь поступающими в них входными данными (например, от глаз, ушей, носа). В свою очередь, ассоциативные зоны выполняют свои функции, потому что связаны с многочисленными сенсорными областями, а «исполнительные» свои — потому что соединяют ассоциативные зоны с двигательными нервами.

Изучение коры головного мозга под микроскопом подтверждает этот вывод. Везде повторяется тот же рисунок соединений: шестислойные колонны, петли обратной связи, ведущие в другую структуру мозга — зрительный бугор, а также повторяющиеся короткие тормозящие пути и более длинные возбуждающие. Имеется некоторое количество вариантов этой схемы, но они скорее похожи на разные параметры, настройки одного и того же алгоритма, чем на разные алгоритмы. Сенсорные зоны низкого уровня отличаются сильнее, но, как показали описанные выше эксперименты, и эти отличия не критичны. Мозжечок, самая древняя эволюционно часть головного мозга, которая отвечает за общую координацию движений, явно имеет другую, очень регулярную архитектуру, основанную на намного меньших нейронах, поэтому может показаться, что по крайней мере обучение движениям происходит по другим алгоритмам. Однако если у человека поврежден мозжечок, кора головного мозга берет на себя его функцию, то есть, вероятно, эволюция сохранила мозжечок не потому, что он делает то, чего не умеет кора, а просто потому, что так эффективнее.

Вычисления, происходящие внутри головного мозга, тоже единообразны. Вся информация представлена в виде электрических импульсов между определенными нейронами. Одинаков и механизм обучения: воспоминания формируются путем биохимического укрепления соединений между действующими вместе нейронами — так называемой долговременной потенциации. Все это верно и для животных: хотя мозг человека

необычно велик, принципы его строения, по-видимому, те же самые.

Еще одна линия аргументов в пользу единообразия коры головного мозга — это, так сказать, бедность генома. Количество соединений в мозге человека более чем в миллион раз превышает количество «букв» в геноме, поэтому геном физически не в состоянии подробно закодировать строение мозга.

Однако самый важный аргумент в пользу того, что мозг — это Верховный алгоритм, заключается в том, что он отвечает за все, что мы способны воспринять и представить. Мы не узнаем о существовании того или иного явления, если мозг не сможет его постичь: либо просто не заметим, либо посчитаем случайностью. Так или иначе, если «встроить» головной мозг в компьютер в виде алгоритма, он сможет узнать все, что можем узнать мы, поэтому один из подходов к разработке Верховного алгоритма — и, пожалуй, самый популярный — обратный инжиниринг головного мозга. Джефф Хокинс затронул его в своей книге On Intelligence[20]. С ним Рэймонд Курцвейл[21] связывает свои надежды на сингулярность — появление искусственного интеллекта, который значительно превосходит человеческий. И даже пробует силы в этом подходе в своей книге How to Create a Mind[22]. Тем не менее, как мы увидим, это лишь один из нескольких возможных подходов, причем не обязательно самый многообещающий, потому что головной мозг невероятно сложен, а мы все еще находимся на очень ранних стадиях его расшифровки. С другой стороны, если мы не сможем отыскать Верховный алгоритм, никакой сингулярности в обозримом будущем не предвидится.

С теорией единого строения коры согласны не все нейробиологи, и для прояснения этого вопроса потребуются дальнейшие исследования — ведь жаркие дебаты вызывает даже вопрос, где пределы способностей мозга. Но если есть то, что знаем мы, а мозг не может узнать, это должна была узнать эволюция.

Аргумент из области эволюции

Нескончаемое разнообразие форм жизни на Земле — результат действия единого механизма: естественного отбора. Что еще

примечательнее, информатикам хорошо знаком механизм такого типа: это итеративный поиск, при котором проблему решают путем перебора множества кандидатов, выбора и модификации лучших и повторения этих шагов столько раз, сколько необходимо. Эволюция *тоже* алгоритм. Перефразируя Чарльза Бэббиджа, пионера вычислительных машин, жившего в Викторианскую эпоху, Бог создал не виды, а алгоритм создания видов. «Бесконечное число самых прекрасных и самых изумительных форм», о котором Дарвин пишет в заключении к «Происхождению видов», скрывает от нас самое прекрасное — единство. Все эти формы закодированы в цепочках ДНК, и все они возникают путем модификации и сочетания этих цепочек. Кто бы подумал, что такой алгоритм способен породить нас с вами? Если механизмы эволюции оказались способны создать человека, они, видимо, смогут узнать все, что только можно узнать, если ввести их в достаточно мощный компьютер. И действительно, эволюционное программирование, основанное на симуляции естественного отбора, — популярная отрасль машинного обучения. Таким образом, эволюция — еще одна многообещающая тропинка, которая может привести нас к Верховному алгоритму.

Эволюция — это высший пример того, на что способен единый алгоритм обучения, если дать ему достаточно данных. Входные данные для эволюции — это опыт и судьба всех когда-либо существовавших живых существ (вот это правда *большие* данные). Но с другой стороны, более трех миллиардов лет на самом большом компьютере на нашей планете — самой планете Земля — работает эволюция. Поэтому хотелось бы, чтобы ее компьютерная копия была быстрее и требовала меньше данных, чем оригинал. Какая модель лучше подходит для Верховного алгоритма: эволюция или мозг? Это похоже на старый спор о «наследственности или воспитании», и, как человека формирует и то и другое, возможно, истинный Верховный алгоритм будет содержать оба элемента.

Аргумент из области физики

В вышедшем в 1959 году знаменитом эссе физик и нобелевский лауреат Юджин Вигнер восхищался «необъяснимой

эффективностью математики в естественных науках». Каким чудом законы, выведенные на основе немногочисленных наблюдений, применимы далеко за их пределами? И почему законы на много порядков точнее, чем данные, на которых они основаны? А самое главное, почему простой, абстрактный язык математики может так точно описывать столь многое в нашем бесконечно сложном мире? Вигнер считал это глубокой тайной, в равной степени радостной и непостижимой. Тем не менее все так и есть, и Верховный алгоритм — логическое продолжение этого феномена.

Если бы мир был просто цветущим и жужжащим хаосом, у нас был бы повод усомниться в существовании универсального обучающегося алгоритма. Однако если все вокруг нас — это следствие нескольких простых законов, вполне может оказаться, что единственный алгоритм может путем индукции сделать все возможные выводы. Все, что ему для этого потребуется, — срезать путь к следствиям законов, заменив невероятно длинные математические выкладки намного более короткими и основанными непосредственно на наблюдениях.

Например, мы полагаем, что законы физики породили эволюцию, но не знаем, как именно. Вместо поиска связывающей их цепочки следствий вывод о естественном отборе можно сделать непосредственно на основе наблюдений, как и поступил Дарвин. На основе тех же наблюдений можно было бы прийти к бесчисленному множеству неверных умозаключений, но большинство из них никогда не придут нам в голову, потому что на наши выводы влияют обширные познания о мире, и полученное знание согласуется с законами природы.

В какой мере характер физических законов распространяется на более высокие области знания, например биологию и социологию, нам еще предстоит узнать, но исследования хаоса дают много завораживающих примеров схожего поведения в очень разных системах, и теория универсальности это объясняет. Красивый пример того, как очень простая процедура итерации может породить неистощимое разнообразие форм, — множество Мандельброта[23]. Если горы, реки, облака и деревья — результат аналогичных процессов, а фрактальная геометрия показывает, что так оно и есть, возможно, эти процессы — просто разная

параметризация одной-единственной процедуры, которую мы можем вывести на их основе.

В физике те же уравнения, примененные к разным параметрам, часто описывают явления в совершенно разных областях, например квантовой механике, электромагнетизме и динамике жидкостей. Волновое уравнение, уравнение диффузии, уравнение Пуассона: если открыть что-то в одной отрасли, будет проще обнаружить аналоги в других, а если научиться решать одно из уравнений, это даст решение для всех сразу. Более того, эти уравнения довольно простые, и в них учитываются те же несколько производных параметров в отношении пространства и времени. Довольно вероятно, что они частные случаи некоего более общего уравнения, и все, что нужно сделать Верховному алгоритму, — выяснить, как конкретизировать его для частных наборов данных.

Еще одну линию доказательств можно найти в оптимизации — математической дисциплине, занимающейся нахождением аргумента, который дает максимальное значение функции. Например, поиск последовательности биржевых сделок, максимизирующей ваш совокупный доход, — это задача по оптимизации. В оптимизации простые функции часто дают удивительно сложные решения. Оптимизация играет выдающуюся роль практически во всех областях науки, технологии и бизнеса, включая машинное обучение. Каждая область оптимизируется в рамках, очерченных оптимизациями в других областях. Мы пытаемся максимизировать наше счастье в рамках экономических ограничений, которые, в свою очередь, становятся лучшими решениями для компаний в пределах доступных технологий, а те представляют собой лучшие решения, которые мы можем найти в рамках биологических и физических ограничений. Биология — результат оптимизации, произведенной эволюцией в рамках ограничений физики и химии, а сами законы физики — те же решения проблем оптимизации. Наверное, все, что существует, — это прогрессирующее решение всеобщей проблемы оптимизации, и Верховный алгоритм следует из формулировки этой проблемы.

Физики и математики — не единственные, кто находит неожиданные связи между разными областями. В своей книге

Consilience («Непротиворечивость») видный биолог Эдвард Уилсон страстно отстаивает единство всего знания — от точных наук до гуманитарных дисциплин. Верховный алгоритм — высочайшее выражение этого единства: если знание объединено общей схемой, значит, Верховный алгоритм существует, и наоборот.

Тем не менее простота физики уникальна. За пределами физики и инженерии достижения математики не так бесспорны: иногда она представляет собой единственный разумный и эффективный путь, а иногда математические модели слишком грубы, чтобы быть полезными. Тенденция к излишнему упрощению вытекает, однако, из ограничений человеческого разума, а не только из ограничений математики как таковой. Жесткий (вернее, студенистый) диск в голове человека в основном занят восприятием и движениями, и для упражнений в математике нам приходится заимствовать области, предназначенные эволюцией для языка. У компьютеров таких ограничений нет, и они могут с легкостью превращать большие объемы данных в очень сложные модели. Машинное обучение — это то, что получается, когда необъяснимая эффективность математики сливается с необъяснимой эффективностью данных. Биология и социология никогда не будут такими простыми, как физика, однако метод, благодаря которому мы откроем их истины, может оказаться несложным.

Аргумент из области статистики

Согласно одной из школ статистики, в основе всего обучения лежит одна простая формула, а именно теорема Байеса, которая определяет, как корректировать предположения при появлении новых доказательств. Байесовский алгоритм начинает с набора гипотез о мире. Когда он видит новые данные, гипотезы, согласующиеся с ними, становятся более вероятными, а те, что с ним не согласуются, — менее вероятными (или даже невозможными). После того как было рассмотрено достаточно данных, начинает доминировать одна или несколько гипотез. Например, я ищу программу, которая точно предсказывает движение курсов акций, и, если акции, которым программа-кандидат предсказывала падение, пойдут вверх, эта программа

потеряет доверие. После того как я рассмотрю некоторое число кандидатов, останутся лишь некоторые достоверные, и они будут воплощать мои знания о рынке акций.

Теорема Байеса — это машина, которая превращает данные в знания. Ее сторонники полагают, что это вообще *единственно* верный способ превращать данные в знания. Если они правы, Верховным алгоритмом будет либо сама теорема Байеса, либо он будет на ней основан. У других специалистов по статистике имеются серьезные сомнения в отношении того, как пользуются теоремой Байеса, и они предпочитают другие способы обучения на основе данных. До появления компьютеров теорему Байеса можно было применять только к очень простым проблемам, и предположение, что она может быть универсальным алгоритмом машинного обучения, казалось весьма натянутым. Однако при большом объеме данных и высокой эффективности вычислений теорема Байеса может найти применение в обширных областях гипотез и распространиться на все области знания, какие только можно себе представить. Если у байесовского обучения и есть какие-то границы, пока они неизвестны.

Аргумент из области информатики

На старших курсах колледжа я любил поиграть в тетрис. Игра очень затягивала: сверху падали разные фигуры, и их нужно было уместить как можно плотнее. Когда гора блоков достигала верхней границы экрана, игра заканчивалась. Тогда я и не подозревал, что это было мое введение в самую важную в теоретической информатике NP-полную задачу[24]. Оказывается, овладеть тетрисом — *по-настоящему* его постичь — не пустяковое дело, а одна из самых полезных вещей, которую только можно сделать. Справившись с задачей тетриса, можно одним ударом решить тысячи сложнейших, невероятно важных проблем науки, технологии и менеджмента. Дело в том, что по сути они *одна и та же* проблема, и это один из самых захватывающих фактов во всей науке.

Как белки принимают характерную для них форму? Как воссоздавать историю эволюции видов по их ДНК? Как доказывать

теоремы с помощью пропозициональной логики? Как выявлять возможности для скупки ценных бумаг с учетом транзакционных издержек? Как определять трехмерную форму по двухмерному изображению? Сжатие данных на дисках, формирование стабильных коалиций в политике, моделирование турбулентности в сдвиговых потоках, нахождение самого безопасного портфеля инвестиций с заданной выручкой и кратчайшего пути, чтобы посетить ряд городов, оптимальное расположение элементов на микросхемах, лучшая расстановка сенсоров в экосистеме, транспортные потоки, социальное обеспечение и (самое главное) как выиграть в тетрис — все это NP-полные задачи. Если получится решить одну из них, можно будет эффективно решать все задачи класса NP. Кто бы мог предположить, что все эти проблемы, такие разные на вид, — в действительности одно и то же? Но если это так, то вполне возможно, что их все (или, точнее, все частные случаи, имеющие эффективное решение) может научиться решать один алгоритм.

P и NP (к сожалению, названия не самые очевидные) — важнейшие классы проблем в информатике. Проблема относится к группе P, если ее можно эффективно решить, а к NP — если можно эффективно проверить ее решение. Знаменитый вопрос о равенстве классов P и NP — каждая ли эффективно проверяемая проблема эффективно решаема. Благодаря NP-полноте все, что нужно для ответа на этот вопрос, — доказать, что *одна* NP-полная задача эффективно решаема (или нет). NP — не самый сложный класс проблем в информатике, но, по-видимому, самый сложный из «реалистичных»: если нельзя даже проверить решение проблемы до скончания времен, какой смысл пытаться ее решить? Люди хорошо научились приблизительно решать NP-задачи, и, наоборот, проблемы, которые нам кажутся интересными (тетрис, например), имеют в себе что-то от NP-класса. Согласно одному из определений искусственного интеллекта, он заключается в нахождении эвристических решений для NP-полных задач. Часто мы решаем такие задачи, редуцируя их до выполнимости. Классическая NP-полная задача звучит так: может ли данная логическая формула в принципе быть истинной или она противоречит самой себе? Если бы мы изобрели обучающийся алгоритм, способный научиться

решать проблему выполнимости, он стал бы хорошим кандидатом на звание Верховного.

Но и без NP-полных задач само наличие компьютеров — серьезнейший признак существования Верховного алгоритма. Если бы вы отправились в начало XX века и рассказали, что вскоре будет изобретена машина, которая сможет решать проблемы во всех сферах человеческой деятельности — *одна и та же* машина для *всех* проблем, — никто бы не поверил. Вам бы объяснили, что машины могут делать что-то одно: сеялки не печатают, а пишущие машинки не сеют. Затем, в 1936 году, Алан Тьюринг[25] придумал любопытное устройство с лентой и головкой, которая читает и пишет символы. Сегодня оно известно как машина Тьюринга. С ее помощью может быть решена каждая проблема, какую только можно решить с помощью логической дедукции. Более того, так называемая универсальная машина Тьюринга может симулировать любую другую, прочтя с ленты ее спецификацию, — другими словами, ее можно запрограммировать делать что угодно.

Верховный алгоритм предназначен для индукции, то есть процесса обучения, точно так же как машина Тьюринга для дедукции. Он может научиться симулировать любые другие алгоритмы путем чтения примеров их поведения на входе и выходе. Равно как многие модели вычислений эквивалентны машине Тьюринга, вероятно, существует много эквивалентных формулировок универсального обучающегося алгоритма. Суть в том, чтобы найти первую такую формулировку, как Тьюринг в свое время нашел первый вариант многоцелевого компьютера.

Алгоритмы машинного обучения против инженерии знаний

Конечно, к Верховному алгоритму скептически относятся столько же людей, сколько испытывают по поводу его существования энтузиазм. Сомнения — это естественно, особенно когда речь идет о своего рода «серебряной пуле». Самое решительное сопротивление оказывает вековечный враг машинного обучения — инженерия знаний. Ее адепты считают, что знание нельзя получить автоматически: его должны вложить в компьютер эксперты.

Конечно, обучающиеся алгоритмы тоже могут извлечь кое-что из данных, но это никоим образом не *настоящее* знание. Для инженеров знаний большие данные — не золотая жила, а обманка.

На заре искусственного интеллекта машинное обучение представлялось очевидным путем к компьютерам с разумом, подобным человеческому. Тьюринг и другие ученые думали, что это *единственный* приемлемый путь. Однако затем инженеры знаний нанесли ответный удар, и к 1970 году машинное обучение было жестко оттеснено на второй план. В какой-то момент, в 1980-х годах, казалось, что инженерия знаний вот-вот завоюет мир, а компании и целые государства вкладывали в нее огромные инвестиции. Но вскоре пришло разочарование, и машинное обучение начало свой неумолимый рост — сначала тихо, а потом на гребне растущего вала данных.

Тем не менее все успехи машинного обучения не убедили инженеров знаний. Они уверены, что вскоре ограничения этого подхода станут очевидными и маятник качнется в другую сторону. Эту точку зрения разделяет Марвин Минский, профессор Массачусетского технологического института и пионер в области искусственного интеллекта. Минский не просто скептически относится к машинному обучению как альтернативе инженерии знаний: он *вообще* не верит, что в науке об искусственном интеллекте можно что-то объединить. Теория интеллекта по Минскому изложена в его книге The Society of Mind («Общество разума»), где он замечает, что «разум — это одна вещь за другой и ничего больше». Вся книга — длинный перечень, сотни отдельных идей, к каждой из которых дается краткое описание. Проблема такого подхода к искусственному интеллекту — в том, что он не работает. Это как коллекционирование марок компьютером. Без машинного обучения количество идей, необходимых, чтобы построить интеллектуальный агент, бесконечно. Если у робота будут все человеческие умения, кроме способности учиться, человек вскоре оставит его позади.

Минский яро поддерживал проект «Сайк»[26], самый известный провал в истории искусственного интеллекта. Целью «Сайка» было создание искусственного интеллекта путем ввода в компьютер всего необходимого знания. Когда в 1980-х годах проект стартовал,

его руководитель Дуглас Ленат уверенно предрекал успех в течение десяти лет. Три десятилетия спустя «Сайк» продолжает расти, а здравый смысл и рассуждения все еще от него ускользают. По иронии, Ленат запоздало согласился заполнять «Сайк» данными, полученными из интернета, но не потому, что «Сайк» научился читать, а потому, что другого выхода не было.

Даже если каким-то чудом удастся закодировать все необходимое, проблемы только начнутся. Многие годы множество исследовательских групп пытались построить полные интеллектуальные агенты, складывая алгоритмы зрения, распознавания речи, понимания языка, рассуждения, планирования, навигации, манипуляций и так далее. Но без объединяющих рамок все эти попытки вскоре наталкивались на непреодолимую стену сложности: слишком много движущихся элементов, слишком много взаимодействий, слишком много ошибок, а разработчики программного обеспечения — всего лишь люди и не могут со всем этим совладать. Инженеры знаний убеждены, что искусственный интеллект — очередная инженерная проблема, однако человечество пока еще не достигло точки, в которой инженерия поможет нам дойти до финишной черты. В 1962 году, когда Кеннеди произнес свою знаменитую речь в честь запуска человека на Луну, этот полет был инженерной проблемой. В 1662 году — нет. В области искусственного интеллекта мы сегодня ближе к XVII веку.

Нет никаких признаков, что инженерия знаний когда-либо будет в состоянии соревноваться с машинным обучением за пределами нескольких ниш. Зачем платить экспертам за медленное, муторное превращение знаний в понятную компьютерам форму, если компьютер сам может извлечь их из данных гораздо дешевле? А как насчет всего того, что эксперты просто не знают, но что можно открыть на основе данных? А если данные недоступны, стоимость инженерии знаний редко превышает пользу. Представьте, что фермерам приходилось бы проектировать каждый початок кукурузы, вместо того чтобы засеять семена и дать им вырасти: мы все умерли бы от голода.

Другой выдающийся ученый, не верящий в машинное обучение, — лингвист Ноам Хомский[27]. Хомский уверен, что язык

обязательно должен быть врожденным, потому что примеров грамматически правильных предложений, которые слышат дети, недостаточно, чтобы научиться грамматике. Однако это только перекладывает бремя обучения языку на эволюцию, и это аргумент не против Верховного алгоритма, а лишь против того, что он похож на головной мозг. Более того, если универсальная грамматика существует (как полагает Хомский), пролить на нее свет — значит сделать шаг к прояснению вопроса о Верховном алгоритме. Это было бы не так, лишь если бы язык не имел ничего общего с другими когнитивными способностями, но это неправдоподобно, учитывая, что в ходе эволюции он появился недавно.

В любом случае, если формализовать аргумент Хомского о «бедности стимула», мы обнаружим, что он очевидно ложен. В 1969 году Джим Хорнинг[28] доказал, что стохастические контекстно-свободные грамматики можно выучить на одних положительных примерах, а затем последовали еще более сильные результаты. (Контекстно-свободная грамматика — хлеб насущный для лингвистов, а их стохастические версии моделируют, с какой вероятностью следует использовать каждое правило.) Кроме того, обучение языку не происходит в вакууме: дети получают от родителей и среды всевозможные подсказки. То, что язык можно выучить на примерах всего за несколько лет, отчасти возможно благодаря сходству между его структурой и структурой мира. Эта общая структура — именно то, что нас интересует, и от Хорнинга и других мы знаем, что ее будет достаточно.

Если говорить более обобщенно, Хомский критически относится к статистическому обучению любого рода. У него есть список того, что не могут делать статистические обучающиеся алгоритмы, однако этот список устарел полвека назад. Хомский, по-видимому, приравнивает машинное обучение к бихевиоризму, в котором поведение животных сводится к ассоциативным реакциям на награды. Но машинное обучение не бихевиоризм. Современные алгоритмы обучения могут научиться богатым внутренним представлениям, а не только парным ассоциациям между стимулами.

В конце концов, практика — критерий истины. Статистические алгоритмы обучения языку работают, а построенные вручную

языковые системы — нет. Первое прозрение пришло в 1970-х годах, когда DARPA (Defense Advanced Research Projects Agency — Агентство передовых оборонных исследовательских проектов, научно-исследовательское крыло Пентагона) запустило первый широкомасштабный проект по распознаванию речи. Ко всеобщему удивлению, простой последовательный обучающийся алгоритм того типа, который высмеивал Хомский, ловко победил сложную систему, основанную на знаниях. Такие обучающиеся алгоритмы теперь используются практически во всех распознавателях речи, включая Siri. Фред Елинек, глава группы распознавания речи в IBM, как-то пошутил: «Всякий раз, когда я увольняю лингвиста, программа начинает работать эффективнее». Увязнув в трясине инженерии знаний, специалисты по компьютерной лингвистике чуть не вымерли в конце 1980-х годов. С тех пор в этой области безраздельно господствуют методы, основанные на машинном обучении: на конференциях по компьютерной лингвистике сложно найти доклад, в котором бы не было чего-нибудь на эту тему. Парсеры статистики анализируют язык с точностью, близкой к человеческой, оставляя далеко позади написанные вручную программы. Машинный перевод, исправление орфографии, определение частей речи, разрешение лексической многозначности, ответы на вопросы, диалоги, подведение итогов — все лучшие системы в этих областях используют машинное обучение. Watson — компьютер, выигравший в Jeopardy! — своим появлением обязан именно ему.

На это Хомский мог бы ответить, что инженерные успехи еще не доказательство научной обоснованности. Однако если ваши дома разваливаются, а двигатели не работают, видимо, с вашей физической теорией что-то не так. Хомский полагает, что лингвисты должны сосредоточиться на «идеальных», по его собственному определению, носителях языка, и это дает ему право игнорировать необходимость в статистике при обучении языку. Неудивительно, что лишь немногие экспериментаторы теперь принимают его теории всерьез.

Еще один потенциальный источник возражений против Верховного алгоритма — это мнение, популяризированное психологом Джерри Фодором[29]: разум состоит из набора модулей,

взаимодействие между которыми ограничено. Например, когда вы смотрите телевизор, ваш «высокоуровневый мозг» понимает, что это всего лишь световые вспышки на плоской поверхности, однако система восприятия зрения по-прежнему видит трехмерные формы. Но даже если сознание модулярно, это еще не значит, что в разных модулях используются разные алгоритмы обучения. Может быть, для работы, скажем, со зрительной и вербальной информацией достаточно одного алгоритма.

Критики вроде Минского, Хомского и Фодора когда-то торжествовали, но их влияние испарилось. Это хорошо, но тем не менее нельзя забывать об их аргументах, когда будем прокладывать путь к Верховному алгоритму. На то есть две причины. Первая — инженеры знаний сталкивались со многими проблемами, стоящими перед машинным обучением, и даже если они не преуспели в их решении, то извлекли много ценных уроков. Вторая — машинное обучение и инженерия знаний, как мы вскоре выясним, переплетены неожиданными и хитроумными связями. К сожалению, оба лагеря часто не слышат друг друга и говорят на разных языках: специалисты по машинному обучению мыслят в категориях вероятностей, а инженеры знаний — в категориях логики. Ниже мы посмотрим, что с этим сделать.

Лебедь кусает робота

«Как бы ни был умен алгоритм, всегда есть то, что он не может узнать». Это утверждение в разных формулировках — самое частое возражение против машинного обучения за пределами науки об искусственном интеллекте и когнитивистики. Нассим Талеб[30] изо всех сил напирал на него в своей книге The Black Swan: The Impact of the Highly Improbable[31]. Некоторые события просто непредсказуемы: если человек видел только белых лебедей, он будет считать, что вероятность когда-нибудь встретить черного равна нулю. Финансовый крах 2008 года оказался как раз таким «черным лебедем».

Действительно, некоторые вещи можно предсказать, а некоторые нельзя, и отличать одно от другого — первейшая задача алгоритма машинного обучения. Однако цель Верховного

алгоритма — узнать все, что *можно* узнать, и этих знаний намного больше, чем может себе представить Талеб и не только он. Спад жилищного рынка совсем не был черным лебедем: его многократно предсказывали. Большинство банковских моделей не смогли его предвидеть исключительно из-за их довольно очевидных ограничений, а не в силу ограниченности машинного обучения как такового. Обучающиеся алгоритмы вполне способны точно предсказать редкие, никогда до этого не происходившие события: можно даже сказать, что в этом весь их смысл. Какова вероятность существования черного лебедя, если его никогда не видели? А как насчет доли известных науке видов, которые, как оказалось, имеют черных представителей? Это очень грубый пример — в этой книге мы увидим гораздо более глубокие.

Еще одно схожее и часто повторяемое возражение: «Данные не могут заменить человеческой интуиции». На самом деле это человеческая интуиция не может заменить данных. К интуиции мы прибегаем, когда не знаем фактов, а поскольку фактов часто не хватает, интуицией люди очень дорожат. Но если перед вами доказательства, разве вы станете их отрицать? Статистический анализ побеждает искателей талантов в бейсболе (это замечательно описано в книге Майкла Льюиса MoneyBall[32]), он превосходит знатоков в дегустации вин, и каждый день мы видим все новые примеры его способностей. Вследствие наплыва данных граница между доказательствами и интуицией очень быстро смещается, и, как при любой революции, въевшиеся привычки надо преодолеть. Если я эксперт по теме *X* в компании *Y*, мне, конечно не понравится, когда меня обойдет какой-то парень с данными. Есть профессиональная поговорка: «Слушай своих клиентов, а не HiPPO[33]». HiPPO — это «мнение самого высокооплачиваемого человека». Если вы хотите быть авторитетом и завтра, пользуйтесь данными, а не боритесь с ними.

«Ладно, — скажет кто-то. — Машинное обучение может находить статистические закономерности в данных, но оно никогда не откроет ничего серьезного, например законов Ньютона». Возможно, пока не откроет, но ручаюсь, в будущем все изменится. Если не брать истории про падающие яблоки, глубокие научные истины найти совсем не легко. Наука в своем развитии проходит

через три этапа, которые можно назвать фазами Браге, Кеплера и Ньютона. В фазе Браге мы собираем много данных, как Тихо Браге, который ночь за ночью, год за годом кропотливо записывал положение планет. В фазе Кеплера мы подбираем к данным эмпирические законы: Кеплер это делал с движением планет. В фазе Ньютона мы открываем глубокие истины. Наука в значительной степени состоит из работы, подобной труду Браге и Кеплера, а ньютоновские проблески — редкость. Сегодня большие данные делают работу миллиардов Браге, а машинное обучение трудится, как миллионы Кеплеров. Если — будем надеяться — человечество еще ждут великие озарения, их с равной вероятностью могут породить и обучающиеся алгоритмы, и еще более занятые ученые будущего, и совместные усилия ученых и алгоритмов. (Конечно, Нобелевскую премию получат ученые, независимо от того, предложили они ключевые идеи или просто нажали на кнопку. У алгоритмов машинного обучения нет никаких амбиций.) В этой книге мы увидим, на что могут быть похожи эти алгоритмы, и порассуждаем о том, что они могут открыть — например, лекарство от рака.

Верховный алгоритм — лиса или еж?

Нам надо рассмотреть еще одно потенциальное возражение против Верховного алгоритма. Наверное, самое серьезное. Его выдвигают не инженеры знаний и не рассерженные эксперты, а сами практики машинного обучения. На секунду поставив себя на их место, я мог бы сказать: «Послушайте, Верховный алгоритм совершенно не похож на мою повседневную работу! Я перепробовал сотни алгоритмов для каждой проблемы, и для разных задач лучше подходят разные алгоритмы. Разве может один заменить все это многообразие?»

На это я отвечу: вы правы. Но разве не лучше вместо сотен вариантов многих алгоритмов пробовать сотни вариантов одного-единственного? Если выяснить, что в каждом алгоритме важно, а что нет, найти у важных элементов общее и посмотреть, как они дополняют друг друга, можно сложить из них Верховный алгоритм. Именно этим мы и займемся на страницах этой книги или хотя бы

попытаемся как можно ближе к этому подойти. Наверное, у вас, дорогой читатель, по мере чтения тоже возникнут какие-то идеи на этот счет.

Насколько сложен будет Верховный алгоритм? Тысячи строк кода? Миллионы? Мы пока не знаем, но в машинном обучении бывало, что простые алгоритмы чудесным образом побеждали очень замысловатые. В известном эпизоде книги The Sciences of the Artificial[34] пионер искусственного интеллекта и нобелевский лауреат Герберт Саймон просит представить себе муравья, который упорно бежит по пляжу к себе домой. Путь муравьишки сложен не потому, что сложен он сам, а потому что вокруг полно маленьких дюн, на которые надо взбираться, и гальки, которую приходится обегать. Попытки смоделировать муравья, запрограммировав все возможные пути, будут обречены на провал. Аналогично самое сложное в машинном обучении — это данные. Все, что должен сделать Верховный алгоритм, — усвоить их, поэтому не надо удивляться, если сам он окажется несложным. Человеческая рука проста: четыре пальца вместе плюс отведенный в сторону большой. И несмотря на это, рука может делать и использовать бесконечное разнообразие инструментов. Верховный алгоритм по отношению к алгоритмам — то же, что рука по отношению к карандашам, мечам, отверткам и вилкам.

Как заметил Исайя Берлин[35], некоторые мыслители подобны лисам и знают много разного, а некоторые — ежам, которые знают что-то одно, но важное. То же самое с обучающимися алгоритмами. Я надеюсь, что Верховный алгоритм окажется ежом, но, даже если это лиса, ее все равно надо поскорее поймать. Самая большая проблема сегодняшних обучающихся алгоритмов не в том, что их много, а в том, что они, хоть и полезны, не делают всего, что мы от них хотим. И прежде чем начать открывать глубокие истины при помощи машинного обучения, надо как следует разобраться в самом машинном обучении.

Что на кону?

Предположим, человеку поставили диагноз «рак» и традиционные методы лечения — хирургия, химио- и лучевая терапия —

не принесли желаемого эффекта. Дальнейший ход лечения станет для него вопросом жизни и смерти. Первый шаг — это секвенировать геном опухоли. Есть компании, например Foundation Medicine в Кембридже, которые этим занимаются: отправьте им образец опухоли, и они пришлют вам список мутаций в ее геноме, достоверно связанных с раком. Без этого не обойтись, потому что каждая раковая опухоль индивидуальна и нет лекарства, которое поможет во всех случаях. Распространяясь по организму человека, рак мутирует, и вследствие естественного отбора, скорее всего, будут выживать и размножаться клетки, наиболее стойкие к назначенным лекарствам. Возможно, нужный препарат помогает только пяти процентам пациентов, или необходимо сочетание лекарств, которое пока вообще не применяли. Может быть, придется разработать совершенно новое лекарство конкретно для данного случая или комплекс препаратов, чтобы подавить способность опухоли к адаптации. С другой стороны, у лекарств могут иметься побочные эффекты, смертельно опасные для данного пациента, но безвредные для большинства других людей. Ни один врач не может уследить за всей информацией, необходимой для выработки оптимальной терапии с учетом истории болезни и генома опухоли. Это идеальная работа для машинного обучения, и тем не менее на сегодняшний день обучающиеся алгоритмы не могут с ней справиться. У каждого из них есть какие-то из необходимых способностей, но не хватает других. У Верховного алгоритма будет все. Если применить его к большому объему данных о пациентах и лекарствах, а также информации, почерпнутой из литературы по биологии и медицине, мы сможем победить рак.

Универсальный алгоритм машинного обучения остро необходим во многих других областях и ситуациях — от невероятно важных до самых обыденных. Представьте себе, например, идеальную рекомендующую систему, которая посоветует именно те книги, фильмы и гаджеты, которые вы сами бы выбрали, будь у вас время проверить все варианты. Алгоритм Amazon очень далек от идеала. Отчасти дело в том, что у него просто недостаточно данных: в целом он знает только, какие предметы вы раньше покупали на этом сайте. Но если разойтись и предоставить ему полный

доступ к потоку сознания человека начиная с рождения, он не будет знать, что с этим делать. Как преобразовать в связную картину мириады решений, калейдоскоп жизни? Как понять, кто этот человек и чего он хочет? Это выходит далеко за пределы кругозора сегодняшних обучающихся алгоритмов, но, если дать все эти данные Верховному алгоритму, он поймет вас примерно так же, как лучший друг.

В один прекрасный день в каждом доме появится робот. Он будет мыть посуду, заправлять кровать, даже присматривать за детьми, пока родители на работе. Как скоро это произойдет — зависит от того, как тяжело окажется отыскать Верховный алгоритм. Если лучшее, на что мы способны, — соединить много разных алгоритмов, каждый из которых решает лишь малую долю проблем искусственного интеллекта, вскоре мы наткнемся на стену сложности. Такой фрагментарный подход сработал в Jeopardy!, но лишь немногие верят, что домашние роботы будущего будут внуками компьютера Watson, победителя этой игры. Дело не в том, что Верховный алгоритм одной левой решит проблему искусственного интеллекта: нам по-прежнему понадобятся чудеса инженерии, и Watson в этом отношении — хороший пример. Однако здесь действует правило 80/20: Верховный алгоритм даст 80 процентов решения, и останется приложить 20 процентов труда, поэтому, несомненно, с него и надо начинать.

Влияние Верховного алгоритма на технологию не ограничится искусственным интеллектом. Универсальный обучающийся алгоритм — невероятно мощное оружие против Монстра Сложности. Нам поддадутся системы, которые сегодня слишком трудно построить. Компьютеры начнут делать больше и требовать меньше помощи с нашей стороны. Они не станут снова и снова повторять те же ошибки, а будут учиться на практике, как люди. Иногда, как старые дворецкие, они даже смогут угадывать, чего вы хотите, еще до того, как вы это выразите. Если компьютеры делают нас умнее, компьютеры с установленным Верховным алгоритмом заставят нас почувствовать себя настоящими гениями. Технологический прогресс заметно ускорится, причем не только в компьютерных науках, но и во многих других областях. Это, в свою очередь, будет способствовать экономическому росту

и уменьшит нищету. С Верховным алгоритмом, помогающим синтезировать и распределять знания, интеллект организаций будет больше, а не меньше суммы интеллектов их подразделений. Типовые задачи станут автоматизированы, а люди найдут себе занятия поинтереснее. Все виды деятельности будут выполняться качественнее, чем сейчас: лучше обученными людьми, компьютерами или и теми и другими. Падения на рынках ценных бумаг будут происходить реже и без тяжелых последствий. Благодаря сети сенсоров, которые опутают нашу планету, и обученным моделям, которые станут моментально обрабатывать их данные, прогресс больше не будет идти вперед на ощупь: здоровье планеты пойдет на поправку. Модели начнут договариваться с миром от вашего имени, играя в замысловатые игры с моделями людей и организаций. А в результате всех этих улучшений мы окажемся счастливее, продуктивнее и долговечнее.

Поскольку потенциальная отдача так велика, нам стоит попробовать изобрести Верховный алгоритм, даже если шансы на успех невысоки. И даже если это займет много времени, поиски могут принести нам непосредственную пользу. Например, мы будем гораздо лучше понимать машинное обучение благодаря единому подходу к этой проблеме. Сегодня очень много деловых решений принимается на основе слабого понимания аналитики, но все может быть иначе. Чтобы пользоваться технологиями, не обязательно разбираться в механизмах их действия, однако нужно иметь хорошую концептуальную модель: это примерно то же, что уметь настроиться на радиостанцию и регулировать громкость. Сегодня люди, которые не занимаются машинным обучением, не имеют даже общего представления о том, что делают обучающиеся алгоритмы. Алгоритмы, которыми мы управляем, пользуясь Google, Facebook или современными аналитическими пакетами, немного похожи на загадочный черный лимузин с тонированными стеклами, который однажды вечером подъезжает к нашей двери. Стоит ли в него садиться? Куда он нас повезет? Настало время занять место водителя. Знание допущений, которые делают разные алгоритмы машинного обучения, поможет подобрать правильные инструменты для решения конкретной задачи, а не хвататься за первые попавшиеся и потом годами

с ними мучиться, болезненно пытаясь открыть то, что надо было знать с самого начала. Понимая, что именно оптимизирует обучающийся алгоритм, можно гарантировать, что он будет оптимизировать важные вещи, а не что попадется под руку. Наверное, самое главное вот что: если знать, как именно пришел к выводам данный обучающийся алгоритм, легче понять, что делать с полученной информацией — чему верить, от чего отказываться, как получить в следующий раз лучший результат. А с универсальным обучающимся алгоритмом, который мы разработаем в этой книге в виде концептуальной модели, все это можно будет сделать без лишнего напряжения. Машинное обучение в своей основе — простая вещь. Надо всего лишь снять один за другим слои математики и научного жаргона и добраться до самой маленькой матрешки.

Все эти преимущества относятся и к личной, и к профессиональной жизни. Как лучше воспользоваться цепочкой данных, которые оставляет каждый наш шаг в современном мире? Любой поступок действует сразу на двух уровнях: дает нам непосредственный результат и учит систему, с которой мы взаимодействовали. Осознание этого — первый шаг к счастливой жизни в XXI веке. Научите алгоритмы, и они будут служить вам, но вначале их надо понять. Что в вашей работе можно сделать с помощью алгоритма, а что нет? И — самое важное — как воспользоваться машинным обучением, чтобы делать это еще лучше? Компьютер — инструмент, а не противник. Вооруженный машинным обучением менеджер становится сверхменеджером, ученый — сверхученым, инженер — сверхинженером. Будущее принадлежит тем, кто глубоко понимает, как сочетать свои уникальные знания и навыки с тем, что алгоритмы делают лучше всего.

Но, может быть, Верховный алгоритм — это ящик Пандоры, который лучше не открывать? Не поработят ли нас компьютеры и не захотят ли от нас избавиться? Не станет ли машинное обучение прислуживать тиранам и зловещим корпорациям? Благодаря пониманию, в каком направлении развивается машинное обучение, мы сможем разобраться, о чем надо волноваться, а о чем не стоит и как поступать в таких случаях. С теми видами обучающихся

алгоритмов, которые мы встретим в этой книге, сценарий «Терминатора», где искусственный сверхинтеллект обретает разум и покоряет человечество с помощью армии роботов, просто невозможен. Если компьютеры умеют учиться, это еще не значит, что они волшебным образом обретут собственную волю. Обучающиеся алгоритмы учатся достигать целей, которые ставим им мы сами, и не могут эти цели менять. Скорее, нам надо позаботиться о том, чтобы они не оказали нам медвежью услугу, а для этого их надо лучше учить.

Прежде всего нам надо подумать, что будет, если Верховный алгоритм попадет в плохие руки. Первая линия защиты — позаботиться, чтобы хорошие ребята получили его раньше остальных, а если непонятно, кто хороший, а кто нет, обеспечить к нему открытый доступ. Вторая линия — осознать, что, как бы ни был совершенен обучающийся алгоритм, он хорош ровно настолько, насколько хороши предоставляемые ему данные. Тот, кто контролирует данные, контролирует и алгоритм. Реакцией на «датификацию» жизни должен стать не уход в джунгли — даже в лесу будет полно сенсоров, — а скорее активное стремление держать под контролем чувствительные для вас данные. Хорошо иметь советчиков, которые найдут и принесут вам то, что вы пожелаете. Без них можно потеряться. Однако они должны приносить вам то, что хотите *вы сами*, а не то, чем хочет снабдить вас кто-то посторонний. Вокруг контроля над данными и владения моделями, обучающимися на их основе, в XXI веке будет сломано немало копий: за них станут сражаться правительства, корпорации, организации и отдельные лица. Но с другой стороны, у вас будет и этическая обязанность делиться информацией ради общего блага. Машинное обучение само по себе не вылечит рак, поэтому больные раком люди принесут пользу будущим пациентам, поделившись информацией о себе.

Другая теория всего

Наука сегодня очень напоминает Балканский полуостров — настоящую Вавилонскую башню, где каждое сообщество говорит на собственном языке и способно видеть только несколько соседних

мини-сообществ. Верховный алгоритм станет единым взглядом на науку в целом и даже, не исключено, приведет к созданию новой теории всего. Это может показаться странным заявлением — ведь машинное обучение просто строит теории на основе данных. Каким образом сам Верховный алгоритм может вырасти в теорию? Разве теория всего — это не теория струн[36]? Верховный алгоритм совершенно на нее не похож!

Для ответа на эти вопросы сначала надо разобраться, что такое научная теория. Теория — это не полное описание мира, а набор ограничений в отношении того, каким он может быть. Чтобы получить полное описание, теорию нужно объединить с данными. Возьмем, например, второй закон Ньютона. Он гласит, что сила равна массе, умноженной на ускорение, то есть $F = ma$. Он не указывает, какова масса или ускорение какого-либо тела или каковы действующие на него силы, а только требует, чтобы в случае, когда масса объекта m, а его ускорение — a, равнодействующая сила составляла ma. Этот закон убирает некоторые степени свободы Вселенной, но не все. То же верно для любой другой физической теории, включая относительность, квантовую механику и теорию струн, которые, в сущности, уточнения законов Ньютона.

Сила теорий в том, что они значительно упрощают описание мира. Если мы вооружены законами Ньютона, достаточно знать только массу, положение и скорости всех предметов в определенный момент времени, чтобы вывести их положения и скорости во все другие моменты. Таким образом, законы Ньютона уменьшают наше описание мира на порядок, равный числу различимых случаев в истории Вселенной в прошлом и будущем. Поразительно! Конечно, законы Ньютона — лишь приближение истинных законов физики, поэтому давайте вместо них возьмем теорию струн, игнорируя все ее проблемы и вопрос, можно ли ее вообще когда-нибудь проверить эмпирически. Разве можно достичь большего? Да, можно. По двум причинам.

Первая заключается в том, что в реальности у нас никогда не будет достаточно данных, чтобы полностью описать наш мир. Даже игнорируя принцип неопределенности, точно знать положение и скорости всех частиц в мире в какой-то момент времени совершенно невозможно. А поскольку законы физики

хаотичны, неопределенность со временем только накапливается, и очень скоро они определяют очень немного. Для точного описания мира нужны регулярные порции свежих данных. Это приводит к тому, что законы физики говорят нам только о локальных событиях, а это резко уменьшает их мощь.

Вторая проблема в следующем: даже если бы мы получили всю полноту знаний о мире в какой-то момент, законы физики по-прежнему не позволяли бы нам узнать его прошлое и будущее. Дело в том, что объем вычислений, необходимых для такого рода предсказаний, превышает способности любого компьютера, какой только можно себе представить, и для идеальной симуляции Вселенной потребовалась бы еще одна идентичная вселенная. Вот почему теория струн за пределами физики в основном неприменима, а теории биологии, психологии, социологии и экономики не выводятся из законов физики: их приходится создавать с нуля. Мы допускаем, что они приближение того, что предсказали бы законы физики в масштабе клеток, головного мозга и общества, но знать этого не можем.

В отличие от локальных теорий, которые имеют силу только в конкретных дисциплинах, Верховный алгоритм властен везде. В области *X* у него будет меньше возможностей, чем у превалирующей в ней теории, но в масштабе всей науки — когда мы рассматриваем мир в целом — он намного сильнее, чем любая другая теория. Верховный алгоритм — это зародыш всех теорий. Все, что нам нужно, чтобы получить теорию *X*, — это минимальное количество данных, необходимое для ее выведения путем индукции. (В случае физики это просто результаты, наверное, нескольких сотен ключевых экспериментов.) Достоинство Верховного алгоритма в том, что он вполне может оказаться лучшей отправной точкой для поиска теории всего, какую мы только можем получить. При всем уважении к Стивену Хокингу[37], Верховный алгоритм может в конце концов рассказать нам о Божественном замысле больше, чем теория струн.

Некоторые могут возразить, что поиски универсального обучающегося алгоритма — типичный пример научной гордыни. Но мечты не гордыня. Может быть, Верховный алгоритм займет свое место среди великих химер, рядом с философским камнем

и вечным двигателем. А может быть, его поиск больше похож на попытки определить долготу в океане: такие расчеты долго считались слишком сложными, от них все отмахивались, а потом пришел одинокий гений и решил проблему. Скорее всего, создание Верховного алгоритма потребует усилий нескольких поколений, и величественный собор будет строиться камень за камнем. Единственный способ проверить — однажды утром встать пораньше и отправиться в путь.

Кандидаты, которые не оправдали надежд

Итак, если Верховный алгоритм существует, на что он похож? На первый взгляд, очевидный ответ — на запоминание. Просто запоминай все, что видишь, и через некоторое время увидишь все, что только можно увидеть, и таким образом узнаешь все, что только можно узнать. Проблема в том, что, как сказал Гераклит, в ту же реку нельзя войти дважды. В мире куда больше вещей, чем мы в состоянии увидеть. Неважно, сколько снежинок вы исследуете: следующая будет другой. Даже если бы вы присутствовали при Большом взрыве и после этого везде и всюду, вы все равно увидели бы лишь крохотную долю того, что могли бы увидеть в будущем. Если бы вы десять тысяч лет наблюдали за жизнью на Земле, это не подготовило бы вас к тому, что еще предстоит. Человек, выросший в одном городе, не впадает в ступор, когда переезжает в другой, однако робот, способный только запоминать, впал бы. Кроме того, знание — это не просто длинный список фактов. Знание бывает обобщенным и структурированным. «Все люди смертны» — намного более емкое утверждение, чем семь миллиардов свидетельств о смерти, по одному на каждого человека. Запоминание же не даст нам ни обобщенности, ни структуры.

Другой кандидат в Верховные алгоритмы — микропроцессор. В принципе, процессор в вашем компьютере можно рассматривать как единый алгоритм, работа которого — выполнять другие алгоритмы, подобно универсальной машине Тьюринга, и он может выполнять любые мыслимые алгоритмы до границ своей памяти и производительности. Для микропроцессора алгоритм — просто еще один вид данных. Проблема в том, что сам по себе

микропроцессор ничего делать не умеет: он просто сидит весь день без дела. Откуда берутся алгоритмы, которые он выполняет? Если они были закодированы программистом-человеком, никакого обучения нет. Тем не менее в каком-то отношении микропроцессор — удачный аналог Верховного алгоритма. Микропроцессор — не самое оптимальное оборудование для запуска отдельных алгоритмов, для этого гораздо больше подходят разработанные для конкретной задачи интегральные схемы специального назначения (application-specific integrated circuit, ASIC). Однако почти для всех приложений мы используем именно микропроцессоры, потому что их гибкость с лихвой компенсирует относительную неэффективность. Если бы нам приходилось разрабатывать ASIC для каждого нового приложения, информационная революция никогда бы не состоялась. Верховный алгоритм — тоже не лучший алгоритм для изучения конкретного элемента знаний. Эффективнее был бы алгоритм, в который уже заложена бо́льшая часть этого знания (или знание целиком: тогда данные будут избыточны). Однако вся суть в том, чтобы вывести знание из данных путем индукции. Это легче и дешевле, поэтому чем более обобщен алгоритм машинного обучения, тем лучше.

Еще более радикальный кандидат — скромный вентиль ИЛИ-НЕ, логический переключатель, который на выходе дает единицу, если на входе два нуля. Не забывайте, что все компьютеры построены из логических вентилей в виде транзисторов и все вычисления можно свести к комбинациям элементов И, ИЛИ и НЕ. Вентиль ИЛИ-НЕ — это просто элемент ИЛИ, за которым следует элемент НЕ: отрицание дизъюнкции[38], как в предложении «Я счастлив, если не голоден и не болен». Элементы И, ИЛИ и НЕ можно реализовать с использованием вентилей ИЛИ-НЕ, поэтому этот вентиль может делать все. Вообще говоря, некоторые микропроцессоры только его и используют. Так почему же он не может стать Верховным алгоритмом? Ведь он, безусловно, непревзойден в своей простоте. К сожалению, вентиль ИЛИ-НЕ — Верховный алгоритм не в большей степени, чем кубик лего — универсальная игрушка. Конечно, детали конструктора как кирпичики и из них многое можно построить, но гора элементов самопроизвольно ни во что не сложится. То же

относится к другим простым вычислительным схемам, например сетям Петри[39] и клеточным автоматам[40].

Перейдем к более сложным кандидатам. Например, к запросам, на которые может ответить любой хороший движок базы данных, или простых алгоритмов в статистическом пакете. Разве их недостаточно? Это более крупные детали лего, но по-прежнему всего лишь кирпичики. Движок базы данных никогда не откроет ничего нового: он просто сообщает то, что знает. Даже если все люди в базе данных смертны, ему не придет в голову экстраполировать эту черту на других людей (проектировщики баз данных побледнели бы от самой этой мысли). Статистика в основном заключается в проверке гипотез, которые кто-то сначала должен сформулировать. Статистические пакеты умеют выполнять линейную регрессию и другие простые процедуры, но они мало чему могут научиться, сколько бы данных им ни предоставили. Качественные пакеты входят в серую зону между статистикой и машинным обучением, но все равно остается множество видов знания, которое они не могут открыть.

Ладно, давайте начистоту. Верховный алгоритм — это уравнение $U(X) = 0$. Он уместится не то что на футболке, а даже на почтовой марке! Уравнение $U(X) = 0$ говорит, что определенная (возможно, очень сложная) функция U какой-то (возможно, очень сложной) переменной X равна нулю. К этой форме можно свести любое уравнение. Например, $F = ma$ можно записать в виде $F - ma = 0$, поэтому, если считать $F - ma$ функцией U переменной F — вуаля: $U(F) = 0$. В целом X может быть любыми вводными данными, а U — любым алгоритмом, поэтому, конечно, Верховный алгоритм не может быть более общим, чем это уравнение, а поскольку мы ищем самый общий алгоритм из всех возможных, это должен быть он. Конечно, я шучу, но конкретно этот неудачный кандидат указывает на одну из реальных опасностей в машинном обучении: создание настолько общего обучающегося алгоритма, что он окажется недостаточно содержательным, чтобы быть полезным.

Так какое же минимальное содержание может иметь обучающийся алгоритм, чтобы оставаться полезным? Законы физики? В конце концов, все в этом мире им подчиняется (по крайней мере, мы так думаем), они породили эволюцию, а в ходе

эволюции — головной мозг. Может быть, Верховный алгоритм и правда скрыт в законах физики, но, если это так, нам надо выразить его явно. Если просто подбрасывать законам физики данные, новых законов не получишь. На это можно посмотреть следующим образом: возможно, основная теория какой-то дисциплины — просто законы физики, облеченные в удобную для этой дисциплины форму. Но если это действительно так, нам нужен алгоритм, который найдет кратчайший путь из данных этой дисциплины к ее теории, и непонятно, смогут ли законы физики в этом помочь. Еще один аспект заключается в следующем: если бы законы физики были иными, Верховный алгоритм все равно во многих случаях смог бы их открыть. Математики любят говорить, что Бог может нарушать законы физики, но даже он не бросает вызов законам логики. Возможно, это так, но законы логики предназначены для дедукции, а нам нужно что-то подобное для индукции.

Пять «племен» машинного обучения

Конечно, охоту за Верховным алгоритмом не надо начинать с нуля. У нас за плечами несколько десятилетий исследований машинного обучения, на которые можно опереться. Лучшие умы планеты посвятили свои жизни разработке обучающихся алгоритмов, а кто-то даже утверждает, что универсальный алгоритм уже у него в руках. Хотя мы стоим на плечах гигантов, такие заявления надо принимать с долей скептицизма, и тогда возникает вопрос: как понять, что Верховный алгоритм найден? Мы поймем это тогда, когда один и тот же обучающийся алгоритм, в котором допустимо только менять параметры, на основе минимальных исходных данных сможет научиться понимать видео и текст так же хорошо, как человек, сделает важные открытия в биологии, социологии и других науках. Очевидно, что по этим стандартам ни один алгоритм машинного обучения пока нельзя признать Верховным, даже в том маловероятном случае, что он уже найден.

Крайне важно понимать, что от Верховного алгоритма не требуется уметь с чистого листа решать новую задачу. Это, наверное, была бы слишком высокая планка для *любого*

обучающегося алгоритма, и это, разумеется, совершенно не похоже на то, как работают сами люди. Например, язык не существует в вакууме, и мы не поймем фразу без знания мира, к которому она относится. Таким образом, когда Верховный алгоритм будет учиться читать, он может опираться на то, что до этого он уже научился видеть, слышать и управлять роботами. Точно так же ученый не просто вслепую подбирает модели к данным — чтобы решить проблему, он оперирует всеми знаниями в данной области. Делая открытия в биологии, Верховный алгоритм тоже сначала может прочитать всю литературу по предмету, какую пожелает, полагаясь на уже освоенный навык чтения. Верховный алгоритм — не просто пассивный потребитель данных. Он может взаимодействовать с окружающей средой и активно искать данные, которые ему нужны, как робот-ученый Адам, о котором мы упоминали выше, или просто ребенок, исследующий окружающий мир.

Поиски Верховного алгоритма сложны, но их оживляет соперничество разных научных школ, действующих в области машинного обучения. Важнейшие из них — символисты, коннекционисты, эволюционисты, байесовцы и аналогисты. У каждого «племени» есть набор фундаментальных постулатов и конкретная проблема, которая больше всего его волнует. «Племя» находит решение для этой проблемы в идеях союзных научных дисциплин, и у него есть верховный алгоритм, который воплощает это решение.

Для символистов интеллект сводится к манипулированию символами — так математики решают уравнения, заменяя одни выражения другими. Символисты понимают, что нельзя учиться с нуля: данные должны сопровождаться исходными знаниями. Они научились встраивать уже имеющееся знание в машинное обучение и на лету соединять фрагменты знания, чтобы решать новые задачи. Их верховный алгоритм — это обратная дедукция: она определяет недостающее для дедукции знание, а затем как можно в большей степени его обобщает.

Для коннекционистов обучение — то, чем занимается головной мозг, и поэтому они считают, что этот орган надо воспроизвести путем обратной инженерии. Мозг учится, корректируя силу

соединений между нейронами, поэтому ключевая проблема — понять, какие соединения за какие ошибки отвечают, и соответствующим образом их изменить. Верховный алгоритм коннекционистов — метод обратного распространения ошибки, который сравнивает выходные данные системы с желаемыми, а потом последовательно, слой за слоем, меняет соединения между нейронами, чтобы сделать результат ближе к тому, что требуется.

Эволюционисты верят, что мать учения — естественный отбор. Если он создал нас самих, значит, он может все, и нам остается только симулировать его на компьютере. Ключевая проблема, которую решает эта научная школа, — обучающаяся структура: требуется не просто подобрать параметры, как при обратном распространении ошибки, а создать мозг, который эти уточнения будет тонко настраивать. Верховный алгоритм эволюционистов — это генетическое программирование, соединяющее и развивающее компьютерные программы точно так же, как природа сводит и развивает живые организмы.

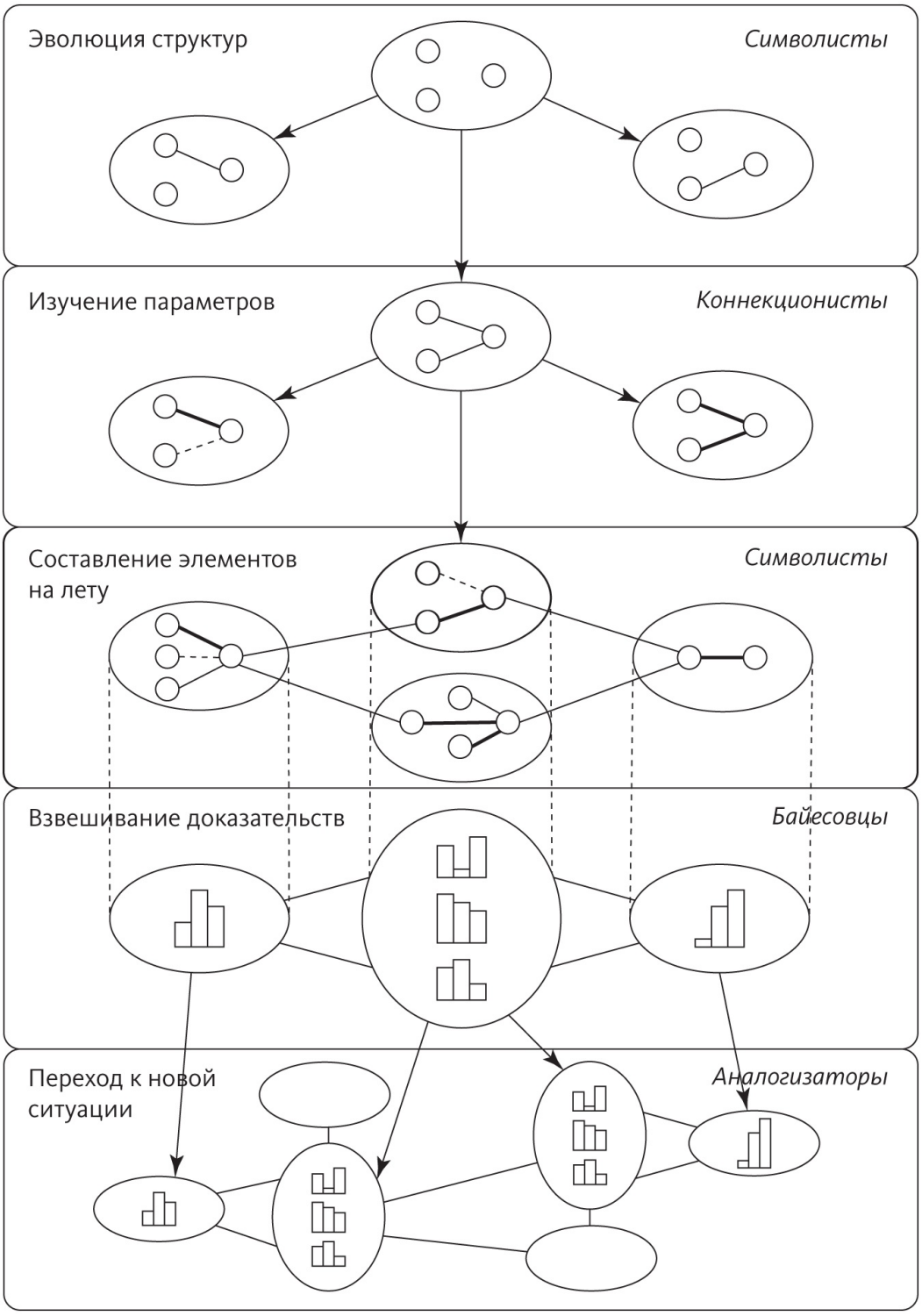
Байесовцы озабочены прежде всего неопределенностью. Все усвоенное знание неопределенно, и само обучение — это форма недостоверного вывода. Проблема, таким образом, заключается в следующем: как работать с зашумленной, неполной, даже противоречивой информацией и не потерять голову? Выходом становится вероятностный вывод, а верховным алгоритмом — теорема Байеса и ее производные. Теорема Байеса объясняет, как встраивать в наши убеждения новые доказательства, а алгоритмы вероятностного вывода делают это с максимальной эффективностью.

Для аналогистов ключ к обучению — находить сходства между разными ситуациями и тем самым логически выводить другие сходства. Если у двух пациентов схожие симптомы, вероятно, у них одинаковое заболевание. Ключевая проблема — оценить, насколько похожи два случая. Верховный алгоритм аналогистов — это метод опорных векторов, который определяет, какой опыт надо запомнить и как соединить опыт, чтобы делать новые прогнозы.

Решения центральных проблем каждого из «племен» — блестящие, с трудом завоеванные успехи. Но подлинный Верховный алгоритм должен решить все пять проблем, а не одну. Например,

чтобы лечить рак, необходимо разобраться в метаболических путях в клетке: каковы механизмы регуляции генов, за какие химические реакции отвечают кодируемые этими генами белки, как добавление новой молекулы повлияет на эти цепочки реакций. Было бы глупо пытаться узнать все это с нуля, игнорируя знания, которые биологи накапливали десятилетиями. Символисты знают, как объединить эти знания с данными секвенсоров ДНК, микрочипами экспрессии генов и так далее и получить результаты, которые нельзя иметь из каждого источника по отдельности. Однако знание, полученное путем обратной дедукции, — качественное, а не количественное, поэтому нам придется не просто разобраться, что с чем взаимодействует, но и понять степень этого взаимодействия, и здесь может пригодиться метод обратного распространения ошибки. Тем не менее и обратная дедукция, и обратное распространение будут подвешены в пространстве без какой-то базовой структуры, к которой можно привязать найденные ими взаимодействия и параметры. Такую структуру может открыть генетическое программирование. С этого момента, если у нас будет полное знание метаболизма и все данные пациента, можно попробовать найти лечение. Однако в реальности имеющаяся информация всегда неполная, а местами и ошибочная. Приходится прокладывать путь к цели, несмотря на эти препятствия, и именно для этого пригодится вероятностное заключение. В самых сложных случаях рак у пациента будет очень сильно отличаться от того, что было описано, и полученное знание не поможет. Спасти положение смогут алгоритмы, основанные на сходстве: они заметят аналогии между очень разными на первый взгляд ситуациями, потому что сосредоточатся на существенных моментах и проигнорируют остальное.

В этой книге мы создадим единый алгоритм со всеми этими возможностями.



В наших поисках мы пройдем по землям каждого из пяти «племен». Пограничные переходы, где они встречаются, ведут переговоры и вступают в схватки, будут самыми непростыми отрезками пути. У каждого «племени» есть свой кусочек мозаики,

которую мы обязаны собрать. Специалисты по машинному обучению, как и все ученые, напоминают слепцов рядом со слоном: один щупает хобот и думает, что это змея. Другой прислонился к ноге и считает, что это дерево. Еще один потрогал бивень и решил, что это бык. Наша цель — дотронуться до всех элементов, не спеша с выводами. Прикоснувшись ко всему, мы попытаемся нарисовать в воображении слона целиком. Соединить все фрагменты в одно решение — далеко не банальная задача. Некоторые полагают, что это вообще невозможно. Но именно это мы сделаем.

Алгоритм, к которому мы придем, пока не будет Верховным (мы увидим почему), но так далеко еще никто не заходил. А в пути нас ждет столько сокровищ, что позавидовал бы сам Крез[41]. Тем не менее эта книга — лишь первая часть саги о Верховном алгоритме. Героем второй части станете вы, дорогой читатель. Ваша миссия, если вы решитесь взять ее на себя, — пройти остаток пути и вернуться с наградой. Я буду вашим скромным проводником по первой части, отсюда и до границы известного мира. Что? Вы говорите, что знаете слишком мало и не сильны в алгоритмах? Не пугайтесь. Информатика еще молода, и здесь, в отличие от физики или биологии, вам не надо быть доктором наук, чтобы совершить в ней революцию. (Не верите — спросите Билла Гейтса, а еще Сергея Брина, Ларри Пейджа и Марка Цукерберга[42].) Важны идеи и упорство.

Итак, вы готовы? Наш путь начнется с визита к символистам, «племени» с самой солидной родословной.

ГЛАВА 3

ПРОБЛЕМА ИНДУКЦИИ ЮМА

Вы рационалист или эмпирик?

Рационалисты считают, что чувства обманчивы и единственный верный путь к знанию — логическое рассуждение. Эмпирики уверены, что рассуждения подвержены ошибкам и знание должно быть получено из наблюдений и экспериментов. Французы — рационалисты. Англосаксы (как их называют французы) — эмпирики. Мыслители, юристы и математики — рационалисты. Журналисты, врачи и ученые — эмпирики. «Она написала убийство» — рационалистический криминальный телесериал. «C.S.I.: Место преступления» — эмпирический. В мире информатики теоретики и инженеры знаний — рационалисты. Хакеры и специалисты по машинному обучению — эмпирики.

Рационалисты любят планировать все заранее, еще до того, как сделают первый шаг. Эмпирики предпочитают пробовать и смотреть, что получится. Не знаю, существует ли ген рационализма или эмпиризма, но, глядя на моих коллег-информатиков, я пришел к выводу, что это почти черты характера: некоторые рационалистичны до мозга костей и не могут быть другими, а другие — насквозь эмпирики и всегда такими были. Представители обоих полюсов могут разговаривать друг с другом и иногда пользоваться полученными другим лагерем результатами, но понимают друг друга лишь отчасти. В глубине души каждый из них верит, что то, чем занимается оппонент, — вторично и не очень интересно.

Рационалисты и эмпирики, наверное, существовали с самого зарождения Homo sapiens. Перед тем как выйти на охоту, Пещерный Бобби долго сидел у костра и размышлял, где его поджидает добыча. Тем временем Пещерная Алиса систематически прочесывала территорию. Поскольку оба вида дошли до наших дней, наверное, будет правильно сказать, что ни один подход не лучше другого. Вы можете подумать, что машинное обучение — это окончательный триумф эмпириков, но скоро мы увидим, что все не так однозначно.

«Рационализм или эмпиризм?» — любимый вопрос философов. Платон был ранним рационалистом, а Аристотель — ранним эмпириком. Но по-настоящему дебаты разгорелись в эпоху Просвещения, когда по каждую сторону встали по три великих мыслителя: Декарт, Спиноза и Лейбниц были ведущими рационалистами; Локк, Беркли и Юм — их соперниками-эмпириками. Доверяя своей силе рассуждения, рационалисты сочиняли теории Вселенной, которые, мягко говоря, не прошли проверку временем, но помимо этого они изобрели фундаментальные математические методики, например математический анализ и аналитическую геометрию. Эмпирики были гораздо практичнее, и их влияние прослеживается везде, начиная с научного метода и заканчивая Конституцией США.

Выдающимся эмпириком и величайшим англоязычным философом всех времен был Дэвид Юм. О его серьезнейшем влиянии говорили такие ученые, как Адам Смит и Чарльз Дарвин, а еще его можно назвать святым покровителем символистов. Юм родился в Шотландии в 1711 году и большую часть своей жизни провел в Эдинбурге, который в XVIII веке процветал и бурлил интеллектуальной жизнью. Юм был человеком добродушным, но при этом строгим скептиком и много времени посвящал разрушению мифов своего времени. Он довел начатые Локком рассуждения об эмпирике до логического завершения и задал вопрос, который с тех пор, как дамоклов меч, висит над любым знанием, от самого банального до самого сложного: как в принципе можно оправдать экстраполяцию того, что мы видели, на то, чего мы не видели? Каждый обучающийся алгоритм в каком-то смысле — попытка ответить на этот вопрос.

Вопрос Юма — отправная точка нашего путешествия. Начнем с того, что проиллюстрируем его примером из повседневной жизни и встретим ее современное воплощение в знаменитой теореме No free lunch — «Бесплатных обедов не бывает»[43]. Затем мы посмотрим, что отвечают Юму символисты. Это подведет нас к самой важной проблеме машинного обучения: проблеме переобучения, то есть выделения фантомных закономерностей, которых на самом деле нет. Мы посмотрим, как ее решают символисты и почему машинное обучение — сердце своего рода алхимии, философский камень

превращения данных в знания. Для символистов этот камень — само знание. В следующих четырех главах мы увидим решения алхимиков из других «племен».

Быть или не быть свиданию?

У вас есть знакомая девушка, которая вам очень нравится. Вы хотите пригласить ее на свидание, однако вам уже приходилось сталкиваться с отказами, и вы решили задать вопрос, только если будете твердо уверены, что она скажет «да». Пятничным вечером вы сидите с мобильником в руке и пытаетесь решить, звонить или не звонить. Вы помните, что в прошлый раз она не согласилась. Но почему? До этого она два раза сказала «да», потом «нет». Может быть, есть какие-то дни, когда она не хочет никуда ходить? Или, может быть, она любит клубы, а рестораны, напротив, ей не нравятся? Вы человек, необычайно любящий систему, поэтому откладываете телефон в сторону и набрасываете на листке бумаги все, что помните по прошлым встречам.

Попытка	День недели	Тип свидания	Погода	Телепрограмма	Согласилась?
1	Будни	Ужин	Тепло	Скучная	Нет
2	Выходные	Клуб	Тепло	Скучная	Да
3	Выходные	Клуб	Тепло	Скучная	Да
4	Выходные	Клуб	Холодно	Интересная	Нет
Сегодня	Выходные	Клуб	Холодно	Скучная	?

Итак, что вас ждет? Быть свиданию или не быть? Есть ли какая-то закономерность во всех этих «да» и «нет»? И самое главное — что эта схема скажет о сегодняшнем дне?

Понятно, что одного фактора для прогнозирования мало. В какие-то выходные она хотела куда-нибудь сходить, а в другие — нет. Иногда ей хотелось развлечься в клубе, а иногда не хотелось и так далее. А как насчет сочетания факторов? Может быть, она любит по выходным ходить в клуб? Нет, не то: случай номер четыре

перечеркивает эту догадку. А может быть, она любит гулять только в теплые вечера? В точку! Сработало! В таком случае, учитывая, что на улице морозец, сегодня вечером шансов маловато. Погодите! А что если она любит ходить в клуб, когда по телевизору нет ничего интересного? Это тоже обоснованное предположение, и в таком случае сегодня вас ждет «да»! Быстрее, надо позвонить ей, пока не очень поздно. Стоп. Как узнать, что эта закономерность правильная? Целых два варианта согласуются с вашим прошлым опытом, но они дают противоположные прогнозы. Подумаем еще раз: а если она ходит в клуб только в хорошую погоду? Или она выходит из дома по выходным, когда по телевизору нечего смотреть? Или…

Тут вы в отчаянии комкаете листок бумаги и швыряете его в мусорную корзину. Ничего не получается! Как быть?! Дух Юма печально кивает у вас за плечом. *У вас нет никаких оснований предпочесть одно обобщение другому.* «Да» и «нет» — одинаково допустимые ответы на вопрос «Что она скажет?». А часы тикают. С горечью вы вытаскиваете из кармана пятак и почти готовы его подбросить.

Вы не одиноки в своем затруднении — оно знакомо и нам. Мы буквально только что отправились в путь навстречу Верховному алгоритму и, похоже, уже наткнулись на непреодолимое препятствие. Есть *хоть какой-нибудь* способ научиться чему-то на прошлом опыте, чтобы с уверенностью применять знание в будущем? А если такого способа нет, не станет ли машинное обучение безнадежным предприятием? Если уж на то пошло, не построена ли вся наука или даже все человеческое знание на довольно шаткой почве?

Непохоже, чтобы проблему решало увеличение объема данных. Вы можете быть супер-Казановой и встречаться с миллионами женщин, по тысяче раз с каждой, но ваш обширный архив все равно не ответит на вопрос, что *эта* женщина ответит в *этот* раз. Даже если сегодняшний случай в точности напоминает тот, когда она сказала «да» — тот же день недели, тот же вид свидания, та же погода и те же шоу по телевизору, — это все еще не означает, что она согласится. Вполне может быть, что ее ответ определяется каким-то фактором, о котором вы не подумали или который

не можете оценить. Или, может быть, в ее ответах нет ни ладу, ни складу: они случайные, и вы просто ставите себе палки в колеса, пытаясь отыскать в них какую-то схему.

Философы спорили о проблеме индукции Юма с тех самых пор, как он ее сформулировал, но так и не пришли к удовлетворительному ответу. Бертран Рассел[44] любил иллюстрировать эту проблему историей об индюке-индуктивисте. В первое утро индюку дали корм в девять утра. Но он был хорошим индуктивистом и не спешил с выводами. Он много дней собирал наблюдения при всевозможных обстоятельствах, однако его раз за разом кормили в девять утра. Наконец он сделал вывод: да, его всегда будут кормить в девять утра. А потом наступил канун Рождества и ему перерезали горло.

Было бы очень хорошо, если бы проблема Юма была всего лишь философским ребусом, который можно и проигнорировать. Но проигнорировать проблему Юма не получится. Например, бизнес Google основан на угадывании, какие страницы вы ищете, когда вписываете в строку поиска определенные слова. Ключевое преимущество этого поисковика — огромный массив запросов, которые люди вводили в прошлом, и ссылок, на которые они кликали на соответствующих страницах результатов. Но что делать, если кто-то вписывает сочетание ключевых слов, которого нет в архивах? А даже если они и есть, разве можно с уверенностью сказать, что текущий пользователь хочет найти те же страницы, что и все его предшественники?

Как насчет того, чтобы *предположить*, что будущее будет похоже на прошлое? Это, безусловно, рискованное допущение (у индюка-индуктивиста, например, оно не сработало). С другой стороны, без него знание невозможно, да и жизнь тоже. Мы предпочитаем жить, пусть и без уверенности. К сожалению, даже с таким предположением мы по-прежнему блуждаем в тумане. Оно работает в «тривиальных» случаях: если я врач, а у пациента *B* точно такие же симптомы, как у пациента *A*, я предположу, что диагноз будет такой же. Однако если симптомы соответствуют не точно, я по-прежнему ничего не узнаю. Это проблема машинного обучения: обобщение случаев, которые *мы еще не видели*.

Но, может быть, все не так страшно? Разве с достаточным количеством данных большинство случаев не попадает в категорию «тривиальных»? Нет, не попадает. В предыдущей главе мы уже разобрались, почему запоминание не может быть универсальным обучающимся алгоритмом, но теперь давайте посмотрим на это с количественной точки зрения. Предположим, у вас есть база данных с триллионом записей по тысяче булевых полей в каждой (булево поле — это ответ на вопрос «да или нет»). Это довольно много. Какую долю возможных случаев вы увидели? (Попробуйте угадать, прежде чем читать дальше.) Итак, число возможных ответов — два на каждый вопрос, поэтому для двух вопросов это дважды два (да-да, да-нет, нет-да и нет-нет), для трех вопросов — это два в кубе ($2 \times 2 \times 2 = 2^3$), а для тысячи вопросов — это два в тысячной степени (2^{1000}). Триллион записей в нашей базе данных — это ничтожно малая доля процента от 2^{1000}, а именно «ноль, запятая, 286 нулей, единица». Итого: неважно, сколько у вас будет данных — тера-, пета-, экса-, зетта- или иоттабайты. Вы *вообще ничего* не видели. Шансы, что новый случай, который вам нужен для принятия решения, уже есть в базе данных, так исчезающе малы, что без обобщения вы даже не сдвинетесь с места.

Если все это звучит немного абстрактно, представьте, что вы крупный провайдер электронной почты и вам надо пометить каждое входящее письмо как спам или не спам. Даже если у вас есть база данных с триллионом уже помеченных писем, она вас не спасет, потому что шанс, что очередное письмо будет точной копией какого-то из предыдущих, практически равен нулю. У вас нет выбора: надо попытаться более обобщенно определить, чем спам отличается от не-спама. И, согласно Юму, сделать это никак нельзя.

Теорема «Бесплатных обедов не бывает»

Через 250 лет после того, как Юм подбросил нам свою гранату, ей придал элегантную математическую форму Дэвид Уолперт, физик, ставший специалистом по машинному обучению. Его результаты, известные как уже упомянутая выше теорема «Бесплатных обедов не бывает», ставят ограничения на то, как хорош может быть

обучающийся алгоритм. Ограничения довольно серьезные: никакой обучающийся алгоритм не может быть лучше случайного угадывания! Вот и приехали: Верховный алгоритм, оказывается, — это просто подбрасывание монетки. Но если серьезно, как может быть, что никакой обучающийся алгоритм не в состоянии победить угадывание с помощью орла или решки? И почему тогда мир полон очень успешных алгоритмов, от спам-фильтров до самоуправляющихся машин (они вот-вот появятся)?

Теорема «Бесплатных обедов не бывает» очень сильно напоминает причину, по которой в свое время Паскаль проиграл бы пари. В своей книге «Мысли», опубликованной в 1669 году, он заявил, что нам надо верить в христианского Бога, потому что, если он существует, это дарует нам вечную жизнь, а если нет — мы мало что теряем. Это был замечательно утонченный аргумент для того времени, но, как заметил на это Дидро, имам может привести точно такой же довод в пользу веры в Аллаха, а если выбрать неправильного бога, придется расплачиваться вечными муками в аду. В целом, учитывая огромное количество мыслимых богов, вы ничего не выиграете, выбрав в качестве объекта своей веры одного из них в пользу любого другого, потому что на любого бога, который говорит «делай то-то», найдется еще один, который потребует нечто противоположное. С тем же успехом можно просто забыть о богах и наслаждаться жизнью без религиозных предрассудков.

Замените «бога» на «обучающийся алгоритм», а «вечную жизнь» — на «точный прогноз», и вы получите теорему «Бесплатных обедов не бывает». Выберите себе любимый алгоритм машинного обучения (мы их много увидим в этой книге), и на каждый мир, где он справляется лучше случайного угадывания, я, адвокат дьявола, коварно создам другой мир, где он справляется ровно настолько же хуже: все, что мне надо сделать, — перевернуть ярлыки на всех случаях, которых вы *не видели*. Поскольку ярлыки на увиденных случаях совпадают, ваш обучающийся алгоритм никак не сможет различить мир и антимир, и в среднем из двух случаев он будет так же хорош, как случайное угадывание. Следовательно, если совместить все возможные миры с их антимирами, в среднем ваш обучающийся алгоритм будет равен подбрасыванию монетки.

Однако не торопитесь сдаваться и списывать со счетов машинное обучение и Верховный алгоритм. Дело в том, что нас заботят не все возможные миры, а только тот, в котором живем мы с вами. Если мы уже знаем что-то об этом мире и введем это в наш обучающийся алгоритм, у него появится преимущество перед произвольным угадыванием. На это Юм ответил бы, что знание как таковое тоже должно быть получено путем логической индукции и, следовательно, ненадежно. Это верно, даже если знание закодировано в наш мозг эволюцией. Однако нам приходится идти на этот риск. Еще можно задуматься: есть ли бесспорный, фундаментальный самородок знаний, на котором можно построить всю свою индукцию? (Что-то вроде Декартова «Я мыслю, следовательно, я существую», хотя сложно придумать, как превратить конкретно это утверждение в обучающийся алгоритм.) Я думаю, ответ — «да, есть», и мы увидим этот самородок в главе 9.

Практическое следствие теоремы «Бесплатных обедов не бывает» — то, что обучение без знаний невозможно. Одних данных недостаточно. Если начинать с чистого листа, мы придем к чистому листу. Машинное обучение — своего рода насос знаний. С помощью машинного обучения можно «выкачать» из данных много знаний, но сначала нам надо его заполнить данными, как насос перед пуском заполняют водой.

Машинное обучение с точки зрения математики относится к категории некорректно поставленных задач, так как единственного решения не существует. Вот простой пример: сумма каких двух чисел равна 1000? Если исходить из того, что числа положительные, у этой задачи 500 возможных ответов: 1 и 999, 2 и 998 и так далее. Чтобы решить некорректно поставленную задачу, придется ввести дополнительные условия. Если я скажу, что второе число в три раза больше первого, — все станет просто! Ответ — 250 и 750.

Том Митчелл, ведущий символист, называет это «тщетностью беспристрастного обучения». В обычной жизни слово «пристрастный» имеет негативный оттенок: предвзятость суждений — это плохо. Однако в машинном обучении предвзятые суждения необходимы. Без них нельзя учиться. На самом деле они незаменимы и для человеческого познания, но при этом так жестко

встроены в наш мозг, что мы принимаем их как должное. Вопросы вызывает только пристрастность, выходящая за эти рамки.

Аристотель говорил, что в разуме нет ничего такого, чего не было бы в органах чувств. Лейбниц добавил: «Кроме самого разума». Человеческий мозг — это не tabula rasa, потому что это совсем не доска: доска пассивна, на ней пишут, а мозг активно обрабатывает получаемую информацию. Доска, на которой он пишет, — это память, и она и впрямь сначала чиста. С другой стороны, компьютер — *действительно* чистая доска, до тех пор пока его не запрограммируют: активный процесс надо заложить в память, прежде чем что-нибудь произойдет. Наша цель — найти простейшую программу, какую мы только можем написать, чтобы она продолжала писать саму себя путем неограниченного чтения данных, пока не узнает все, что можно узнать.

У машинного обучения имеется неотъемлемый элемент азартной игры. В конце первого фильма про Грязного Гарри Клинт Иствуд гонится за ограбившим банк бандитом и раз за разом в него стреляет. Наконец грабитель повержен. Он лежит рядом с заряженным ружьем и не знает, хватать его или нет. Было шесть выстрелов или только пять? Гарри сочувствует (если можно так выразиться): «Тебе надо лишь спросить: “Повезет или нет?” Ну как, мерзавец?» Этот вопрос специалисты по машинному обучению должны задавать себе каждый день, когда они приходят на работу. Повезет или нет? Как и эволюция, машинное обучение не будет каждый раз попадать в десятку. Вообще говоря, ошибки — не исключение, а правило. Но это нормально, потому что промахи мы отбрасываем, а попаданиями пользуемся, и важен именно совокупный результат. Когда мы получаем новую частицу знаний, она становится основой для логической индукции еще большего знания. Единственный вопрос — с чего начать.

Подготовка насоса знаний

В «Математических началах натуральной философии» наряду с законами движения Ньютон формулирует четыре правила индукции. Они далеко не так известны, как физические законы, но,

пожалуй, не менее важны. Ключевое правило — третье, которое можно перефразировать так:

> Принцип Ньютона: то, что верно для всего, что мы видели, верно для всего во Вселенной.

Не будет преувеличением сказать, что это невинное вроде бы утверждение — сердце ньютоновской революции и современной науки. Законы Кеплера применялись ровно к шести сущностям — планетам Солнечной системы, которые в то время были известны. Законы Ньютона применимы ко всем до единой частицам материи во Вселенной. Прыжок в обобщении между этими законами просто колоссальный, и это прямое следствие сформулированного Ньютоном правила. Приведенный выше принцип сам по себе — насос знаний невероятной мощи. Без него не было бы законов природы, а только вечно неполные заплатки из небольших закономерностей.

Принцип Ньютона — первое неписаное правило машинного обучения. Путем индукции мы выводим самые широко применимые законы, какие только возможно, и сужаем их действие, только если данные вынуждают нас это сделать. На первый взгляд это может показаться чрезмерной, даже нелепой самоуверенностью, но в науке такой подход работает уже более трех сотен лет. Безусловно, можно представить вселенную настолько разнородную и капризную, что Ньютонов принцип будет систематически терпеть поражение, но наша Вселенная не такая.

Тем не менее принцип Ньютона лишь первый шаг. Нам все еще надо найти истину во всем том, что мы увидели: извлечь закономерности из сырых данных. Стандартное решение: предположить, что *форму* истины мы знаем, а работа алгоритма машинного обучения — это облечь ее в плоть. Например, в описанной выше проблеме со свиданием можно предположить, что ответ девушки будет определяться чем-то одним. В таком случае обучение заключается просто в рассмотрении всех известных факторов (день недели, тип свидания, погода, телепрограмма) и проверке, всегда ли корректно они предопределяют ответ. Сложность в том, что ни один фактор не подходит! Вы рискнули

и проиграли, поэтому немного ослабляете условия. Что если ответ девушки определяется сочетанием двух факторов? Четыре фактора по два возможных значения для каждого — это 24 варианта для проверки (шесть пар факторов, из которых можно выбирать, умноженные на два варианта для каждого значения фактора). Теперь у нас глаза разбегаются: целых четыре сочетания двух факторов корректно предсказывают результат! Что делать? Если вы чувствуете удачу, можете выбрать какой-то из них и надеяться на лучшее. Однако более разумный подход — демократический: дайте им «проголосовать» и выберите победивший прогноз.

Если все сочетания двух факторов проигрышные, можно попробовать все сочетания любого числа факторов. Специалисты по машинному обучению и психологи называют это «конъюнктивными понятиями». К таким понятиям относятся словарные определения: «У стула есть сиденье, *и* спинка, *и* некоторое число ножек». Уберите любое из этих условий, и это уже будет не стул. Конъюнктивное понятие можно найти у Толстого в первой строке «Анны Карениной»: «Все счастливые семьи похожи друг на друга, каждая несчастливая семья несчастлива по-своему». То же верно и для отдельных людей. Чтобы быть счастливым, нужны здоровье, любовь, друзья, деньги, любимая работа и так далее. Уберите что-то из этого списка, и человек будет несчастлив.

В машинном обучении примеры концепции называют положительными примерами, а контрпримеры — отрицательными. Если вы пытаетесь научиться узнавать кошек на картинке, изображения кошек будут положительными примерами, а собак — отрицательными. Если составить базу данных семей из мировой литературы, Каренины будут отрицательным примером счастливой семьи, но найдется и некоторое количество драгоценных положительных примеров.

Для машинного обучения типично начинать с ограничивающих условий и постепенно ослаблять их, если они не объясняют данных. Этот процесс обычно выполняется обучающимся алгоритмом автоматически, без какой-либо помощи со стороны человека. Сначала алгоритм тестирует все отдельные факторы, затем все сочетания двух факторов, потом все сочетания трех и так далее.

Однако здесь мы опять сталкиваемся с проблемой: конъюнктивных понятий *очень много*, а времени, чтобы все перепробовать, недостаточно.

Описанный выше случай со свиданиями несколько обманчив, потому что содержит всего четыре переменных и четыре примера. Поэтому представьте, что вы управляете сайтом знакомств и вам надо разобраться, какие пары познакомить друг с другом. Если каждый пользователь заполнит анкету из 50 вопросов с ответами «да» или «нет», у каждой потенциальной пары будет сотня атрибутов, по 50 на каждого человека. Можно ли дать конъюнктивное определение понятия «подходящая пара» на основе информации о парах, которые сходили на свидание и сообщили о результатах? Для этого пришлось бы перепробовать 3^{100} возможных определений (три варианта для каждого атрибута: «да», «нет» и «не входит в концепцию»). Даже если в вашем распоряжении самый быстрый в мире компьютер, к моменту, когда он закончит, все пары уже давно умрут, а вы разоритесь, если только вам не повезет и в точку не попадет очень короткое определение. Правил много, времени мало. Надо придумать что-то получше.

Вот один из выходов. Предположите на секунду, пусть это и невероятно, что все пары подходят друг другу. Затем попробуйте исключить все пары, которые не удовлетворяют единственному атрибуту определения. Повторите это для всех атрибутов и выберите тот, который исключит больше всего плохих пар и меньше всего хороших. Ваше определение теперь будет выглядеть, например, так: «Они подходят друг другу, только если он открытый человек». Теперь попробуйте прибавить к этому определению каждый из оставшихся атрибутов и выберите тот, что исключит больше всего оставшихся плохих пар и меньше всего хороших. Возможно, определение станет таким: «Они подходят друг другу, только если оба — открытые люди». Попробуйте прибавить к найденным чертам третий атрибут и так далее. Когда вы исключите все плохие пары — готово: у вас в руках определение концепции, которое включает все положительные примеры и исключает все отрицательные. Например: «Они хорошая пара, если оба — открытые люди, он любит собак, а она не любительница

кошек». Теперь можно выбросить все данные и оставить только это определение, потому что оно включает все, что важно для ваших задач. К этому алгоритму вы гарантированно придете в разумные сроки, а еще это первый настоящий обучающийся алгоритм, с которым мы познакомились в этой книге!

Как править миром

Тем не менее на конъюнктивных понятиях далеко не уедешь. Проблема, как выразился Редьярд Киплинг, в том, что «путей в искусстве есть семь и десять раз по шесть, и любой из них для песни — лучше всех». В реальной жизни понятия дизъюнктивны. У стульев может быть и четыре ножки, и одна, а иногда они вообще без ножек. В шахматы можно выиграть бесчисленным количеством способов. Электронные письма со словом «виагра», скорее всего, спам, но то же самое можно сказать о письмах со словом «БЕСПЛАТНО!!!». Кроме того, у всех правил бывают исключения. Некоторые неблагополучные семьи умудряются быть счастливыми. Все птицы летают, только если это не пингвины, страусы, казуары и киви (а также если у них не сломано крыло, они не сидят в клетке и так далее).

Так что нам нужно находить концепции, которые заданы целым набором правил, а не одним, например:

Если вам нравятся IV—VI эпизоды «Звездных войн», значит, вам понравится «Аватар».

Если вам нравятся «Звездный путь: Следующее поколение» и «Титаник», вам понравится «Аватар».

Если вы член экологической организации Sierra Club и любите научную фантастику, вам понравится «Аватар».

Или:

Если вашей кредитной карточкой вчера пользовались в Китае, Канаде и Нигерии, значит, ее украли.

Если вашей кредитной карточкой пользовались два раза после 11 вечера в будний день, значит, ее украли.

Если с вашей кредитной карточки купили бензин на один доллар,
значит, ее украли.

(Если вы не поняли последнее правило, небольшое пояснение: раньше воры обычно покупали бензин на доллар, чтобы убедиться, что украденная карточка работает. Потом специалисты по добыче данных раскусили этот прием.)

С помощью алгоритма для нахождения конъюнктивных понятий, с которым мы познакомились выше, можно составлять подобные наборы по одному правило за правилом. Когда мы нашли правило, можно отбросить положительные примеры, которые оно включает, поэтому следующее правило будет пытаться охватить как можно больше оставшихся положительных примеров и так далее, пока все не будет включено. Это применение принципа «разделяй и властвуй», древнейшей стратегии в научном арсенале. Кроме того, мы можем улучшить алгоритм поиска отдельных правил, если будем иметь в запасе не одну, а n гипотез и на каждом этапе расширять их всеми возможными способами, сохраняя n лучших результатов.

Открытием такого способа поиска правил мы обязаны польскому информатику Рышарду Михальскому. Его родной город Калуш в разное время входил в состав Польши, СССР, Германии и Украины, и, возможно, именно это повлияло на его склонность к дизъюнктивным понятиям. Эмигрировав в 1970 году в США, он вместе с Томом Митчеллом и Джейми Карбонеллом основал символистскую школу машинного обучения. У Михальского был весьма деспотичный характер. Выступавшие на конференциях по машинному обучению не были застрахованы от того, что в конце он не поднимет руку и не заявит, что только что услышал повторение одной из своих старых идей.

Наборы правил популярны в торговых сетях: с их помощью определяют, какие товары надо закупать. Как правило, ретейлеры используют более всесторонний подход, чем «разделяй и властвуй», и ищут все правила, которые с большой вероятностью прогнозируют спрос. Пионер в этой области — Walmart. Еще на заре применения этого метода они открыли, что с подгузниками часто покупают пиво. Звучит странно? Одна из интерпретаций такая:

молодые матери посылают мужей в супермаркет за подгузниками, а те в качестве компенсации за моральный ущерб покупают себе ящик пива. Зная это, супермаркеты теперь могут продавать больше пенного напитка, выставляя его на полках по соседству с подгузниками. К такому выводу никогда не придешь без поиска правил: «закон пива и подгузников» стал легендой среди специалистов по добыче данных (некоторые, правда, утверждают, что это скорее городская легенда). Как бы то ни было, все это довольно далеко от проблем разработки цифровых схем, которые были на уме у Михальского, когда он в 1960-х впервые начал задумываться о логическом поиске правил. Изобретая новый алгоритм машинного обучения, нельзя даже представить себе все области, в которых он может найти применение.

Первый практический урок в области обучения правилам я получил, когда только переехал в США, чтобы поступить в аспирантуру, и подал заявку на получение кредитной карточки. Банк прислал мне письмо, в котором говорилось: «К сожалению, ваше заявление отклонено по следующим причинам: НЕДОСТАТОЧНО ДОЛГОЕ ПРОЖИВАНИЕ ПО ТЕКУЩЕМУ АДРЕСУ И ОТСУТСТВИЕ КРЕДИТНОЙ ИСТОРИИ» (или еще что-то в том же духе заглавными буквами). Тогда я понял, что в области машинного обучения предстоит еще немало работы.

Между слепотой и галлюцинациями

Наборы правил намного мощнее, чем конъюнктивные понятия. Вообще говоря, они настолько сильны, что с их помощью можно выразить *любое* понятие. Почему — понять несложно: если вы дадите мне полный список всех примеров какого-то понятия, я могу просто превратить каждый из них в правило, которое описывает все его атрибуты, и набор таких правил станет определением понятия. Если вернуться к нашей проблеме свидания, одним из правил будет такое: *Если сегодня выходной, на улице тепло, по телевизору не показывают ничего хорошего и я предложу сходить в клуб, она скажет «да».* В таблице содержится лишь несколько примеров, но, если в нее внести все $2 \times 2 \times 2 \times 2 = 16$ возможных и каждому

присвоить ярлык «Есть свидание» или «Нет свидания», превращение каждого положительного примера в правило решит проблему.

Наборы правил — мощный, но обоюдоострый меч. Их достоинство в том, что всегда можно найти набор правил, который идеально подойдет к имеющимся данным. Однако не спешите радоваться, что поймали удачу за хвост. Не забывайте: есть серьезнейший риск столкнуться с совершенно бессмысленным правилом. Помните теорему о бесплатных обедах? Учиться без знаний нельзя. Предположение, что понятие можно определить набором правил, — пустое предположение.

Один из частных случаев бесполезного набора правил просто включает все положительные примеры, которые вы видели, и ничего больше. Он может показаться стопроцентно точным, но это иллюзия: по его предсказаниям, каждый новый пример будет отрицательным, поэтому на каждом положительном он будет ошибаться. Если в целом положительных примеров больше, чем отрицательных, получится даже хуже, чем подбрасывать монетку. Представьте себе фильтр, который будет отправлять письма в спам, только если они точная копия сообщения, ранее помеченного как спам. Научить этому легко, это здорово работает с уже помеченной выборкой, но с тем же успехом можно вообще не иметь спам-фильтра. К сожалению, наш алгоритм «разделяй и властвуй» легко может научиться набору правил вроде этого.

В рассказе «Фунес памятливый» Хорхе Луис Борхес повествует о встрече с молодым человеком с идеальной памятью. Сначала такой дар может показаться редким везением, но на самом деле это ужасное проклятье. Фунес может вспомнить точную форму туч в небе в произвольный момент времени в прошлом, но ему сложно понять, что собака, которую он видел сбоку в 15:14, — та же самая собака, которую он видел спереди в 15:15, и он каждый раз удивляется собственному отражению в зеркале. Фунес неспособен обобщать, поэтому для него две вещи одинаковы, только если они выглядят идентично, вплоть до мелочей. Неограниченное обучение правилам похоже на Фунеса и совершенно неработоспособно. Учиться — значит забывать о подробностях в той же степени, как помнить о важных элементах. Компьютеры — высшее проявление

синдрома саванта[45]: они без малейших проблем запоминают все, но хотим мы от них не этого.

Проблема не ограничивается массовым запоминанием частностей. Каждый раз, когда обучающийся алгоритм находит в данных закономерность, которая в реальном мире ошибочна, мы говорим, что он «подогнал под ответ». Переобучение — центральная проблема машинного обучения: ей посвящено больше статей, чем любой другой теме. Каждый мощный обучающийся алгоритм — символистов, коннекционистов или любой другой — должен беспокоиться о паттернах-галлюцинациях, и единственный безопасный способ их избежать — серьезно ограничить то, чему обучающийся алгоритм может научиться: например, требовать, чтобы это были короткие конъюнктивные понятия. К сожалению, с водой можно выплеснуть и ребенка, и тогда алгоритм машинного обучения будет неспособен увидеть в данных большинство истинных схем. Таким образом, хороший обучающийся алгоритм всегда станет балансировать на узкой тропинке между слепотой и галлюцинациями.

Люди тоже не застрахованы от переобучения. Можно даже сказать, что это корень многих наших бед. Представьте себе ситуацию: маленькая белая девочка видит в торговом центре девочку-мексиканку и кричит: «Мама, смотри, ребенок-служанка!» (это реальный случай). Дело не в прирожденном расизме. Скорее, она слишком обобщила представление о тех немногих латиноамериканках, которых успела увидеть за свою короткую пока жизнь, — в мире полно представительниц этой этнической группы, не работающих прислугой, но девочка их пока не встретила. Наши убеждения основаны на опыте, а опыт дает очень неполную картину мира, поэтому перепрыгнуть к ложным выводам несложно. Ум и эрудиция тоже не панацея. Именно переобучением было утверждение Аристотеля, что для того, чтобы объект продолжал двигаться, к нему должна быть приложена сила. Лишь гениальный Галилей интуитивно почувствовал, что невозмущенные тела тоже продолжают двигаться, хотя не был в открытом космосе и собственными глазами этого не видел.

Однако обучающиеся алгоритмы, с их почти неограниченной способностью находить закономерности в данных, особенно

уязвимы для переобучения. За время, пока человек будет искать одну закономерность, компьютер найдет миллионы. В машинном обучении величайшая сила компьютера — способность обрабатывать огромное количество данных и бесконечно, без устали повторять одно и то же — одновременно становится его ахиллесовой пятой. Просто удивительно, сколько всего можно найти, если хорошенько поискать. В бестселлере 1998 года The Bible Code[46] утверждается, что Библия содержит предсказания будущих событий, которые можно прочитать, если брать буквы через определенные интервалы и составлять из них слова. К сожалению, есть столько способов это сделать, что «предсказания» обязательно найдутся в любом достаточно длинном тексте. Скептики ответили автору пророчествами из «Моби Дика» и постановлений Верховного суда, а также нашли в Книге Бытия упоминания о Розуэлле и летающих тарелках[47]. Джон фон Нейман, один из основоположников информатики, как-то точно заметил: «С четырьмя параметрами я могу подогнать слона, а с пятью заставлю его махать хоботом». Сегодня мы каждый день учим модели с миллионами параметров. Этого достаточно, чтобы каждый слон в мире махал хоботом по-своему. Кто-то даже сказал, что «добывать данные — значит пытать их до тех пор, пока они не признаются».

Переобучение серьезно усугубляется шумом. Шум в машинном обучении означает просто ошибки в данных или случайные события, которые нельзя предвидеть. Представьте, что ваша знакомая, которую вы собираетесь пригласить на свидание, очень любит ходить по клубам, когда по телевизору нет ничего интересного, но вы неправильно запомнили случай номер три и записали, что в тот вечер по телевидению *показывали* что-то хорошее. Если теперь вы попытаетесь составить набор правил, который делает исключение для того вечера, результат, вероятно, будет хуже, чем если просто его проигнорировать. Или представьте, что у девушки было похмелье после предыдущего вечера и она сказала «нет», хотя при обычных обстоятельствах согласилась бы. Если вы не знаете о ее состоянии, обучение набору правил, который верно учитывает этот пример, будет контрпродуктивным: целесообразнее «неправильно классифицировать» его как «нет». Все

очень плохо: шум может сделать невозможным составление *любого* связного набора правил. Обратите внимание, что случаи два и три на самом деле неразличимы: у них точно такие же атрибуты. Если ваша знакомая сказала «да» во втором случае и «нет» в третьем, отсутствует правило, которое верно учло бы оба.

Переобучение возникает, когда у вас слишком много гипотез и недостаточно данных, чтобы их различить. Проблема в том, что даже в простых конъюнктивных обучающихся алгоритмах число гипотез растет экспоненциально с числом атрибутов. Экспоненциальный рост — страшная сила. Бактерия *E. coli* может делиться надвое примерно каждые 15 минут. Если бы у нее было достаточно питательных веществ, она бы за день разрослась в бактериальную массу размером с нашу планету. Когда количество элементов, необходимых алгоритму для работы, растет в геометрической прогрессии с увеличением размера вводных данных, информатики называют это комбинаторным взрывом и бегут в укрытие. В машинном обучении количество возможных частных случаев какого-либо понятия — экспоненциальная функция числа атрибутов: если атрибуты булевы, с каждым новым атрибутом число возможных частных случаев удваивается, каждый случай расширяется для «да» или «нет» этого атрибута. В свою очередь, число возможных понятий — это экспоненциальная функция числа возможных частных случаев: поскольку понятие отмечает каждый случай как положительный или отрицательный, добавление частного случая удваивает число возможных понятий. В результате число понятий — это экспоненциальная функция экспоненциальной функции числа атрибутов! Другими словами, машинное обучение — комбинаторный взрыв комбинаторных взрывов. Может, лучше просто сдаться и не тратить времени на такую безнадежную проблему?

К счастью, при обучении получается «отрубить голову» одной из экспонент и оставить только «обычную» единичную неразрешимую экспоненциальную проблему. Представьте, что у вас полная сумка листочков с определениями понятий и вы достаете наугад одно из них, чтобы посмотреть, насколько хорошо оно подходит к данным. Вероятность, что плохое определение подойдет всей тысяче примеров в ваших данных, не больше, чем ситуация,

когда монетка тысячу раз подряд падает орлом вверх. «У стула четыре ноги, и он красный, либо у него есть сиденье, но нет ножек», вероятно, подойдет к некоторым, но не ко всем стульям, которые вы видели, а также подойдет к некоторым, но не ко всем другим предметам. Поэтому, если случайное определение корректно подходит к тысяче примеров, крайне маловероятно, что оно неправильное. По крайней мере, оно достаточно близко к истине. А если определение согласуется с миллионом примеров, оно практически наверняка верно, иначе почему оно подходит ко всем этим примерам?

Конечно, реальный алгоритм машинного обучения не просто берет из мешка одно произвольное определение: он пробует их целую охапку, и отбор не происходит произвольным образом. Чем больше определений пробует алгоритм, тем больше вероятность, что одно из них подойдет ко всем примерам хотя бы случайно. Если сделать миллион повторений по тысяче бросков монетки, практически наверняка хотя бы одно повторение даст все орлы, а миллион — это достаточно скромное число гипотез для рассмотрения: оно примерно соответствует числу возможных конъюнктивных понятий, если у примеров всего 13 атрибутов. (Обратите внимание, что вам не надо явно пробовать понятия одно за другим. Если лучшие, которые вы нашли с использованием конъюнктивного обучающегося алгоритма, подходят ко всем примерам, результат будет тот же самый.)

Итого: обучение — гонка между количеством данных, имеющихся в вашем распоряжении, и количеством рассматриваемых вами гипотез. Увеличение объема данных экспоненциально уменьшает количество прошедших проверку гипотез, но, если гипотез изначально много, в конце все равно может остаться некоторое количество плохих. Есть золотое правило: если обучающийся алгоритм учитывает только экспоненциальное число гипотез (например, все возможные конъюнктивные понятия), то экспоненциальный выигрыш от данных решает проблему, и все в порядке, при условии, что у вас множество примеров и не очень много атрибутов. С другой стороны, если алгоритм рассматривает дважды экспоненциальное число (например, все возможные наборы правил), тогда данные

отменяют только одну из экспонент, и трудности остаются. Можно даже заранее решить, сколько примеров понадобится, чтобы быть достаточно уверенным, что выбранная обучающимся алгоритмом гипотеза очень близка к истинной, при условии, что к ней подходят все данные. Другими словами, чтобы гипотеза была, вероятно, приблизительно правильной. За изобретение этого типа анализа гарвардский ученый Лесли Вэлиант получил премию Тьюринга — Нобелевскую премию по информатике. Его книга на эту тему называется Probably Approximately Correct («Возможно, приблизительно верно»).

Точность, которой можно доверять

На практике анализ по Вэлианту обычно дает очень пессимистичные результаты и требует больше данных, чем есть в наличии. Как же решить, верить ли обучающемуся алгоритму? Все просто: не верьте, пока не проверите результаты на данных, которые *обучающийся алгоритм не видел*. Если схемы, выдвинутые им в качестве гипотезы, окажутся верны и для новых данных, можно быть уверенным, что они реальные. В противном случае вы будете знать, что имело место переобучение. Это просто применение научного метода к машинному обучению: новой теории мало объяснить прошлый опыт (такую состряпать несложно) — она должна также делать новые предсказания, а принимают ее только после того, как предсказания были экспериментально подтверждены. (И даже тогда лишь предварительно, потому что будущие данные по-прежнему могут ее фальсифицировать.)

Общая теория относительности Эйнштейна стала общепринятой, только когда Артур Эддингтон эмпирически подтвердил, что Солнце отклоняет свет далеких звезд. Но вам не надо ждать, пока поступят новые данные, чтобы решить, можно ли доверять алгоритму машинного обучения. Лучше взять все данные, которые у вас есть, и произвольно разделить их на обучающее множество, которое вы дадите алгоритму, и тестовое множество, которое надо спрятать от него и использовать для верификации точности. Точность на скрытых данных — золотой

стандарт в машинном обучении. Можно написать статью о том, какой прекрасный новый обучающийся алгоритм вы придумали, но, если на скрытых данных он значительно не превосходит уже имеющиеся, статью никто не опубликует.

Точность на данных, которые алгоритм еще не видел, — настолько строгий критерий, что многие научные теории его не проходят. От этого они не становятся бесполезными, ведь наука — это не только предсказания, но и объяснение и понимание, однако в итоге, если модели не делают точных прогнозов на новых данных, нельзя быть уверенным, что лежащие в основе явления по-настоящему поняты и объяснены. А для машинного обучения тестирование на скрытых данных незаменимо, потому что это единственный способ определить, случилось ли с обучающимся алгоритмом переобучение.

Но и точность на тестовой выборке не гарантия от ошибок. Согласно легенде, один из ранних простых обучающихся алгоритмов со стопроцентной точностью отличал танки и в обучающей, и в тестовой выборке, каждая из которых состояла из 100 изображений. Удивительно или подозрительно? Оказалось, что все картинки с танками были светлее, и это все, что увидел обучающийся алгоритм. В наши дни выборки данных крупнее, но качество сбора данных не обязательно лучше, поэтому нельзя терять бдительность. Реалистичная эмпирическая оценка сыграла важную роль в превращении машинного обучения из молодой дисциплины в зрелую науку. До конца 1980-х исследователи каждого из «племен» в основном верили собственным аргументам, исходили из того, что их парадигма фундаментально лучше, и мало общались с другими лагерями. Затем символисты, например Рэй Муни и Джуд Шавлик, начали систематически сравнивать разные алгоритмы на тех же наборах данных, и — вот сюрприз! — оказалось, что однозначного победителя нет. Сегодня соперничество продолжается, но перекрестное опыление встречается гораздо чаще. Общие экспериментальные стандарты и большой банк наборов данных, поддерживаемый группой машинного обучения в Калифорнийском университете в Ирвайне, творят чудеса и толкают науку вперед. Как мы увидим, лучшие

шансы создать универсальный обучающийся алгоритм — у синтеза идей из разных парадигм.

Конечно, мало уметь выявить переобучение: прежде всего надо научиться его избегать. Это означает вовремя остановить даже потенциально превосходную подгонку под данные. Один из методов — применение тестов статистической значимости для проверки того, что схемы, которые мы видим, действительно существуют. Например, и правило, включающее 300 положительных примеров против 100 отрицательных, и правило, включающее три положительных примера против одного отрицательного, на обучающих данных точны в 75 процентах, однако первое правило почти наверняка лучше, чем бросок монетки, в то время как второе — нет, поскольку четыре броска «правильной» монетки легко могут дать три орла. Если в какой-то момент при составлении правила не получается найти условия, которые значительно улучшили бы его точность, нужно просто остановиться, даже если оно все еще охватывает некоторые отрицательные примеры. Точность правила на обучающей выборке окажется меньше, но, вероятно, оно будет более точным обобщением, а именно это нас на самом деле интересует.

Но и это еще не все. Если я попробую одно правило и оно окажется в 75 процентах точным на 400 примерах, я, вероятно, ему поверю. Но если я перепробую миллион правил и лучшее будет точным в 75 процентах из 400 примеров, я, вероятно, ему не поверю, потому что это вполне могло произойти случайно. Это та же проблема, с которой вы сталкиваетесь при выборе паевого фонда. Фонд «Ясновидец» десять лет подряд был лидером рынка. Ух ты! Наверное, у них гениальный управляющий! Или нет? Если у вас есть возможность выбирать из тысячи фондов, велик шанс, что десять лет лидером будет даже такой фонд, которым тайно управляют бросающие дротики мартышки. Научная литература тоже страдает от этой проблемы. Тесты статистической значимости — золотой стандарт при допуске результатов исследований к публикации, но, если эффект ищет несколько коллективов, а находит его только один, есть вероятность, что произошла ошибка, хотя по солидной на вид статье этого никак не определить. Одним из решений была бы публикация

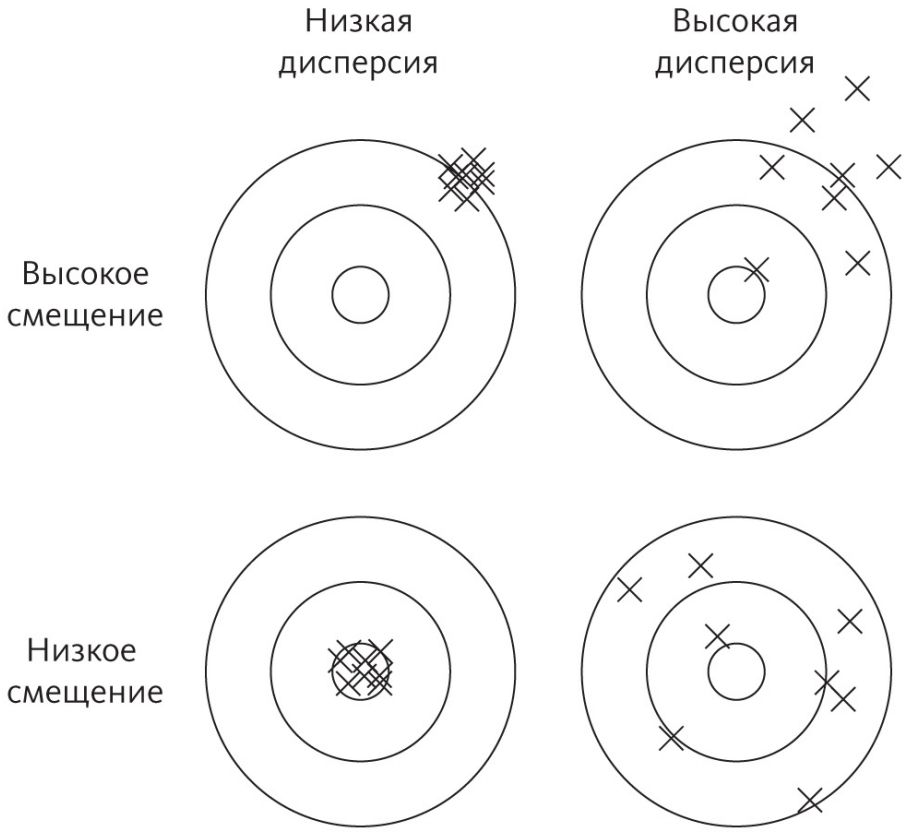
и положительных, и отрицательных результатов, чтобы читатель знал обо всех неудачных попытках, но такой подход не прижился. В машинном обучении можно отслеживать, сколько правил было испробовано, и соответствующим образом подбирать тесты значимости, однако тогда появляется тенденция выбрасывать много хороших правил, а не только плохих. Метод немного лучше — признать, что некоторые ложные гипотезы неизбежно прокрадутся, и держать их количество под контролем, отбрасывая гипотезы с низкой значимостью и тестируя оставленные на дальнейших данных.

Еще один популярный подход — отдавать предпочтение более простым гипотезам. Алгоритм «разделяй и властвуй» косвенно предпочитает простые правила, потому что условия перестают прибавляться к правилу, как только оно охватывает только положительные примеры, и перестает добавлять правила, как только все положительные примеры охвачены. Тем не менее для борьбы с переобучением нужно более сильное предпочтение простым правилам, которое остановит добавление условий еще до того, как будут охвачены все негативные примеры. Например, можно вычитать из точности штрафные очки, пропорциональные длине правила, и использовать это как средство оценки.

Предпочтение более простым гипотезам широко известно как бритва Оккама[48], однако в контексте машинного обучения этот принцип немного обманчив. «Не множить сущее без необходимости», как часто перефразируют бритву, означает только то, что нужно выбирать самую простую теорию, которая подходит к данным. Оккам, наверное, пришел бы в недоумение от мысли, что нам надо отдавать предпочтение теории, которая не идеально подходит к доказательствам, только на том основании, что она более качественно обобщает. Простые теории предпочтительнее не потому, что они обязательно точнее, а потому, что они означают меньшую когнитивную нагрузку (для нас) и меньшие вычислительные затраты (для наших алгоритмов). Более того, даже самые замысловатые модели — обычно лишь существенное упрощение реальности. Из теоремы о бесплатных обедах мы знаем, что даже в случае теорий, идеально подходящих к данным, нет гарантии, что простейшая обобщает лучше всего, а на практике

одни из лучших обучающихся алгоритмов — например, бустинг и метод опорных векторов — извлекают на первый взгляд необоснованно сложные модели. (Мы посмотрим, почему они работают, в главах 7 и 9.)

Если точность обучающегося алгоритма в тестовой выборке разочаровывает, надо диагностировать проблему: дело в слепоте или галлюцинациях? В машинном обучении для этих проблем существуют специальные термины: *смещение* и *дисперсия*. Часы, которые постоянно опаздывают на час, имеют большое смещение, но низкую дисперсию. Если часы беспорядочно идут то быстро, то медленно, но в среднем показывают правильное время, дисперсия высокая, но смещение низкое. Представьте, что вы сидите в баре с друзьями, выпиваете и играете в дартс. Вы втайне от них годами тренируетесь, добились мастерства, и все дротики попадают прямо в яблочко. У вас низкое смещение и низкая дисперсия, что показано в нижнем левом углу этой диаграммы:



Ваш друг Бен тоже очень хорош, но сегодня вечером немного перебрал. Его дротиками утыкана вся мишень, но тем не менее он громко заявляет, что в среднем попал в десятку. (Может быть, ему надо было посвятить себя статистике.) Это случай низкого смещения и высокой дисперсии, показанный в правом нижнем углу. Подруга Бена Эшли попадает стабильно, но у нее есть склонность метить слишком высоко и вправо. Дисперсия у нее низкая, а смещение высокое (левый верхний угол). Коди никогда до этого не играл в дартс. Он попадает куда угодно, только не в центр. У него и высокое смещение, и высокая дисперсия (вверху справа).

Вы можете оценить смещение и дисперсию обучающегося алгоритма, сравнив его прогнозы после обучения на случайных вариациях обучающей выборки. Если он продолжает повторять те же самые ошибки, проблема в смещении и нужно сделать его эластичнее (или просто взять другой). Если в ошибках алгоритма нет никакой схемы, проблема в дисперсии и надо либо попробовать менее гибкий, либо получить больше данных. У большинства обучающихся алгоритмов есть «ручка», с помощью которой можно отрегулировать гибкость: это, например, порог значимости и штрафы за размер модели. Подстройка — первое, что нужно попробовать.

Индукция — противоположность дедукции

Более глубокая проблема, однако, заключается в том, что большинство обучающихся алгоритмов начинают с очень скромного объема знаний, и никакая подстройка не сможет вывести их к финишной черте. Без руководства знаниями, равными по объему содержимому мозга взрослого человека, они легко сбиваются с курса. Простое допущение, что вы знаете форму правды (например, что это маленький набор правил), — совсем немного, хотя из этого исходит большинство алгоритмов. Строгий эмпирик заметил бы, что это все, что закодировано в архитектуре головного мозга новорожденного. И действительно, дети подвержены переобучению чаще, чем взрослые, однако мы хотели бы учиться быстрее, чем младенцы (даже если не считать колледж, 18 лет — это все равно долго). Верховный алгоритм должен уметь начинать

с большого объема знаний, заложенных людьми или выученных в предыдущие заходы, и использовать его для извлечения из данных новых обобщений. Этот подход практикуют ученые, и это далеко не «чистая доска». Индукционный алгоритм, основанный на правиле «разделяй и властвуй», на это не способен, но это может сделать другой способ формулировки правил.

Главное — понять, что индукция — просто обратный дедукции процесс, точно так же как вычитание — это противоположность деления, а интегрирование — противоположность дифференцирования. Идея была впервые предложена Уильямом Стэнли Джевонсом[49] в конце первого десятилетия XIX века. В 1988 году англичанин Стив Магглтон и австралиец Рэй Бантайн разработали первый практический алгоритм, основанный на этом принципе. Стратегия брать хорошо известную операцию и выводить ее противоположность имеет в математике долгую историю. Применение этого принципа к сложению привело к изобретению целых чисел, потому что без отрицательных чисел сложение не всегда имеет противоположность (3 – 4 = –1). Аналогично применение его к умножению привело к открытию рациональных чисел, а возведение в квадрат дало комплексные числа. Давайте посмотрим, можно ли применить этот принцип к дедукции. Вот классический пример дедуктивного рассуждения:

Сократ — человек.
Все люди смертны.
Следовательно...

Первое утверждение — факт о Сократе, а второе — общее правило о людях. Что из этого следует? Конечно, что Сократ смертен, если применить это правило к Сократу. При индуктивном рассуждении мы вместо этого начинаем с исходного факта следствия и ищем правило, которое позволило бы вывести второе из первого:

Сократ — человек.
.....
Следовательно, Сократ смертен.

Одним из таких правил будет: *если Сократ — человек, значит, он смертен.* Это правило соответствует условиям задачи, но не очень полезно, потому что не специфично для Сократа. Однако теперь мы применим принцип Ньютона и обобщим правило до всех сущностей: *если сущность — человек, значит, она смертна.* Или, более сжато: *все люди смертны.* Конечно, было бы опрометчиво выводить это правило на примере одного только Сократа, однако нам известны аналогичные факты о других людях:

Платон — человек. Платон смертен.
Аристотель — человек. Аристотель смертен.
И так далее.

Для каждой пары фактов мы формулируем правило, которое позволяет нам вывести второй факт из первого и обобщить его благодаря принципу Ньютона. Если одно и то же общее правило выводится снова и снова, можно с определенной уверенностью сказать, что оно верно.

Пока что мы еще не сделали ничего такого, чего бы не умел алгоритм «разделяй и властвуй». Однако предположим, что вместо информации, что Сократ, Платон и Аристотель — люди, мы знаем только, что они философы. Мы по-прежнему хотим сделать вывод, что они смертны, и до этого сделали вывод или нам сказали, что все люди смертны. Чего не хватает теперь? Другого правила: *все философы — люди.* Это тоже вполне обоснованное обобщение (как минимум пока мы не решим проблему искусственного интеллекта и роботы не начнут философствовать), и оно заполняет пробел в наших рассуждениях:

Сократ — философ.
Все философы — люди.
Все люди смертны.
Следовательно, Сократ смертен.

Кроме того, мы можем выводить правила исключительно на основе других правил. Если мы знаем, что все философы — люди и все философы смертны, то можем индуцировать, что все люди смертны. (Мы не можем, однако, сделать вывод, что все смертные —

люди, потому что нам известны другие смертные существа, например кошки и собаки. С другой стороны, ученые, люди искусства и так далее — тоже люди и тоже смертны, а это укрепляет правило.) В целом чем больше правил и фактов у нас есть изначально, тем больше возможностей индуцировать новые правила путем обратной дедукции. А чем больше правил мы индуцируем, тем больше можем индуцировать. Это положительная спираль создания знаний, которая ограничена только риском переобучения и вычислительной сложностью. Но от этих проблем исходные знания тоже помогают: если вместо одной большой дыры надо заполнить много маленьких, этапы индукции будут менее рискованными и, следовательно, менее подверженными переобучению. (Например, при том же количестве примеров выведение путем индукции правила, что все философы — люди, менее рискованно, чем вывод, что все люди смертны.)

Обратить операцию часто бывает сложно, потому у нее может быть несколько противоположностей: например, у положительного числа есть два квадратных корня — положительный и отрицательный ($2^2 = (-2)^2 = 4$). Самый знаменитый пример — то, что интеграл производной функции воссоздает эту функцию лишь до постоянной. Производная функции говорит нам, насколько она идет вверх и вниз в данной точке. Сложение всех этих изменений возвращает нам эту функцию, за исключением того, что мы не знаем, где она началась. Мы можем «проматывать» интеграл функции вверх или вниз без изменения производной. Чтобы упростить проблему, функцию можно «сжать», предположив, что аддитивная постоянная равна нулю. У обратной дедукции схожая проблема, и одно из ее решений — принцип Ньютона. Например, из правил *«Все греческие философы — люди»* и *«Все греческие философы смертны»* можно сделать вывод *«Все люди смертны»* или просто *«Все греки смертны»*. Однако зачем довольствоваться более скромным обобщением? Вместо этого лучше предположить, что все люди смертны, пока не появится исключение. (Которое, по мнению Рэя Курцвейла, скоро появится.)

Одна из важных областей применения обратной дедукции — прогнозирование наличия у новых лекарств вредных побочных эффектов. Неудачное тестирование на животных и клинические

испытания — главная причина, по которой разработка новых лекарств занимает много лет и стоит миллиарды долларов. Путем обобщения молекулярных структур известных токсичных веществ можно будет создать правила, которые быстро «прополют» предположительно многообещающие соединения и значительно увеличат шанс успешного прохождения испытаний оставшимися.

Как научиться лечить рак

В целом обратная дедукция — прекрасный путь к новым знаниям в биологии, а это первый шаг к лечению рака. Центральная догма гласит, что все, что происходит в живой клетке, в итоге контролируется генами посредством белков, синтез которых гены инициируют. В результате клетка похожа на крохотный компьютер, а ДНК — на действующую в нем программу: измените ДНК, и клетка кожи может стать нейроном, а мышиная клетка — превратиться в человеческую. В компьютерной программе все ошибки на совести программиста, но в клетке сбои происходят спонтанно, например под действием радиации или из-за ошибок при копировании: гены могут меняться, удваиваться и так далее. В большинстве случаев такие мутации приводят к тихой смерти клетки, но иногда она начинает расти, бесконтрольно делиться, и человек заболевает раком.

Чтобы вылечить рак, нужно остановить воспроизведение больных клеток, не повредив при этом здоровые. Для этого необходимо знать, чем они отличаются и, в частности, чем отличаются их геномы, поскольку все остальное — следствие. К счастью, секвенирование генов становится рутинной и доступной процедурой, а с его помощью можно научиться предсказывать, какие лекарства будут работать против конкретных генов рака: это совсем не похоже на традиционную химиотерапию, при которой уничтожаются все клетки без разбора. Чтобы узнать, какие лекарства сработают против определенных мутаций, требуется база данных пациентов, геномов их опухолей, проверенных лекарств и исходов. Простейшие правила кодируют прямые соответствия между генами и лекарствами. Когда секвенирование геномов

опухолей и сопоставление исходов лечения станет стандартной практикой, будет открыто много подобных правил.

Однако это только начало. Большинство видов рака представляют собой комбинацию мутаций, и лекарства для их лечения пока еще не изобретены. Поэтому следующий шаг — сформулировать правила с более сложными условиями, учитывающими геном рака, геном пациента, историю болезни, известные побочные эффекты препаратов и так далее. Однако важнейшая цель — составить полную модель функционирования клетки. Это позволит нам симулировать на компьютере результаты последствия мутаций у конкретного пациента, а также действие различных комбинаций лекарств, уже существующих и потенциально возможных. Главные источники информации для построения таких моделей — это секвенсоры ДНК, микрочипы для анализа экспрессии генов и биологическая литература. Соединить эту информацию — очень подходящее задание для обратной дедукции.

Адам, робот-ученый, с которым мы уже знакомы, дает представление о том, как это может выглядеть. Его задача — разобраться, как работает дрожжевая клетка. Все начинается с базовых знаний о генетике и метаболизме дрожжей и уже собранных данных об экспрессии генов в дрожжевых клетках. Затем Адам с помощью обратной дедукции выдвигает гипотезы о том, какие гены кодируют какие белки, проектирует эксперименты с ДНК-микрочипами, чтобы проверить гипотезы, затем корректирует их и переходит к следующему циклу. Будет ли происходить экспрессия данного гена, зависит от других генов и средовых факторов, и итоговую сеть взаимодействий можно представить в виде набора правил, например:

Если температура высокая, ген *A* активен.
Если ген *A* активен, а ген *B* — нет, происходит экспрессия гена *C*.
Если ген *C* активен, экспрессии гена *D* не будет.

Если бы мы знали первое и третье правила, но не знали второго и у нас были бы данные с ДНК-микрочипа, где при высокой температуре экспрессии *B* и *D* не наблюдается, то могли бы вывести

второе правило путем обратной дедукции. Когда у нас будет это правило и, возможно, мы подтвердим его путем эксперимента с микрочипом, его можно будет использовать как основу для дальнейших выводов путем индукции. Аналогичным путем можно собрать воедино цепочки химических реакций, благодаря которым белки выполняют свою функцию.

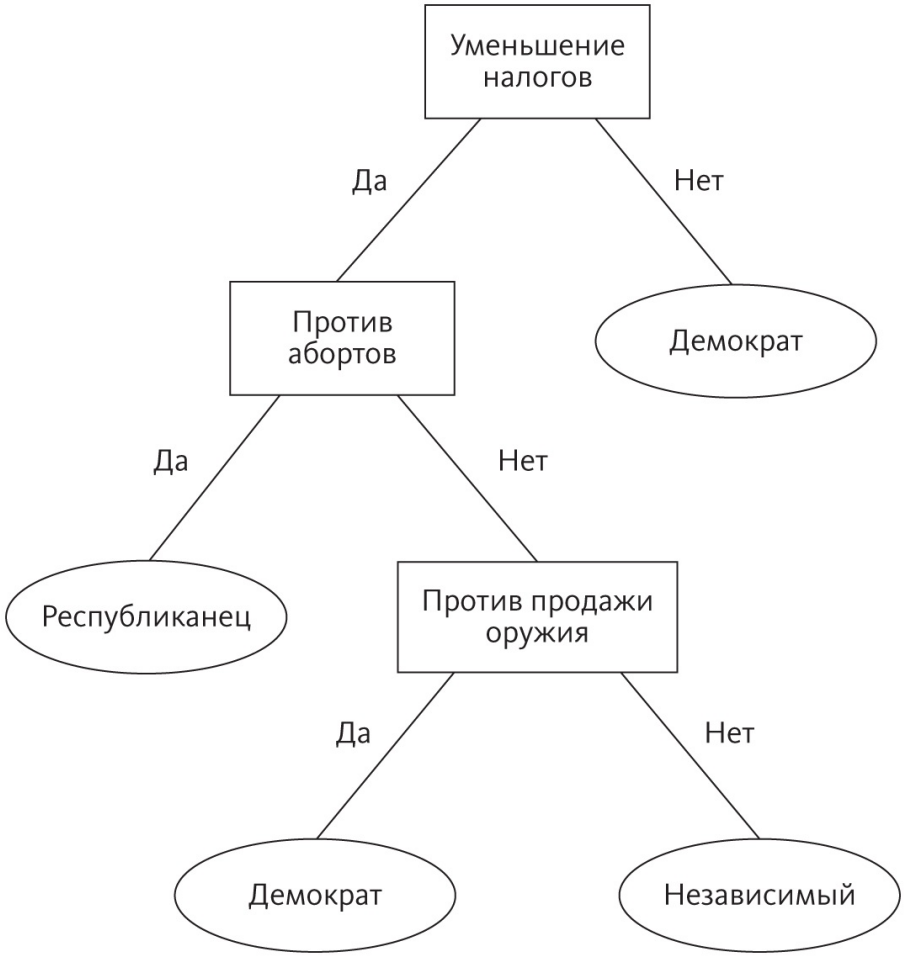
Однако недостаточно просто знать, как происходит взаиморегуляция генов и как организована сеть белковых реакций в клетке. Нужна информация, сколько именно вырабатывается молекул каждого вещества. Микрочипы ДНК и другие эксперименты могут предоставить такую количественную информацию, но обратная дедукция с ее логическим характером «все или ничего» не очень хорошо подходит для подобных задач. Для этого нам понадобятся коннекционистские методы, с которыми мы познакомимся в следующей главе.

Игра в двадцать вопросов[50]

Другое ограничение обратной дедукции заключается в том, что оно требует большого объема вычислений и из-за этого его трудно применять к масштабным наборам данных. Для решения этой проблемы символисты прибегают к индукции с помощью дерева решений. Деревья решений можно считать ответом на вопрос, что делать, если к какому-то частному случаю применимы правила не одного, а целого ряда понятий. Как в таком случае решить, к какому понятию принадлежит этот случай? Если перед нами частично скрытый предмет с плоской поверхностью и четырьмя ножками, как понять, стол это или стул? Один из вариантов — упорядочить правила, например, в порядке уменьшения точности и выбрать первое подходящее. Другой — дать правилам проголосовать. Деревья решений же априори гарантируют, что к каждому случаю будет подобрано ровно одно правило. Это будет так, если каждая пара правил отличается как минимум в одном тестировании атрибутов и такой набор правил можно выстроить в виде дерева решений. Например, посмотрите на следующий набор:

Если вы за уменьшение налогов и против абортов, вы республиканец.
Если вы против уменьшения налогов, вы демократ.
Если вы за уменьшение налогов, за право на аборт и за свободный оборот оружия, вы независимый кандидат.
Если вы за уменьшение налогов, за право на аборт и против свободного оборота оружия, вы демократ.

Все это можно организовать в виде следующего дерева решений:



Дерево решений — как игра в «20 вопросов» с каждым случаем. Начиная с корня каждый узел спрашивает про значение одного атрибута, и, в зависимости от ответа, мы следуем по той или иной ветви. Когда мы достигаем «листа» дерева, на нем нас ждет предсказанное понятие. Каждый путь от корня до листа соответствует правилу. Если принцип напоминает вам о длинной

серии вопросов, через которые приходится проходить, чтобы дозвониться в клиентскую службу, это не случайно: раздражающие голосовые меню тоже деревья решений. Компьютер на другом конце провода играет с вами в ту же самую игру, чтобы понять, чего вы хотите. Каждый пункт меню — это вопрос.

Согласно дереву решений выше, вы либо республиканец, либо демократ, либо независимый кандидат. Невозможна ситуация, когда этих вариантов больше чем один или ни одного. Наборы понятий, обладающие этим свойством, называют наборами классов, а алгоритмы, которые их определяют, — классификаторами. Каждое понятие косвенно определяет два класса: оно само и его отрицание (например, спам и не-спам). Классификаторы — самая широко распространенная форма машинного обучения.

Обучать деревья решений можно с помощью одного из вариантов алгоритма «разделяй и властвуй». Сначала надо выбрать атрибут, который будет протестирован у корня. Затем мы сосредоточимся на примерах с нисходящих ветвей и выберем для них следующие тесты (например, проверим, за или против абортов сторонники уменьшения налогов). Процесс будет повторяться для каждого нового узла, который мы получим путем индукции, пока все примеры в ветви не будут принадлежать к одному классу. В этот момент мы присвоим этой ветви данный класс.

Напрашивается вопрос: как выбрать лучший атрибут для тестирования в узле? Точность — количество правильно предсказанных примеров — работает не очень хорошо, потому что мы не пытаемся предсказать конкретный класс, а, скорее, стремимся постепенно разделять классы, пока не «очистим» все ветви. Это заставляет вспомнить понятие энтропии[51] из теории информации. Энтропия набора предметов — мера его неупорядоченности. Если в группе из 150 человек будет 50 республиканцев, 50 демократов и 50 независимых кандидатов, ее политическая энтропия максимальна. С другой стороны, если в группе одни республиканцы, энтропия будет равна нулю, во всяком случае, в отношении партийной принадлежности. Поэтому, чтобы получить хорошее дерево решений, мы выберем в каждом узле атрибут, который в среднем даст самую низкую

энтропию классов по всем ее ветвям, с учетом количества примеров в каждой из ветвей.

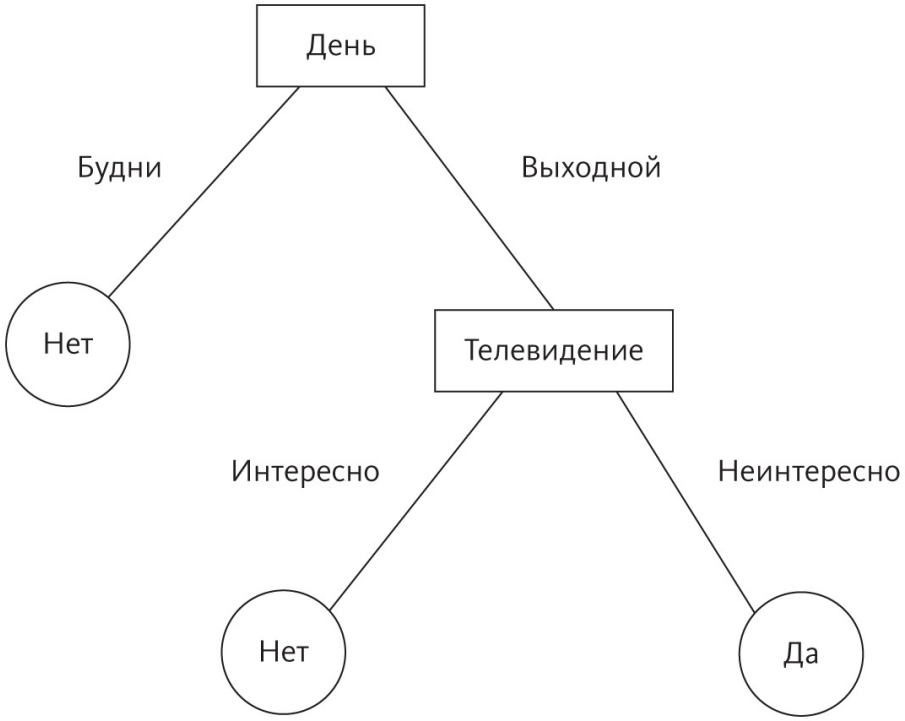
Как и в случае обучения правилам, мы не хотим получить дерево, которое будет идеально предсказывать классы всех примеров в обучающей выборке, потому что это будет, вероятно, переобучением. Для его предотвращения мы, опять же, можем использовать тесты значимости или штрафные очки для больших размеров дерева.

Иметь отдельную ветвь для каждого значения атрибута неплохо, если они дискретные. А как насчет числовых атрибутов? Если выделять ветвь для каждого значения непрерывной переменной, дерево окажется бесконечно широким. Простое решение — выбрать ряд ключевых порогов на основе энтропии и использовать их. Например, «температура пациента выше или ниже 37,7 °C?». Для выявления у человека инфекции этой информации в сочетании с другими симптомами может быть достаточно.

Деревья решений находят применение во многих областях. Так, они делают важную работу в психологии. Эрл Хант[52] и его коллеги пользовались деревьями решений в 1960 году для моделирования усвоения человеком новых концепций, а один из магистрантов Ханта, Джон Росс Куинлан, попробовал использовать их в шахматах. Его первоначальная цель была скромной: предсказать результаты эндшпилей «король и ладья против короля и ферзя» на основе ситуации на доске. Теперь же дерево решений, согласно опросам, стало самым широко используемым алгоритмом машинного обучения, что неудивительно: эту методику легко понять и освоить, и обычно она дает довольно точный результат без лишних настроек. Куинлан — самый выдающийся исследователь в школе символистов. Этот невозмутимый прагматичный австралиец год за годом неустанно улучшал деревья решений, сделал их золотым стандартом в области классификации и пишет о них удивительно ясные статьи.

Что бы вы ни хотели предсказать, очень вероятно, что кто-то уже использовал для этого деревья решений. С их помощью разработанный Microsoft игровой контроллер Kinect определяет положение частей тела, получая сигналы от сенсоров глубины, и передает информацию в приставку Xbox. В 2002 году деревья

решений обошли группу экспертов, правильно предсказав три из каждых четырех постановлений Верховного суда, в то время как люди дали менее 60 процентов правильных ответов. «Тысячи поклонников деревьев решений не могут ошибаться!» — думаете вы и набрасываете свое дерево, чтобы угадать ответ девушки на ваше приглашение:



Получается, что сегодня вечером она скажет «да». Вы делаете глубокий вдох, достаете телефон и набираете ее номер.

Символисты

Важнейшее убеждение символистов заключается в том, что интеллект можно свести к манипулированию символами. Математик решает уравнения, переставляя символы и заменяя одни другими согласно заранее определенным правилам. Так же поступает логик, когда делает выводы путем дедукции. Согласно этой гипотезе, интеллект не зависит от носителя: можно писать символы мелом на доске, включать и выключать транзисторы, выражать их импульсами между нейронами или с помощью конструктора Tinkertoys. Если у вас есть структура, обладающая

мощью универсальной машины Тьюринга, вы сможете сделать все что угодно. Программное обеспечение можно вообще отделить от «железа», и, если вы хотите просто разобраться, как могут учиться машины, вам (к счастью) не надо волноваться о машинах как таковых, за исключением приобретения ПК или циклов на облаке Amazon.

Веру символистов в мощь манипуляций символами разделяют многие другие информатики, психологи и философы. Психолог Дэвид Марр утверждает, что любую систему обработки информации нужно рассматривать на трех уровнях: фундаментальные свойства проблемы, которую она решает, алгоритмы и представления, которые используются для ее решения, и их физическое воплощение. Например, сложение можно определить набором аксиом, не зависящих от того, как оно выполняется. Числа можно выразить по-разному (например, римскими и арабскими цифрами) и складывать с использованием разных алгоритмов, а алгоритмы могут выполняться на абаке, карманном калькуляторе или даже — что очень неэффективно — в уме. Обучение — яркий пример когнитивной способности, которую мы можем плодотворно изучать с точки зрения уровней Марра.

Символистское машинное обучение — ответвление инженерии знаний, одной из школ искусственного интеллекта. В 1970-х у так называемых систем на основе знаний были очень впечатляющие успехи, в 1980-х они быстро распространились, но потом вымерли. Главная причина — печально известное «узкое горло» приобретения знаний: получать информацию от экспертов и кодировать в виде правил слишком сложное, трудоемкое и подверженное ошибкам занятие, поэтому для большинства проблем такой подход нецелесообразен. Оказалось, что намного легче позволить компьютеру автоматически учиться, скажем, диагностировать заболевания путем просмотра в базах данных симптомов и исходов, чем без конца опрашивать врачей. Внезапно работы таких первопроходцев, как Рышард Михальский, Том Митчелл и Росс Куинлан, приобрели новую значимость, и с тех пор дисциплина непрерывно развивается. (Еще одной важной проблемой систем, основанных на знаниях, было то, что им сложно работать с неопределенностью. Подробнее мы поговорим об этом в главе 6.)

Благодаря своему происхождению и основополагающим принципам символистское машинное обучение ближе к другим областям науки об искусственном интеллекте, чем другие школы машинного обучения. Если информатику представить в виде континента, у символизма будет длинная граница с инженерией знаний. Обмен информацией происходит в обоих направлениях: обучающиеся алгоритмы используют введенное вручную знание, а знание, полученное путем индукции, пополняет базы знаний. Тем не менее вдоль этой границы проходит разлом между рационалистами и эмпириками, и пересечь ее непросто.

Символизм — кратчайший путь к Верховному алгоритму. Он не требует разбираться, как работает эволюция или головной мозг, и позволяет обойтись без сложной математики байесианства. Наборы правил и деревья решений просты для понимания, и поэтому пользователь представляет себе, что замышляет обучающийся алгоритм, ему легче отличить правильные действия от неправильных, при необходимости внести поправки и быть уверенным в результатах.

Но несмотря на популярность деревьев решений, более удобный исходный пункт для поисков Верховного алгоритма — обратная дедукция. У нее есть критически важное качество: в нее легко встраивать знания, а, как нам уже известно, из-за проблемы Юма это существенное преимущество. Кроме того, наборы правил — экспоненциально более компактный способ представления большинства понятий, чем деревья решений. Превратить дерево решений в набор правил несложно: каждый путь от корня к листу становится правилом, и нет никаких подводных камней. С другой стороны, если нужно превратить в дерево решений набор правил, в худшем случае придется разворачивать каждое из них в мини-дерево решений, а затем заменять каждый листок дерева, полученного из правила один, копией дерева для правила два, каждый листок каждой копии правила два копией правила три и так далее, что порождает серьезные проблемы.

Обратная дедукция как сверхученый. Он будет систематически рассматривать доказательства, взвешивать возможные выводы, сопоставлять лучшие и использовать их вместе с другими доказательствами для формулировки дальнейших гипотез, и все это

с компьютерной скоростью. Это чисто и изящно, по крайней мере на вкус символиста. С другой стороны, у метода есть ряд серьезных недостатков. Количество возможных выводов очень велико, и, чтобы не заблудиться, приходится не держаться близко к исходному знанию. Обратную дедукцию легко запутать шумом: как разобраться, каких шагов в дедукции не хватает, если предположения или заключения ложны? Еще более серьезно то, что реальные понятия очень часто не определяются сжатым набором правил. Они не черно-белые, а находятся в большой серой зоне между, скажем, спамом и не-спамом, поэтому приходится взвешивать и накапливать слабые доказательства, пока картина не прояснится. В частности, при диагностике заболеваний одним симптомам придается большее значение, чем другим, и неполные доказательства — это нормально. Никто еще не преуспел в обучении набору правил, которое будет определять кошку, глядя на пиксели на картинке, и, наверное, это просто невозможно.

Очень критично по отношению к символистскому обучению настроены коннекционисты. Они считают, что понятия, которые можно определить с помощью логических правил, лишь вершина айсберга, а в глубине есть много такого, что формальные рассуждения просто неспособны увидеть, точно так же как значительная часть работы мозга скрыта в подсознании. Нельзя построить бесплотного автоматического ученого и надеяться, что он сделает что-то полезное: сначала надо одарить его чем-то вроде настоящего мозга, соединенного с настоящими органами чувств, вырастить в реальном мире, возможно, даже ставить ему время от времени подножки. Как же построить такой мозг? Путем обратной инженерии. Если вы решили построить путем обратной инженерии автомобиль, придется заглянуть под капот. Если вы хотите таким же образом создать мозг, надо заглянуть в черепную коробку.

ГЛАВА 4

КАК УЧИТСЯ НАШ МОЗГ?

С момента своего открытия правило Хебба — краеугольный камень коннекционизма. Своим названием это научное направление обязано представлению, что знания хранятся в соединениях между нейронами. В вышедшей в 1949 году книге The Organization of Behavior («Организация поведения») канадский психолог Дональд Хебб описывал это следующим образом: «Если аксон[53] клетки *A* расположен достаточно близко к клетке *B* и неоднократно или постоянно участвует в ее стимуляции, то в одной или обеих клетках будут иметь место процессы роста или метаболические изменения, которые повышают эффективность возбуждения клеткой *A* клетки *B*». Это утверждение часто перефразируют как «нейроны, которые срабатывают вместе, связываются друг с другом».

В правиле Хебба слились идеи психологии, нейробиологии и немалая доля домыслов. Ассоциативное обучение было любимой темой британских эмпириков начиная с Локка, Юма и Джона Стюарта Милля. В Principles of Psychology («Принципы психологии») Уильям Джеймс[54] сформулировал общий принцип ассоциации, который замечательно похож на правило Хебба, но вместо нейронов в нем присутствуют процессы в головном мозге, а вместо эффективности стимуляции — распространение возбуждения. Примерно в то же самое время великий испанский нейробиолог Сантьяго Рамон-и-Кахаль провел первые подробные исследования мозга, окрашивая нейроны по недавно изобретенному методу Гольджи[55], и каталогизировал свои наблюдения, как ботаники классифицируют новые виды деревьев. Ко времени Хебба нейробиологи в общих чертах понимали, как работают нейроны, однако именно он первым предложил механизм, согласно которому нейроны могут кодировать ассоциации.

В символистском обучении между символами и понятиями, которые они представляют, существует однозначное соответствие. Коннекционистские же представления распределены: каждое понятие представлено множеством нейронов, и каждый нейрон

участвует в представлении многих концепций. Нейроны, которые возбуждают друг друга, образуют, в терминологии Хебба, «ансамбли клеток». С помощью таких собраний в головном мозге представлены понятия и воспоминания. В каждый ансамбль могут входить нейроны из разных областей мозга, ансамбли могут пересекаться. Так, клеточный ансамбль для понятия «нога» включает ансамбль для понятия «ступня», в который, в свою очередь, входят ансамбли для изображения ступни и звучания слова «ступня». Если вы спросите символистскую систему, где находится понятие «Нью-Йорк», она укажет точное место его хранения в памяти. В коннекционистской системе ответ будет «везде понемногу».

Еще одно отличие между символистским и коннекционистским обучением заключается в том, что первое — последовательное, а второе — параллельное. В случае обратной дедукции мы шаг за шагом разбираемся, какое правило необходимо ввести, чтобы от посылок прийти к желаемым выводам. В коннекционистской модели все нейроны учатся одновременно, согласно правилу Хебба. В этом нашли отражение различия между компьютерами и мозгом. Компьютеры даже совершенно обычные операции — например, сложение двух чисел или переключение выключателя — делают маленькими шажочками, поэтому им нужно много этапов. При этом шаги могут быть очень быстрыми, потому что транзисторы способны включаться и выключаться миллиарды раз в секунду. Мозг же умеет выполнять большое количество вычислений параллельно благодаря одновременной работе миллиардов нейронов. При этом нейроны могут стимулироваться в лучшем случае тысячу раз в секунду, и каждое из этих вычислений медленное.

Количество транзисторов в компьютере приближается к количеству нейронов в головном мозге человека, однако мозг безусловно выигрывает в количестве соединений. Типичный транзистор в микропроцессоре непосредственно связан лишь с немногими другими, и применяемая технология планарных полупроводников жестко ограничивает потенциал совершенствования работы компьютера. А у нейрона — тысячи синапсов. Если вы идете по улице и увидели знакомую, вам

понадобится лишь десятая доля секунды, чтобы ее узнать. Учитывая скорость переключения нейронов, этого времени едва хватило бы для сотни шагов обработки информации, но за эти сотни шагов мозг способен просканировать всю память, найти в ней самое подходящее и адаптировать найденное к новому контексту (другая одежда, другое освещение и так далее). Каждый шаг обработки может быть очень сложным и включать большой объем информации.

Это не значит, что с помощью компьютера нельзя симулировать работу мозга: в конце концов, именно это делают коннекционистские алгоритмы. Поскольку компьютер — универсальная машина Тьюринга, он может выполнять вычисления, происходящие в мозге, как и любые другие, при условии, что у него есть достаточно памяти и времени. В частности, недостаток связности можно компенсировать скоростью: использовать одно и то же соединение тысячу раз, чтобы имитировать тысячу соединений. На самом деле сегодня главный недостаток компьютеров заключается в том, что в отличие от мозга они потребляют энергию: ваш мозг использует примерно столько мощности, сколько маленькая лампочка, в то время как электричеством, питающим компьютер Watson, о котором мы рассказывали выше, можно осветить целый бизнес-центр.

Тем не менее для имитации работы мозга одного правила Хебба мало: сначала надо разобраться с устройством головного мозга. Каждый нейрон напоминает крохотное деревце с огромной корневой системой из дендритов[56] и тонким волнистым стволом — аксоном. Мозг в целом похож на лес из миллиардов таких деревьев, однако лес этот необычный: ветви деревьев соединены в нем с корнями тысяч других деревьев (такие соединения называются синапсами), образуя колоссальное, невиданное хитросплетение. У одних нейронов аксоны короткие, у других — чрезвычайно длинные, простирающиеся от одного конца мозга к другому. Если расположить аксоны мозга друг за другом, они займут расстояние от Земли до Луны.

Эти джунгли потрескивают от электрических разрядов. Искры бегут по стволам и порождают в соседних деревьях еще больший сонм искр. Время от времени лес неистово вспыхивает, потом снова

успокаивается. Когда человек шевелит пальцем на ноге, серии электрических разрядов — так называемых потенциалов действия — бегут вниз по спинному мозгу, пока не достигнут мышц пальца и не прикажут ему двигаться. Работа мозга похожа на симфонию таких электрических разрядов. Если бы можно было посмотреть изнутри на то, что происходит в тот момент, когда вы читаете эту страницу, сцена затмила бы самые оживленные мегаполисы из фантастических романов. Этот невероятно сложный узор нейронных искр в итоге порождает человеческое сознание.

Во времена Хебба еще не умели измерять силу синапсов и ее изменения, не говоря уже о том, чтобы разбираться в молекулярной биологии синаптических процессов. Сегодня мы знаем, что синапсы возникают и развиваются, когда вскоре после пресинаптических нейронов возбуждаются постсинаптические. Как и во всех других клетках, концентрация ионов внутри и за пределами нейрона отличается, и из-за этого на клеточной мембране имеется электрическое напряжение. Когда пресинаптический нейрон возбуждается, в синаптическую щель выделяются крохотные пузырьки с молекулами нейротрансмиттеров. Они заставляют открыться каналы в мембране постсинаптического нейрона, из которых выходят ионы калия и натрия, меняющие напряжение на мембране. Если одновременно возбуждается достаточное количество близко расположенных пресинаптических нейронов, напряжение подскакивает и по аксону постсинаптического нейрона проходит потенциал действия. Благодаря этому ионные каналы становятся восприимчивее, а также появляются новые, усиливающие синапс каналы. Насколько нам известно, нейроны учатся именно так.

Следующий шаг — превратить все это в алгоритм.

Взлет и падение перцептрона

Первая формальная модель нейрона была предложена в 1943 году Уорреном Маккаллоком[57] и Уолтером Питтсом[58]. Она была во многом похожа на логические вентили, из которых состоят компьютеры. Вентиль ИЛИ включается, когда как минимум один из его входов включен, а вентиль И — когда включены все. Нейрон

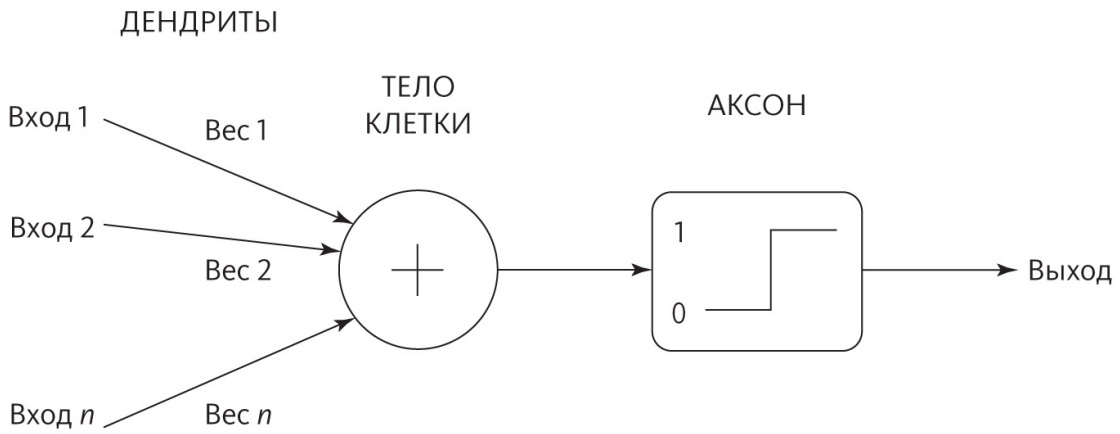
Маккаллока–Питтса включается, когда количество его активных входов превышает определенное пороговое значение. Если порог равен единице, нейрон действует как вентиль ИЛИ. Если порог равен числу входов — как вентиль И. Кроме того, один нейрон Маккаллока–Питтса может не давать включаться другому: это моделирует и ингибирующие синапсы, и вентиль НЕ. Таким образом, нейронные сети могут совершать все операции, которые умеет делать компьютер. Поначалу компьютер часто называли электронным мозгом, и это была не просто аналогия.

Однако нейрон Маккаллока–Питтса не умеет учиться. Для этого соединениям между нейронами надо присвоить переменный вес, и в результате получится так называемый перцептрон. Перцептроны были изобретены в конце 1950-х Фрэнком Розенблаттом[59], психологом из Корнелльского университета. Харизматичный оратор и очень живой человек, Розенблатт сделал для зарождения машинного обучения больше, чем кто бы то ни было. Своим названием перцептроны обязаны его интересу к применению своих моделей в проблемах восприятия (перцепции), например распознавания речи и символов. Вместо того чтобы внедрить перцептроны в компьютерные программы, которые в те дни были очень медлительными, Розенблатт построил собственные устройства: вес был представлен в них в виде переменных резисторов, как те, что стоят в переключателях с регулируемой яркостью, а для взвешенного обучения использовались электромоторы, которые крутили ручки резисторов. (Как вам такие высокие технологии?)

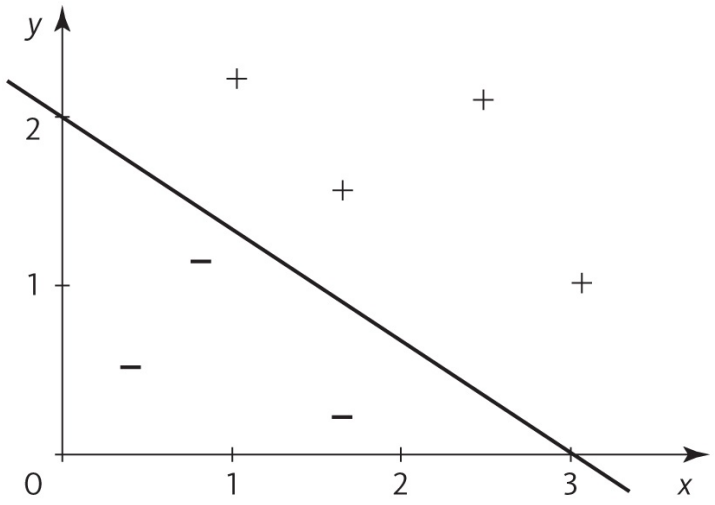
В перцептроне положительный вес представляет возбуждающее соединение, а отрицательный — ингибирующее. Если взвешенная сумма входов перцептрона выше порогового значения, он выдает единицу, а если ниже — ноль. Путем варьирования весов и порогов можно изменить функцию, которую вычисляет перцептрон. Конечно, много подробностей работы нейронов игнорируется, но ведь мы хотим все максимально упростить, и наша цель — не построить реалистичную модель мозга, а разработать обучающийся алгоритм широкого применения. Если какие-то из проигнорированных деталей окажутся важными, их всегда можно будет добавить. Несмотря на все упрощения и абстрактность,

можно заметить, что каждый элемент этой модели соответствует элементу нейрона:

Чем больше вес входа, тем сильнее соответствующий синапс. Тело клетки складывает все взвешенные входы, а аксон применяет к результату ступенчатую функцию. На рисунке в рамке аксона показан график ступенчатой функции: ноль для низких значений входа резко переходит в единицу, когда вход достигает порогового значения.



Представьте, что у перцептрона есть два непрерывных входа *x* и *y* (это значит, что *x* и *y* могут принимать любые числовые значения, а не только 0 и 1). В таком случае каждый пример можно представить в виде точки на плоскости, а границей между положительными (для которых перцептрон выдает 1) и отрицательными (выход 0) примерами будет прямая линия:



Дело в том, что граница — это ряд точек, в которых взвешенная сумма точно соответствует пороговому значению, а взвешенная сумма — линейная функция. Например, если вес x — 2, вес y — 3, а порог — 6, граница будет задана уравнением $2x + 3 = 6$. Точка $x = 0$, $y = 2$ лежит на границе, и, чтобы удержаться на ней, нам надо делать три шага вперед для каждых двух шагов вниз: тогда прирост x восполнит уменьшение y. Полученные в результате точки образуют прямую.

Нахождение весов перцептрона подразумевает варьирование направления прямой до тех пор, пока с одной стороны не окажутся все положительные примеры, а с другой — все отрицательные. В одном измерении граница — это точка, в двух измерениях — прямая, в трех — плоскость, а если измерений больше трех — гиперплоскость. Визуализировать что-то в гиперпространстве сложно, однако математика в нем работает точно так же: в n измерениях у нас будет n входов, а у перцептрона — n весов. Чтобы решить, срабатывает перцептрон или нет, надо умножить каждый вес на значение соответствующего входного сигнала и сравнить их общую сумму с пороговым значением.

Если веса всех входов равны единице, а порог — это половина числа входов, перцептрон сработает в случае, если срабатывает больше половины входов. Иными словами, перцептрон похож на крохотный парламент, в котором побеждает большинство (хотя, наверное, не такой уж и крохотный, учитывая, что в нем могут быть тысячи членов). Но при этом парламент не совсем демократический, поскольку в целом не все имеют равное право голоса. Нейронная сеть в этом отношении больше похожа на Facebook, потому что несколько близких друзей стоят тысячи френдов, — именно им вы больше всего доверяете, и они больше всего на вас влияют. Если друг порекомендует вам фильм, вы посмотрите его и вам понравится, в следующий раз вы, вероятно, снова последуете его совету. С другой стороны, если подруга постоянно восторгается фильмами, которые не доставляют вам никакого удовольствия, вы начнете игнорировать ее мнение (и не исключено, что дружба поостынет).

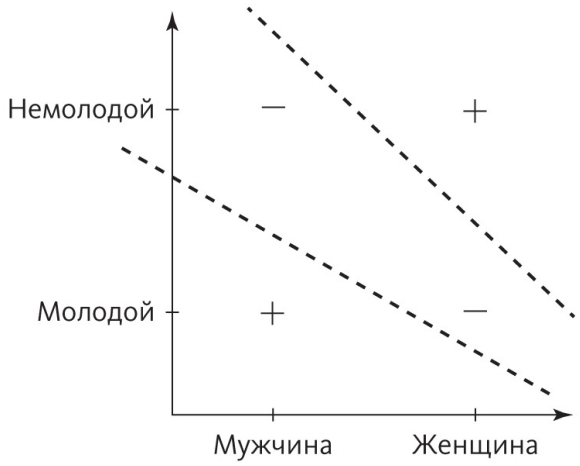
Именно так алгоритм перцептрона Розенблатта узнает вес входов.

Давайте рассмотрим «бабушкину клетку», излюбленный мысленный эксперимент когнитивных нейробиологов. «Бабушкина клетка» — это нейрон в вашем мозге, который возбуждается тогда и только тогда, когда вы видите свою бабушку. Есть ли такая клетка на самом деле — вопрос открытый, но давайте изобретем ее специально для машинного обучения. Перцептрон учится узнавать бабушку следующим образом. Входные сигналы для этой клетки — либо необработанные пиксели, либо различные жестко прошитые свойства изображения, например *карие глаза*: вход будет равен 1, если на изображении есть карие глаза, и 0 — если нет. Вначале вес всех соединений, ведущих от свойств к нейронам, маленький и произвольный, как у синапсов в мозге новорожденного. Затем мы показываем перцептрону ряд картинок: на одних есть ваша бабушка, а на других нет. Если перцептрон срабатывает при виде бабушки или не срабатывает, когда видит кого-то еще, значит, никакого обучения не нужно (не чини то, что работает). Но если перцептрон не срабатывает, когда смотрит на бабушку, это значит, что взвешенная сумма значений его входов должна быть выше и вес*а* активных входов надо увеличить (например, если бабушка кареглазая, вес этой черты повысится). И наоборот, если перцептрон срабатывает, когда не надо, вес*а* активных входов следует уменьшить. Ошибки — двигатель обучения. Со временем черты, которые указывают на бабушку, получат большой вес, а те, что не указывают, — маленький. Как только перцептрон начнет всегда срабатывать при виде вашей бабушки и ошибочные срабатывания исчезнут, обучение завершится.

Перцептрон вызвал восторг в научном сообществе. Он был простым, но при этом умел узнавать печатные буквы и звуки речи: для этого требовалось только обучение на примерах. Коллега Розенблатта по Корнелльскому университету доказал: если положительные и отрицательные примеры можно разделить гиперплоскостью, перцептрон эту плоскость найдет. Розенблатту и другим ученым казалось вполне достижимым истинное понимание принципов, по которым учится мозг, а с ним — мощный многоцелевой обучающийся алгоритм.

Но затем перцептрон уперся в стену. Инженеров знаний раздражали заявления Розенблатта: они завидовали вниманию

и финансированию, которое привлекали нейронные сети в целом и перцептроны в частности. Одним из таких критиков был Марвин Минский, бывший одноклассник Розенблатта по Научной средней школе в Бронксе, руководивший к тому времени группой искусственного интеллекта в Массачусетском технологическом институте. (Любопытно, что его диссертация была посвящена нейронным сетям, но потом он в них разочаровался.) В 1969 году Минский и его коллега Сеймур Пейперт[60] опубликовали книгу Perceptrons: an Introduction to Computational Geometry[61], где подробно, один за другим описали простые вещи, которым одноименный алгоритм не в состоянии научиться. Самый простой и потому самый убийственный пример — это функция «исключающее ИЛИ» (сокращенно XOR), которая верна, если верен один, но не оба входа. Например, две самые лояльные группы покупателей продукции Nike — это, видимо, мальчики-подростки и женщины среднего возраста. Другими словами, вы, скорее всего, купите кроссовки Nike, если вы молоды XOR женщина. Молодость подходит, женский пол тоже, но не оба фактора сразу. Если вы не молоды и вы не женщина, для рекламы Nike вы тоже неперспективная цель. Проблема с XOR в том, что не существует прямой линии, способной отделить положительные примеры от отрицательных. На рисунке показаны два неподходящих кандидата:



Поскольку перцептроны могут находить только линейные границы, XOR для них недоступен, а если они неспособны даже

на это, значит, перцептрон — не лучшая модель того, как учится мозг, и неподходящий кандидат в Верховные алгоритмы.

Перцептрон моделирует только обучение отдельного нейрона. Минский и Пейперт признавали, что слои взаимосвязанных нейронов должны быть способны на большее, но не понимали, как такие слои обучить. Другие ученые тоже этого не знали. Проблема в том, что не существует четкого способа изменить вес нейронов в «скрытых» слоях, чтобы уменьшить ошибки нейронов в выходном слое. Каждый скрытый нейрон влияет на выход множеством путей, и у каждой ошибки — тысячи отцов. Кого винить? И наоборот, кого благодарить за правильный выход? Задача присвоения коэффициентов доверия появляется каждый раз, когда мы пытаемся обучить сложную модель, и представляет собой одну из центральных проблем машинного обучения.

Книга Perceptrons была пронзительно ясной, безупречной с точки зрения математики и оказала катастрофическое воздействие на машинное обучение, которое в те годы ассоциировалось в основном с нейронными сетями. Большинство исследователей (не говоря уже о спонсорах) пришли к выводу, что единственный способ построить интеллектуальную систему — это явно ее запрограммировать, поэтому в науке на 15 лет воцарилась инженерия знаний, а машинное обучение, казалось, было обречено остаться на свалке истории.

Физик делает мозг из стекла

Если об истории машинного обучения снять голливудский блокбастер, Марвин Минский был бы главным злодеем — злой королевой, которая дает Белоснежке отравленное яблоко и бросает ее в лесу (в написанном в 1988 году эссе Сеймур Пейперт даже в шутку сравнивал себя с охотником, которого королева послала в лес убить Белоснежку). Принцем же на белом коне был бы физик из Калифорнийского технологического института по имени Джон Хопфилд[62]. В 1982 году Хопфилд заметил поразительное сходство между мозгом и спиновыми стеклами — экзотическим материалом, который очень любят специалисты по статистической физике. Это открытие привело к возрождению коннекционизма, пиком

которого несколько лет спустя стало изобретение первых алгоритмов, способных решать проблему коэффициентов доверия. Кроме того, оно положило начало новой эры, в которой машинное обучение вытеснило инженерию знаний с положения доминирующей парадигмы в науке об искусственном интеллекте.

Спиновые стекла на самом деле не стекла, хотя некоторые стеклоподобные свойства у них есть. Скорее, они магнитные материалы. Каждый электрон — это крохотный магнит, так как у него есть спин[63], который может указывать «вверх» или «вниз». В таких материалах, как железо, спины электронов обычно выстраиваются в одном направлении: если электрон со спином «вниз» окружен электронами со спином «вверх», он, вероятно, перевернется. Когда большинство спинов в куске железа выстраивается, он превращается в магнит. В обычных магнитах сила взаимодействия между соседними спинами одинакова для всех пар, однако в спиновом стекле она может отличаться и даже бывает негативной, из-за чего расположенные рядом спины принимают противоположные направления. Энергия обычного магнита ниже всего, если все спины выровнены, но в спиновом стекле все не так просто: вообще говоря, нахождение состояния наименьшей энергии для спинового стекла — это NP-полная проблема, то есть к ней можно свести практически любую другую сложную проблему оптимизации. В результате спиновое стекло не обязательно приходит в состояние наименьшей энергии: оно может застрять в локальном, а не глобальном минимуме, то есть состоянии меньшей энергии, чем все состояния, в которые можно из него перейти, поменяв спин. Во многом это похоже на дождевую воду, которая стекает в озеро, а не прямо в океан.

Хопфилд заметил интересное сходство между спиновым стеклом и нейронными сетями. Спин электрона отвечает на поведение своих соседей во многом так же, как нейрон: он переворачивается вверх, если взвешенная сумма соседей превышает пороговое значение, и вниз (или не меняется), если не превышает. Вдохновленный этим фактом, Хопфилд определил тип нейронной сети, которая со временем эволюционирует таким же образом, как спиновое стекло, и постулировал, что состояния минимальной энергии для этой сети — это ее воспоминания.

Каждое такое состояние представляет собой «область притяжения» для исходных состояний, которые в нее сходятся, и благодаря этому нейронная сеть способна распознавать паттерны: например, если одно из воспоминаний — черно-белые пиксели, образующие цифру девять, а на изображении — искаженная девятка, сеть сведет ее к «идеальной» цифре и узнает. Внезапно к машинному обучению стало можно применить широкий спектр физических теорий, в эту дисциплину пошел поток статистических физиков, помогая вытащить ее из локального минимума, в котором она застряла.

Однако спиновое стекло — это все еще очень нереалистичная модель мозга. Во-первых, спиновые взаимодействия симметричны, а соединения между нейронами головного мозга — нет. Другой большой проблемой, которую модель Хопфилда игнорировала, было то, что настоящие нейроны действуют по законам статистики: они не детерминистски включаются и выключаются в зависимости от входа, а скорее включаются с большей вероятностью, но не обязательно, при повышении взвешенной суммы входов. В 1985 году исследователи Дэвид Окли, Джеффри Хинтон и Терри Сейновски заменили детерминистские нейроны в сетях Хопфилда вероятностными. Нейронная сеть получила вероятностное распределение по своим состояниям, и состояния высокой энергии стали экспоненциально менее вероятны, чем низкоэнергетические. Вероятность нахождения сети в конкретном состоянии была задана хорошо известным в термодинамике распределением Больцмана, поэтому ученые называли свою сеть машиной Больцмана.

Машина Больцмана состоит из смеси сенсорных и скрытых нейронов (аналогично, например, сетчатке глаза и мозгу) и учится путем попеременного сна и пробуждения, как человек. В разбуженном состоянии сенсорные нейроны срабатывают в соответствии с данными, а скрытые эволюционируют согласно динамике сети и сенсорным входам. Например, если сети показать изображение девятки, нейроны, соответствующие черным пикселям изображения, включатся, другие останутся выключенными, и скрытые нейроны будут произвольно включаться по распределению Больцмана для этих значений пикселей. Во время сна сенсорные и скрытые нейроны свободно блуждают, а перед рассветом нового дня машина сравнивает статистику своих

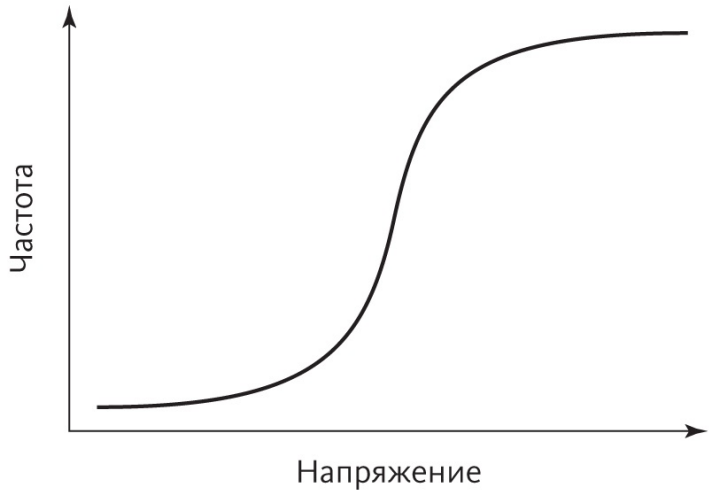
состояний во время сна и во время вчерашней активности и изменяет вес*а* связей так, чтобы согласовать эти состояния. Если в течение дня два нейрона обычно срабатывали вместе, а во сне реже, вес их соединения увеличится. Если наоборот — уменьшится. День за днем предсказанные корреляции между сенсорными нейронами эволюционируют, пока не начнут совпадать с реальными: в этот момент машина Больцмана получает хорошую модель данных, то есть проблема присвоения коэффициентов доверия эффективно решается.

Джефф Хинтон продолжил исследования и в следующие десятилетия перепробовал много вариантов машины Больцмана. Хинтон — психолог, ставший информатиком, и праправнук Джорджа Буля, изобретателя логического исчисления, используемого во всех цифровых компьютерах, — ведущий коннекционист в мире. Он дольше и упорнее других пытался разобраться, как работает мозг. Хинтон рассказывает, что как-то пришел домой с работы и возбужденно крикнул: «Есть! Я понял, как работает мозг!» На что дочь ему ответила: «Папа, опять?!» В последнее время он увлекся глубоким обучением, о котором мы поговорим дальше в этой главе, а также участвовал в разработке метода обратного распространения ошибки — более совершенного, чем машины Больцмана, алгоритма, решающего проблему присвоения кредитов доверия (об этом пойдет речь в следующей главе). Машины Больцмана могут решать эту задачу в принципе, но на практике обучение идет очень медленно и трудно, поэтому такой подход в большинстве случаев нецелесообразен. Для следующего прорыва нужно было отказаться от еще одного чрезмерного упрощения, которое восходит к Маккаллоку и Питтсу.

Самая важная кривая в мире

По отношению к соседям нейрон может быть только в одном из двух состояний — активным и неактивным. Однако здесь не хватает важного нюанса. Потенциалы действия длятся недолго: напряжение подскакивает всего на долю секунды и немедленно возвращается в состояние покоя. Этот скачок едва регистрируется принимающим нейроном: чтобы разбудить клетку, нужна череда

скачков с короткими промежутками. Обычные нейроны периодически возбуждаются и без всякой стимуляции. Когда стимуляция накапливается, нейрон возбуждается все чаще и чаще, а затем достигает насыщения — самой высокой частоты скачков напряжения, на которую он способен, после которой увеличение стимуляции не оказывает эффекта. Нейрон больше напоминает не логический вентиль, а преобразователь напряжения в частоту. Кривая зависимости частоты от напряжения выглядит следующим образом:



Эту похожую на вытянутую букву S кривую называют по-разному: логистической, S-образной, сигмоидой. Присмотритесь к ней повнимательнее, потому что это самая важная кривая в мире. Сначала выход медленно растет вместе с входом: так медленно, что кажется постоянным. Затем он начинает меняться быстрее, потом очень быстро, а после все медленнее и медленнее и наконец вновь становится почти постоянным. Кривая транзистора, которая связывает входящее и выходящее напряжение, тоже S-образна, поэтому и компьютеры, и головной мозг наполнены S-кривыми. Но это еще не все. Форму сигмоиды имеют всевозможные фазовые переходы: вероятность, что электрон сменит спин в зависимости от приложенного поля, намагничивание железа, запись бита памяти на твердый диск, открытие ионного канала в клетке, таяние льда, испарение воды, инфляционное расширение молодой Вселенной, прерывистое равновесие в эволюции, смена научных парадигм, распространение новых технологий, бегство белого населения

из смешанных районов, слухи, эпидемии, революции, падения империй и многое другое. Книгу The Tipping Point: How Little Things Can Make a Big Difference[64] можно было бы (хотя и менее заманчиво) назвать «Сигмоида». Землетрясение — это фазовый переход в относительном положении двух прилегающих тектонических плит, а стук, который мы иногда слышим ночью, — просто сдвиг микроскопических «тектонических плит» в стенах дома, так что не пугайтесь. Йозеф Шумпетер[65] говорил, что экономика развивается трещинами и скачками: творческое разрушение тоже имеет S-образную форму. Финансовые приобретения и потери тоже воздействуют на человеческое счастье по сигмоиде, поэтому не стоит излишне надрываться и переживать. Вероятность, что произвольная логическая формула будет выполнимой — самая суть NP-полных проблем, — следует фазовому переходу от почти единицы к почти нулю по мере увеличения длины формулы. Статистические физики могут изучать фазовые переходы всю жизнь.

В романе Хемингуэя «И восходит солнце» Майка Кэмпбелла спрашивают, как он обанкротился, и тот отвечает: «Двумя способами. Сначала постепенно, а потом сразу». То же самое могли бы сказать в банке Lehman Brothers. В этом суть сигмоиды. Одно из правил прогнозирования, сформулированных футуристом Полом Саффо, гласит: ищите S-образные кривые. Если не получается «поймать» комфортную температуру в душе — вода сначала слишком холодная, а потом сразу слишком горячая, — вините S-кривую. Развитие по S-образной кривой хорошо видно, когда готовишь воздушную кукурузу: сначала ничего не происходит, затем лопается несколько зерен, потом сразу много, потом почти все взрываются фейерверком, потом еще немного — и можно есть. Движения мышц тоже следуют сигмоиде: медленно, быстро и опять медленно: мультфильмы стали гораздо естественнее, когда диснеевские мультипликаторы поняли это и начали имитировать. По S-кривой движутся глаза, фиксируясь вместе с сознанием то на одном, то на другом предмете. Согласно фазовому переходу меняется настроение. То же самое с рождением, половым созреванием, влюбленностью, браком, беременностью, поступлением на работу и увольнением, переездом в другой город,

повышением по службе, выходом на пенсию и смертью. Вселенная — огромная симфония фазовых переходов, от космических до микроскопических, от самых обыденных до меняющих нашу жизнь.

Сигмоида важна не просто как модель. В математике она трудится не покладая рук. Если приблизить ее центральный отрезок, он будет близок прямой. Многие явления, которые мы считаем линейными, на самом деле представляют собой S-образные кривые, потому что ничто не может расти бесконечно. В силу относительности и вопреки Ньютону ускорение не увеличивается линейно с увеличением силы, а следует по сигмоиде, центрированной на нуле. Аналогичная картина наблюдается с зависимостью электрического тока от напряжения в резисторах электрических цепей и в лампочках (пока нить не расплавится, что само по себе очередной фазовый переход). Если посмотреть на S-образную кривую издалека, она будет напоминать ступенчатую функцию, в которой выход в пороговом значении внезапно меняется с нуля до единицы. Поэтому, в зависимости от входящего напряжения, работу транзистора в цифровых компьютерах и аналоговых устройствах, например усилителях и тюнерах, будет описывать та же самая кривая. Начальный отрезок сигмоиды по существу экспоненциальный, а рядом с точкой насыщения она приближается к затуханию по экспоненте. Когда кто-то говорит об экспоненциальном росте, спросите себя: как скоро он перейдет в S-образную кривую? Когда замедлится взрывной рост населения, закон Мура исчерпает свои возможности, а сингулярность так и не наступит? Дифференцируйте сигмоиду, и вы получите гауссову кривую: «медленно — быстро — медленно» превратится в «низко — высоко — низко». Добавьте последовательность ступенчатых S-образных кривых, идущих то вверх, то вниз, и получится что-то близкое к синусоиде. На самом деле каждую функцию можно близко аппроксимировать суммой S-образных кривых: когда функция идет вверх, вы добавляете сигмоиду, когда вниз — отнимаете. Обучение ребенка — это не постепенное улучшение, а накопление S-образных кривых. Это относится и к технологическим изменениям. Взгляните на Нью-Йорк издали, и вы увидите, как вдоль горизонта

разворачивается совокупность сигмоид, острых, как углы небоскребов.

Для нас самое главное то, что S-образные кривые ведут к новому решению проблемы коэффициентов доверия. Раз Вселенная — это симфония фазовых переходов, давайте смоделируем ее с помощью фазового перехода. Именно так поступает головной мозг: подстраивает систему фазовых переходов внутри к аналогичной системе снаружи. Итак, давайте заменим ступенчатую функцию перцептрона сигмоидой и посмотрим, что произойдет.

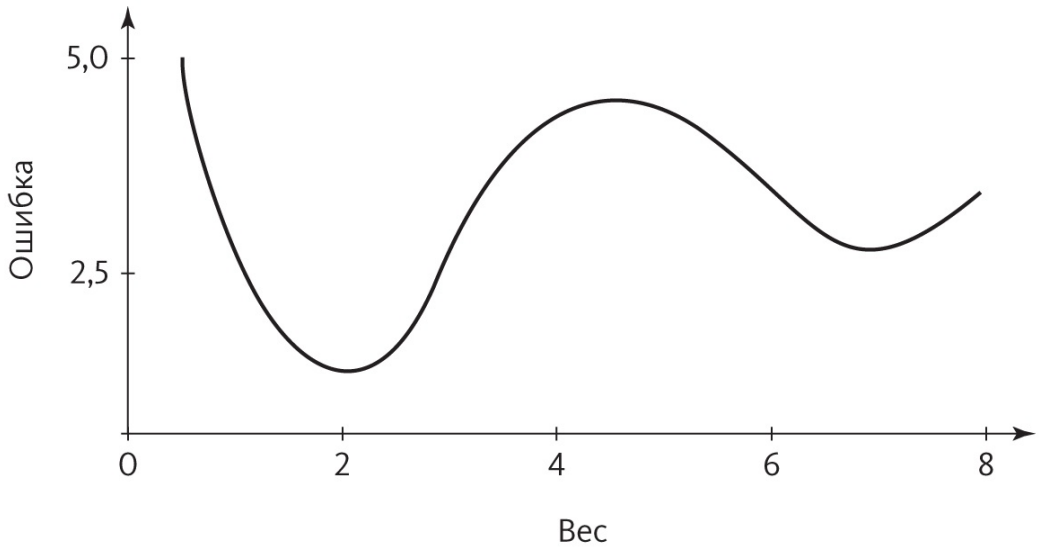
Альпинизм в гиперпространстве

В алгоритме перцептрона сигнал ошибки действует по принципу «все или ничего»: либо правильно, либо неправильно. Негусто, особенно в случае сетей из многих нейронов. Можно понять, что ошибся нейрон на выходе (ой, это была не ваша бабушка?), но как насчет какого-то нейрона в глубинах мозга? И вообще, что значат правота и ошибка для глубинного нейрона? Однако если выход нейрона непрерывный, а не бинарный, картина меняется. Прежде всего мы можем оценить, *насколько* ошибается выходной нейрон, по разнице между получаемым и желаемым выходом. Если нейрон должен искрить активностью («Ой, бабушка! Привет!») и он немного активен, это лучше, чем если бы он не срабатывал вовсе. Еще важнее то, что теперь можно распространить эту ошибку на скрытые нейроны: если выходной нейрон должен быть активнее и с ним связан нейрон *A*, то чем более активен нейрон *A*, тем больше мы должны усилить соединение между ними. Если *A* подавляется нейроном *B*, то *B* должен быть менее активным и так далее. Благодаря обратной связи от всех нейронов, с которыми он связан, каждый нейрон решает, насколько больше или меньше надо активироваться. Это, а также активность *его собственных* входных нейронов диктует ему, усиливать или ослаблять соединения с ними. Мне надо быть активнее, а нейрон *B* меня подавляет? Значит, его вес надо снизить. А нейрон *C* очень активен, но его соединение со мной слабое? Усилим его. В следующем раунде нейроны-«клиенты», расположенные дальше в сети, подскажут, насколько хорошо я справился с задачей.

Всякий раз, когда «сетчатка» обучающегося алгоритма видит новый образ, сигнал распространяется по всей сети, пока не даст выход. Сравнение полученного выхода с желаемым выдает сигнал ошибки, который затем распространяется обратно через все слои и достигает сетчатки. На основе возвращающегося сигнала и вводных, полученных во время прохождения вперед, каждый нейрон корректирует вес *a*. По мере того как сеть видит все новые и новые изображения вашей бабушки и других людей, веса постепенно сходятся со значениями, которые позволяют отличить одно от другого. Метод обратного распространения ошибки, как называется этот алгоритм, несравнимо мощнее перцептрона. Единичный нейрон может найти только прямую линию, а так называемый многослойный перцептрон — произвольно запутанные границы, при условии что у него есть достаточно скрытых нейронов. Это делает обратное распространение ошибки верховным алгоритмом коннекционистов.

Обратное распространение — частный случай стратегии, очень распространенной в природе и в технологии: если вам надо быстро забраться на гору, выбирайте самый крутой склон, который только найдете. Технический термин для этого явления — «градиентное восхождение» (если вы хотите попасть на вершину) или «градиентный спуск» (если смотреть на долину внизу). Бактерии умеют искать пищу, перемещаясь согласно градиенту концентрации, скажем, глюкозы, и убегать от ядов, двигаясь против их градиента. С помощью градиентного спуска можно оптимизировать массу вещей, от крыльев самолетов до антенных систем. Обратное распространение — эффективный способ такой оптимизации в многослойном перцептроне: продолжайте корректировать вес *a*, чтобы снизить возможность ошибки, и остановитесь, когда станет очевидно, что корректировки ничего не дают. В случае обратного распространения не надо разбираться, как с нуля корректировать вес каждого нейрона (это было бы слишком медленно): это можно делать слой за слоем, настраивая каждый нейрон на основе уже настроенных, с которыми он соединен. Если в чрезвычайной ситуации вам придется выбросить весь инструментарий машинного обучения и спасти что-то одно, вы, вероятно, решите спасти градиентный спуск.

Так как же обратное распространение решает проблему машинного обучения? Может быть, надо просто собрать кучу нейронов, подождать, пока они наколдуют все, что надо, а потом по дороге в банк заехать получить Нобелевскую премию за открытие принципа работы мозга? К сожалению, в жизни все не так просто. Представьте, что у вашей сети только один вес; зависимость ошибки от него показана на этом графике:



Оптимальный вес, в котором ошибка самая низкая, — это 2,0. Если сеть начнет работу, например, с 0,75, обратное распространение ошибки за несколько шагов придет к оптимуму, как катящийся с горки мячик. Однако если начать с 5,5, мы скатимся к весу 7,0 и застрянем там. Обратное распространение ошибки со своими поэтапными изменениями весов не сможет найти глобальный минимум ошибки, а локальные минимумы могут быть сколь угодно плохими: например, бабушку можно перепутать со шляпой. Если вес всего один, можно перепробовать все возможные значения с шагом 0,01 и таким образом найти оптимум. Но когда весов тысячи, не говоря уже о миллионах или миллиардах, это не вариант, потому что число точек на сетке будет увеличиваться экспоненциально с числом весов. Глобальный минимум окажется скрыт где-то в бездонных глубинах гиперпространства — ищи иголку в стоге сена.

Представьте, что вас похитили, завязали глаза и бросили где-то в Гималаях. Голова раскалывается, с памятью не очень, но вы твердо знаете, что надо забраться на вершину Эвереста. Как быть? Вы делаете шаг вперед и едва не скатываетесь в ущелье. Переведя дух, вы решаете действовать систематичнее и осторожно ощупываете ногой почву вокруг, чтобы определить самую высокую точку. Затем вы робко шагаете к ней, и все повторяется. Понемногу вы забираетесь все выше и выше. Через какое-то время любой шаг начинает вести вниз, и вы останавливаетесь. Это градиентное восхождение. Если бы в Гималаях существовал один Эверест, причем идеальной конической формы, все было бы прекрасно. Но, скорее всего, место, где все шаги ведут вниз, будет все еще очень далеко от вершины: вы просто застрянете на каком-нибудь холме у подножья. Именно это происходит с обратным распространением ошибки, только на горы оно взбирается в гиперпространстве, а не в трехмерном пространстве, как наше. Если ваша сеть состоит из одного нейрона и вы будете шаг за шагом подниматься к наилучшим весам, то придете к вершине. Но в многослойном перцептроне ландшафт очень изрезанный — поди найди высочайший пик.

Отчасти поэтому Минский, Пейперт и другие исследователи не понимали, как можно обучать многослойные перцептроны. Они могли представить себе замену ступенчатых функций S-образными кривыми и градиентный спуск, но затем сталкивались с проблемой локальных минимумов ошибки. В то время ученые не доверяли компьютерным симуляциям и требовали математических доказательств работоспособности алгоритма, а для обратного распространения ошибки такого доказательства не было. Но, как мы уже видели, в большинстве случаев локального минимума достаточно. Поверхность ошибки часто похожа на дикобраза: много крутых пиков и впадин, и на самом деле неважно, найдем ли мы самую глубокую, абсолютную впадину — сойдет любая. Еще лучше то, что локальный минимум бывает даже предпочтительнее, потому что он меньше подвержен переобучению, чем глобальный.

Гиперпространство — обоюдоострый меч. С одной стороны, чем больше количество измерений, тем больше места для очень сложных поверхностей и локальных экстремумов. С другой

стороны, чтобы застрять в локальном экстремуме, надо застрять во *всех* измерениях, а во многих одновременно застрять сложнее, чем в трех. В гиперпространстве есть перевалы, проходящие через всю (гипер)местность, поэтому с небольшой помощью со стороны человека обратное распространение ошибки зачастую способно найти путь к идеально хорошему набору весов. Может быть, это не уровень моря, а только легендарная долина Шангри-Ла, но на что жаловаться, если в гиперпространстве миллионы таких долин и к каждой ведут миллиарды перевалов?

Тем не менее придавать слишком большое значение весам, которые находит обратное распространение ошибки, не стоит. Помните, что есть, вероятно, много очень разных, но одинаково хороших вариантов. Обучение многослойного перцептрона хаотично в том смысле, что, начав из слегка отличающихся мест, он может привести к весьма различным решениям. Этот феномен проявляется в случае незначительных отличий как в исходных весах, так и в обучающих данных и имеет место во всех мощных обучающихся алгоритмах, а не только в обратном распространении ошибки.

Мы *могли бы* избавиться от проблемы локальных экстремумов, убрав наши сигмоиды и позволив каждому нейрону просто выдавать взвешенную сумму своих входов. Поверхность ошибки стала бы в этом случае очень гладкой, и остался бы всего один минимум — глобальный. Дело, однако, в том, что линейная функция линейных функций — по-прежнему линейная функция, поэтому сеть линейных нейронов ничем не лучше, чем единичный нейрон. Линейный мозг, каким бы большим он ни был, будет глупее червяка. S-образные кривые — просто хороший перевалочный пункт между глупостью линейных функций и сложностью ступенчатых функций.

Перцептроны наносят ответный удар

Метод обратного распространения ошибки был изобретен в 1986 году Дэвидом Румельхартом, психологом из Калифорнийского университета в Сан-Диего, в сотрудничестве с Джеффом Хинтоном и Рональдом Уильямсом[66]. Они доказали,

кроме всего прочего, что обратное распространение способно справиться с исключающим ИЛИ, и тем самым дали коннекционистам возможность показать язык Минскому и Пейперту. Вспомните пример с кроссовками Nike: подростки и женщины среднего возраста — их наиболее вероятные покупатели. Это можно представить с помощью сети из трех нейронов: один срабатывает, когда видит подростка, другой — женщину среднего возраста, а третий — когда активизируются оба. Благодаря обратному распространению ошибки можно узнать соответствующие веса и получить успешный детектор предполагаемых покупателей Nike. (Вот так-то, Марвин.)

В первых демонстрациях мощи обратного распространения Терри Сейновски и Чарльз Розенберг обучали многослойный перцептрон читать вслух. Их система NETtalk сканировала текст, подбирала фонемы согласно контексту и передавала их в синтезатор речи. NETtalk не только делал правильные обобщения для новых слов, чего не умели системы, основанные на знаниях, но и научился говорить очень похоже на человека. Сейновски любил очаровывать публику на научных мероприятиях, пуская запись обучения NETtalk: сначала лепет, затем что-то более внятное и наконец вполне гладкая речь с отдельными ошибками. (Поищите примеры на YouTube по запросу sejnowski nettalk.)

Первым большим успехом нейронных сетей стало прогнозирование на фондовой бирже. Поскольку сети умеют выявлять маленькие нелинейности в очень зашумленных данных, они приобрели популярность и вытеснили распространенные в финансах линейные модели. Типичный инвестиционный фонд тренирует сети для каждой из многочисленных ценных бумаг, затем позволяет выбрать самые многообещающие, после чего люди-аналитики решают, в какую из них инвестировать. Однако ряд фондов пошел до конца и разрешил алгоритмам машинного обучения осуществлять покупки и продажи самостоятельно. Сколько именно из них преуспело — тайна за семью печатями, но, поскольку специалисты по обучающимся алгоритмам в устрашающем темпе исчезают в недрах хеджевых фондов, вероятно, в этом что-то есть.

Нелинейные модели важны далеко не только на фондовой бирже. Ученые повсеместно используют линейную регрессию, потому что хорошо ее знают, но изучаемые явления чаще нелинейные, и многослойный перцептрон умеет их моделировать. Линейные модели не видят фазовых переходов, а нейронные сети впитывают их как губка.

Другим заметным успехом ранних нейронных сетей стало обучение вождению машины. Беспилотные автомобили впервые привлекли всеобщее внимание на соревнованиях DARPA Grand Challenge[67] в 2004-м и 2005 годах, но за десять с лишним лет до этого ученые Университета Карнеги–Меллон успешно обучили многослойный перцептрон водить машину: узнавать дорогу на видео и поворачивать руль в нужном месте. С небольшой помощью человека — второго пилота — этот автомобиль сумел проехать через все Соединенные Штаты от океана до океана, хотя «зрение» у него было очень мутное (30 × 32 пикселя), а мозг меньше, чем у червяка. (Проект назвали No Hands Across America.) Может быть, это не была первая по-настоящему беспилотная машина, но даже она выгодно отличалась от большинства подростков за рулем.

У метода обратного распространения ошибки несметное количество применений. По мере того как росла его слава, становилось все больше известно о его истории. Оказалось, что, как это часто бывает в науке, метод изобретали несколько раз: французский информатик Ян Лекун и другие ученые наткнулись на него примерно в то же время, что и Румельхарт. Еще в 1980-е годы сообщение о методе обратного распространения отклонили на ведущей конференции по проблемам искусственного интеллекта, потому что, по мнению рецензентов, Минский и Пейперт доказали, что перцептроны не работают. Вообще говоря, Румельхарт считается изобретателем метода скорее по «тесту Колумба»: Колумб не был первым человеком, который открыл Америку, но он был последним. Оказалось, что Пол Вербос, аспирант Гарвардского университета, предложил схожий алгоритм в своей диссертации в 1974 году, а самая большая ирония в том, что Артур Брайсон и Хэ Юци, специалисты по теории управления, добились этого в 1969 году — именно когда Минский и Пейперт

публиковали свою книгу Perceptrons! Так что сама история машинного обучения показывает, зачем нам нужны обучающиеся алгоритмы: если бы алгоритмы автоматически выявили, что статьи по теме есть в научной литературе с 1969 года, мы бы не потратили впустую десятилетия, и кто знает, какие открытия были бы сделаны быстрее.

В истории перцептрона много иронии, но печально то, что Фрэнк Розенблатт так и не увидел второго акта своего творения: он утонул в Чесапикском заливе в том же 1969 году.

Полная модель клетки

Живая клетка — прекрасный пример нелинейной системы. Она выполняет все свои функции благодаря сложной сети химических реакций, превращающих сырье в конечные продукты. Как мы видели в предыдущей главе, структуру этой сети можно открыть символистскими методами, например обратной дедукцией, но для построения полной модели работы клетки нужен количественный подход: надо узнать параметры, которые связывают уровень экспрессии различных генов, соотносят переменные окружающей среды с внутренними переменными и так далее. Это непросто, потому что между этими величинами нет простой линейной зависимости. Свою стабильность клетка скорее поддерживает благодаря пересекающимся петлям обратной связи, и ее поведение очень сложно. Для решения этой проблемы хорошо подходит метод обратного распространения ошибки, который способен эффективно учиться нелинейным функциям. Если бы у нас в руках была полная карта метаболических цепочек и мы располагали достаточными данными наблюдений за всеми соответствующими переменными, обратное распространение теоретически могло бы получить подробную модель клетки и многослойный перцептрон предсказывал бы любую переменную как функцию ее непосредственных причин.

Однако в обозримом будущем у нас будет только частичное понимание клеточного метаболизма и мы сможем наблюдать лишь долю нужных параметров. Для получения полезных моделей в условиях недостатка информации и неизбежных противоречий

нужны байесовские методы, в которые мы погрузимся в главе 6. То же касается прогнозов для конкретного пациента, если модель уже имеется: байесовский вывод извлечет максимум из неизбежно неполной и зашумленной картины. Хорошо то, что для лечения рака не обязательно понимать функционирование опухолевых клеток полностью и во всех подробностях: достаточно просто обезвредить их, не повреждая нормальные клетки. В главе 6 мы увидим, как правильно сориентировать обучение, обходя то, чего мы не знаем и не обязательно должны знать.

На нынешнем этапе нам известно, что на основе данных и предыдущего знания можно с помощью обратной дедукции сделать вывод о структуре клеточных сетей, однако количество способов его применения порождает комбинаторный взрыв, так что требуется какая-то стратегия. Поскольку метаболические сети были разработаны эволюцией, возможно, симулирование эволюции в обучающихся алгоритмах как раз подойдет. В следующей главе мы посмотрим, как это сделать.

В глубинах мозга

Когда метод обратного распространения ошибки «пошел в народ», коннекционисты рисовали в воображении быстрое обучение все больших и больших сетей до тех пор, пока, если позволит «железо», они не сравняются с искусственным мозгом. Оказалось, все не так. Обучение сетей с одним скрытым слоем проходило хорошо, но после этого все резко усложнялось. Сети с несколькими слоями работали только в случае, если их тщательно разрабатывали под конкретное применение (скажем, распознавание символов), а за пределами этих рамок метод обратного распространения терпел неудачу. По мере добавления слоев сигнал ошибки расходился все больше и больше, как река, ветвящаяся на мелкие протоки вплоть до отдельных незаметных капелек. Обучение с десятками и сотнями скрытых слоев, как в мозге, оставалось отдаленной мечтой, и к середине 1990-х восторги по поводу многослойных перцептронов поутихли. Стойкое ядро коннекционистов не сдавалось, но в целом внимание

переместилось в другие области машинного обучения (мы увидим их в главах 6 и 7).

Однако сегодня коннекционизм возрождается. Мы обучаем более глубокие сети, чем когда бы то ни было, и они задают новые стандарты в зрении, распознавании речи, разработке лекарственных средств и других сферах. Новая область — глубокое обучение — появилась даже на первой странице New York Times, но, если заглянуть под капот, мы с удивлением увидим, что там гудит все тот же старый добрый двигатель — метод обратного распространения ошибки. Что изменилось? В общем-то, ничего нового, скажут критики: просто компьютеры сделались быстрее, а данных стало больше. На это Хинтон и другие ответят: «Вот именно! Мы были совершенно правы!»

По правде говоря, коннекционисты добились больших успехов. Одним из героев последнего взлета на американских горках коннекционизма стало непритязательное маленькое устройство под названием автокодировщик — многослойный перцептрон, который на выходе выдает то же, что получил на входе. Он получает изображение вашей бабушки и выдает ту же самую картинку. На первый взгляд это может показаться дурацкой затеей: где вообще можно применить эту штуку? Но вся суть в том, чтобы скрытый слой был намного меньше, чем входной и выходной, то есть чтобы сеть не могла просто научиться копировать вход в скрытый слой, а скрытый слой — в выходной, потому что в таком случае устройство вообще никуда не годится. Однако если скрытый слой маленький, происходит интересная вещь: сеть вынуждена кодировать вход всего несколькими битами, чтобы представить его в скрытом слое, а затем эти биты декодируются обратно до полного размера. Система может, например, научиться кодировать состоящее из миллиона пикселей изображение бабушки всего лишь семью буквами — словом «бабушка» — или каким-то коротким кодом собственного изобретения и одновременно научиться раскодировать это слово в картинку милой вашему сердцу пенсионерки. Таким образом, автокодировщик похож на инструмент для сжатия файлов, но имеет два преимущества: сам разбирается, как надо сжимать, и, как сети Хопфилда, умеет

превращать зашумленное, искаженное изображение в хорошее и чистое.

Автокодировщики были известны еще в 1980-х, но тогда их было очень сложно учить, несмотря на всего один скрытый слой. Разобраться, как упаковать большой объем информации в горсть битов, — чертовски сложная проблема (один код для вашей бабушки, немного другой — для дедушки, еще один — для Дженнифер Энистон[68] и так далее): ландшафт гиперпространства слишком изрезан, чтобы забраться на хороший пик, а скрытые элементы должны узнать, из чего складывается избыток исключающих ИЛИ на входе. Из-за этих проблем автокодировщики тогда по-настоящему не привились. Чтобы преодолеть сложности, потребовалось больше десятилетия. Был придуман следующий трюк: скрытый слой надо сделать больше, чем входной и выходной. Что это даст? На самом деле это только половина решения: вторая часть — заставить все, кроме некоторого количества скрытых единиц, быть выключенными в данный момент. Это все еще не позволяет скрытому слою просто копировать вход и, что самое главное, сильно облегчает обучение. Если мы позволим разным битам представлять разные входы, входы перестанут конкурировать за настройку одних и тех же битов. Кроме того, у сети появится намного больше параметров, поэтому у гиперпространства будет намного больше измерений, а следовательно, и способов выбраться из того, что могло бы стать локальными максимумами. Этот изящный трюк называется разреженным автокодировщиком.

Однако по-настоящему глубокого обучения мы пока не видели. Следующая хитрая идея — поставить разреженные автокодировщики друг на друга, как большой сэндвич. Скрытый слой первого становится входом/выходом для второго и так далее. Поскольку нейроны нелинейные, каждый скрытый слой учится более сложным представлениям входа, основываясь на предыдущем. Если имеется большой набор изображений лиц, первый автокодировщик научится кодировать мелкие элементы, например уголки и точки, второй использует это для кодирования черт лица, например кончика носа и радужки глаза, третий займется целыми носами и глазами и так далее. Наконец, верхний

слой может быть традиционным перцептроном — он научится узнавать вашу бабушку по чертам высокого уровня, которые дает нижележащий слой. Это намного легче, чем использовать только сырые данные одного скрытого слоя или пытаться провести обратное распространение сразу через все слои. Сеть Google Brain, прорекламированная New York Times, представляет собой бутерброд из девяти слоев автокодировщиков и других ингредиентов, который учится узнавать кошек на видеороликах на YouTube. На тот момент эта сеть была крупнейшей, которую когда-либо обучали: в ней был миллиард соединений. Неудивительно, что Эндрю Ын, один из руководителей проекта, — горячий сторонник идеи, что человеческий разум сводится к одному алгоритму и достаточно просто его найти. Ын, за обходительными манерами которого скрывается невероятная амбициозность, убежден, что многоярусные разреженные автокодировщики могут привести нас ближе к разгадке искусственного интеллекта, чем все, что мы имели раньше.

Многоярусные автокодировщики — не единственная разновидность глубоких обучающихся алгоритмов. Еще одна основана на машинах Больцмана, встречаются модели зрительной коры на сверточных нейронных сетях. Однако, несмотря на замечательные успехи, все это пока еще очень далеко от головного мозга. Сеть Google умеет распознавать кошачьи мордочки только анфас, а человек узнает кота в любой позе, даже если тот вообще отвернется. Кроме того, сеть Google все еще довольно мелкая: автокодировщики составляют всего три из девяти ее слоев. Многослойный перцептрон — удовлетворительная модель мозжечка — части мозга, ответственной за низкоуровневый контроль движений. Однако кора головного мозга — совсем другое дело. В ней нет, например, обратных связей, необходимых для распространения ошибки, и тем не менее именно в коре происходит настоящее волшебство обучения. В своей книге On Intelligence («Об интеллекте») Джефф Хокинс отстаивает разработку алгоритмов, основанных на близком воспроизведении строения коры головного мозга, но ни один из этих алгоритмов пока не может соперничать с сегодняшними глубокими сетями.

По мере того как мы будем лучше понимать мозг, ситуация может измениться. Вдохновленная проектом «Геном человека», новая дисциплина — коннектомика — стремится составить карту всех мозговых синапсов. В построение полноценной модели Евросоюз вложил миллиарды евро, а американская программа BRAIN, имеющая схожие цели, только в 2014 году получила 100 миллионов долларов финансирования. Тем не менее символисты очень скептически смотрят на этот путь к Верховному алгоритму. Даже если мы будем представлять себе весь мозг на уровне отдельных синапсов, понадобятся (какая ирония) более совершенные алгоритмы машинного обучения, чтобы превратить эту картину в монтажные схемы: о том, чтобы сделать это вручную, не может быть и речи. Хуже то, что, даже получив полную карту головного мозга, мы все еще будем теряться в догадках, как он работает. Нервная система червя Caenorhabditis elegans, состоящая всего из 302 нейронов, была полностью картирована еще в 1986 году, однако мы по-прежнему понимаем ее работу лишь фрагментарно. Чтобы что-то понять в болоте мелких деталей и «выполоть» специфичные для человека подробности и просто причуды эволюции, нужны более высокоуровневые концепции. Мы не строим самолеты путем обратной инженерии птичьих перьев, и самолеты не машут крыльями, однако в основе конструкции самолета лежат принципы аэродинамики, единые для всех летающих объектов. Аналогичных принципов мышления мы все еще не имеем.

Может быть, коннектомика впадает в крайности: некоторые коннекционисты, по слухам, утверждают, что метод обратного распространения и есть Верховный алгоритм: надо просто увеличить масштаб. Но символисты высмеивают эти взгляды и предъявляют длинный перечень того, что люди делать умеют, а нейронные сети — нет. Взять хотя бы «здравый смысл», требующий соединять фрагменты информации, до этого, может быть, никогда и рядом не стоявшие. Ест ли Мария на обед ботинки? Не ест, потому что она человек, люди едят только съедобные вещи, а ботинки несъедобные. Символические системы справляются с этим без проблем — они просто составляют цепочки соответствующих правил, — а многослойные перцептроны этого

делать не умеют и, обучившись, будут раз за разом вычислять одну и ту же фиксированную функцию. Нейронные сети — не композиционные, а композиционность — существенный элемент человеческого познания. Еще одна большая проблема в том, что и люди, и символические модели, например наборы правил и деревья решений, способны объяснять ход своих рассуждений, в то время как нейронные сети — большие горы чисел, которые никто не может понять.

Но если у человека есть все эти способности и мозг не выучивает их путем подбора синапсов, откуда они берутся? Вы не верите в волшебство? Тогда ответ — «эволюция». Убежденный критик коннекционизма просто обязан разобраться, откуда эволюция узнала все, что ребенок знает при рождении, — и чем больше мы списываем на врожденные навыки, тем труднее задача. Если получится все это понять и запрограммировать компьютер выполнять такую задачу, будет очень неучтиво отказывать вам в лаврах изобретателя Верховного алгоритма — по крайней мере, одного из его вариантов.

ГЛАВА 5

ЭВОЛЮЦИЯ: ОБУЧАЮЩИЙСЯ АЛГОРИТМ ПРИРОДЫ

Перед вами Robotic Park — огромная фабрика по производству роботов. Вокруг нее — тысяча квадратных миль джунглей, каменных и не очень. Джунгли окружает самая высокая и толстая в мире стена, утыканная наблюдательными вышками, прожекторами и орудийными гнездами. У стены две задачи: не пустить на фабрику нарушителей и не выпустить ее обитателей — миллионы роботов, сражающихся за выживание и власть. Роботы-победители размножаются путем доступа к программированию 3D-принтеров. Шаг за шагом роботы становятся умнее, быстрее и смертоноснее. Robotic Park принадлежит Армии США и призван путем эволюции вывести совершенного солдата.

Пока такой фабрики не существует, но однажды она может появиться. Несколько лет назад на мастер-классе DARPA я предложил эту идею в рамках мысленного эксперимента, и один из присутствующих в зале высших чинов сухо заметил: «Да, это реализуемо». Его решимость будет выглядеть не такой пугающей, если вспомнить, что для обучения своих подразделений американская армия построила в калифорнийской пустыне полноценный макет афганской деревни вместе с жителями, так что несколько миллиардов долларов — небольшая цена за идеального бойца.

Первые шаги в этом направлении уже сделаны. В лаборатории Creative Machines Lab в Корнелльском университете, которой руководит Ход Липсон, роботы причудливых форм учатся плавать и летать — возможно, прямо сейчас, когда вы читаете эти строки. Один из них похож на ползающую башню из резиновых блоков, другой — на вертолет со стрекозиными крыльями, еще один — на меняющий форму конструктор Tinkertoy. Эти роботы созданы не инженерами, а эволюцией — тем самым процессом, который породил разнообразие жизни на Земле. Изначально роботы

эволюционируют внутри компьютерной симуляции, но, как только они становятся достаточно перспективными, чтобы выйти в реальный мир, их автоматически печатают на 3D-принтере. Творения Липсона пока не готовы захватить мир, но они уже далеко ушли от первобытного набора элементов в компьютерной программе, в которой они родились.

Алгоритм, обеспечивший эволюцию этих роботов, изобрел в XIX веке Чарльз Дарвин. В то время он не воспринимал эволюцию как алгоритм, отчасти потому, что в ней недоставало ключевой подпрограммы. Как только Джеймс Уотсон и Фрэнсис Крик[69] в 1953 году открыли ее, все было готово для второго пришествия: эволюция *in silico* вместо *in vivo*[70], происходящая в миллиард раз быстрее. Ее пророком стал Джон Холланд — румяный улыбчивый парень со Среднего Запада[71].

Алгоритм Дарвина

Как и многие другие ученые, работавшие над ранними этапами машинного обучения, Холланд начинал с нейронных сетей, но, после того как он — тогда еще студент Мичиганского университета — прочитал классический трактат Рональда Фишера The Genetical Theory of Natural Selection[72], его интересы приобрели другое направление. В своей книге Фишер, который также был основателем современной статистики, сформулировал первую математическую теорию эволюции. Теория Фишера была блестящей, но Холланд чувствовал, что в ней не хватает самой сути эволюции: автор рассматривал каждый ген изолированно, а ведь приспособленность организма — комплексная функция всех его генов. Если бы гены были независимы, частотность их вариантов очень быстро сошлась бы в точку максимальной приспособленности и после этого оставалась бы в равновесии. Но если гены взаимодействуют, эволюция — поиск максимальной приспособленности — становится невообразимо сложнее. Когда в геноме тысяча генов и у каждого два варианта, это даст 2^{1000} возможных состояний: во Вселенной нет такой древней и большой планеты, чтобы все перепробовать. И тем не менее эволюция на Земле сумела создать ряд замечательно

приспособленных организмов, и теория естественного отбора Дарвина объясняет, как именно это происходит, по крайней мере качественно, а не количественно. Холланд решил превратить все это в алгоритм.

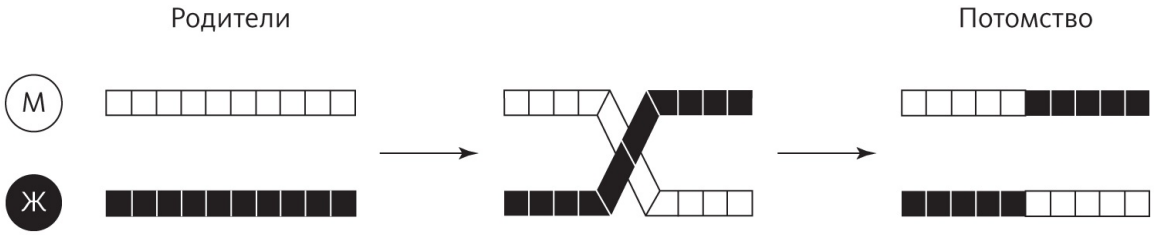
Но сначала ему надо было окончить университет. Он благоразумно выбрал более традиционную тему — булевы схемы с циклами — и в 1959 году защитил первую в мире диссертацию по информатике. Научный руководитель Холланда Артур Бёркс[73] поощрял интерес к эволюционным вычислениям: помог ему устроиться по совместительству на работу в Мичиганском университете и защищал его от нападок старших коллег, которые вообще не считали эту тему информатикой. Сам Бёркс был таким открытым для новых идей, потому что тесно сотрудничал с Джоном фон Нейманом[74], доказавшим принципиальную возможность существования самовоспроизводящихся машин. Бёркcу выпало завершить эту работу после того, как в 1957 году фон Нейман умер от рака. То, что фон Нейману удалось доказать возможность существования таких машин, — замечательное достижение, учитывая примитивное состояние генетики и информатики в то время, однако его автомат просто делал точные копии самого себя: эволюционирующие автоматы ждали Холланда.

Ключевой вход генетического алгоритма, как назвали творение Холланда, — функция приспособленности. Если имеется программа-кандидат и некая цель, которую эта программа должна выполнить, функция приспособленности присваивает программе баллы, показывающие, насколько хорошо она справилась с задачей. Можно ли так интерпретировать приспособленность в естественном отборе — большой вопрос: приспособленность крыла к полету интуитивно понятна, однако цель эволюции как таковой неизвестна. Тем не менее в машинном обучении необходимость чего-то похожего на функцию приспособленности не вызывает никаких сомнений. Если нам нужно поставить диагноз, то программа, которая дает правильный результат у 60 процентов пациентов в нашей базе данных, будет лучше, чем та, что попадает в точку только в 55 процентах случаев, и здесь возможной функцией приспособленности станет доля правильно диагностированных случаев.

В этом отношении генетические алгоритмы во многом похожи на искусственную селекцию. Дарвин открывает «Происхождение видов» дискуссией на эту тему, чтобы, оттолкнувшись от нее, перейти к более сложной концепции естественного отбора. Все одомашненные растения и животные, которые мы сегодня воспринимаем как должное, появились в результате многих поколений отбора и спаривания организмов, лучше всего подходящих для наших целей: кукурузы с самыми крупными початками, деревьев с самыми сладкими фруктами, самых длинношерстных овец, самых выносливых лошадей. Генетические алгоритмы делают то же самое, только выращивают они не живых существ, а программы, и поколение длится несколько секунд компьютерного времени, а не целую жизнь.

Функция приспособленности воплощает роль человека в этом процессе, но более тонкий аспект — это роль природы. Начав с популяции не очень подходящих кандидатов — возможно, совершенно случайных, — генетический алгоритм должен прийти к вариантам, которые затем можно будет отобрать на основе приспособленности. Как это делает природа? Дарвин этого не знал. Здесь в игру вступает генетическая часть алгоритма. Точно так же как ДНК кодирует организм в последовательности пар азотистых оснований, программу можно закодировать в строке битов. Вместо нулей и единиц алфавит ДНК состоит из четырех символов — аденина, тимина, гуанина и цитозина. Но различие лишь поверхностное. Вариативность последовательности ДНК, или строки битов, можно получить несколькими способами. Самый простой подход — это точечная мутация, смена значения произвольного бита в строке или изменение одного основания в спирали ДНК. Но Холланд видел настоящую мощь генетических алгоритмов в более сложном процессе: половом размножении.

Если снять с полового размножения все лишнее (не хихикайте), останется суть: обмен генетического материала между хромосомами отца и матери. Этот процесс называется кроссинговер, и его результат — появление двух новых хромосом. Первая состоит из материнской хромосомы до точки перекреста, после которой идет отцовская, вторая — наоборот:



Генетический алгоритм основан на подражании этому процессу. В каждом поколении он сводит друг с другом самые приспособленные особи, перекрещивает их битовые строки в произвольной точке и получает двух потомков от каждой пары родителей. После этого алгоритм делает в новых строках точечные мутации и отпускает в виртуальный мир. Когда строки возвращаются с присвоенным значением приспособленности, процесс повторяется заново. Каждое новое поколение более приспособлено, чем предыдущее, и процесс прерывается либо после достижения желаемой приспособленности, либо когда заканчивается время.

Представьте, например, что нам нужно вывести правило для фильтрации спама. Если в обучающих данных десять тысяч разных слов, каждое правило-кандидат можно представить в виде строки из 20 тысяч битов, по два для каждого слова. Первый бит для слова «бесплатно» будет равен единице, если письмам, содержащим слово «бесплатно», разрешено соответствовать правилу, и нулю, если не разрешено. Второй бит противоположен: один, если письма, не содержащие слова «бесплатно», соответствуют правилу, и ноль — если не соответствуют. Если единице равны оба бита, письмо будет соответствовать правилу вне зависимости от того, содержит оно слово «бесплатно» или нет, то есть правило, по сути, не содержит условий для этого слова. С другой стороны, если оба бита равны нулю, правилу не будут соответствовать никакие письма, поскольку либо один, либо другой бит всегда ошибается и такой фильтр пропустит любые письма (ой!). В целом письмо соответствует правилу, только если оно разрешает весь паттерн содержащихся и отсутствующих в нем слов. Приспособленностью правила может быть, например, процент писем, который оно правильно классифицирует. Начиная с популяции произвольных строк, каждая из которых представляет собой правило с произвольными

условиями, генетический алгоритм будет выводить все более хорошие правила путем повторяющегося кроссинговера и мутаций самых подходящих строк в каждом поколении. Например, если в текущей популяции есть правило «Если письмо содержит слово “бесплатный” — это спам» и «Если письмо содержит слово “легко” — это спам», перекрещивание их даст, вероятно, более подходящее правило «Если письмо содержит слова “бесплатный” и “легко” — это спам», при условии, что перекрест не придется между двумя битами, соответствующими одному из этих слов. Кроссинговер также породит правило «Все письма — спам», которое появится в результате отбрасывания обоих условий. Но у этого правила вряд ли будет много потомков в следующем поколении.

Поскольку наша цель — создать лучший спам-фильтр из всех возможных, мы не обязаны честно симулировать настоящий естественный отбор и можем свободно хитрить, подгоняя алгоритм под свои нужды. Одна из частых уловок — допущение бессмертия (жаль, что в реальной жизни его нет): хорошо подходящая особь будет конкурировать за размножение не только в своем поколении, но и с детьми, внуками, правнуками и так далее — до тех пор пока остается одной из самых приспособленных в популяции. В реальном мире все не так. Лучшее, что может сделать очень приспособленная особь, — передать половину своих генов многочисленным детям, каждый из которых будет, вероятно, менее приспособлен, так как другую половину генов унаследует от второго родителя. Бессмертие позволяет избежать отката назад, и при некотором везении алгоритм быстрее достигнет желаемой приспособленности. Конечно, если оценивать по количеству потомков, самые приспособленные индивидуумы в истории были похожи на Чингисхана — предка одного из двух сотен живущих сегодня людей. Так что, наверное, не так плохо, что в реальной жизни бессмертие не дозволено.

Если мы хотим вывести не одно, а целый набор правил фильтрации спама, можно представить набор — кандидат из n правил в виде строки $n \times 20\,000$ битов (20 тысяч для каждого правила, если в данных, как раньше, 10 тысяч разных слов). Правила, содержащие для каких-то слов 00, выпадают из набора, поскольку они, как мы видели выше, не подходят ни к каким

письмам. Если письмо подходит к любому правилу в наборе, оно классифицируется как спам. В противном случае оно допустимо. Мы по-прежнему можем сформулировать приспособленность как процент правильно классифицированных писем, но для борьбы с переобучением, вероятно, будет целесообразно вычитать из результата штраф, пропорциональный сумме активных условий в наборе правил.

Мы поступим еще изящнее, если разрешим выводить правила для промежуточных концепций, а затем выстраивать цепочки из этих правил в процессе работы. Например, мы можем получить правила «Если письмо содержит слово “кредит” — это мошенничество» и «Если письмо — мошенничество, значит, это письмо — спам». Поскольку теперь следствие из правила не всегда «спам», требуется ввести в строки правил дополнительные биты, чтобы представить эти следствия. Конечно, компьютер не использует слово «мошенничество» буквально: он просто выдает некую строку битов, представляющую это понятие, но для наших целей этого вполне достаточно. Такие наборы правил Холланд называет системами классификации. Они «рабочие лошадки» эволюционистов — основанного им племени машинного обучения. Как и многослойные перцептроны, системы классификации сталкиваются с проблемой присвоения коэффициентов доверия — какова приспособленность правил к промежуточным понятиям? — и для ее решения Холланд разработал так называемый алгоритм пожарной цепочки. Тем не менее системы классификации используются намного реже, чем многослойные перцептроны.

По сравнению с простой моделью, описанной в книге Фишера, генетические алгоритмы — довольно большой скачок. Дарвин жаловался, что ему не хватает математических способностей, но, живи он на столетие позже, то, вероятно, горевал бы из-за неумения программировать. Поймать естественный отбор в серии уравнений действительно крайне сложно, однако выразить его в виде алгоритма — совсем другое дело, и это могло бы пролить свет на многие мучающие человечество вопросы. Почему виды появляются в палеонтологической летописи внезапно? Где доказательства, что они постепенно эволюционировали из более ранних видов? В 1972 году Нильс Элдридж и Стивен Джей Гулд[75]

предположили, что эволюция состоит из ряда «прерывистых равновесий»: перемежающихся длинных периодов застоя и коротких всплесков быстрых изменений, одним из которых стал кембрийский взрыв[76]. Эта теория породила жаркие дебаты: критики прозвали ее «дерганной эволюцией». На это Элдридж и Гулд отвечали, что постепенную эволюцию можно назвать «ползучей». Эксперименты с генетическими алгоритмами говорят в пользу скачков. Если запустить такой алгоритм на 100 тысяч поколений и понаблюдать за популяцией в тысячепоколенных отрезках, график зависимости приспособленности от времени будет, вероятно, похож на неровную лестницу с внезапными скачками улучшений, за которыми идут плоские периоды затишья, со временем длящиеся все дольше. Несложно догадаться, почему так происходит: когда алгоритм достигнет локального максимума — пика на ландшафте приспособленности, — он будет оставаться там до тех пор, пока в результате счастливой мутации или кроссинговера какая-то особь не окажется на склоне более высокого пика: в этот момент такая особь начнет размножаться и с каждым поколением взбираться по склону все выше. Чем выше текущий пик, тем дольше приходится ждать такого события. Конечно, в природе эволюция сложнее: во-первых, среда может меняться — как физически, так и потому, что другие организмы тоже эволюционируют, и особь на пике приспособленности вдруг может почувствовать давление и будет вынуждена эволюционировать снова. Так что текущие генетические алгоритмы полезны, но конец пути еще очень далеко.

Дилемма изучения–применения

Обратите внимание, насколько генетические алгоритмы отличаются от многослойных перцептронов. Метод обратного распространения в любой момент времени рассматривает одну гипотезу, и эта гипотеза постепенно меняется, пока не найдет локальный оптимум. Генетические алгоритмы на каждом этапе рассматривают всю популяцию гипотез и благодаря кроссинговеру способны делать большие скачки от одного поколения к другому. После установления небольших произвольных исходных весов

обратное распространение действует детерминистски, а в генетических алгоритмах много случайных выборов: какие гипотезы оставить в живых и подвергнуть кроссинговеру (более вероятные кандидаты — лучше приспособленные гипотезы), где скрестить две строки, какие биты мутировать. Процесс обратного распространения получает вес*а* для заранее определенной архитектуры сети: более густые сети эластичнее, но их сложнее обучать. Генетические алгоритмы не делают априорных допущений об изучаемой структуре, кроме общей формы.

Благодаря этому генетические алгоритмы с намного меньшей вероятностью, чем обратное распространение ошибки, застревают в локальном оптимуме и в принципе более способны прийти к чему-то по-настоящему новому. В то же время их намного сложнее анализировать. Откуда нам знать, что генетический алгоритм получит что-то осмысленное, а не будет, как пьяный, слоняться вокруг да около? Главное здесь — мыслить в категориях кирпичиков. Каждый поднабор битов в строке потенциально кодирует полезный кирпичик, и, если скрестить две строки, кирпичики сольются в более крупный блок, который мы будем использовать в дальнейшем. Иллюстрировать силу кирпичиков Холланд любил с помощью фотороботов. До появления компьютеров полицейские художники умели по показаниям свидетелей быстро рисовать портрет подозреваемого: подбирали бумажную полоску из набора типичных форм рта, потом полоску с глазами, носом, подбородком и так далее. Система из десяти элементов по десять вариантов каждого позволяла составить 10 миллиардов разных лиц. Это больше, чем людей на планете.

В машинном обучении и вообще в информатике нет ничего лучше, чем сделать комбинаторный взрыв не врагом, а союзником. В генетических алгоритмах интересно то, что каждая строка косвенно содержит экспоненциальное число строительных блоков — схем, — поэтому поиск намного эффективнее, чем кажется. Дело в том, что каждый поднабор битов в строке — это схема, представляющая некую потенциально подходящую комбинацию свойств, а количество поднаборов в строке экспоненциально. Схему можно представить, заменив не входящие в нее биты звездочкой: например, строка 110 содержит схемы ***,

0, *1*, 1, *10, 11*, 1*0 и 110. Каждый выбор битов дает свою схему, а поскольку для каждого бита у нас есть два варианта действий (включать / не включать в схему), это дает 2^n схем. И наоборот, конкретную схему в популяции могут представлять многие строки, и каждый раз схема неявно оценивается. Представьте, что вероятность выживания и перехода гипотезы в следующее поколение пропорциональна ее приспособленности. Холланд показал, что в таком случае чем более приспособлены представления схемы в одном поколении по сравнению со средним значением, тем с большей вероятностью можно ожидать увидеть их в следующем поколении. Поэтому, хотя генетический алгоритм явно оперирует строками, косвенно он ищет в гораздо более обширном пространстве схем. Со временем в популяции начинают доминировать более приспособленные схемы, и, в отличие от пьяного, генетические алгоритмы находят дорогу домой.

Одна из самых важных проблем как машинного обучения, так и жизни — дилемма изучения–применения. Если вы нашли что-то работающее, лучше просто это использовать или попытаться поискать дальше, зная, что это может оказаться пустой тратой времени, а может привести к более хорошему решению. Лучше быть ковбоем или фермером? Основать новую компанию или управлять уже существующей? Поддерживать постоянные отношения или непрерывно искать свою половинку? Кризис среднего возраста — тяга к изучению после многих лет применения. Человек срывается с места и летит в Лас-Вегас, полный решимости потратить сбережения всей жизни ради шанса стать миллионером. Он входит в первое попавшееся казино и видит ряд игровых автоматов. Сыграть надо в тот, который даст в среднем лучший выигрыш, но где он стоит — неизвестно. Чтобы разобраться, надо попробовать каждый автомат достаточное количество раз, но, если заниматься этим слишком долго, все деньги уйдут на проигрышные варианты. И наоборот, если перестать пробовать слишком рано и выбрать автомат, который случайно показался удачным в первые несколько раундов, но на самом деле не самый хороший, можно играть на нем остаток ночи и просадить все деньги. В этом суть дилеммы изучения–применения: каждый раз приходится выбирать между повторением хода, принесшего наибольший на данный момент

выигрыш, и другими шагами, которые дадут информацию, ведущую, возможно, к еще большему выигрышу. Холланд доказал, что для двух игровых автоматов оптимальная стратегия — каждый раз подбрасывать неправильную монету, которая по мере развития событий становится экспоненциально более неправильной. (Только не подавайте на меня в суд, если этот метод не сработает. Помните, что в конечном счете казино всегда выигрывает.) Чем перспективнее выглядит игровой автомат, тем дольше вы должны в него играть, но нельзя совсем отказываться от второго на случай, если в конце концов он окажется лучшим.

Генетический алгоритм похож на вожака банды профессиональных игроков, которые играют в автоматы сразу во всех казино города. Две схемы конкурируют друг с другом, если включают одни и те же биты, но отличаются по крайней мере одним из них, например *10 и *11, а *n* конкурирующих схем — как *n* игровых автоматов. Каждый набор конкурирующих схем — это казино, и генетический алгоритм одновременно определяет автомат-победитель в каждом из них, следуя оптимальной стратегии: игре на самых перспективных машинах с экспоненциально увеличивающейся частотой. Хорошо придумано.

В книге «Автостопом по галактике»[77] инопланетная цивилизация строит огромный суперкомпьютер, чтобы ответить на Главный Вопрос, и спустя долгое время компьютер выдает ответ: «42». При этом компьютер добавляет, что инопланетяне не знают, в чем заключается этот вопрос, поэтому те строят еще больший компьютер, чтобы это определить. К сожалению, этот компьютер, известный также как планета Земля, уничтожают, чтобы освободить место для космического шоссе, за несколько минут до завершения вычислений, длившихся много миллионов лет. Теперь можно только гадать, какой это был вопрос, но, наверное, он звучал так: «На каком автомате мне сыграть?»

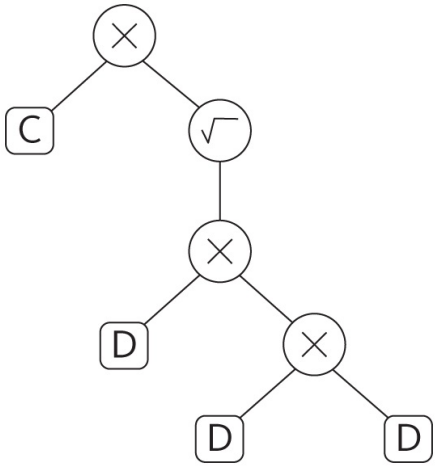
Выживание самых приспособленных программ

Первые несколько десятилетий сообщество, занимавшееся генетическими алгоритмами, состояло в основном из самого Джона Холланда, его студентов и студентов его студентов. Примерно

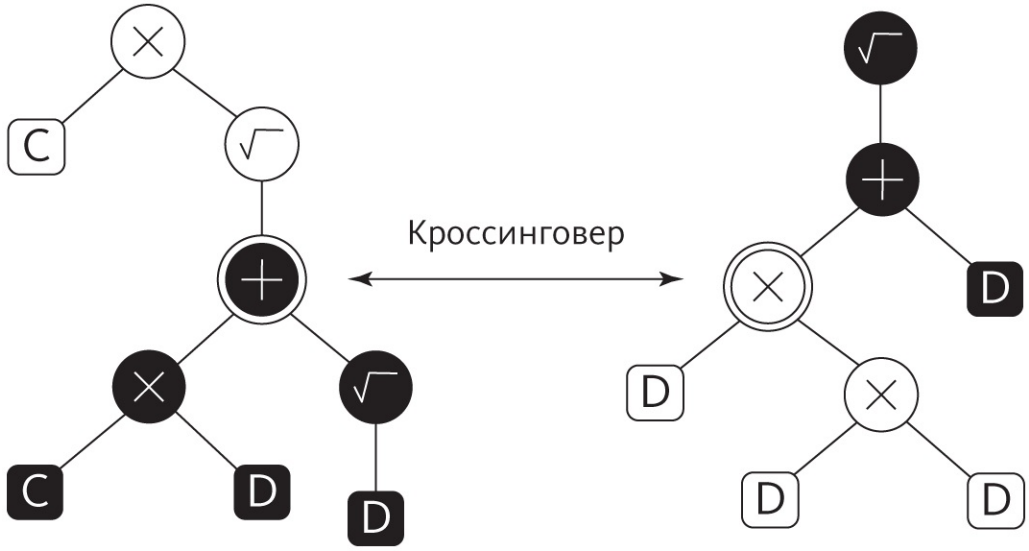
в 1983 году самой большой проблемой, которую умели решать генетические алгоритмы, было обучение управлению системами газопроводов. Но затем, примерно в период второго пришествия нейронных сетей, интерес к эволюционным вычислениям начал набирать обороты. Первая международная конференция по генетическим алгоритмам состоялась в 1985 году в Питтсбурге, а потом произошел кембрийский взрыв разновидностей генетических алгоритмов. Некоторые из них пробовали моделировать эволюцию более точно: в конце концов, базовый генетический алгоритм был только грубым ее приближением. Другие шли в самых разных направлениях, скрещивая эволюционные идеи с концепциями из области информатики, которые смутили бы Дарвина.

Одним из самых выдающихся студентов Холланда был Джон Коза[78]. В 1987 году он возвращался на самолете в Калифорнию с конференции в Италии, и его озарило: почему бы не получать путем эволюции полноценные компьютерные программы, а не сравнительно простые вещи вроде правил «Если..., то...» и контроллеров газопроводов? А если поставить себе такую цель, зачем держаться за битовые строки как их представление? Программа на самом деле — дерево обращений к подпрограммам, поэтому лучше непосредственно скрещивать эти поддеревья, а не втискивать их в битовые строки и рисковать разрушить отличные подпрограммы, перекрещивая их в произвольной точке.

Например, представьте, что вы хотите вывести программу, вычисляющую длину года на некой планете *(T)* на основе ее расстояния от солнца *(D)*. По третьему закону Кеплера, *T* — это квадратный корень из *D* в кубе, умноженный на постоянную *C*, которая зависит от используемых единиц времени и расстояния. Генетический алгоритм должен уметь работать на основе данных Тихо Браге о планетарных движениях, как когда-то сам Кеплер работал. В подходе Коза *D* и *C* — это листья программного дерева, а операции, которые их соединяют, например умножение и извлечение квадратного корня, — это внутренние узлы. Вот программное дерево, которое правильно вычисляет *T*:



В генетическом программировании, как Коза назвал свой метод, мы скрещиваем два программных дерева, произвольно меняя местами два их поддерева. Например, одним из результатов кроссинговера деревьев на рисунке ниже, проведенного в выделенных узлах, будет правильная программа для вычисления *T*:



Мы можем измерить приспособленность программы (или ее неприспособленность) по расстоянию между ее фактическим выходом и правильным выходом на обучающих данных. Например, если программа говорит, что на Земле в году 300 дней, это вычтет из ее приспособленности 65 пунктов. Генетическое программирование начинает с популяции случайных программных

деревьев, а потом использует кроссинговер, мутации и выживание для постепенного выведения лучших программ, пока не будет удовлетворено результатом.

Конечно, расчет продолжительности планетарного года — очень простая задача: в ней есть только умножение и квадратный корень. В целом программные деревья могут включать полный спектр элементов программирования, например утверждения «если… то…», циклы и рекурсию. Более показательный пример возможностей генетического программирования — это определение последовательности действий робота для достижения определенной цели. Представьте, что я попросил своего офисного робота принести мне степлер из кладовки. У робота для этого есть большой набор возможных действий: пройти по коридору, открыть дверь, взять предмет и так далее. Каждое из них, в свою очередь, может состоять из различных поддействий: например, протянуть манипулятор к предмету, схватить его в самых разных точках. В зависимости от результатов предыдущих действий новые действия могут выполняться или не выполняться, иногда нужно их повторить некоторое количество раз и так далее. Задача состоит в том, чтобы составить правильную структуру действий и поддействий, а также определить параметры каждого из них, например, как далеко выдвинуть манипулятор. Из «атомарных», мельчайших действий робота и их допустимых комбинаций генетическое программирование может собрать сложное поведение для достижения желаемой цели. Многие ученые таким образом выводили стратегии для роботов-футболистов.

Одно из последствий применения кроссинговера к программным деревьям вместо битовых строк заключается в том, что получающиеся в результате программы могут быть любого размера и обучение делается более гибким. Однако такие программы имеют тенденцию к разбуханию: по мере эволюции деревья становятся все больше и больше (это еще называют «выживанием толстейших»). Эволюционисты могут утешиться фактом, что программы, написанные человеком, в этом отношении не лучше — Microsoft Windows содержит 45 миллионов строк кода, — к тому же к созданному человеком коду нельзя применить

такие простые решения, как прибавление к функции приспособленности штрафа за сложность.

Первым успехом генетического программирования стала в 1995 году разработка электронных схем. Начав с кучи элементов — транзисторов, резисторов и конденсаторов, — система Коза вновь изобрела запатентованный ранее фильтр нижних частот, который можно использовать, например, для усиления басов в танцевальной музыке. С тех пор Коза превратил повторное изобретение запатентованных устройств в своего рода спорт и начал выдавать их дюжинами. Следующей вехой стал патент на разработанную таким образом промышленную систему оптимизации, выданный в 2005 году Ведомством США по патентам и торговым знакам. Если бы тест Тьюринга[79] заключался в обмане патентного эксперта, а не собеседника, 25 января 2005 года вошло бы в учебники истории.

Своей самоуверенностью Коза выделяется даже среди представителей дисциплины, где скромность не в чести. Он видит в генетическом программировании машину для изобретений, Кремниевого Эдисона XXI столетия. Коза и другие эволюционисты верят, что такой алгоритм может получить любую программу, и ставят на него в гонке за Верховным алгоритмом. В 2004 году они учредили ежегодную премию Humies, которую вручают за «соперничающие с человеком» генетические творения. На сегодняшний день присуждено 39 премий.

Зачем нужен секс?

Несмотря на все успехи и озарения, которые принесли нам генетические алгоритмы в таких вопросах, как градуализм против прерывистого равновесия, одна великая тайна остается неразгаданной: какую роль в эволюции играет половое размножение. Эволюционисты возлагают большие надежды на кроссинговер, однако представители других «племен» считают, что игра не стоит свеч. Ни один из теоретических результатов Холланда не показывает, что кроссинговер полезен: чтобы со временем экспоненциально увеличить частоту наиболее подходящих схем в популяции, достаточно мутаций, а логика «строительных блоков» привлекательна, но быстро сталкивается

с проблемами даже при использовании генетического программирования. Когда в ходе эволюции появляются крупные блоки, кроссинговер со все большей вероятностью начинает их разбивать, а если удается вывести очень приспособленную особь, ее потомки, как правило, быстро захватывают популяцию и закрывают собой потенциально более перспективные схемы, содержащиеся в менее приспособленных в целом особях. В результате поиск вариантов сводится к определению чемпиона по приспособленности. Исследователи придумали много приемов сохранения разнообразия в популяции, но результаты пока неубедительны. Инженеры, несомненно, широко используют строительные блоки, но для их правильного соединения нужно, скажем так, много инженерии: сложить их вместе любым старым способом непросто, и неясно, может ли кроссинговер здесь помочь.

Если исключить половое размножение, у эволюционистов в качестве двигателя останутся одни мутации. Когда размер популяции значительно превышает количество генов, есть шанс, что все точечные мутации в ней уже представлены, и тогда поиск становится разновидностью восхождения по выпуклой поверхности: попробуй все возможные шаги, выбери лучший и повтори. (Или выбери несколько лучших вариантов: в таком случае это называется лучевой поиск.) Символисты, например, постоянно используют такой подход для получения наборов правил и не считают его формой эволюции. Чтобы не застрять в локальном максимуме, восхождение можно усилить случайностью (допустить с некоторой вероятностью движение вниз) и случайными перезапусками (через некоторое время перепрыгнуть в произвольное состояние и продолжать восхождение там). Этого достаточно, чтобы найти удачные решения проблем. Оправдывает ли польза от добавления кроссинговера лишние затраты на вычисления — вопрос открытый.

Никто точно не знает, почему половое размножение так распространено в природе. Было выдвинуто несколько теорий, но ни одна из них не стала общепринятой. Ведущее положение занимает гипотеза Черной Королевы[80], популяризированная Мэттом Ридли в книге The Red Queen: Sex and the Evolution of Human Nature («Черная королева. Секс и эволюция человеческой

натуры»). Черная Королева говорит Алисе: «Приходится бежать со всех ног, чтобы только остаться на том же месте». Организмы находятся в постоянной гонке с паразитами, и половое размножение помогает популяции быть настолько разнообразной, чтобы ни один микроб не смог заразить ее целиком. Если дело в этом, тогда половое размножение не имеет отношения к машинному обучению, по крайней мере пока полученные путем эволюции программы не начнут соревноваться с компьютерными вирусами за время работы процессора и память. (Что интересно, Дэнни Хиллис[81] утверждает, что целенаправленное введение в генетический алгоритм попутно эволюционирующих паразитов может помочь избежать локальных максимумов путем постепенного наращивания сложности, однако пока по этому пути никто не пошел.) Христос Пападимитриу[82] и его коллеги показали, что половое размножение оптимизирует не приспособленность, а то, что они назвали смешиваемостью: способность гена в среднем хорошо справляться при соединении с другими генами. Это может пригодиться, если функция приспособленности либо неизвестна, либо непостоянна, как в естественном отборе, однако в машинном обучении и оптимизации, как правило, лучше справляется восхождение на выпуклые поверхности.

На этом проблемы генетического программирования не заканчиваются. Получается, что даже его успехи, возможно, совсем не такие «генетические», как хотелось бы эволюционистам. Возьмем разработку электронных схем — знаковый успех генетического программирования. Как правило, даже относительно простые схемы требуют огромного объема поиска, причем неясно, насколько мы обязаны результатами грубой силе, а насколько — генетическому интеллекту. Чтобы ответить растущему хору критиков, Коза включил в свою опубликованную в 1992 году книгу Genetic Programming эксперименты, показывающие, что генетическое программирование превосходит случайно сгенерированных кандидатов в проблеме синтеза булевых схем, но отрыв был небольшой. В 1995 году на Международной конференции по машинному обучению (International Conference on Machine Learning, ICML) в Лейк-Тахо Кевин Лэнг опубликовал статью о том, что восхождение на выпуклые поверхности побеждает

генетическое программирование в тех же самых программах, причем часто со значительным перевесом. Коза и другие эволюционисты неоднократно пытались опубликовать свои работы в материалах ICML, ведущем мероприятии в этой области, но, к их растущему разочарованию, их постоянно отклоняли из-за недостаточной эмпирической обоснованности. Коза и так был раздосадован тем, что его не публикуют, поэтому работа Лэнга просто вывела его из равновесия. На скорую руку он написал статью на 23 страницах в два столбца, в которой опровергал выводы Лэнга и обвинял рецензентов ICML в нарушении научной этики, а затем положил по экземпляру на каждое кресло в конференц-зале. Статья Лэнга (а может, и ответ Коза — как посмотреть) стали последней каплей: инцидент в Тахо привел к окончательному расхождению между эволюционистами и остальным сообществом ученых, занимающихся машинным обучением. Эволюционисты хлопнули дверью. Специалисты по генетическому программированию начали проводить собственные конференции, которые впоследствии слились с конференциями по генетическим алгоритмам в GECCO — Genetic and Evolutionary Computing Conference. А мейнстрим машинного обучения во многом просто забыл об их существовании. Печальная развязка, но не первый случай в истории, когда секс приводит к разводу.

Может быть, секс не преуспел в машинном обучении, но в утешение можно сказать, что он все же сыграл видную роль в эволюции технологий. Порнография стала непризнанным «приложением-приманкой» Глобальной сети, не говоря уже о печатной прессе, фотографии и видео. Вибратор был первым ручным электрическим устройством, на столетие опередившим мобильные телефоны. Мотороллеры получили распространение в послевоенной Европе, особенно в Италии, потому что на них молодые пары могли скрыться от своих семей. Одной из «приманок» огня, который миллион лет назад открыл Homo erectus, было, несомненно, то, что с его помощью легче стало назначать свидания. Несомненно и то, что индустрия секс-ботов станет мотором, толкающим человекоподобных роботов ко все большей реалистичности. Просто секс, по-видимому, не средство, а цель технологической эволюции.

Воспитание природы

У эволюционистов и коннекционистов есть одно важное сходство: и те и другие разрабатывают обучающиеся алгоритмы, вдохновленные природой. Однако потом их пути расходятся. Эволюционисты сосредоточены на получении структур: для них тонкая настройка результата путем оптимизации параметров имеет второстепенное значение. Коннекционисты же предпочитают брать простые, вручную написанные структуры со множеством соединений и предоставлять весовому обучению делать всю работу. Это все тот же извечный вопрос о приоритете природы и воспитания, на этот раз в машинном обучении, и у обоих оппонентов имеются веские аргументы.

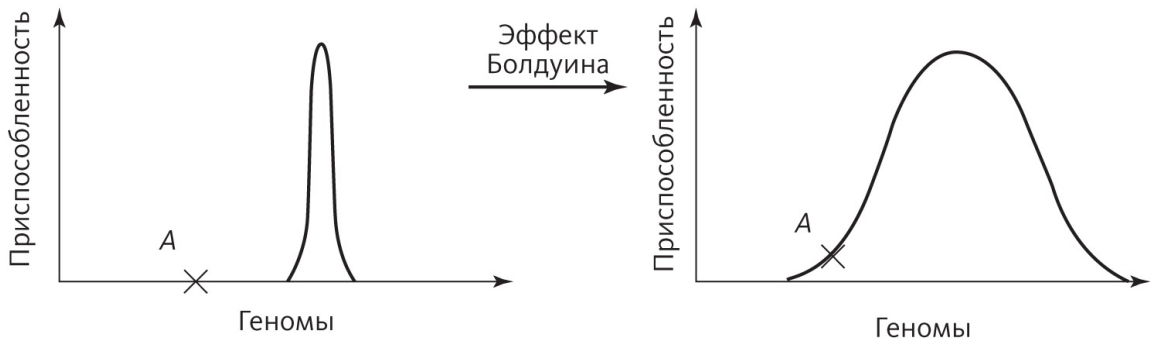
С одной стороны, эволюция породила много удивительных вещей, самая чудесная из которых — вы сами. С кроссинговером или без него, получение структур путем эволюции — существенный элемент Верховного алгоритма. Мозг может узнать все, но он не может получить еще один мозг. Если как следует разобраться в его архитектуре, можно просто воплотить его в «железе», но пока мы очень далеки от этого, поэтому однозначно надо обратиться за поддержкой к компьютерной симуляции эволюции. Более того, путем эволюции мы хотим получать мозг для роботов, системы с произвольными сенсорами и искусственный сверхинтеллект: нет причин держаться за устройство человеческого мозга, если для этих целей что-то подойдет лучше. С другой стороны, эволюция работает ужасно медленно. Вся жизнь организма дает всего лишь один фрагмент информации о его геноме: приспособленность, выраженную в числе потомков. Это колоссальная расточительность, которую нейронное обучение избегает путем получения информации в месте использования (если можно так выразиться). Как любят подчеркивать коннекционисты, например Джефф Хинтон, нет смысла носить в геноме информацию, если мы легко можем получить ее из органов чувств. Когда новорожденный открывает глаза, в его мозг начинает потоком литься видимый мир, и нужно просто все организовать, а в геноме должна быть задана архитектура машины, которая займется этой организацией.

Как и в дебатах по поводу наследственности и воспитания, ни у одной стороны нет полного ответа, и нужно понять, как соединить оба фактора. Верховный алгоритм — это не генетическое программирование и не обратное распространение ошибки, однако он должен включать основные элементы обоих подходов: обучение структурам и весам. С традиционной точки зрения первую часть дает природа, которая создает мозг в ходе эволюции, а затем за дело берется воспитание, заполняя мозг информацией. Это можно легко воспроизвести в алгоритмах машинного обучения. Сначала происходит обучение структуре сети с использованием (например) восхождения на выпуклые поверхности для определения, какие нейроны соединены друг с другом: надо попробовать добавить в сеть все возможные новые соединения, сохранить те, которые больше всего улучшают ее результативность, и повторить процедуру. Затем нужно узнать вес соединений методом обратного распространения ошибки — и новенький мозг готов к использованию.

Однако в этом месте и в естественной, и в искусственной эволюции появляется важная тонкость: вес надо узнать для всех рассматриваемых структур-кандидатов, а не только для последней, чтобы посмотреть, как хорошо она будет справляться с борьбой за выживание (в природе) или с обучающими данными (в искусственной системе). На каждом этапе нам будут нужны структуры, которые работают лучше всех не до, а после нахождения весов. Поэтому в реальности природа не предшествует воспитанию: они скорее перемежаются, и каждый раунд обучения «воспитанием» готовит сцену для следующего раунда обучения «природой», и наоборот. Природа эволюционирует ради воспитания, которое получает. Эволюционный рост ассоциативных зон коры головного мозга основан на нейронном обучении в сенсорных зонах — без этого он был бы бесполезным. Гусята постоянно ходят за своей мамой (поведение, сформировавшееся в ходе эволюции), но для этого они должны ее узнавать (выученная способность). Если вместо гусыни вылупившиеся птенцы увидят человека, они будут следовать за ним: это замечательно показал Конрад Лоренц[83]. В мозге новорожденного свойства среды уже закодированы, но косвенно: эволюция оптимизирует мозг для извлечения этих свойств

из ожидаемых вводных. Аналогично для алгоритма, который итерационно учится новым структурам и весам, каждая новая структура неявно — функция весов, которые он получил в предыдущих раундах.

Из всех возможных геномов лишь немногие соответствуют жизнеспособным организмам, поэтому типичный ландшафт приспособленности представляет собой обширные равнины с периодическими резкими пиками, что очень затрудняет эволюцию. Если начать в Канзасе путь с завязанными глазами, не имея представления, в какой стороне Скалистые горы, можно очень долго блуждать в поисках предгорий и только потом начать восхождение. Однако если соединить эволюцию с нейронным обучением, результат будет очень интересный. Если вы стоите на плоской поверхности, но горы не слишком далеко, нейронное обучение может вас туда привести, причем чем ближе вы к горам, тем с большей вероятностью до них доберетесь. Это как способность видеть горизонт: в степях Уичито такая способность вам не пригодится, зато в Денвере вы увидите вдали Скалистые горы и направитесь к ним. Денвер, таким образом, станет намного более подходящим местом, чем Канзас, где у вас на глазах была повязка. Суммарный эффект — расширение пиков приспособленности и возможность найти путь к ним из мест, которые раньше были проблемными, например точки *A* на графике ниже:



В биологии это называется эффектом Болдуина, в честь Джеймса Марка Болдуина[84], предложившего его в 1896 году. В эволюции, по Болдуину, выученное поведение впоследствии становится генетически обусловленным: если похожие на собак

млекопитающие способны научиться плавать, у них больше шансов эволюционировать в морских котиков (как это и произошло), чем у животных, которые плавать не умеют. Таким образом, индивидуальное обучение может повлиять на эволюцию и без ламаркистских теорий. Джефф Хинтон и Стивен Нолан продемонстрировали эффект Болдуина в машинном обучении путем применения генетических алгоритмов: они получили с помощью эволюции структуру нейронной сети и обнаружили, что ее приспособленность со временем увеличивается, только если разрешено индивидуальное обучение.

Побеждает тот, кто быстрее учится

Эволюция ищет удачные структуры, а нейронное обучение их заполняет: такое сочетание — самый легкий шаг к Верховному алгоритму. Этот подход может удивить любого, кто знаком с бесконечными перипетиями спора о роли природы и воспитания, который не утихает две с половиной тысячи лет. Однако если смотреть на жизнь глазами компьютера, многое проясняется. «Природа» для компьютера — это программа, которую он выполняет, а «воспитание» — получаемые им данные. Вопрос, что важнее, очевидно абсурден. И без программы, и без данных никакого результата не будет, и нельзя сказать, что программа дает 60 процентов результата, а данные — 40. Знакомство с машинным обучением — прививка от такого прямолинейного мышления.

С другой стороны, возникает вопрос: почему наши поиски до сих пор не увенчались успехом? Ведь если соединить два верховных алгоритма природы — эволюцию и головной мозг, — большего и пожелать нельзя! К сожалению, пока мы имеем лишь очень грубую картину того, как учится природа: достаточно хорошую для множества применений, но все еще бледную тень реальности. Например, критически важная часть жизни — развитие эмбриона, а в машинном обучении ему нет аналога: «организм» — самая непосредственная функция генома, и, возможно, здесь нам не хватает чего-то важного. Но еще одна причина в том, что даже полное понимание того, как учится природа, будет недостаточным. Во-первых, природа работает слишком медленно: у эволюции

обучение отнимает миллиарды лет, а у мозга — всю жизнь. Культура в этом отношении лучше: результат целой жизни обучения можно дистиллировать в книге, которую человек прочтет за несколько часов. Но обучающиеся алгоритмы должны уметь учиться за минуты или секунды. Побеждает тот, кто учится быстрее, будь то эффект Болдуина, ускоряющий эволюцию, устное общение, ускоряющее человеческое обучение, или компьютеры, открывающие паттерны со скоростью света. Машинное обучение — последняя глава в гонке жизни на Земле, и более быстрое «железо» — лишь половина успеха. Вторая часть — это более умное программное обеспечение.

Важнейшая цель машинного обучения — любой ценой найти лучший обучающийся алгоритм из всех возможных, и эволюция и головной мозг вряд ли на это способны. У порождений эволюции много очевидных изъянов. Например, зрительный нерв млекопитающих связан с передней, а не с задней частью сетчатки, из-за чего рядом с центральной ямкой, областью самого четкого зрения, появляется просто вопиюще ненужное слепое пятно. Молекулярная биология живых клеток — это хаос, поэтому специалисты часто шутят, что верить в разумный замысел могут только люди, которые об этом не подозревают. В архитектуре головного мозга тоже могут быть недостатки: у мозга много ограничений, которых лишены компьютеры — например, очень ограниченная краткосрочная память, — и нет причин их сохранять. Более того, известно много ситуаций, в которых люди постоянно поступают неправильно, и Даниэль Канеман пространно иллюстрирует это в своей книге Thinking, Fast and Slow[85].

В отличие от коннекционистов и эволюционистов, символисты и байесовцы не верят в подражание природе и скорее хотят чисто теоретически понять, что надо делать при обучении — и алгоритмам, и людям. Например, если мы хотим научиться диагностировать рак, недостаточно сказать: «Вот так учится природа, давайте сделаем то же самое». Ставки слишком высоки: ошибки стоят жизней. Врачи должны диагностировать болезнь самым надежным способом, какой только можно придумать, и методы должны быть схожими с теми, которыми математики доказывают теоремы, или хотя бы максимально близкими к ним,

учитывая, что такая строгость встречается нечасто. Надо взвешивать доказательства, чтобы свести к минимуму вероятность неверного диагноза, или, точнее, чтобы чем дороже была ошибка, тем меньше была бы вероятность ее совершить. (Например, неспособность найти имеющуюся опухоль потенциально намного опаснее, чем ложное подозрение.) Врачи должны принимать *оптимальные* решения, а не просто такие, которые кажутся удачными.

Это частный случай линии разлома, проходящего через значительную часть науки и философии: различия между дескриптивными и нормативными теориями, между «есть вот так» и «должно быть вот так». В то же время символисты и байесовцы любят подчеркивать, что попытки понять, как мы должны учиться, могут помочь разобраться, как мы учимся на самом деле, потому что и то и другое предположительно очень даже взаимосвязано. В частности, поведение, которое важно для выживания и которое долго эволюционировало, должно быть близко к оптимальному. Человек не очень хорошо умеет отвечать на письменные вопросы о вероятностях, зато прекрасно, не задумываясь выбирает движение руки и кисти, чтобы попасть в мишень. Многие психологи применяли символистские и байесовские модели для объяснения некоторых аспектов человеческого поведения. Символисты доминировали в первые несколько десятилетий когнитивной психологии. В 1980-х и 1990-х власть захватили коннекционисты, а теперь на взлете сторонники байесовского подхода.

Для самых сложных проблем — тех, которые мы по-настоящему хотим, но не можем решить, например для лечения рака, — истинные «природные» подходы, вероятно, слишком просты и не принесут успеха, даже если дать им огромное количество данных. В принципе можно узнать полную модель метаболической сети клетки путем сочетания поиска структур, с кроссинговером или без, и подбора параметров методом обратного распространения ошибки, однако есть слишком много локальных экстремумов, в которых можно крепко увязнуть. Рассуждать нужно более крупными блоками, собирая и переставляя их при необходимости и используя обратную дедукцию, чтобы заполнить пробелы.

А направлять обучение должна цель — оптимальная диагностика рака и нахождение наилучших лекарств для его лечения.

Оптимальное обучение — это главная цель байесовцев, и они не сомневаются, что поняли, как ее достичь. Сюда, пожалуйста…

ГЛАВА 6

В СВЯТИЛИЩЕ ПРЕПОДОБНОГО БАЙЕСА

Из ночной тьмы выступает глыба кафедрального собора. Мозаичные окна льют свет на мостовую и соседние здания, проецируя замысловатые уравнения. Вы подходите ближе и слышите, что изнутри доносятся песнопения. Кажется, это латынь или, может быть, язык математики, но «вавилонская рыбка»[86] у вас в ухе переводит слова на понятный язык: «Поверни ручку! Поверни ручку!» Как только вы входите, пение переходит во вздох удовлетворения. По толпе проносится ропот: «Постериор! Постериор!» Вы проталкиваетесь вперед. Над алтарем возвышается массивная каменная таблица. На ней трехметровыми буквами выгравирована формула:

$$P(A|B) = P(A)\,P(B|A) / P(B)$$

Вы непонимающе смотрите на нее, но очки Google Glass услужливо подсказывают: «Теорема Байеса». Толпа начинает петь: «Больше данных! Больше данных!» Вереницу жертв безжалостно толкают к алтарю. Вдруг вы понимаете, что вы тоже среди них, но слишком поздно. Над вами нависла ручка. Вы кричите: «Нет! Я не хочу быть точкой данных! Пусти-и-ите!» И — просыпаетесь в холодном поту. На коленях у вас лежит книга под названием «Верховный алгоритм». Трясясь от пережитого кошмара, вы продолжаете читать с того места, где остановились.

Теорема, которая правит миром

О формуле, с которой начинается путь к оптимальному обучению, многие слышали: это теорема Байеса. Но в этой главе мы посмотрим на нее в совершенно другом свете и увидим, что она намного мощнее, чем может показаться, если судить по ее повседневному применению. По правде говоря, теорема Байеса — это просто несложное правило обновления уровня доверия к гипотезе при получении новых доказательств: если свидетельство

совпадает с гипотезой, ее вероятность идет вверх, если нет — вниз. Например, если тест на СПИД положительный, вероятность соответствующего диагноза повышается. Но когда доказательств — например, результатов анализов — много, все становится интереснее. Чтобы соединить их без риска комбинаторного взрыва, нужно сделать упрощающие допущения. Еще любопытнее рассматривать одновременно большое количество гипотез, например все возможные диагнозы у пациента. Вычисление на основе симптомов вероятности каждого заболевания за разумное время — серьезный интеллектуальный вызов. Когда мы поймем, как это сделать, мы будем готовы учиться по-байесовски. Для этого «племени» обучение — это «просто» еще одно применение теоремы Байеса, где целые модели — гипотезы, а данные — доказательства: по мере накопления данных некоторые модели становятся более вероятными, а некоторые — менее, пока в идеале одна модель не побеждает вчистую. Байесовцы изобрели дьявольски хитрые разновидности моделей, так что давайте приступим.

Томас Байес — английский священник, живший в XVIII веке, — сам того не подозревая, стал центром новой религии. Такой поворот может показаться удивительным, но стоит заметить, что то же самое произошло и с Иисусом: христианство в том виде, в котором мы его знаем, изобрел апостол Павел, а сам Иисус видел в себе вершину иудейской веры. Аналогично байесианство в привычном для нас виде было изобретено Пьер-Симоном де Лапласом — французом, родившимся на пять десятилетий позже Байеса. Байес был проповедником и первым описал новый подход к вероятностям, но именно Лаплас выразил его идеи в виде теоремы.

Лаплас, один из величайших математиков всех времен и народов, наверное, больше всего известен своей мечтой о ньютоновском детерминизме:

> Разум, которому в каждый определенный момент были бы известны все силы, приводящие природу в движение, и положение всех тел, из которых она состоит, будь он также достаточно обширен, чтобы подвергнуть эти данные анализу, смог бы объять единым законом движение величайших тел

> Вселенной и мельчайшего атома; для такого разума ничего не было бы неясного и будущее существовало бы в его глазах точно так же, как прошлое.

Это забавно, поскольку Лаплас был отцом теории вероятностей, которая, как он полагал, представляла собой просто здравый смысл, сведенный к вычислениям. В сердце его исследований вероятности лежала озабоченность вопросом Юма. Откуда, например, мы знаем, что завтра взойдет солнце? Каждый день, вплоть до сегодняшнего, это происходило, но ведь нет никаких гарантий, что так будет и впредь. Ответ Лапласа состоит из двух частей. Первая — то, что мы теперь называем принципом безразличия или принципом недостаточного основания. Однажды — скажем, в начале времен, которое для Лапласа было приблизительно 5 тысяч лет назад, — мы просыпаемся, прекрасно проводим день, а вечером видим, что солнце заходит. Вернется ли оно? Мы никогда не видели восхода, и у нас нет причин полагать, что оно взойдет или не взойдет. Таким образом мы должны рассмотреть два одинаково вероятных сценария и сказать, что солнце снова взойдет с вероятностью 1⁄2. Но, продолжал Лаплас, если прошлое хоть как-то указывает на будущее, каждый день, когда солнце восходит, должен укреплять нашу уверенность, что так будет происходить и дальше. Спустя пять тысячелетий вероятность, что солнце завтра снова взойдет, должна быть очень близка единице, но не равняться ей, потому что полной уверенности никогда не будет. Из этого мысленного эксперимента Лаплас вывел свое так называемое правило следования, согласно которому вероятность, что солнце снова взойдет после n восходов, равна $(n + 1) / (n + 2)$. Если $n = 0$, то это просто 1⁄2, а когда n увеличивается, растет и вероятность, стремясь к единице, когда n стремится к бесконечности.

Это правило вытекает из более общего принципа. Представьте, что вы проснулись посреди ночи на чужой планете. Хотя над вами только звездное небо, у вас есть причины полагать, что солнце в какой-то момент взойдет, поскольку большинство планет вращаются вокруг своей оси и вокруг звезды. Поэтому оценка соответствующей вероятности должна быть больше 1⁄2 (скажем, 2⁄3). Это называется *априорной вероятностью* восхода солнца,

поскольку она предшествует любым доказательствам: вы не исходите из подсчета восходов на этой планете в прошлом, потому что вас там не было и вы их не видели. Априорная вероятность скорее отражает наши убеждения по поводу того, что может произойти, основанные на общих знаниях о Вселенной. Но вот звезды начинают бледнеть, и уверенность, что солнце на этой планете действительно восходит, начинает расти, подкрепляемая земным опытом. Ваша уверенность теперь — это *апостериорная вероятность*, поскольку она возникает после того, как вы увидели некие доказательства. Небо светлеет, и апостериорная вероятность подскакивает еще раз. Наконец над горизонтом показывается краешек яркого диска, и солнечный луч «ловит башню гордую султана… в огненный силок», как в «Рубаи» Омара Хайяма. Если вы не страдаете галлюцинациями, то можете быть уверены, что солнце взойдет.

Возникает важнейший вопрос: как именно должна меняться апостериорная вероятность при появлении все большего объема доказательств? Ответ дает теорема Байеса. Мы можем рассматривать ее в причинно-следственных категориях. Восход солнца заставляет звезды угасать и делает небо светлее, однако последний факт — более сильное доказательство в пользу рассвета, поскольку звезды могут угасать и посреди ночи, скажем, из-за спустившегося тумана. Поэтому после того, как посветлело небо, вероятность рассвета должна увеличиться больше, чем после того, как вы заметили блекнущие звезды. Математики скажут, что *P(рассвет | светлое-небо)*, то есть условная вероятность восхода солнца при условии, что небо светлеет, будет больше, чем *P(рассвет | гаснущие-звезды)* — условной вероятности при условии, что звезды блекнут. Согласно теореме Байеса, чем более вероятно, что следствие вызвано причиной, тем более вероятно, что причина связана со следствием: если *P(светлое-небо | рассвет)* выше, чем *P(гаснущие-звезды | рассвет)*, — может быть, потому что некоторые планеты настолько удалены от своих солнц, что звезды там продолжают светить и после восхода, — следовательно, *P(рассвет | светлое-небо)* тоже будет выше, чем *P(рассвет | гаснущие-звезды)*.

Однако это еще не все. Если мы наблюдаем следствие, которое может произойти даже без данной причины, то, несомненно,

имеется недостаточно доказательств наличия этой причины. Теорема Байеса учитывает это, говоря, что *P(причина | следствие)* уменьшается вместе с *P(следствие)*, априорной вероятностью следствия (то есть ее вероятностью при отсутствии какого-либо знания о причинах). Наконец, при прочих равных, чем выше априорная вероятность причины, тем выше должна быть апостериорная вероятность. Если собрать все это вместе, получится теорема Байеса, которая гласит:

$$P(\text{причина} \mid \text{следствие}) = P(\text{причина}) \times P(\text{следствие} \mid \text{причина}) / P(\text{следствие}).$$

Замените слово «причина» на *A*, а «следствие» на *B*, для краткости опустите все знаки умножения, и вы получите трехметровую формулу из кафедрального собора.

Это, конечно, просто формулировка теоремы, а не ее доказательство. Но и доказательство на удивление простое. Мы можем проиллюстрировать его на примере из медицинской диагностики, одной из «приманок» байесовского вывода. Представьте, что вы врач и за последний месяц поставили диагноз сотне пациентов. Четырнадцать из них болели гриппом, у 20 была высокая температура, а у 11 — и то и другое. Условная вероятность температуры при гриппе, таким образом, составляет одиннадцать из четырнадцати, или 11⁄14. Обусловленность уменьшает размеры рассматриваемой нами вселенной, в данном случае от всех пациентов только до пациентов с гриппом. Во вселенной всех пациентов вероятность высокой температуры составляет 20⁄100, а во вселенной пациентов, больных гриппом, — 11⁄14. Вероятность того, что у пациента грипп *и* высокая температура, равна доле пациентов, больных гриппом, умноженной на долю пациентов с высокой температурой: $P(\text{грипп, температура}) = P(\text{грипп}) \times P(\text{температура} \mid \text{грипп}) = 14⁄100 \times 11⁄14 = 11⁄100$. Но верно и следующее: $P(\text{грипп, температура}) = P(\text{температура}) \times P(\text{грипп} \mid \text{температура})$. Таким образом, поскольку и то, и другое равно $P(\text{грипп, температура})$, то $P(\text{температура}) \times P(\text{грипп} \mid \text{температура}) = P(\text{грипп}) \times P(\text{температура} \mid \text{грипп})$. Разделите обе

стороны на *P(температура)*, и вы получите *P(грипп | температура)* = *P(грипп)* × *P(температура | грипп)* / *P(температура)*.

Вот и все! Это теорема Байеса, где грипп — это причина, а высокая температура — следствие.

Люди, оказывается, не очень хорошо владеют байесовским выводом, по крайней мере в устных рассуждениях. Проблема в том, что мы склонны пренебрегать априорной вероятностью причины. Если анализ показал наличие ВИЧ и этот тест дает только один процент ложных положительных результатов, стоит ли паниковать? На первый взгляд может показаться, что, увы, шанс наличия СПИДа — 99 процентов. Но давайте сохраним хладнокровие и последовательно применим теорему Байеса: *P(ВИЧ | положительный)* = *P(ВИЧ)* × *P(положительный | ВИЧ)* / *P(положительный)*. *P(ВИЧ)* — это распространенность данного вируса в общей популяции, которая в США составляет 0,3 процента. *P(положительный)* — это вероятность, что тест даст положительный результат независимо от того, есть у человека СПИД или нет. Скажем, это 1 процент. Поэтому *P(ВИЧ | положительный)* = 0,003 × 0,99 / 0,01 = 0,297. Это далеко не 0,99! Причина в том, что ВИЧ в общей популяции встречается редко. Положительный результат теста на два порядка увеличивает вероятность, что человек болен СПИДом, но она все еще меньше 1⁄2. Так что, если анализы дали положительный результат, разумнее будет сохранить спокойствие и провести еще один, более доказательный тест. Есть шанс, что все будет хорошо.

Теорема Байеса полезна, потому что обычно известна вероятность следствий при данных причинах, а узнать хотим вероятность причин при данных следствиях. Например, мы знаем процент пациентов с гриппом, у которых повышена температура, но на самом деле нам нужно определить вероятность, что пациент с температурой болен гриппом. Теорема Байеса позволяет нам перейти от одного к другому. Ее значимость, однако, этим далеко не ограничивается. Для байесовцев эта невинно выглядящая формула — настоящее *F* = *ma* машинного обучения, основа, из которой вытекает множество результатов и практических применений. Каков бы ни был Верховный алгоритм, он должен быть «всего лишь» вычислительным воплощением теоремы Байеса.

Я пишу «всего лишь» в кавычках, потому что применение теоремы Байеса в компьютерах оказалось дьявольски непростой задачей для всех проблем, кроме простейших. Скоро мы увидим почему.

Теорема Байеса как основа статистики и машинного обучения страдает не только от вычислительной сложности, но и от крайней противоречивости. Вы можете удивиться: разве она не прямое следствие идеи условной вероятности, как мы видели на примере гриппа? Действительно, с формулой как таковой ни у кого проблем не возникает. Противоречие заключается в том, как именно байесовцы получают вероятности, которые в нее включены, и что эти вероятности означают. Для большинства статистиков единственный допустимый способ оценки вероятностей — вычисление частоты соответствующего события. Например, вероятность гриппа равна 0,2, потому что им болело 20 из 100 обследованных пациентов. Это «частотная» интерпретация вероятности, и она дала название господствующему учению в статистике. Но обратите внимание, что в принципе безразличия Лапласа и в примере с восходом солнца мы просто высасываем вероятность из пальца. Чем оправдано априорное предположение, что вероятность восхода солнца равна одной второй, двум третьим или еще какой-то величине? На это байесовцы отвечают, что вероятность — это не частота, а субъективная степень убежденности, поэтому вам решать, какая она будет, а байесовский вывод просто позволяет обновлять априорные убеждения после появления новых доказательств, чтобы получать апостериорные убеждения (это называется «провернуть ручку Байеса»). Поклонники теоремы Байеса верят в эту идею с почти религиозным рвением и 200 лет выдерживают нападки и возражения. С появлением на сцене достаточно мощных компьютеров и больших наборов данных байесовский вывод начал брать верх.

Все модели неверны, но некоторые полезны

Настоящие врачи не диагностируют грипп на основе высокой температуры, а учитывают целый комплекс симптомов, включая боль в горле, кашель, насморк, головную боль, озноб и так далее.

Поэтому, когда нам действительно надо вычислить по теореме Байеса *P(грипп | температура, кашель, больное горло, насморк, головная боль, озноб, …)*, мы знаем, что эта вероятность пропорциональна *P(температура, кашель, больное горло, насморк, головная боль, озноб, … | грипп)*. Но здесь мы сталкиваемся с проблемой. Как оценить эту вероятность? Если каждый симптом — булева переменная (он либо есть, либо нет) и врач учитывает n симптомов, у пациента может быть 2^n комбинаций симптомов. Если у нас, скажем, 20 симптомов и база данных из 10 тысяч пациентов, мы увидим лишь малую долю из примерно миллиона возможных комбинаций. Еще хуже то, что для точной оценки вероятности конкретного сочетания симптомов нужны как минимум десятки его наблюдений, а это значит, что база данных должна включать десятки миллионов пациентов. Добавьте еще десяток симптомов, и нам понадобится больше пациентов, чем людей на Земле. Если симптомов сто и мы каким-то чудом получим такие данные, не хватит места на всех жестких дисках в мире, чтобы сохранить все эти вероятности. А если в кабинет войдет пациент с не встречавшимся ранее сочетанием симптомов, будет непонятно, как поставить ему диагноз. То есть мы столкнемся с давним врагом: комбинаторным взрывом.

Поэтому мы поступим так, как всегда стоит поступать: пойдем на компромисс. Нужно сделать упрощающие допущения, которые срежут количество подлежащих оценке вероятностей до уровня, с которым под силу справиться. Одно из простых и очень популярных допущений заключается в том, что все следствия данной причины независимы. Это значит, например, что наличие высокой температуры не влияет на вероятность кашля, если уже известно, что у больного грипп. Математически это значит, что *P(температура, кашель | грипп)* — просто *P(температура | грипп)* × *P(кашель | грипп)*. Получается, что и то и другое легко оценить на основании небольшого количества наблюдений. В предыдущем разделе мы уже сделали это для высокой температуры, и для кашля или любого другого симптома все будет так же. Необходимое количество наблюдений больше не растет экспоненциально с количеством симптомов. На самом деле оно вообще не растет.

Обратите внимание: речь идет о том, что высокая температура и кашель независимы не в принципе, а только при условии, что у больного грипп. Если неизвестно, есть грипп или нет, температура и кашель будут очень сильно коррелировать, поскольку вероятность кашля при высокой температуре намного выше. *P(температура, кашель)* не равно *P(температура)* × *P(кашель)*. Смысл в том, что если уже известно, что у больного грипп, то информация о высокой температуре не даст никакой *дополнительной* информации о том, есть ли у него кашель. Аналогично, если вы видите, что звезды гаснут, но не знаете, должно ли взойти солнце, ваши ожидания, что небо прояснится, должны возрасти. Но если вы уже знаете, что восход неизбежен, гаснущие звезды не важны.

Следует отметить, что такой фокус можно проделать только благодаря теореме Байеса. Если бы мы хотели прямо оценить *P(грипп | температура, кашель и так далее)* без предварительного преобразования с помощью теоремы в *P(температура, кашель и так далее | грипп)*, по-прежнему требовалось бы экспоненциальное число вероятностей, по одной для каждого сочетания «симптомы — грипп / не грипп».

Алгоритм машинного обучения, который применяет теорему Байеса и исходит из того, что следствия данной причины независимы, называется наивный байесовский классификатор. Дело в том, что такое допущение, прямо скажем, довольно наивное: в реальности температура увеличивает вероятность кашля, даже если уже известно, что у больного грипп, потому что она, например, повышает вероятность тяжелого гриппа. Однако машинное обучение — это искусство безнаказанно делать ложные допущения, а как заметил статистик Джордж Бокс, «все модели неверны, но некоторые полезны». Чрезмерно упрощенная модель, для оценки которой у вас есть достаточно данных, лучше, чем идеальная, для которой данных нет. Просто поразительно, насколько ошибочны и одновременно полезны бывают некоторые модели. Экономист Милтон Фридман в одном очень влиятельном эссе даже утверждал, что чрезмерно упрощенные теории — лучшие, при условии, что они дают точные предсказания: они объясняют больше с наименьшими усилиями. По-моему, это перебор, однако это хорошая иллюстрация того, что, вопреки Эйнштейну, наука часто развивается

по принципу «упрощай до тех пор, пока это возможно, а потом упрости еще немного».

Точно неизвестно, кто изобрел наивный байесовский алгоритм. Он был упомянут без автора в 1973 году в учебнике по распознаванию паттернов, но распространение получил лишь в 1990-е, когда исследователи заметили, что, как ни странно, он часто бывает точнее, чем намного более сложные обучающиеся алгоритмы. В то время я был старшекурсником. Запоздало решил включить наивный байесовский алгоритм в свои эксперименты и был шокирован, потому что оказалось, что он работает лучше, чем все другие, которые я сравнивал. К счастью, было одно исключение — алгоритм, который я разрабатывал для дипломной работы, — иначе, наверное, я не писал бы эти строки.

Наивный байесовский алгоритм сейчас используется очень широко: на нем основаны, например, многие спам-фильтры. Это применение придумал Дэвид Хекерман, выдающийся ученый-байесовец, врач. Ему пришла в голову мысль, что к спаму надо относиться как к заболеванию, симптомы которого — слова в электронном письме. «Виагра» — это симптом, и «халява» тоже, а имя лучшего друга, вероятно, указывает, что письмо настоящее. Затем можно использовать наивный байесовский алгоритм для классификации писем на спам и не-спам, но только при условии, что спамеры генерируют свои письма путем произвольного выбора слова. Это, конечно, странное допущение, которое было бы верно, не будь у предложений ни синтаксиса, ни содержания. Но тем же летом Мехран Сахами[87], тогда еще студент Стэнфорда, попробовал применить этот подход во время стажировки в Microsoft Research, и все отлично сработало. Когда Билл Гейтс спросил Хекермана, как это может быть, тот ответил, что для выявления спама вникать в подробности сообщения не обязательно: достаточно просто уловить суть, посмотрев, какие слова оно содержит.

В простейших поисковых системах алгоритмы, довольно похожие на наивный байесовский, используются, чтобы определить, какие сайты выдавать в ответ на запрос. Основное различие в том, что вместо классификации на спам и не спам нужно определить, подходит сайт к запросу или не подходит. Список проблем прогнозирования, к которым был применен наивный

байесовский алгоритм, можно продолжать бесконечно. Питер Норвиг, директор по исследованиям в Google, как-то рассказал, что у них этот подход используется шире всего, а ведь Google применяет машинное обучение везде где только можно. Несложно догадаться, почему наивный Байес приобрел в Google такую популярность. Даже не говоря о неожиданной точности, он позволяет легко увеличить масштаб. Обучение наивного байесовского классификатора сводится к простому подсчету, сколько раз каждый атрибут появляется с каждым классом, а это едва ли занимает больше времени, чем считывание данных с диска.

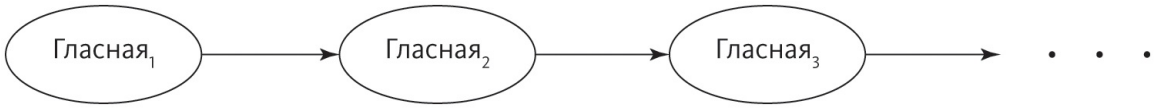
Наивный байесовский алгоритм в шутку можно использовать даже в большем масштабе, чем Google, и смоделировать с его помощью всю Вселенную. Действительно, если верить в существование всемогущего бога, можно создать модель Вселенной как обширного распределения, где все происходит независимо при условии божьей воли. Ловушка, конечно, в том, что божественный разум нам недоступен, но в главе 8 мы узнаем, как получать наивные байесовские модели, даже не зная классов примеров.

Поначалу может показаться, что это не так, но наивный байесовский алгоритм тесно связан с перцептронами. Перцептрон добавляет веса, а байесовский алгоритм умножает вероятности, но, если логарифмировать, последнее сведется к первому. И то и другое можно рассматривать как обобщение простых правил «Если..., то...», где каждый предшественник может вносить больший или меньший вклад в вывод, вместо подхода «все или ничего». Этот факт — еще один пример глубокой связи между алгоритмами машинного обучения и намек на существование Верховного алгоритма. Вы можете не знать теорему Байеса (хотя теперь уже знаете), но в какой-то степени каждый из десяти миллиардов нейронов в вашем мозге — это ее крохотный частный случай.

Наивный байесовский алгоритм — хорошая концептуальная модель обучающегося алгоритма для чтения прессы: улавливает попарные корреляции между каждым из входов и выходов. Но машинное обучение, конечно, не просто парные корреляции, не в большей степени, чем мозг — это нейрон. Настоящее действие начинается, если посмотреть на более сложные паттерны.

От «Евгения Онегина» до Siri

В преддверии Первой мировой войны русский математик Андрей Марков опубликовал статью, где вероятности применялись, помимо всего прочего, к поэзии. В своей работе он моделировал классику русской литературы — пушкинского «Евгения Онегина» — с помощью подхода, который мы сейчас называем цепью Маркова. Вместо того чтобы предположить, что каждая буква сгенерирована случайно, независимо от остальных, Марков ввел абсолютный минимум последовательной структуры: допустил, что вероятность появления той или иной буквы зависит от буквы, непосредственно ей предшествующей. Он показал, что, например, гласные и согласные обычно чередуются, поэтому, если вы видите согласную, следующая буква (если игнорировать знаки пунктуации и пробелы) с намного большей вероятностью будет гласной, чем если бы буквы друг от друга не зависели. Может показаться, что это невеликое достижение, но до появления компьютеров требовалось много часов вручную подсчитывать символы, и идея была довольно новой. Если *гласная*$_i$ — это булева переменная, которая верна, если *i*-я по счету буква «Евгения Онегина» — гласная, и неверна, если она согласная, можно представить модель Маркова как похожий на цепочку график со стрелками между узлами, указывающими на прямую зависимость между соответствующими переменными:

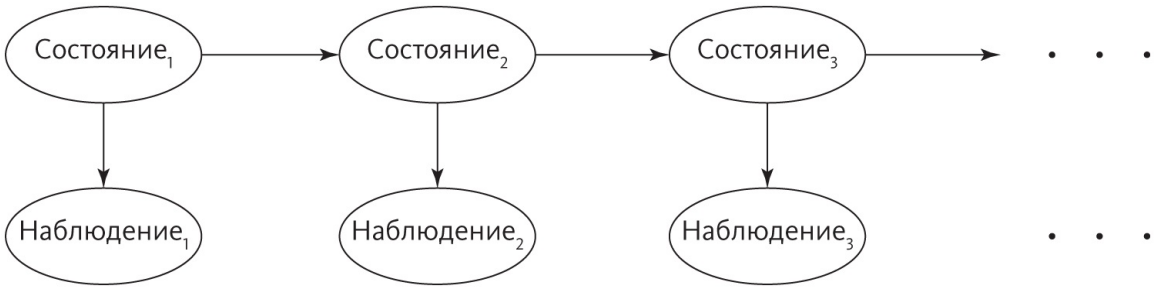


Марков сделал предположение (неверное, но полезное), что в каждом месте текста вероятности одинаковы. Таким образом нам нужно оценить только три вероятности: *P(гласная*$_1$ *= верно)*; *P(гласная*$_{i+1}$ *= верно | гласная*$_i$ *= верно)* и *P(гласная*$_{i+1}$ *= верно | гласная*$_i$ *= верно)*. (Поскольку сумма вероятностей равна единице, из этого можно сразу получить *P(гласная*$_1$ *= ложно)* и так далее.) Как и в случае наивного байесовского алгоритма, переменных можно взять сколько угодно, не опасаясь, что число вероятностей, которые надо оценить, пробьет потолок, однако теперь переменные зависят друг от друга.

Если измерять не только вероятность гласных в зависимости от согласных, но и вероятность следования друг за другом для всех букв алфавита, можно поиграть в генерирование новых текстов, имеющих ту же статистику, что и «Евгений Онегин»: выбирайте первую букву, потом вторую, исходя из первой, и так далее. Получится, конечно, полная чепуха, но, если мы поставим буквы в зависимость от нескольких предыдущих букв, а не от одной, текст начнет напоминать скорее бессвязную речь пьяного — местами разборчиво, хотя в целом бессмыслица. Все еще недостаточно, чтобы пройти тест Тьюринга, но модели вроде этой — ключевой компонент систем машинного перевода, например Google Translate, которые позволяют увидеть весь интернет на английском (или почти английском), независимо от того, на каком языке написана исходная страница.

PageRank — алгоритм, благодаря которому появился Google, — тоже представляет собой марковскую цепь. Идея Ларри Пейджа заключалась в том, что веб-страницы, к которым ведут много ссылок, вероятно, важнее, чем страницы, где их мало, а ссылки с важных страниц должны сами по себе считаться больше. Из-за этого возникает бесконечная регрессия, но и с ней можно справиться с помощью цепи Маркова. Представьте, что человек посещает один сайт за другим, случайно проходя по ссылкам. Состояния в этой цепи Маркова — это не символы, а веб-страницы, что увеличивает масштаб проблемы, однако математика все та же. Суммой баллов страницы тогда будет доля времени, которую человек на ней проводит, либо вероятность, что он окажется на этой странице после долгого блуждания вокруг нее.

Цепи Маркова появляются повсюду, это одна из самых активно изучаемых тем в математике, но это все еще сильно ограниченная разновидность вероятностных моделей. Сделать шаг вперед можно с помощью такой модели:



Состояния, как и ранее, образуют марковскую цепь, но мы их не видим, и надо вывести их из наблюдений. Это называется скрытой марковской моделью, сокращенно СММ (название немного неоднозначное, потому что скрыта не модель, а состояния). СММ — сердце систем распознавания речи, например Siri. В задачах такого рода скрытые состояния — это написанные слова, наблюдения — это звуки, которые слышит Siri, а цель — определить слова на основе звуков. В модели есть два элемента: вероятность следующего слова при известном текущем, как в цепи Маркова, и вероятность услышать различные звуки, когда произносят слово. (Как именно сделать такой вывод — интересная проблема, к которой мы обратимся после следующего раздела.)

Кроме Siri, вы используете СММ каждый раз, когда разговариваете по мобильному телефону. Дело в том, что ваши слова передаются по воздуху в виде потока битов, а биты при передаче искажаются. СММ определяет, какими они должны быть (скрытые состояния), на основе полученных данных (наблюдений), и, если испортилось не слишком много битов, у нее обычно все получается.

Скрытая марковская модель — любимый инструмент специалистов по вычислительной биологии. Белок представляет собой последовательность аминокислот, а ДНК — последовательность азотистых оснований. Если мы хотим предсказать, например, в какую трехмерную форму сложится белок, можно считать аминокислоты наблюдениями, а тип складывания в каждой точке — скрытым состоянием. Аналогично можно использовать СММ для определения мест в ДНК, где инициируется транскрипция генов, а также многих других свойств.

Если состояния и наблюдения — не дискретные, а непрерывные переменные, СММ превращается в так называемый фильтр

Калмана[88]. Экономисты используют эти фильтры, чтобы убрать шум из временных рядов таких величин, как внутренний валовой продукт (ВВП), инфляция и безработица. «Истинные» значения ВВП — это скрытые состояния. На каждом временном отрезке истинное значение должно быть схоже и с наблюдаемым, и с предыдущим истинным значением, поскольку в экономике резкие скачки встречаются нечасто. Фильтр Калмана находит компромисс между этими условиями и позволяет получить более гладкую, но соответствующую наблюдениям кривую. Кроме того, фильтры Калмана не дают ракетам сбиться с курса, и без них человек не побывал бы на Луне.

Все связано, но не напрямую

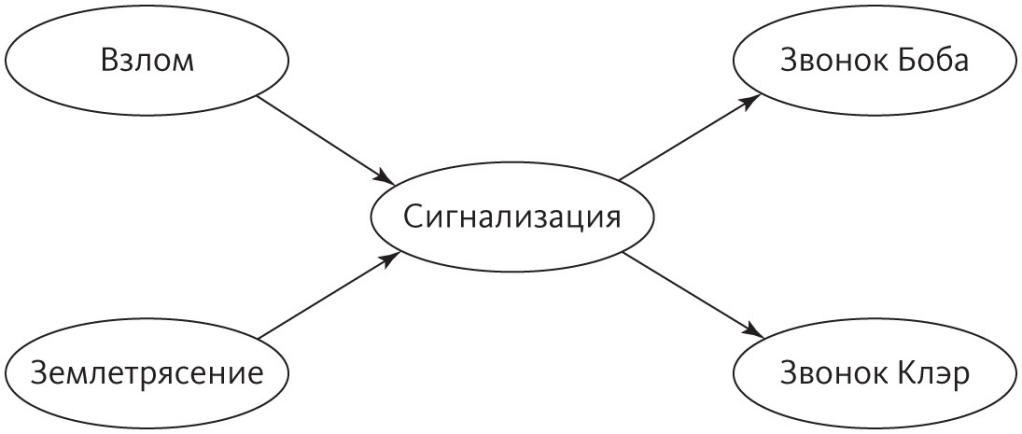
Скрытые марковские модели хорошо подходят для моделирования последовательностей всех видов, но им все еще очень далеко до гибкости символистских правил типа «если…, то…», где условием может быть все, а вывод может стать условием в любом последующем правиле. Однако если допустить такую произвольную структуру на практике, это приведет к взрывному росту количества вероятностей, которые нам надо определить. Ученые долго не могли справиться с этой квадратурой круга и прибегали к ситуативным схемам, например приписывали правилам оценочную достоверность и кое-как их соединяли. Если из *A* с достоверностью 0,8 следует *B*, а из *B* с достоверностью 0,7 вытекает *C*, то, наверное, *C* следует из *A* с достоверностью 0,8 × 0,7.

Проблема таких схем в том, что они могут приводить к сильным искажениям. Из двух совершенно разумных правил «если ороситель включен, трава будет мокрая» и «если трава мокрая, значит шел дождь» я могу вывести бессмысленное правило «если ороситель включен, значит шел дождь». Еще более коварная проблема заключается в том, что при применении правил с оценками достоверности одни и те же доказательства могут засчитываться дважды. Представьте, что вы читаете в New York Times сообщение о приземлении инопланетян. Не исключено, что это розыгрыш, хотя сегодня не первое апреля. Потом вы видите подобные заголовки

в Wall Street Journal, USA Today и Washington Post и начинаете паниковать, как слушатели печально известной передачи Орсона Уэллса, которые приняли радиоспектакль «Война миров» за чистую монету[89]. Если, однако, вы обратите внимание на мелкий шрифт и поймете, что все четыре газеты получили новость от Associated Press, можно снова заподозрить розыгрыш, на этот раз со стороны репортера новостного агентства. Системы правил неспособны справиться с этой проблемой, равно как и наивный байесовский алгоритм: если в качестве предикторов того, что новость правдива, используются такие свойства, как «сообщила New York Times», он может только добавить «сообщило агентство Associated Press», а это лишь испортит дело.

Прорыв был сделан в начале 1980-х годов, когда Джуда Перл, профессор информатики в Калифорнийском университете в Лос-Анджелесе, изобрел новое представление — байесовские сети. Перл — один из самых заслуженных авторитетов в компьютерных науках, и его методы оставили отпечаток в машинном обучении, искусственном интеллекте и многих других дисциплинах. В 2012 году ему была присуждена премия Тьюринга.

Перл понял, что вполне допустимо иметь сложную сеть зависимостей между случайными переменными при условии, что каждая переменная прямо зависит только от нескольких других. Эти зависимости можно представить в виде графика, аналогичного тому, который мы видели при обсуждении цепей Маркова и СММ, однако теперь он может иметь любую структуру, если только стрелки не образуют замкнутые петли. Один из любимых примеров Перла — охранная сигнализация. Она должна сработать, если в дом пытается влезть грабитель, но может включиться и при землетрясении. (В Лос-Анджелесе, где живет Перл, землетрясения почти так же часты, как кражи со взломом.) Представьте, что вы засиделись на работе и вам позвонил сосед Боб, чтобы предупредить, что у вас сработала сигнализация. Соседка Клэр не звонит. Надо ли звонить в полицию? Вот график зависимостей:



Если на этом графике есть стрелка от одного узла к другому, мы говорим, что первый узел — родительский для второго. Следовательно, родители узла «Сигнализация» — «Взлом» и «Землетрясение», а «Сигнализация» будет единственным родителем узлов «Звонок Боба» и «Звонок Клэр». Байесовская сеть — это аналогичный график зависимостей, но каждой переменной присвоена табличка с ее вероятностью при всех сочетаниях ее родителей. У узлов «Взлом» и «Землетрясение» родителей нет, поэтому вероятность у них будет только одна. У «Сигнализации» их будет уже четыре: вероятность срабатывания, если нет ни взлома, ни землетрясения; вероятность срабатывания при взломе, но без землетрясения, и так далее. У узла «Звонок Боба» будут две вероятности (при наличии и отсутствии срабатывания сигнализации), и то же самое для звонка Клэр.

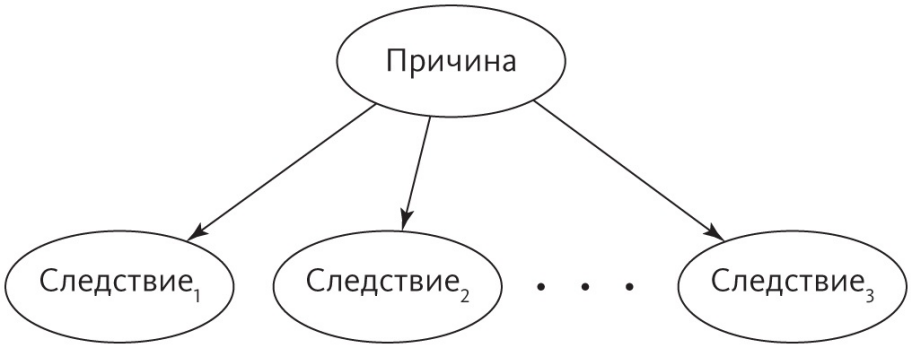
Это ключевой момент. Звонок Боба зависит от узлов «Взлом» и «Землетрясение», но только посредством узла «Сигнализация», то есть условно независим от взлома и землетрясения при условии срабатывания сигнализации, и то же самое с Клэр. Если сигнализация не сработала, соседи будут крепко спать и грабитель спокойно сделает свое дело. Также при условии срабатывания сигнализации звонки Боба и Клэр будут независимы друг от друга. Если бы этой структуры независимости не было, пришлось бы определить $2^5 = 32$ вероятности, по одной для каждого возможного состояния пяти переменных. (Или 31, если вы педант, поскольку последнюю можно оставить неявной.) Если ввести условную независимость, останется определить всего $1 + 1 + 4 + 2 + 2 = 10$ вероятностей, то есть экономия составит 68 процентов. И это

только в этом маленьком примере. Когда переменных сотни и тысячи, экономия может приближаться к 100 процентам.

Первый закон экологии, согласно биологу Барри Коммонеру, заключается в том, что все взаимосвязано. Может быть, это действительно так, но в таком случае мир был бы непостижим, если бы не спасительная условная независимость: все связано, но лишь косвенно. Если что-то происходит на расстоянии километра, повлиять на меня это может только посредством чего-нибудь в моем соседстве, пусть даже это будут световые лучи. Как заметил один шутник, благодаря пространству с нами происходит не все сразу. Иначе говоря, структура пространства — это частный случай условной независимости.

В примере с ограблением полная таблица из 32 вероятностей не представлена явно, но она содержится в наборе меньших таблиц и структуре графа. Чтобы получить *P(взлом, землетрясение, сигнализация, звонок Боба, звонок Клэр)*, нужно только перемножить *P(взлом)*, *P(землетрясение)*, *P(сигнализация | взлом, землетрясение)*, *P(звонок Боба | сигнализация)* и *P(звонок Клэр | сигнализация)*. То же самое в любой байесовской сети: чтобы получить вероятность полного состояния, просто перемножьте вероятности соответствующих линий в таблицах отдельных переменных. Поэтому при условной независимости информация не теряется из-за перехода на более компактное представление, и можно легко вычислить вероятности крайне необычных состояний, включая те, что до этого никогда не наблюдались. Байесовские сети показывают ошибочность расхожего мнения, будто машинное обучение неспособно предсказывать очень редкие события — «черных лебедей», как их называет Нассим Талеб.

Оглядываясь назад, можно заметить, что наивный байесовский алгоритм, цепи Маркова и СММ — это частные случаи байесовских сетей. Вот структура наивного Байеса:



Цепи Маркова кодируют допущение, что будущее условно независимо от прошлого при известном настоящем, а СММ дополнительно предполагает, что каждое наблюдение зависит только от соответствующего состояния. Байесовские сети для байесовцев — то же самое, что логика для символистов: лингва франка[90], который позволяет элегантно кодировать головокружительное разнообразие ситуаций и придумывать алгоритмы, которые будут работать в каждой из них.

Байесовские сети можно воспринимать как «порождающую модель», рецепт вероятностного генерирования состояния мира: сначала надо независимо решить, что произошло — взлом и/или землетрясение, — затем, исходя из этого, понять, срабатывает ли сигнализация, а затем, на этой основе, звонят ли Боб и Клэр. Байесовская сеть как будто рассказывает историю: происходит *A*, которое ведет к *B*. Одновременно с *B* происходит *C*, и вместе они вызывают *D*. Чтобы вычислить вероятность конкретной истории, надо просто перемножить все вероятности в разных ее цепочках.

Одна из самых потрясающих областей применения байесовских сетей — моделирование взаимной регуляции генов в клетке. Попытки открыть парные корреляции между конкретными генами и заболеваниями стоили миллиарды долларов, но дали обидно скромные результаты. Теперь это не удивляет — ведь поведение клетки складывается из сложнейших взаимодействий между генами и средой, и единичный ген имеет ограниченную прогностическую силу. Однако благодаря байесовским сетям мы можем открыть эти взаимодействия при условии, что в нашем распоряжении имеются необходимые данные, а с распространением ДНК-микрочипов данных появляется все больше.

После новаторского применения машинного обучения для фильтрации спама Дэвид Хекерман решил попробовать использовать байесовские сети в борьбе со СПИДом. ВИЧ — сильный противник. Он очень быстро мутирует, поэтому к нему сложно подобрать вакцину и надолго сдержать его с помощью лекарств. Хекерман заметил, что это те же кошки-мышки, как между спамом и фильтрами, и решил применить аналогичный прием: атаковать самое слабое звено. В случае спама слабыми звеньями были в том числе URL, которые приходится использовать для приема платежа от клиента. В случае ВИЧ ими оказались небольшие участки вирусного белка, которые не могут меняться без ущерба для вируса. Если бы получилось натренировать иммунную систему узнавать такие области и атаковать содержащие их клетки, было бы несложно разработать вакцину от СПИДа. Хекерман и его коллеги использовали байесовскую сеть, чтобы выявить эти уязвимые области, и разработали механизм доставки вакцины, которая способна научить иммунную систему атаковать их. Система работает у мышей, и в настоящее время готовятся клинические исследования.

Часто бывает, что даже после того, как все условные независимости учтены, у некоторых узлов байесовской сети все равно остается слишком много родителей. В некоторых сетях так густо от стрелок, что, если их распечатать, страница будет черной (физик Марк Ньюман называет их «абсурдограммы»). Врачу нужно одновременно диагностировать все возможные у пациента заболевания, а не только одно, а каждая болезнь — это родительский узел многих симптомов. Высокая температура может быть вызвана не только гриппом, а любым количеством состояний, и совершенно безнадежно пытаться предсказать ее вероятность при каждом возможном их сочетании. Но не все потеряно. Вместо таблицы, в которой указаны условные вероятности узла для каждого состояния родителей, можно получить более простое распределение. Самый популярный вариант — это вероятностная версия операции «логическое ИЛИ»: любая причина сама по себе может вызвать высокую температуру, но каждая причина с определенной вероятностью этого не сделает, даже если обычно ее достаточно. Хекерман и другие обучили байесовские сети, которые

диагностируют таким образом сотни инфекционных заболеваний, а Google применяет гигантские сети такого рода в AdSense — системе автоматического подбора рекламы для размещения на веб-страницах. Сети связывают миллионы относящихся к контенту переменных друг с другом и с 12 миллионами слов и фраз более чем 300 миллионами стрелок, каждая из которых получена на основе сотни миллиардов отрывков текстов и поисковых запросов.

Более веселый пример — сервис Microsoft Xbox Live, в котором байесовская сеть используется для оценки игроков и подбора игроков схожего уровня. Результат игры — вероятностная функция уровня навыков противника, и благодаря теореме Байеса можно сделать вывод о навыках игрока на основании его побед и поражений.

Проблема логического вывода

Во всем этом есть, к сожалению, большая загвоздка. Само то, что байесовская сеть позволяет компактно представлять вероятностное распределение, еще не означает, что с ее помощью можно эффективно рассуждать. Представьте, что вы хотите вычислить *P(взлом | звонок Боба, нет звонка Клэр)*. Из теоремы Байеса вы знаете, что это просто *P(взлом) P(звонок Боба, нет звонка Клэр | взлом) / P(звонок Боба, нет звонка Клэр)*, или, эквивалентно, *P(взлом, звонок Боба, нет звонка Клэр) / P(звонок Боба, нет звонка Клэр)*. Если бы в нашем распоряжении была полная таблица вероятностей всех состояний, можно было бы вычислить обе эти вероятности, сложив соответствующие строки в таблице. Например, *P(звонок Боба, нет звонка Клэр)* — это сумма вероятностей во всех линейках, где Боб звонит, а Клэр не звонит. Но байесовская сеть не дает полной таблицы. Ее всегда можно построить на основе отдельных таблиц, но необходимое для этого время и пространство растет экспоненциально. На самом деле мы хотим вычислить *P(взлом | звонок Боба, нет звонка Клэр)* без построения полной таблицы. К этому, в сущности, сводится проблема логического вывода в байесовских сетях.

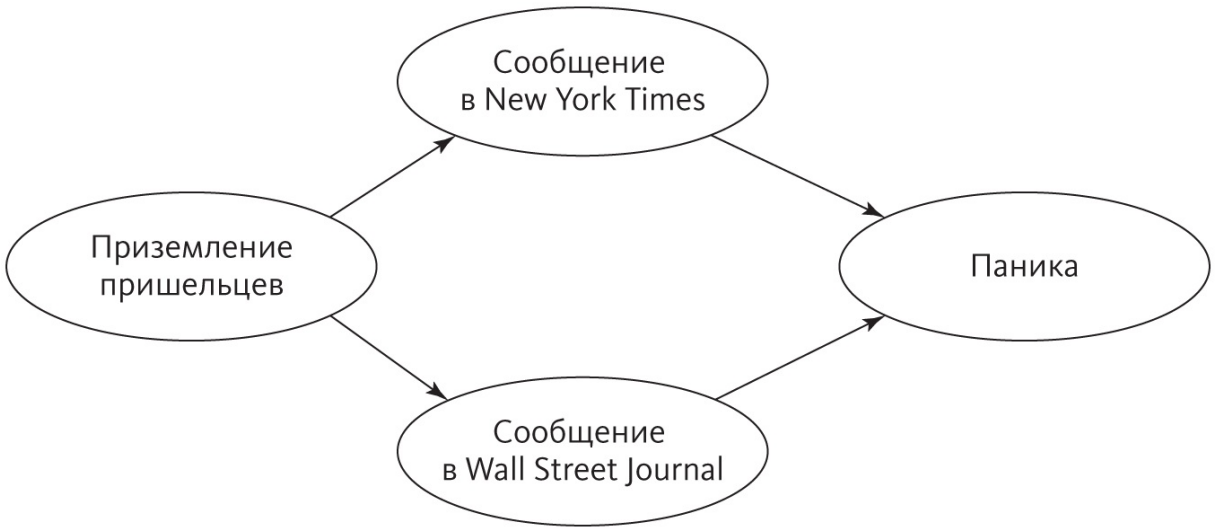
Во многих случаях удается это сделать и без экспоненциального взрыва. Представьте, что вы командир отряда, который колонной

по одному глубокой ночью пробирается по вражеским тылам. Надо убедиться, что никто не отстал и не потерялся. Можно остановиться и всех пересчитать, но нет времени. Более разумное решение — спросить идущего за вами солдата, сколько за ним человек. Солдаты будут задавать тот же самый вопрос по цепочке друг другу, пока последний не скажет: «Никого нет». Теперь предпоследний солдат может сказать: «один», — и так далее обратно к голове колонны, и каждый солдат будет добавлять единицу к количеству за ним. Так вы узнаете, сколько солдат за вами идет, и останавливаться не придется.

В Siri та же самая идея используется, чтобы по звукам, которые она улавливает микрофоном, вычислить вероятность, что вы сказали, например, «позвони в полицию». Эту фразу можно считать отрядом слов, марширующих друг за другом по странице. Слово «полиция» хочет знать вероятность своего появления, но для этого ему надо определить вероятность «в», а предлог, в свою очередь, должен узнать вероятность слова «позвони». Поэтому «позвони» вычисляет собственную вероятность и передает ее предлогу «в», который делает то же самое и передает результат слову «полиция». Теперь «полиция» знает свою вероятность, на которую повлияли все слова в предложении, и при этом не надо составлять полную таблицу из восьми вероятностей (первое слово «позвони» или нет, второе слово «в» или нет, третье слово «полиция» или нет). В реальности Siri учитывает все слова, которые могли бы появиться в каждой позиции, а не просто стоит ли первым слово «позвони» и так далее, однако алгоритм тот же самый. Наверное, Siri слышит звуки и предполагает, что первое слово либо «позвони», либо «позови», второе либо «в», либо «к», а третье — либо «полицию», либо «позицию». Может быть, по отдельности самые вероятные слова — это «позови», «к» и «позицию». Но тогда получится бессмысленное предложение: «Позови к позицию». Поэтому, принимая во внимание другие слова, Siri делает вывод, что на самом деле предложение — «Позвони в полицию». Программа звонит, и, к счастью, полицейские успевают вовремя и ловят забравшегося к вам вора.

Та же идея работает и в случае, если граф[91] представляет собой не цепь, а дерево. Если вместо взвода вы командуете целой армией,

то можете спросить каждого ротного, сколько солдат за ним идет, а потом сложить их ответы. Каждый командир роты, в свою очередь, спросит своих взводных и так далее. Но если граф образует петлю, у вас появятся проблемы. Допустим, какой-то офицер-связной входит сразу в два взвода. Тогда два раза посчитают не только его самого, но и всех идущих за ним солдат. Именно это произойдет в примере с высадкой инопланетян, если вы захотите вычислить, скажем, вероятность паники:



Одно из решений — соединить «Сообщение в New York Times» и «Сообщение в Wall Street Journal» в одну мегапеременную с четырьмя значениями: «ДаДа», если сообщают оба источника, «ДаНет», если о приземлении сообщает New York Times, но не Wall Street Journal, и так далее. Это превратит граф в цепочку из трех переменных, и все хорошо. Однако с добавлением каждого нового источника новости число значений в мегапеременной будет удваиваться. Если вместо двух источников у вас целых 50, мегапеременная получит 2^{50} значений. Поэтому такой метод на большее не способен, а ничего лучше не придумали.

Проблема сложнее, чем может показаться на первый взгляд, потому что у байесовских сетей появляются «невидимые» стрелки, идущие вместе с видимыми. «Взлом» и «землетрясение» априорно независимы, но сработавшая сигнализация их связывает: она заставляет подозревать ограбление. Но если вы услышите по радио, что было землетрясение, то начнете предполагать, что оно

и виновато. Землетрясение *оправдывает* срабатывание сигнализации и делает ограбление менее вероятным, а следовательно, между ними появляется зависимость. В байесовской сети все родители той же переменной таким образом оказываются взаимозависимы, что, в свою очередь, порождает еще больше зависимостей, и результирующий граф часто получается намного плотнее, чем исходный.

Критически важный вопрос для логического вывода: можно ли сделать заполненный график «похожим на дерево», чтобы ствол при этом был не слишком толстый. Если у мегапеременной в стволе слишком много возможных значений, дерево будет расти бесконтрольно, пока не заполонит всю планету, как баобабы в «Маленьком принце». В древе жизни каждый вид — это ветвь, но внутри каждой ветви есть граф, где у каждого создания имеются двое родителей, четыре внука, какое-то количество потомков и так далее. «Толщина» ветви — это размер популяции данного вида. Если ветви слишком толстые, единственный выбор — прибегнуть к приближенному выводу.

Одно из решений, приведенное Перлом в качестве упражнения в его книге о байесовских сетях, — делать вид, что в графе нет петель, и продолжать распространять вероятности туда-сюда, пока они не сойдутся. Такой алгоритм известен как циклическое распространение доверия. Вообще говоря, идея странная, но, как оказалось, во многих случаях она довольно хорошо работает. Например, это один из современных методов беспроводной связи, где случайные переменные — хитрым образом закодированные биты сообщения. Но циклическое распространение доверия может сходиться и к неправильным ответам или бесконечно изменяться (осциллировать). Еще одно решение проблемы возникло в физике, было импортировано в машинное обучение и значительно расширено Майклом Джорданом и другими учеными. Оно заключается в том, чтобы приблизить неразрешимое распределение с помощью разрешимого, оптимизировать параметры последнего и как можно ближе приблизить его к первому.

Но самый популярный вариант — утопить наши печали в вине и бродить всю ночь пьяным. По-научному это называется метод Монте-Карло в марковских цепях, или сокращенно МСМС. «Монте-

Карло» потому, что в методе есть шансы, как в казино в одноименном городе, а марковские цепи упоминаются потому, что в этот метод входит последовательность шагов, каждый из которых зависит только от предыдущего. Идея МСМС заключается в том, чтобы совершать случайные прогулки, как упомянутый пьяница, перепрыгивая из одного состояния сети в другое таким образом, чтобы в долгосрочной перспективе число посещений каждого состояния было пропорционально вероятности этого состояния. Затем мы можем оценить вероятность взлома, скажем, как долю посещений состояния, в котором происходит ограбление. Удобная для анализа цепь Маркова сводится к стабильному распределению и через какое-то время начинает давать приблизительно те же ответы. Например, если вы тасуете колоду карт, через какое-то время все порядки карт станут одинаково вероятными, независимо от исходного порядка, поэтому вы будете знать, что при n возможных вариантов вероятность каждого из них будет равна $1/n$. Весь фокус в МСМС заключается в том, чтобы разработать цепь Маркова, которая сходится к распределению нашей байесовской сети. Простой вариант — многократно циклически проходить через переменные, делая выборку каждой в соответствии с ее условной вероятностью, исходя из состояния соседей. Люди часто говорят об МСМС как о своего рода симуляции, но это не так: цепь Маркова не имитирует какой-то реальный процесс — скорее, она придумана для того, чтобы эффективно генерировать примеры из байесовской сети, которая сама по себе не является последовательной моделью.

Истоки МСМС восходят к Манхэттенскому проекту, в котором физики оценивали вероятность столкновения нейтронов с атомами, вызывающего цепную реакцию. В последующие десятилетия метод произвел такую революцию, что его часто называют одним из самых важных алгоритмов всех времен. МСМС хорош не только для вычисления вероятностей, но и для интегрирования любых функций. Без него ученые были бы ограничены функциями, которые можно интегрировать аналитически, или низкоразмерными, удобными для анализа интегралами, которые можно приблизить методом трапеций. МСМС позволяет свободно строить сложные модели, делая всю трудную работу за вас.

Байесовцы, со своей стороны, обязаны МСМС растущей популярностью своих методов, вероятно, больше, чем чему-то другому.

Отрицательный момент — то, что МСМС зачастую мучительно медленно сходится или начинает обманывать, потому что кажется, что он сошелся, а на самом деле нет. В реальных распределениях обычно очень много пиков, которые, как Эвересты, взлетают над широкой равниной крохотных вероятностей. Цепь Маркова, следовательно, будет сходиться к ближайшему пику и останется там, а оценка вероятности окажется очень пристрастной: как если бы пьяница учуял запах спиртного и завис на всю ночь в ближайшей забегаловке, вместо того чтобы бесцельно слоняться по городу, как нам нужно. С другой стороны, если вместо цепи Маркова сгенерировать независимые пробы, как в более простых методах Монте-Карло, никакого запаха не будет и, вероятно, наш пьяница даже не найдет первый кабак. Это все равно что бросать дротики в карту города, надеясь, что они попадут прямиком в паб.

Логический вывод в байесовских сетях не ограничен вычислением вероятностей. К нему относится и нахождение наиболее вероятного объяснения признаков, например заболевания, которое лучше всего объясняет симптомы, или слов, которые лучше всего объясняют звуки, услышанные Siri. Это не то же самое, что просто выбрать на каждом этапе самое вероятное слово, потому что слова, которые схожи по отдельности исходя из звуков, могут реже встречаться вместе, как в примере «Позови к позицию». Однако и в таких задачах срабатывают аналогичные виды алгоритмов (именно их использует большинство распознавателей речи). Самое главное, что вывод предусматривает принятие наилучших решений не только на основе вероятности разных исходов, но и с учетом соответствующих затрат (или, говоря научным языком, полезности). Затраты, связанные с проигнорированным письмом от начальника, который просит что-то сделать завтра, будут намного выше, чем затраты на ознакомление с ненужным рекламным письмом, поэтому часто целесообразно пропустить письма через фильтр, даже если они довольно сильно напоминают спам.

Беспилотные автомобили и другие роботы — показательный пример работы вероятностного вывода. Машина ездит туда-сюда, создает карту территории и все увереннее определяет свое положение. Согласно недавнему исследованию, у лондонских таксистов увеличиваются размеры задней части гиппокампа — области мозга, участвующей в создании карт и запоминании, — когда они учатся ориентироваться в городе. Наверное, здесь действуют аналогичные алгоритмы вероятностного вывода с той лишь важной разницей, что людям алкоголь, по-видимому, не помогает.

Учимся по-байесовски

Теперь, когда мы знаем, как (более-менее) решать проблему логического вывода, можно приступать к обучению байесовских сетей на основе данных, ведь для байесовцев обучение — это всего лишь очередная разновидность вероятностного вывода. Все что нужно — применить теорему Байеса, где гипотезы будут возможными причинами, а данные — наблюдаемым следствием:

$$P(\text{гипотеза} \mid \text{данные}) = P(\text{гипотеза}) \times P(\text{данные} \mid \text{гипотеза}) / P(\text{данные}).$$

Гипотеза может быть сложна, как целая байесовская сеть, или проста, как вероятность, что монетка упадет орлом вверх. В последнем случае данные — это просто результат серии подбрасываний. Если, скажем, мы получили 70 орлов в сотне подбрасываний, сторонник частотного подхода оценит, что вероятность выпадения орла составляет 0,7. Это оправдано так называемым принципом наибольшего правдоподобия: из всех возможных вероятностей орлов 0,7 — то значение, при котором вероятность увидеть 70 орлов при 100 подбрасываниях максимальна. Эта вероятность — $P(\text{данные} \mid \text{гипотеза})$, и принцип гласит, что нужно выбирать гипотезу, которая ее максимизирует. Байесовцы, однако, поступают разумнее. Они говорят, что никогда точно не известно, какая из гипотез верна, и поэтому нельзя просто выбирать одну гипотезу, например значение 0,7 для вероятности выпадения орла. Надо скорее вычислить апостериорную

вероятность каждой возможной гипотезы и при прогнозировании учесть все. Сумма вероятностей должна равняться единице, поэтому, если какая-то гипотеза становится вероятнее, вероятность других уменьшается. Для байесовца на самом деле не существует такого понятия, как истина: есть априорное распределение гипотез, и после появления данных оно становится апостериорным распределением по теореме Байеса. Вот и все.

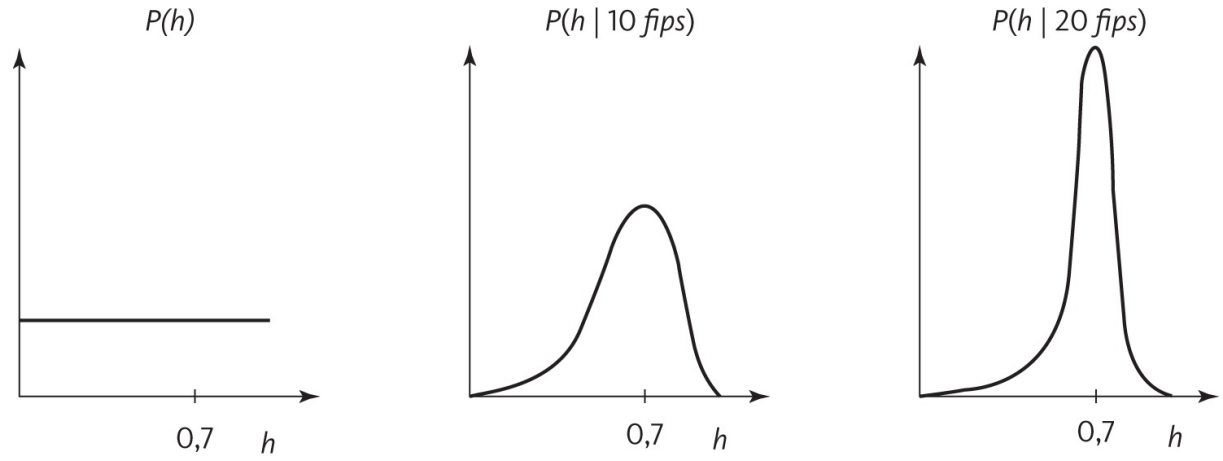
Это радикальный отход от традиционных научных методов. Все равно что сказать: «Вообще, ни Коперник, ни Птолемей не правы. Давайте лучше предскажем будущие траектории планет исходя из того, что Земля вращается вокруг Солнца, а потом — что Солнце вращается вокруг Земли, а результаты усредним».

Конечно, здесь речь идет о взвешенном среднем, где вес гипотезы — это ее апостериорная вероятность, поэтому гипотеза, которая лучше объясняет данные, будет иметь большее значение. Тем не менее ученые шутят, что быть байесовцем — значит никогда не говорить, что ты хоть в чем-то уверен.

Не стоит упоминать, что постоянно таскать за собой множество гипотез вместо одной тяжело. При обучении байесовской сети приходится делать предсказания путем усреднения всех возможных сетей, включая все возможные структуры графов и все возможные параметры значений для каждой структуры. В некоторых случаях можно вычислить среднее по параметрам в замкнутой форме, но с варьирующимися структурами такого везения ждать не приходится. Остается, например, применить МСМС в пространстве сетей, перепрыгивая из одной возможной сети к другой по ходу цепи Маркова. Соедините эту сложность и вычислительные затраты с неоднозначностью байесовской идеи о том, что объективной реальности вообще не существует, и вы поймете, почему в науке последние 100 лет доминирует частотный подход.

У байесовского метода есть, однако, спасительное свойство и ряд серьезных плюсов. В большинстве случаев апостериорная вероятность практически всех гипотез чрезвычайно мала и их можно спокойно проигнорировать: даже рассмотрение одной, наиболее вероятной гипотезы обычно дает очень хорошее приближение. Представьте, что наше априорное распределение для

проблемы броска монетки заключается в том, что все вероятности орлов одинаково правдоподобны. После появления результатов последовательных подбрасываний распределение будет все больше и больше концентрироваться на гипотезе, которая лучше всего согласуется с данными. Например, если *h* пробегает по возможным вероятностям орлов, а монета падает орлом вверх в 70 процентах случаев, получится что-то вроде:



Апостериорная вероятность броска становится априорной для следующего броска, и, бросок за броском, мы все больше убеждаемся, что $h = 0{,}7$. Если просто взять одну наиболее вероятную гипотезу (в данном случае $h = 0{,}7$), байесовский подход станет довольно похож на частотный, но с одним очень важным отличием: байесовцы учитывают априорную *P(гипотеза)*, а не просто вероятность *P(данные | гипотеза)*. (Данные до *P(данные)* можно проигнорировать, потому что они одинаковы для всех гипотез и, следовательно, не влияют на выбор победителя.) Если мы хотим сделать допущение, что все гипотезы априори одинаково вероятны, байесовский подход сведется к принципу наибольшего правдоподобия. Поэтому байесовцы могут заявить сторонникам частотного подхода: «Смотрите, то, что вы делаете, — частный случай того, что делаем мы, но наши допущения хотя бы явные». А если гипотезы не одинаково правдоподобны априори, неявное допущение наибольшей правдоподобности заключается в том, что они ведут к неправильным ответам.

Это может показаться чисто теоретической дискуссией, но на самом деле ее практические последствия огромны. Если мы видели, что монету подбросили только один раз и выпал орел, принцип наибольшего правдоподобия подскажет, что вероятность выпадения орла должна быть равна единице. Это будет крайне неточно, и мы окажемся совершенно неподготовлены, если монетка упадет решкой. После многократных подбрасываний оценка станет надежнее, но во многих проблемах подбрасываний никогда не будет достаточно, как бы ни был велик объем данных. Представьте, что в наших обучающих данных слово «суперархиэкстраультрамегаграндиозно» никогда не появляется в спаме, но однажды встречается в письме про Мэри Поппинс. Спам-фильтр, основанный на наивном байесовском алгоритме с оценкой вероятности наибольшего правдоподобия, решит, что такое письмо не может быть спамом, пусть даже все остальные слова вопиют: «Спам! Спам!» Напротив, сторонник байесовского подхода дал бы этому слову низкую, но не нулевую вероятность появления в спаме, и в таком случае другие слова бы его перевесили.

Проблема лишь усугубится, если попытаться узнать и структуру байесовской сети, и ее параметры. Мы можем сделать это путем восхождения по выпуклой поверхности, начиная с пустой сети (без стрелок) и добавляя стрелки, которые больше всего увеличивают вероятность, пока ни одна из них не будет приводить к улучшению. К сожалению, это быстро вызовет очень сильное переобучение, и получится сеть, приписывающая нулевую вероятность всем состояниям, которые не появляются в данных. Байесовцы могут сделать нечто гораздо более интересное: использовать априорное распределение, чтобы закодировать экспертное знание о проблеме. Это их ответ на вопрос Юма. Например, можно разработать исходную байесовскую сеть для медицинской диагностики, опросив врачей, какие заболевания, по их мнению, вызывают те или иные симптомы, и добавить соответствующие стрелки. Это «априорная сеть», и априорное распределение может штрафовать альтернативные сети по числу стрелок, которые они в нее добавляют или убирают. Тем не менее врачам свойственно ошибаться, и данным разрешено перевесить их мнение: если рост

правдоподобия в результате добавления стрелки перевешивает штраф, она будет добавлена.

Конечно, сторонникам частотного подхода известно об этой проблеме, и у них есть свои решения: например, умножить правдоподобие на фактор, который штрафует более сложные сети. Но в этот момент частотный и байесовский подходы становятся неразличимыми, и как вы назовете функцию, подсчитывающую очки: «оштрафованным правдоподобием» или «апостериорной вероятностью», — просто дело вкуса.

Несмотря на то что частотный и байесовский типы мышления по некоторым вопросам сходятся, между ними остается философское различие в отношении значения вероятности. Многим ученым неприятно рассматривать его как нечто субъективное, хотя благодаря этому становятся возможными многие применения, которые в противном случае запрещены. Если вы сторонник частотного подхода, можно оценивать вероятности только тех событий, которые происходят более одного раза, и вопросы вроде «Какова вероятность, что Хиллари Клинтон победит Джеба Буша на следующих президентских выборах?» не имеют ответа, потому что еще не было выборов, в которых сошлись бы эти кандидаты. Для байесовца же вероятность — субъективная степень веры, поэтому он волен выдвигать обоснованные предположения, и анализ суждений делает все его предположения состоятельными.

Байесовский метод применим не только к обучению байесовских сетей и их частных случаев. (Наоборот, вопреки названию, байесовские сети не обязательно байесовские: сторонники частотного подхода тоже могут их обучать, как мы только что видели.) Можно применить априорное распределение к любому классу гипотез — наборам правил, нейронным сетям, программам, — а затем обновлять правдоподобие гипотез при получении данных. Байесовская точка зрения заключается в том, что вы можете выбирать представление, но затем его надо обучать с помощью теоремы Байеса. В 1990-х годах байесовцы произвели эффектный захват Конференции по системам обработки нейронной информации (Neural Information Processing Systems, NIPS) — главного мероприятия для коннекционистских исследований. Зачинщиками были Дэвид Маккей, Редфорд Нил и Майкл Джордан.

Маккей, британец и студент Джона Хопфилда в Калифорнийском техническом университете, позднее ставший главным научным консультантом Департамента энергетики Великобритании, показал, как обучать по-байесовски многослойные перцептроны, Нил познакомил коннекционистов с MCMC, а Джордан — с вариационным выводом. Наконец, они указали, что в пределе можно «проинтегрировать» нейроны многослойного перцептрона, оставляя тип байесовской модели, которая на них не ссылается. Вскоре после этого слово «нейронный» в заголовках статей, поданных на конференцию по системам обработки нейронной информации, стало резко уменьшать шансы на публикацию. Некоторые шутили, что надо переименовать NIPS в BIPS — «Байесовские системы обработки информации».

Марков взвешивает доказательства

Байесовцы шли к мировому господству, но тут произошло нечто забавное. Ученые, пользующиеся байесовскими моделями, стали постоянно замечать, что результат получается лучше, если манипулировать вероятностями недозволенными методами. Например, возведение *P(слова)* в определенную степень улучшало точность распознавания речи, но тогда это переставало быть теоремой Байеса. Что произошло? Как оказалось, виновата ложная независимость допущений, которые делают порождающие модели. Благодаря упрощенной структуре графа модели становятся обучающимися и стоящими сохранения, но тогда больше даст простое получение наилучших параметров для имеющейся задачи, независимо от того, представляют ли они собой вероятности. Настоящая сила, скажем, наивного байесовского алгоритма заключается в том, что он дает небольшой информативный набор свойств, на основании которого можно предсказать класс, а также быстрый надежный способ узнать соответствующие параметры. В спам-фильтре каждое свойство — это частота определенного слова в спаме, а соответствующий параметр — то, как часто оно встречается. То же самое для не-спама. Если смотреть с этой точки зрения, наивный байесовский алгоритм может оказаться оптимальным в том смысле, что он делает лучшие возможные

предсказания, причем зачастую там, где независимость допущений сильно нарушена. Когда я это понял и в 1996 году опубликовал статью на эту тему, подозрение к наивному Байесу уменьшилось и его популярность выросла. Но это стало шагом на пути к модели другого рода, которая в последние два десятилетия все больше вытесняет байесовские сети из машинного обучения, — к сетям Маркова.

Сеть Маркова — это набор свойств и соответствующих весов, которые вместе определяют распределение вероятности. Свойство может быть простое, например «это медленная песня», или сложное, например «это медленный хип-хоп с саксофонной импровизацией и нисходящим аккордовым пассажем». В сервисе Pandora используется большой набор свойств, который создатели называют Music Genome Project. Он помогает отобрать песни, которые вам стоит послушать. Представьте, что мы подключили ее к сети Маркова. Если вы любите медленные песни, веса соответствующих свойств пойдут вверх и вы с большей вероятностью будете слышать такую музыку, если включите Pandora. Если вы при этом любите исполнителей хип-хопа, вес этого свойства тоже вырастет. Песни, которые вы, скорее всего, услышите, теперь объединят обе черты: это будут медленные композиции в исполнении хип-хоперов. Если вы не любите медленные песни или хип-хоперов как таковых и вам нравится только их сочетание, понадобится более сложная черта — «медленная песня в исполнении хип-хопера». Свойства Pandora созданы вручную, но в сетях Маркова их можно получить путем восхождения на выпуклые поверхности, аналогично выведению правил. В любом случае хороший способ узнать вес*а* — градиентный спуск.

Как и байесовские сети, сети Маркова можно представить в виде графов, но вместо стрелок будут ненаправленные дуги. Когда две переменные соединены, это значит, что они прямо зависят друг от друга, если появляются вместе в каком-то свойстве, как «медленная песня» и «песня в исполнении хип-хопера» в свойстве «медленная песня в исполнении хип-хопера».

Сети Маркова — базовая методика во многих областях, например компьютерном зрении. В частности, беспилотный

автомобиль должен разделить любое увиденное изображение на дорогу, небо и окружающую местность. Один из вариантов — присвоить каждому пикселю, в зависимости от цвета, один из трех ярлыков, но этого совершенно недостаточно. Изображения очень зашумлены и разнообразны, поэтому у машины постоянно будут галлюцинации: разбросанные по всей дороге камни, куски дороги в небе. В то же время известно, что соседние пиксели на изображении обычно входят в один и тот же объект, и можно ввести соответствующий набор свойств: для каждой пары соседних пикселей свойство верно, если пиксели относятся к тому же самому объекту, и ложно, если это не так. Теперь изображения с крупными, сплошными блоками дороги и неба становятся намного более вероятными, чем изображения без них, и машина начинает ехать прямо, вместо того чтобы вилять то влево, то вправо, уворачиваясь от привидевшихся булыжников.

Сети Маркова можно обучить максимизировать либо правдоподобие всех данных, либо условную функцию правдоподобия того, что мы хотим предсказать, исходя из имеющихся знаний. Для Siri вероятность всех данных — это *P(слова, звуки)*, а условное правдоподобие того, что нас интересует, — *P(слова | звуки)*. Оптимизируя последнее, можно проигнорировать *P(звуки)*, потому что оно только отвлекает от цели, и, если его не проигнорировать, оно может быть произвольно сложным. Это намного лучше, чем нереалистичное допущение СММ, что звуки зависят исключительно от соответствующих слов без какого-либо влияния среды. На самом деле Siri важно только понять, какие слова вы произнесли, и, наверное, даже не стоит беспокоиться о вероятностях: достаточно просто убедиться, что при подсчете весов свойств у правильных слов сумма пунктов будет больше, чем у неправильных. В идеале намного больше, просто для безопасности.

Как мы увидим в следующей главе, аналогизаторы довели эту линию рассуждения до логического завершения и в первом десятилетии нового тысячелетия завоевали конференцию NIPS. Коннекционисты еще раз взяли верх, теперь уже под знаменем глубокого обучения. Некоторые говорят, что наука развивается циклами, но она больше похожа на спираль вокруг вектора

прогресса. Спираль машинного обучения сходится в Верховном алгоритме.

Логика и вероятность: несчастная любовь

Вы ошибаетесь, если думаете, что байесовцы и символисты отлично поладят, потому что и те и другие верят в теоретический, а не естественно-научный подход к обучению. Символисты не любят вероятностей и рассказывают анекдоты вроде «Сколько байесовцев нужно, чтобы поменять лампочку? Они сами точно не знают. Если подумать, они даже не уверены, перегорела ли лампочка». А если серьезно, символисты показывают, какую высокую цену приходится платить за вероятность. Логический вывод внезапно становится намного затратнее, все эти числа сложно понять, надо что-то делать с априорной информацией и постоянно убегать от полчищ гипотез-зомби. Пропадает столь милая сердцу символистов способность на лету складывать элементы знаний. Хуже всего то, что неизвестно, как применить распределение вероятностей ко многим проблемам, которые нам надо решить. Байесовская сеть — это распределение по вектору переменных. А что с распределениями по сетям, базам данных, базам знаний, языкам, планам, компьютерным программам и многому другому? Со всем этим легко справляется логика, и, раз алгоритм на это неспособен, это явно не Верховный алгоритм.

Байесовцы, в свою очередь, указывают на хрупкость логики. Если у меня есть правило вроде «Птицы летают», мир даже с одной нелетающей птицей невозможен. Если попытаться залатать все дыры исключениями, например «Птицы летают, если они не пингвины», их получится бесконечно много. (Что со страусами? С птицами в клетке? Мертвыми птицами? Птицами со сломанными крыльями? С промокшими крыльями?) Врач диагностирует рак, и больной решает проконсультироваться еще у одного специалиста. Если второй доктор не согласен с первым, ситуация заходит в тупик. Мнения нельзя взвесить, приходится просто верить обоим. В результате происходит катастрофа: свиньи летают, вечный двигатель возможен, а Земли не существует — потому что в логике из противоречий можно вывести что угодно. Более того, если

знание получено из данных, никогда нельзя быть уверенным, что оно истинно. Почему символисты делают вид, что это не так? Юм, несомненно, не одобрил бы такую беззаботность.

Байесовцы и символисты соглашаются, что априорные допущения неизбежны, но расходятся в том, какое априорное знание разрешено. Для байесовцев знание выражается в априорном распределении по структуре и параметрам модели. Априорными параметрами в принципе может быть все что угодно, но, по иронии, байесовцы, как правило, выбирают неинформативные (например, приписывают всем гипотезам одну и ту же вероятность), потому что им так удобнее делать расчеты. И в любом случае люди не очень хорошо умеют оценивать вероятности. Что касается структуры, байесовские сети предполагают интуитивное инкорпорирование знаний: нарисуй стрелку из *A* в *B*, если думаешь, что *A* прямо вызывает *B*. Символисты намного гибче: можно дать алгоритму машинного обучения в качестве априорного знания все, что можно закодировать путем логики, а логикой можно закодировать практически все при условии, что это «все» — черно-белое.

Очевидно, что нужны и логика, и вероятности. Хороший пример — лечение рака. Байесовская сеть может моделировать отдельный аспект функционирования клеток, например регуляцию генов или фолдинг белка, но только логика может сложить фрагменты в связную картину. С другой стороны, логика не может работать с неполной или зашумленной информацией, которой очень много в экспериментальной биологии, а байесовские сети прекрасно с этим справляются.

Байесовское обучение работает на одной таблице данных, где столбцы представляют переменные (например, уровень экспрессии гена), а строки — случаи (например, наблюдаемый в отдельных экспериментах с микрочипом уровень экспрессии каждого гена). Ничего страшного, если в таблице есть «дыры» и ошибочные измерения, потому что можно применить вероятностный вывод, чтобы заполнить пробелы и сгладить ошибки путем усреднения. Но когда таблиц больше, байесовское обучение заходит в тупик. Оно не знает, например, как соединить данные об экспрессии генов с данными о том, какой сегмент ДНК кодирует белки, и как, в свою очередь, трехмерная форма этих белков заставляет их

прикрепляться к разным частям молекулы ДНК, влияя на экспрессию других генов. В случае логики мы легко можем составить правила, связывающие все эти аспекты, и получить соответствующие комбинации таблиц, но только при условии, что в таблицах нет ошибок и белых пятен.

Соединить коннекционизм и эволюционизм довольно легко: просто эволюционируйте структуру сети и получите параметры путем обратного распространения ошибки. Объединить логику и вероятностный подход намного сложнее. Попытки решить эту проблему предпринимал еще Лейбниц, который был пионером в обеих областях, а после него — лучшие философы и математики XIX и XX века, например Джордж Буль и Рудольф Карнап. Несмотря на все усилия, результаты были очень скромными. Позднее в бой вступили информатики и исследователи искусственного интеллекта, но к началу нового тысячелетия они достигли лишь частичных успехов, например, к байесовским сетям добавили логические конструкты. Большинство экспертов были убеждены, что объединить логику и вероятности вообще невозможно. Перспективы создания Верховного алгоритма выглядели неважно, особенно потому, что существовавшие тогда эволюционистские и коннекционистские алгоритмы тоже не могли справиться с неполной информацией и множественными наборами данных.

К счастью, с тех пор мы продвинулись вперед на пути к решению проблемы, и сегодня Верховный алгоритм кажется намного ближе. Решение мы увидим в главе 9, но сначала надо подобрать все еще недостающую очень важную часть мозаики: как учиться, если данных очень мало. Здесь вступает в игру одна из самых важных идей в машинном обучении: аналогия. У всех «племен», которые мы до сих пор встретили, есть одна общая черта: они получают явную модель рассматриваемого явления, будь то набор правил, многослойный перцептрон, генетическая программа или байесовская сеть. Если у них для этого недостаточно данных, они заходят в тупик. А аналогизаторам для обучения достаточно всего одного примера, потому что они модели не формируют. Давайте посмотрим, чем они вместо этого занимаются.

ГЛАВА 7

ТЫ — ТО, НА ЧТО ТЫ ПОХОЖ

Фрэнк Абигнейл-младший — один из самых знаменитых мошенников в истории, Леонардо Ди Каприо сыграл его в фильме Спилберга «Поймай меня, если сможешь». Абигнейл подделывал чеки на миллионы долларов, прикидывался адвокатом и преподавателем колледжа, путешествовал по миру, выдавая себя за пилота Pan Am, и все это когда ему еще не исполнился 21 год. Но, наверное, самая сногсшибательная его проделка — это когда он в конце 1960-х почти год успешно изображал врача в Атланте. Казалось бы, чтобы заниматься медициной, нужно много лет учиться в медицинском институте, пройти ординатуру, получить лицензию и так далее, но Абигнейлу удалось обойти эти мелочи, и все были довольны.

Представьте на секунду, что вам предстоит провернуть нечто подобное. Вы тайком пробираетесь в пустой медицинский кабинет. Вскоре появляется пациент и рассказывает вам о своих симптомах. Надо поставить ему диагноз, только вот в медицине вы ничего не смыслите. В вашем распоряжении — шкаф с историями болезней: симптомы, диагнозы, назначенное лечение и так далее. Как вы поступите? Самое простое — это заглянуть в документы, поискать пациента с самыми похожими симптомами и поставить такой же диагноз. Если вы умеете вести себя с больным и убедительно говорить, как Абигнейл, этого может оказаться достаточно для успеха. Та же идея успешно применяется и за пределами медицины. Если вы молодой президент и столкнулись с мировым кризисом, как в свое время Кеннеди, когда самолет-разведчик обнаружил на Кубе советские ядерные ракеты, вполне вероятно, что готового сценария у вас не окажется. Вместо этого можно поискать похожие примеры в истории и попытаться сделать из них выводы. Объединенный комитет начальников штабов подталкивал президента напасть на Кубу, но Кеннеди только что прочитал The Guns of August[92] — бестселлер о начале Первой мировой войны — и хорошо осознавал, что такой

шаг легко может вылиться в тотальную войну. Кеннеди предпочел морскую блокаду — и, может быть, спас мир от ядерной катастрофы.

Аналогия была искрой, из которой разгорелись величайшие научные достижения в истории человечества. Теория естественного отбора родилась, когда Дарвин, читая An Essay on the Principle of Population («Опыт о законе народонаселения») Мальтуса, был поражен сходством между борьбой за выживание в экономике и в природе. Модель атома появилась на свет, когда Бор увидел в ней миниатюрную Солнечную систему, где электроны соответствовали планетам, а ядро — Солнцу. Кекуле открыл кольцевую форму молекулы бензола, представив себе змею, пожирающую свой хвост.

У рассуждений по аналогии выдающаяся интеллектуальная родословная. Еще Аристотель выразил их в своем законе подобия: если две вещи схожи, мысль об одной из них будет склонна вызывать мысль о другой. Эмпирики, например Локк и Юм, пошли по этому пути. Истина, говорил Ницше, — это движущаяся армия метафор. Аналогии любил Кант, а Уильям Джеймс полагал, что чувство одинаковости — киль и позвоночник человеческого мышления. Некоторые современные психологи даже утверждают, что человеческое познание целиком соткано из аналогий. Мы полагаемся на них, чтобы найти дорогу в новом городе и чтобы понять такие выражения, как «увидеть свет» и «не терять лица». Подростки, которые в каждое предложение вставляют словечко «типа», согласятся, типа, что аналогия — это, типа, важная штука.

С учетом всего этого неудивительно, что аналогия играет видную роль в машинном обучении. Однако дорогу себе она пробивала медленно, и поначалу ее затмевали нейронные сети. Первое воплощение аналогии в алгоритме появилось в малоизвестном отчете, написанном в 1951 году Эвелин Фикс и Джо Ходжесом — статистиками из Университета Беркли, — и потом десятки лет не публиковалось в мейнстримных журналах. Однако тем временем начали появляться, а потом множиться другие статьи об алгоритме Фикс и Ходжеса, пока он не стал одним из самых исследуемых в информатике. Алгоритм ближайшего соседа — так он называется — будет первым шагом в нашем

путешествии по обучению на основе аналогий. Вторым станет метод опорных векторов, который, как буря, налетел на машинное обучение на переломе тысячелетий и лишь недавно встретил достойного соперника в лице глубокого обучения. Третья и последняя тема — это полноценное аналогическое рассуждение, которое несколько десятилетий было базовым в психологии и искусственном интеллекте и примерно столько же — в машинном обучении.

Аналогизаторы — наименее сплоченное из пяти «племен». В отличие от приверженцев других учений, которых объединяет сильное чувство идентичности и общие идеалы, аналогизаторы представляют собой скорее свободное собрание ученых, согласных с тем, что в качестве основы обучения нужно полагаться на суждения о сходстве. Некоторые, например ребята, занимающиеся методом опорных векторов, могут даже не захотеть встать под общий зонтик. Но за окном идет дождь из глубоких моделей, и мне кажется, действовать сообща им не повредит. Аналогия — одна из центральных идей в машинном обучении, и аналогизаторы всех мастей — ее хранители. Может быть, в грядущем десятилетии в машинном обучении будет доминировать глубокая аналогия, соединяющая в один алгоритм эффективность ближайшего соседа, математическую сложность метода опорных векторов и мощь и гибкость рассуждения по аналогии. (Вот я и выдал один из своих секретных научных проектов.)

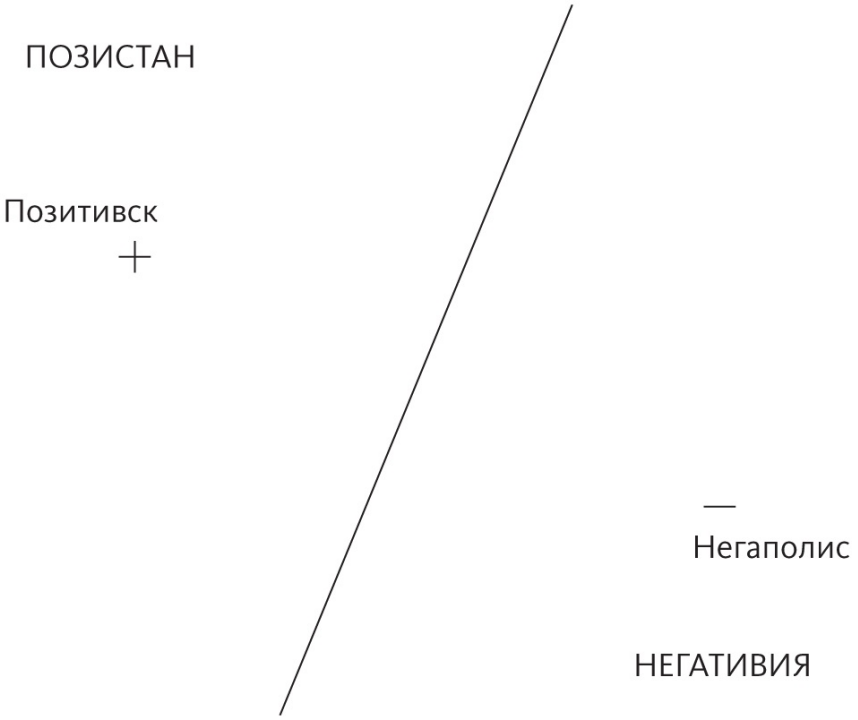
Попробуй подобрать мне пару

Алгоритм ближайшего соседа — самый простой и быстрый обучающийся алгоритм, какой только изобрели ученые. Можно даже сказать, что это вообще самый быстрый алгоритм, который можно придумать. В нем не надо делать ровным счетом ничего, и поэтому для выполнения ему требуется нулевое время. Лучше не бывает. Если вы хотите научиться узнавать лица и в вашем распоряжении есть обширная база данных изображений с ярлыками «лицо / не лицо», просто усадите этот алгоритм за работу, расслабьтесь и будьте счастливы. В этих изображениях уже скрыта модель того,

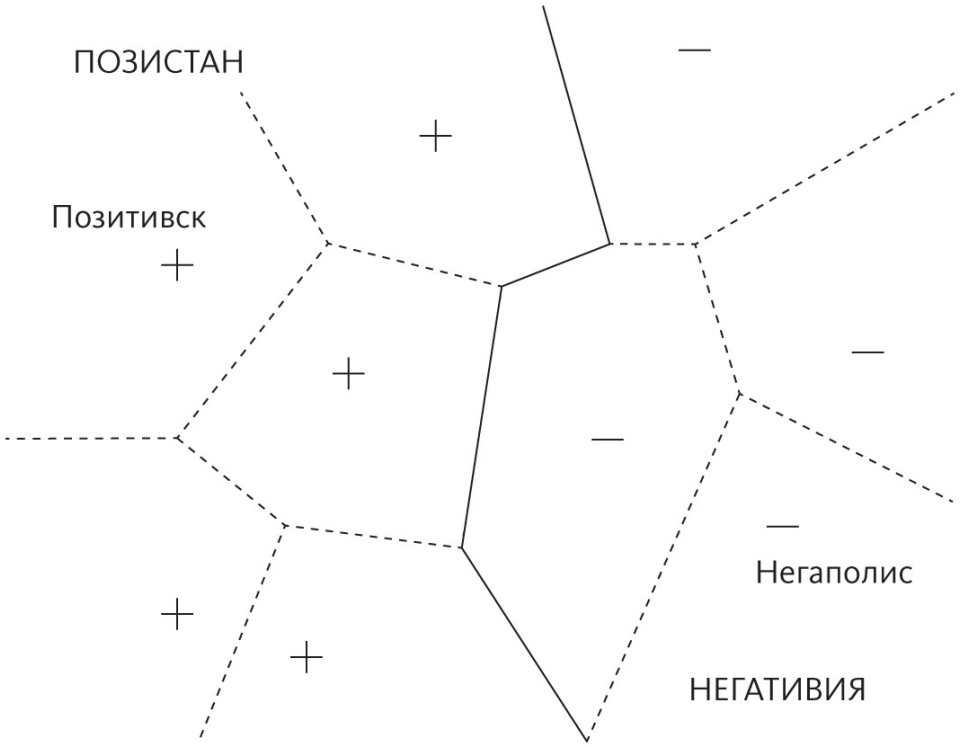
что такое лицо. Представьте, что вы Facebook и хотите автоматически определять лица на фотографиях, которые загружают пользователи, — это будет прелюдией к автоматическому добавлению тегов с именами друзей. Будет очень приятно ничего не делать, учитывая, что ежедневно в Facebook люди загружают свыше трехсот миллионов фотографий. Применение к ним любого из алгоритмов машинного обучения, которые мы до сих пор видели (может быть, кроме наивного байесовского), потребовало бы массы вычислений. А наивный Байес недостаточно сообразителен, чтобы узнавать лица.

Конечно, за все надо платить, и цена в данном случае — это время проверки. Джейн Юзер только что загрузила новую картинку. Это лицо или нет? Алгоритм ближайшего соседа ответит: найди самую похожую картинку во всей базе данных маркированных фотографий — ее «ближайшего соседа». И если на найденной картинке лицо, то и на этой тоже. Довольно просто, но теперь придется за долю секунды (в идеале) просканировать, возможно, миллиарды фотографий. Алгоритм застают врасплох, и, как ученику, который не готовился к контрольной, ему придется как-то выходить из положения. Однако в отличие от реальной жизни, где мама учит не откладывать на завтра то, что можно сделать сегодня, в машинном обучении прокрастинация может принести большую пользу. Вообще говоря, всю область, в которую входит алгоритм ближайшего соседа, называют «ленивым обучением», и в таком термине нет ничего обидного.

Ленивые обучающиеся алгоритмы намного умнее, чем может показаться, потому что их модели, хотя и неявные, могут быть невероятно сложными. Давайте рассмотрим крайний случай, когда для каждого класса у нас есть только один пример. Допустим, мы хотим угадать, где проходит граница между двумя государствами, но знаем мы только расположение их столиц. Большинство алгоритмов машинного обучения зайдет здесь в тупик, но алгоритм ближайшего соседа радостно скажет, что граница — это прямая линия, лежащая на полпути между двумя городами.



Точки на этой линии находятся на одинаковом удалении от обоих столиц. Точки слева от нее ближе к Позитивску, поэтому ближайший сосед предполагает, что они относятся к Позистану, и наоборот. Конечно, если бы это была точная граница, нам бы крупно повезло, но и это приближение, вероятно, намного лучше, чем ничего. Однако по-настоящему интересно становится, когда мы знаем много городов по обеим сторонам границы:



Ближайший сосед способен провести очень сложную границу, хотя он просто запоминает, где находятся города, и в соответствии с этим относит точки к тому или иному государству. «Агломерацией» города можно считать все точки, которые к нему ближе, чем к любому другому. Границы между такими агломерациями показаны на рисунке пунктиром. Теперь и Позистан, и Негативия — просто объединение агломераций всех городов этих стран. В отличие от этого алгоритма, деревья решений (например) способны лишь проводить границы, проходящие попеременно с севера на юг и с востока на запад, что, вероятно, будет намного худшим приближением настоящей границы. Таким образом, хотя алгоритмы на основе дерева решений будут изо всех сил стараться за время обучения определить, где проходит граница, победит «ленивый» метод ближайшего соседа.

Все дело в том, что построить глобальную модель, например дерево решений, намного сложнее, чем просто одно за другим определить положение конкретных элементов запроса. Представьте себе попытку определить с помощью дерева решений, что такое лицо. Можно было бы сказать, что у лица два глаза, нос и рот, но что такое глаз и как его найти на изображении? А если человек закроет

глаза? Дать надежное определение лица вплоть до отдельных пикселей крайне сложно, особенно учитывая всевозможные выражения, позы, контекст, условия освещения. Алгоритм ближайшего соседа этого не делает и срезает путь: если в базе данных изображение, больше всего похожее на то, которое загрузила Джейн, — это лицо, значит на загруженном изображении тоже лицо. Чтобы все работало, в базе данных должна найтись достаточно похожая картинка, например лицо в аналогичном положении, освещении и так далее, поэтому чем больше база данных, тем лучше. Для простой двухмерной проблемы, например угадывания границы между двумя странами, маленькой базы данных будет достаточно. Для очень сложной проблемы, например определения лиц, где цвет каждого пикселя — это измерение вариативности, понадобится огромная база данных. Сегодня такие базы существуют. Обучение с их помощью может быть слишком затратным для трудолюбивых алгоритмов, которые явно проводят границу между лицами и не-лицами, однако для ближайшего соседа граница уже скрыта в расположении точек данных и расстояниях, и единственная затрата — это время запроса.

Та же идея создания локальной модели вместо глобальной работает и за пределами проблем классификации. Ученые повсеместно используют линейную регрессию для прогнозирования непрерывных переменных, несмотря на то что большинство явлений нелинейны. К счастью, явления линейны локально, потому что гладкие кривые локально хорошо аппроксимируются прямыми линиями. Поэтому не пытайтесь подобрать прямую ко всем данным — сначала подгоните ее к точкам рядом с точкой запроса: получится очень мощный алгоритм нелинейной регрессии. Лень оправдывает себя. Если бы Кеннеди захотел получить полную теорию международных отношений, чтобы решить, что делать с ракетами на Кубе, у него были бы проблемы. Но он увидел аналогию между кризисом и ситуацией перед Первой мировой войной, и эта аналогия направила его к правильным решениям.

Как поведал Стивен Джонсон в книге The Ghost Map, алгоритм ближайшего соседа может спасать жизни. В 1854 году Лондон поразила вспышка холеры. В некоторых частях города от нее умер каждый восьмой житель. Господствовавшая тогда теория, что

холера вызвана якобы плохим воздухом, не помогла предотвратить распространение эпидемии. Но Джон Сноу, физик, скептически относившийся к этой теории, придумал кое-что получше. Он отметил на карте Лондона все известные случаи холеры и разделил карту на области, расположенные ближе всего к общественным водокачкам. Эврика! Оказалось, что почти все смерти приходились на «агломерацию» конкретного водозабора, расположенного на Брод-стрит в районе Сохо. Сделав вывод, что вода там заражена, Сноу убедил местные власти выключить насос, и эпидемия сошла на нет. Из этого случая родилась эпидемиология, а еще это пример первого успешного применения алгоритма ближайшего соседа — почти за столетие до его официального открытия.

В алгоритме ближайшего соседа каждая точка данных — это маленький классификатор, предсказывающий класс для всех примеров запросов, на которые она правильно отвечает. Это как армия муравьев: отдельные солдаты сами по себе делают немного, но вместе способны сдвигать горы. Если груз слишком тяжел для одного муравья, он зовет соседей. Метод *k*-ближайших соседей действует в том же духе: тестовый пример классифицируется путем нахождения *k*-ближайших соседей, которые после этого голосуют. Если ближайшее изображение к только что загруженному — это лицо, но следующие два ближайших — нет, третий ближайший сосед решает, что загруженная картинка все же не лицо. Алгоритм ближайшего соседа подвержен переобучению: если точке данных присвоен неправильный класс, он распространится на всю свою агломерацию. Алгоритм *k*-ближайших соседей более устойчив, потому что ошибается только тогда, когда большинство из *k*-ближайших соседей зашумлены. Но за это приходится платить более замутненным зрением: из-за голосования размываются мелкие детали границы. Когда *k* идет вверх, дисперсия уменьшается, но увеличивается смещенность.

Брать *k*-ближайших соседей вместо одного — это еще не все. Интуиция подсказывает, что примеры, ближе всего расположенные к тестовому, должны быть важнее. Это ведет нас к взвешенному алгоритму *k*-ближайших соседей. В 1994 году группа ученых из Миннесотского университета и Массачусетского

технологического института построила рекомендательную систему на основе, по их словам, «обманчиво простой идеи»: люди, которые соглашались на что-то в прошлом, с большей вероятностью согласятся на это и в будущем. Эта мысль вела прямиком к системам коллаборативной фильтрации, которые имеются на всех уважающих себя сайтах электронной торговли. Представьте, что вы, как Netflix, собрали базу данных, где каждый пользователь присваивает просмотренным фильмам рейтинг от одной до пяти звезд. Вы хотите определить, понравится ли вашему пользователю по имени Кен фильм «Гравитация», поэтому ищете пользователей, оценки которых лучше всего коррелируют с оценками Кена. Если все они присвоили «Гравитации» высокий рейтинг, вероятно, так поступит и Кен, и этот фильм можно ему посоветовать. Если, однако, у них нет единого мнения в отношении «Гравитации», все равно нужно как-то выйти из положения, и в данном случае пригодится список пользователей, упорядоченный по их корреляции с Кеном. Если Ли коррелирует с Кеном сильнее, чем Мег, его оценки должны считаться, соответственно, более важными. Спрогнозированная оценка Кена будет таким образом средней взвешенной оценок его соседей, где вес каждого соседа — это его коэффициент корреляции с Кеном.

Однако есть интересный момент. Представьте, что у Ли и Кена очень схожие вкусы, но, когда Кен дает фильму пять звездочек, Ли всегда выставляет три, когда Кен дает три, Ли — только одну и так далее. Нам хотелось бы использовать оценки Ли для прогнозирования оценок Кена, но, если сделать это «в лоб», мы всегда будем отклоняться на две звездочки. Вместо этого нужно предсказать, насколько рейтинги Кена будут выше или ниже его средней, основываясь на таком же показателе для Ли. Теперь видно, что Кен всегда на две звездочки выше своей средней, когда Ли на две звездочки выше своей, и наш прогноз будет попадать в точку.

Кстати говоря, для коллаборативной фильтрации явные оценки не обязательны. Если Кен заказал фильм на Netflix, это значит, что он ожидает, что фильм ему понравится. Так что «оценкой» может быть просто «заказал / не заказал», и два пользователя будут похожи, если они заказывают много одинаковых фильмов. Даже

простой клик на что-то косвенно показывает интерес пользователя. Алгоритм ближайшего соседа работает во всех этих случаях. Сегодня для того, чтобы давать рекомендации посетителям сайта, используются все виды алгоритмов, но взвешенный *k*-ближайший сосед был первым, нашедшим широкое применение в этой области, и его до сих пор сложно победить.

Рекомендательные системы — это большой бизнес: Amazon они дают треть доходов, а Netflix — три четверти. А ведь когда-то метод ближайшего соседа считался непрактичным из-за высоких требований к памяти. В те времена память компьютеров делали из маленьких сердечников, напоминавших железные кольца, по одному для каждого бита, и хранение даже нескольких тысяч примеров было обременительно. Сейчас времена изменились. Тем не менее не всегда целесообразно запоминать все увиденные примеры, а затем искать среди них, особенно потому что большинство из них, вероятно, не имеют отношения к делу. Еще раз взгляните на карту Позистана и Негативии и обратите внимание, что, если Позитивск исчезнет, граница с Негативией не изменится: агломерации близлежащих городов расширятся и займут земли, которые занимала столица, но все они позистанские. Города, которые имеют значение, располагаются исключительно вдоль границы, поэтому все остальные можно опустить. Из этого вытекает простой способ повысить эффективность метода ближайшего соседа: нужно удалить примеры, которые были правильно классифицированы их соседями. Благодаря этому и другим приемам методы ближайшего соседа находят применение в самых неожиданных областях, например в управлении манипулятором робота в реальном времени. Но при этом в таких областях, как высокочастотный трейдинг, где компьютеры покупают и продают ценные бумаги в доли секунды, такие методы по-прежнему не в почете. В гонке между нейронными сетями, которые можно применять к примерам с фиксированным количеством сложений, умножений и сигмоид, и алгоритмом, который должен искать ближайшего соседа в большой базе данных, нейронная сеть обязательно победит.

Другая причина, по которой исследователи поначалу скептически относились к ближайшему соседу, заключается в том,

что было непонятно, может ли он определять истинные границы между понятиями. Но в 1967 году ученые-информатики Том Кавер и Питер Харт доказали, что при наличии достаточного количества данных ближайший сосед в худшем случае подвержен ошибкам всего в два раза больше, чем лучший вообразимый классификатор. Если, скажем, как минимум один процент тестовых примеров неизбежно будет неправильно классифицирован из-за зашумленности данных, ближайший сосед гарантированно получит максимум два процента ошибок. Это было историческое открытие. До этого момента все известные классификаторы исходили из того, что граница имеет очень четкую форму, обычно прямую линию. Это давало и плюсы, и минусы: с одной стороны, можно было доказать правильность, как в случае с перцептроном, но при этом появлялись строгие ограничения на то, что такой классификатор может узнать. Метод ближайшего соседа был первым в истории алгоритмом, который мог воспользоваться преимуществом неограниченного количества данных, чтобы обучаться произвольно сложным понятиям. Человеку не под силу проследить границы, которые он образует в гиперпространстве из миллионов примеров, но благодаря доказательству Кавера и Харта мы знаем, что они, вероятно, недалеки от истины. Рэй Курцвейл считает, что сингулярность начинается, когда люди перестают понимать, что делают компьютеры. По этому стандарту не совсем преувеличением будет сказать, что это уже происходит и началось еще в 1951 году, когда Фикс и Ходжес изобрели метод ближайшего соседа — самый маленький алгоритм, какой только можно изобрести.

Проклятие размерности

Конечно, в райском саду есть и Змей. Его зовут Проклятие Размерности, и, хотя он в большей или меньшей степени поражает все алгоритмы машинного обучения, для ближайшего соседа он особенно опасен. В низких измерениях (например, двух или трех) ближайший сосед обычно работает довольно хорошо, но по мере увеличения количества измерений все довольно быстро начинает рушиться. Сегодня нет ничего необычного в обучении на тысячах или даже миллионах атрибутов. Для коммерческих сайтов,

пытающихся узнать ваши предпочтения, атрибутом становится каждый клик. То же самое с каждым словом на веб-странице, с каждым пикселем в изображении. А у ближайшего соседа проблемы могут появиться даже с десятками или сотнями атрибутов. Первая проблема в том, что большая часть атрибутов не имеет отношения к делу: можно знать миллион фактов о Кене, но, вполне возможно, лишь немногие из них могут что-то сказать (например) о его риске заболеть раком легких. Для конкретно этого предсказания критически важно знать, курит Кен или нет, а информация о курении вряд ли поможет решить, понравится ли ему «Гравитация». Символистские методы со своей стороны довольно хорошо убирают неподходящие атрибуты: если в атрибуте не содержится информация о данном классе, его просто не включают в дерево решений или набор правил. Но метод ближайшего соседа неподходящие атрибуты безнадежно запутывают, потому что все они вносят свой вклад в сходство между примерами. Если не имеющих отношения к делу атрибутов будет достаточно много, случайное сходство в нерелевантных измерениях подавит имеющее значение сходство в важных, и метод ближайшего соседа окажется ничем не лучше случайного угадывания.

Еще одна большая и неожиданная проблема заключается в том, что большое число атрибутов может мешать, даже когда все они имеют отношение к делу. Может показаться, что много информации — это всегда благо. Разве это не лозунг нашего времени? Но по мере увеличения числа измерений начинает экспоненциально расти число обучающих примеров, необходимых для определения границ понятия. Двадцать булевых атрибутов дадут примерно миллион возможных примеров. С двадцать первым примеров станет два миллиона, с соответствующим числом способов прохождения между ними границы. Каждый лишний атрибут делает проблему обучения в два раза сложнее, и это если атрибуты булевы. Если атрибут высокоинформативный, польза от его добавления может превышать затраты. Но когда в распоряжении есть лишь малоинформативные атрибуты, например слова в электронном письме или пиксели изображения, это, вероятно, породит проблемы, несмотря на то что

в совокупности они могут нести достаточно информации, чтобы предсказать то, что вы хотите.

Все даже хуже. Ближайший сосед основан на нахождении схожих объектов, а в высоких измерениях распадается сама идея сходства. Гиперпространство — как сумеречная зона. Наша интуиция, основанная на опыте жизни в трех измерениях, там не действует, и начинают происходить все более и более странные вещи. Представьте себе апельсин: шарик вкусной мякоти, окруженный тонкой кожицей. Мякоть в апельсине занимает, скажем, 90 процентов радиуса, а оставшиеся десять приходятся на кожуру. Это означает, что 73 процента объема апельсина — это мякоть ($0{,}9^3$). Теперь рассмотрим гиперапельсин: если мякоть занимает все те же 90 процентов радиуса, но, скажем, в сотне измерений, то она сократится примерно до всего лишь 3⁄1000 процента объема ($0{,}9^{100}$). Гиперапельсин будет состоять из одной кожуры, и его никогда нельзя будет очистить!

Беспокоит и то, что происходит с нашей старой знакомой, гауссовой кривой. Нормальное распределение говорит, что данные в сущности расположены в какой-то точке (средняя распределения), но с некоторым расхождением вокруг нее (заданным стандартным отклонением). Верно? Да, но не в гиперпространстве. При нормальном распределении в высокой размерности будет выше вероятность получить пример далеко от средней, чем близко к ней. Кривая Гаусса в гиперпространстве больше похожа на пончик, чем на колокол. Когда ближайший сосед входит в этот беспорядочный мир, он безнадежно запутывается. Все примеры выглядят одинаково схожими и при этом слишком далеко отстоят друг от друга, чтобы делать полезные прогнозы. Если случайным образом равномерно рассеять примеры внутри высокоразмерного гиперкуба, большинство окажется ближе к грани этого куба, чем к своему ближайшему соседу. На средневековых картах неисследованные области обозначали драконами, морскими змеями и другими фантастическими существами или просто фразой «Здесь драконы». В гиперпространстве драконы повсюду, в том числе прямо в дверях. Попробуйте прогуляться в гости к соседу, и вы никогда туда не доберетесь: станете вечно блуждать в чужих землях и гадать, куда делись все знакомые предметы.

Деревья решений тоже не застрахованы от проклятия размерности. Скажем, понятие, которое вы пытаетесь получить, представляет собой сферу: точки внутри нее положительные, а снаружи — отрицательные. Дерево решений может приблизить сферу самым маленьким кубом, в который она помещается. Это не идеально, но и не очень плохо: неправильно классифицированы будут только углы. Однако в большем числе измерений почти весь объем гиперкуба окажется вне гиперсферы, и на каждый пример, который вы правильно классифицируете как положительный, будет приходиться много отрицательных, которые вы сочтете положительными, а это резко снижает точность.

На самом деле такая проблема есть у всех обучающихся алгоритмов — это вторая беда машинного обучения после переобучения. Термин «проклятие размерности» был придуман в 50-е годы Ричардом Беллманом[93], специалистом по теории управления. Он заметил, что алгоритмы управления, которые хорошо работают в трех измерениях, становятся безнадежно неэффективными в пространствах с большим числом измерений, например, когда вы хотите контролировать каждый сустав манипулятора или каждую ручку на химическом комбинате. А в машинном обучении проблема не только в вычислительных затратах: с ростом размерности само обучение становится все сложнее и сложнее.

Тем не менее не все потеряно. Во-первых, можно избавиться от не имеющих отношения к делу измерений. Деревья решений делают это автоматически, путем вычисления информационного выигрыша от каждого атрибута и выбора самых информативных. В методе ближайшего соседа мы можем сделать нечто похожее, сначала отбросив все атрибуты, которые дают прирост информации ниже определенного порога, а затем измерив схожесть в пространстве с меньшим числом измерений. В некоторых случаях это быстрый и достаточно хороший прием, но, к сожалению, ко многим понятиям он неприменим. Среди них, например, исключающее ИЛИ: если атрибут говорит что-то о данном классе только в сочетании с другими атрибутами, он будет отброшен. Более затратный, но хитрый вариант — «обернуть» выбор атрибута вокруг самого обучающегося алгоритма с поиском путем

восхождения на выпуклые поверхности, который будет удалять атрибуты, пока это не повредит точности метода ближайшего соседа на скрытых данных. Ньютон многократно выбирал атрибуты и определил, что для предсказания траектории тела важна только его масса, а не цвет, запах, возраст и миллиард других свойств. Вообще говоря, самое важное в уравнении — все те количества, которые в нем не появляются: когда известны самые существенные элементы, часто оказывается легче разобраться, как они зависят друг от друга.

Одно из решений проблемы неважных атрибутов — определение их веса. Вместо того чтобы считать сходство по всем измерениям равноценным, мы «сжимаем» наименее подходящие. Представьте, что обучающие примеры — это точки в комнате и высота для наших целей не требуется. Если ее отбросить, все примеры спроецируются на пол. Произвести понижающее взвешивание — все равно что опустить в комнате потолок. Высота точки все еще засчитывается при вычислении расстояния до других точек, но уже меньше, чем ее горизонтальное положение. И, как и многое другое в машинном обучении, вес атрибутов можно найти путем градиентного спуска.

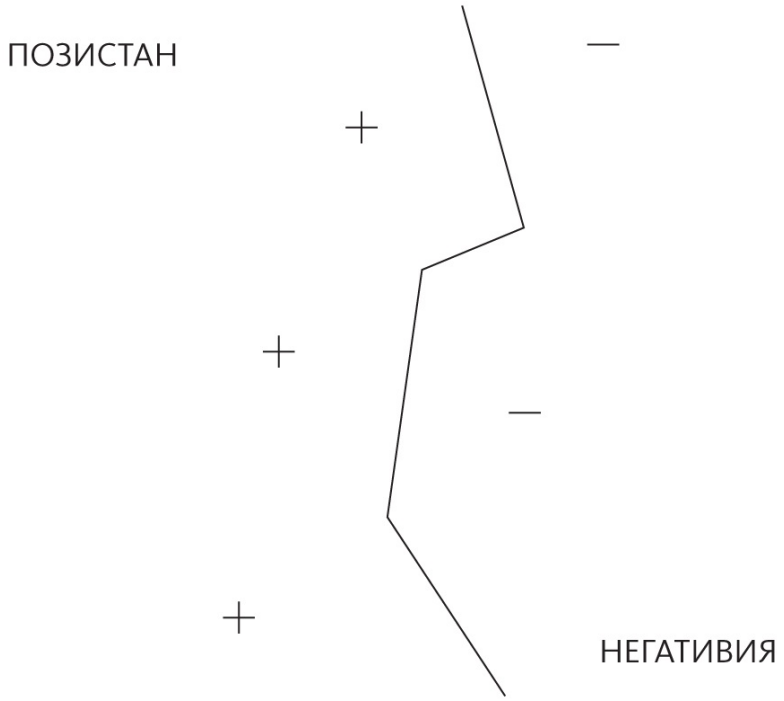
Может случиться, что потолок в комнате высокий, а точки данных лежат рядом с полом, как тонкий слой пыли на ковре. В этом случае нам повезло: проблема выглядит трехмерной, но в сущности она ближе к двухмерной. Мы не будем сокращать высоту, потому что это уже сделала природа. Такое «благословение неравномерности» данных в (гипер)пространстве часто спасает положение. У примеров могут быть тысячи атрибутов, но в реальности все они «живут» в пространстве с намного меньшим числом измерений. Именно поэтому метод ближайшего соседа бывает хорош, например, для распознавания написанных вручную цифр: каждый пиксель — это измерение, поэтому измерений много, но лишь мизерная доля всех возможных изображений — цифры, и все они живут вместе в уютном уголке гиперпространства. Форма низкоразмерного пространства с данными бывает, однако, довольно своенравна. Например, если в комнате стоит мебель, пыль оседает не только на пол, но и на столы, стулья, покрывала и так далее. Если можно определить примерную форму слоя пыли, покрывающей комнату, тогда останется найти координаты каждой

точки на нем. Как мы увидим в следующей главе, целая субдисциплина машинного обучения посвящена открытию форм этих слоев путем, так сказать, прощупывания гиперпространства во тьме.

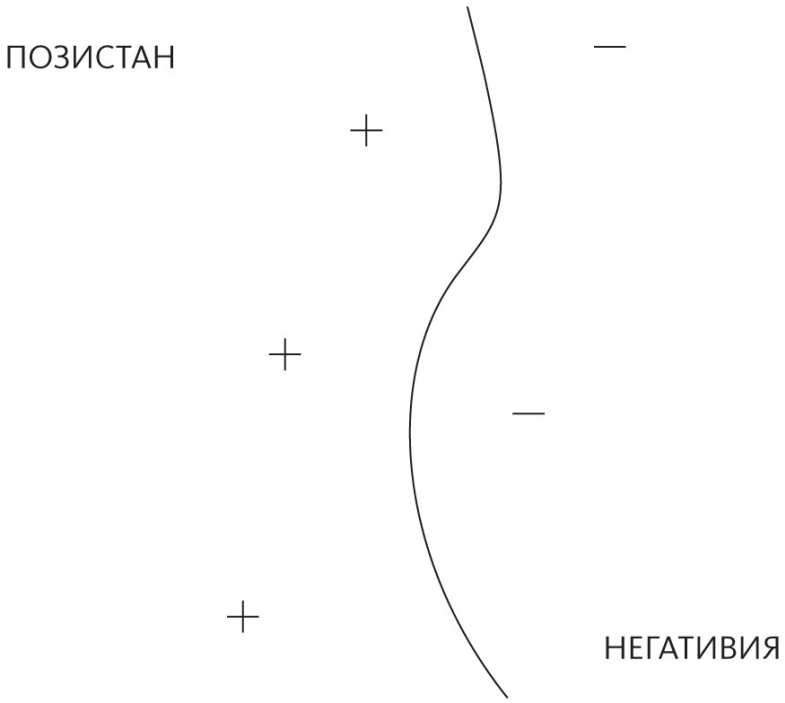
Змеи на плоскости

Метод ближайшего соседа оставался самым широко используемым обучающимся алгоритмом аналогистов вплоть до середины 1990-х, когда его затмили более гламурные кузены из других «племен». Но тут, сметая все на своем пути, на смену ворвался новый алгоритм, основанный на принципах сходства. Можно сказать, что это был еще один «дивиденд от мира», плод окончания холодной войны. Метод опорных векторов был детищем советского специалиста по частотному подходу Владимира Вапника[94]. Вапник большую часть своей карьеры работал в московском Институте проблем управления, но в 1990 году Советский Союз рухнул, и ученый уехал в США, где устроился на работу в легендарную Bell Labs[95]. В России Вапник в основном довольствовался теоретической, бумажной работой, но атмосфера в Bell Labs была иной. Исследователи стремились к практическим результатам, и Вапник наконец решился превратить свои идеи в алгоритм. В течение нескольких лет он с коллегами по лаборатории разработал метод опорных векторов, и вскоре опорные векторы оказались повсюду и принялись ставить новые рекорды точности.

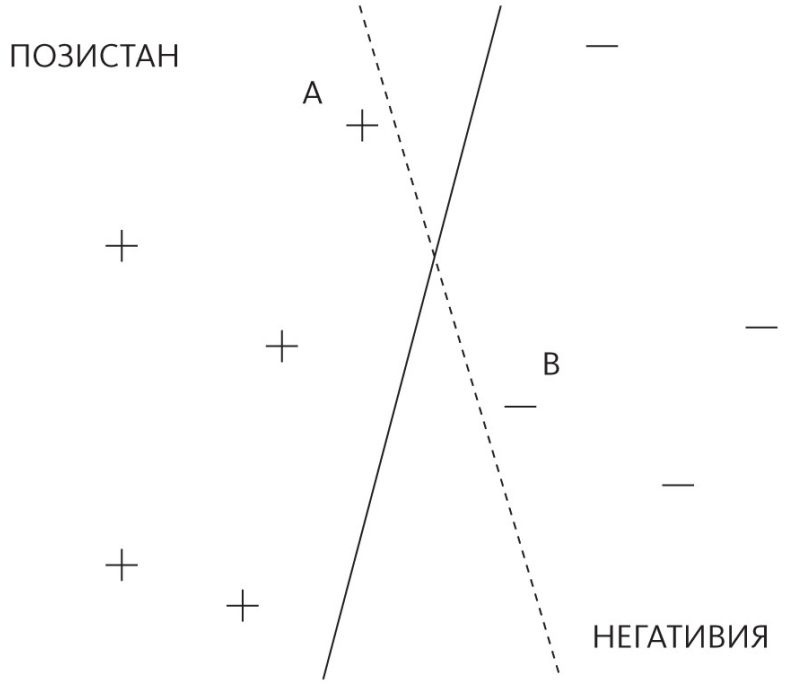
На первый взгляд метод опорных векторов во многом похож на взвешенный алгоритм *k*-ближайших соседей: граница между положительными и отрицательными классами определяется мерой схожести примеров и их весами. Тестовый пример принадлежит к положительному классу, если в среднем он выглядит более похожим на положительные примеры, чем на отрицательные. Среднее взвешивается, и метод опорных векторов помнит только ключевые примеры, необходимые для проведения границы. Если еще раз посмотреть на Позистан и Негативию без городов, не расположенных на границе, останется такая карта:



Примеры здесь называются опорными векторами, потому что это векторы, которые «поддерживают» границу: уберите один, и участок границы соскользнет в другое место. Также можно заметить, что граница представляет собой зубчатую линию с резкими углами, которые зависят от точного расположения примеров. У реальных понятий, как правило, границы более плавные, а это означает, что приближение, сделанное методом ближайшего соседа, вероятно, не идеально. Благодаря методу опорных векторов можно сделать границу гладкой, больше похожей на эту:



Чтобы обучить метод опорных векторов, нужно выбрать опорные векторы и их вес. Меру схожести, которая в мире опорных векторов называется ядром, обычно назначают априорно. Одним из важнейших открытий Вапника было то, что не все границы, отделяющие положительные тренировочные примеры от отрицательных, равноценны. Представьте, что Позистан воюет с Негативией и государства разделены нейтральной полосой с минными полями с обеих сторон. Ваша задача — исследовать эту ничейную землю, пройдя с одного ее конца к другому, и не взлететь на воздух. К счастью, у вас в руках карта с расположением мин. Вы, понятное дело, выберете не просто любую старую тропинку, а станете обходить мины как можно более широким кругом. Именно так поступает метод опорных векторов: мины для него — это примеры, а найденная тропа — выученная граница. Самое близкое место, где граница подходит к примеру, — ее зазор, и метод опорных векторов выбирает опорные векторы и вес*а* так, чтобы зазор был максимальным. Например, сплошная прямая граница на этом рисунке лучше, чем пунктирная:



Пунктирная граница четко разделяет положительные и отрицательные примеры, но опасно близко подходит к минам *A* и *B*. Эти примеры — опорные векторы. Удалите один из них, и граница с максимальным зазором переместится в другое место. Конечно, граница может быть изогнутой, из-за чего зазор сложнее визуализировать, но можно представить себе, как по ничейной земле ползет змея и зазор — ее жировые отложения. Если без риска взорваться на кусочки может проползти очень толстая змея, значит, метод опорных векторов хорошо разделяет положительные и отрицательные примеры, и Вапник показал, что в этом случае можно быть уверенным, что метод не подвержен переобучению. Интуиция подсказывает, что у толстой змеи меньше способов проскользнуть мимо мин, чем у тощей, и точно так же, если зазор большой, у него меньше шансов переобучиться данным, нарисовав слишком замысловатую границу.

Вторая часть истории — это то, как метод опорных векторов находит самую толстую змею, способную проползти между положительными и отрицательными минами. Может показаться, что обучения весам для каждого тренировочного примера путем градиентного спуска будет достаточно. Надо просто найти веса, которые максимизируют зазор, и любой пример, который

заканчивается нулевым весом, можно отбросить. К сожалению, в таком случае веса начали бы расти безгранично, потому что с точки зрения математики чем больше вес*а*, тем больше зазор. Если вы находитесь в метре от мины и удвоите размер всего, включая вас самих, от мины вас станут отделять два метра, но это не уменьшит вероятности, что вы на нее наступите. Вместо этого придется максимизировать зазор под давлением того, что веса могут увеличиться лишь до какой-то фиксированной величины. Или аналогично можно минимизировать вес*а* под давлением того, что все примеры имеют данный зазор, который может быть единицей — точное значение произвольно. Это то, что обычно делает метод опорных векторов.

Оптимизация при наличии ограничений — проблема максимизации или минимизации функции, в отношении которой действуют некие условия. Вселенная, например, максимизирует энтропию при условии сохранения постоянства энергии. Проблемы такого рода широко распространены в бизнесе и технологиях. Можно стремиться максимизировать число единиц продукции, которое производит фабрика, при условии доступного числа машинных инструментов, спецификаций продукции и так далее. С появлением метода опорных векторов оптимизация при наличии ограничений стала критически важной и в машинном обучении. Неограниченная оптимизация сравнима с тем, как добраться до вершины горы, и именно это делает градиентный спуск (в данном случае восхождение). Ограниченная оптимизация — все равно что зайти как можно выше, но при этом не сходить с дороги. Но если дорога будет доходить до самой вершины, ограниченные и неограниченные проблемы будут иметь одно и то же решение. Однако чаще дорога сначала петляет вверх по горе, а затем сворачивает вниз, так и не достигнув вершины. Вы понимаете, что достигли высочайшей точки дороги, и не можете зайти выше, не съехав с нее. Другими словами, путь к вершине находится под прямым углом к дороге. Если дорога и путь к вершине образуют острый или тупой угол, всегда можно забраться еще выше, продолжая ехать по дороге, даже если вы не подниметесь так быстро, как если бы пошли прямо к вершине. Поэтому для решения проблемы ограниченной оптимизации надо следовать

не по градиенту, а по его части, параллельной поверхности ограничения — в данном случае дороги, — и остановиться, когда эта часть будет равна нулю.

В целом нам надо справиться со многими ограничениями сразу (в случае метода опорных векторов — по одному на каждый пример). Представьте, что вы хотите подобраться как можно ближе к Северному полюсу, но не можете выйти из комнаты. Каждая из четырех стен комнаты — это ограничение, поэтому для решения задачи надо идти по компасу, пока вы не упретесь в угол, где встречаются северо-восточная и северо-западная стена. Эти стены будут активными ограничениями, потому что они не дадут вам достичь оптимума, а именно Северного полюса. Если одна из стен вашей комнаты смотрит точно на север, она станет единственным активным ограничением; для решения надо направиться в ее центр. А если вы Санта-Клаус и живете прямо на Северном полюсе, все ограничения окажутся неактивными и можно будет сесть и поразмышлять над проблемой оптимального распределения игрушек. (Бродячим торговцам легче, чем Санте.) В методе опорных векторов активные ограничения — опорные векторы, поскольку их зазоры уже наименьшие из разрешенных. Перемещение границы нарушило бы одно или больше ограничений. Все другие примеры не имеют отношения к делу и их вес равен нулю.

В реальности методу опорных векторов обычно разрешается нарушать некоторые ограничения, то есть классифицировать некоторые примеры неправильно или менее чем на зазор, потому что в противном случае будет возникать переобучение. Если где-то в центре положительной области есть негативный пример, создающий шум, нам не надо, чтобы граница вилась внутри положительной зоны просто ради того, чтобы правильно классифицировать этот пример. Однако за каждый неправильно определенный пример начисляются штрафные баллы, и это стимулирует метод опорных векторов сводить их к минимуму. Так что данный метод как песчаные черви в «Дюне»: большие, мощные и способные пережить довольно много взрывов, но не слишком много.

В поисках применения своему алгоритму Вапник и его сотрудники вскоре вышли на распознавание написанных от руки

цифр, в котором их коллеги-коннекционисты в Bell Labs были мировыми экспертами. Ко всеобщему удивлению, метод опорных векторов с ходу справился не хуже многослойного перцептрона, который тщательно, годами оттачивали для распознавания цифр. Это подготовило почву для долгой интенсивной конкуренции между методами. Метод опорных векторов можно рассматривать как обобщение перцептрона, использование специфической меры сходства (скалярного произведения векторов) даст гиперплоскостную границу между классами. Но у метода опорных векторов имеется большое преимущество по сравнению с многослойными перцептронами: у весов есть единичный оптимум, а не много локальных, и поэтому их намного легче надежно найти. Несмотря на это, опорные векторы не менее выразительны, чем многослойные перцептроны: опорные векторы фактически действуют как скрытый слой, а их взвешенное среднее — как выходной слой. Например, метод опорных векторов может легко представлять функцию исключающего ИЛИ, имея один опорный вектор для каждой из четырех возможных конфигураций. Но и коннекционисты не сдавались без боя. В 1995 году Ларри Джекел, глава отдела Bell Labs, в котором работал Вапник, поспорил с ним на хороший обед, что к 2000 году нейронные сети будут так же понятны, как метод опорных векторов. Он проиграл. В ответ Вапник поспорил, что к 2005 году никто не будет пользоваться нейронными сетями. И тоже проиграл. (Единственным, кто бесплатно пообедал, был Янн Лекун, их свидетель.) Более того, с появлением глубокого обучения коннекционисты снова взяли верх. При условии обучаемости, сети со многими слоями могут выражать многие функции компактнее, чем метод опорных векторов, у которого всегда только один слой, а это иногда имеет решающее значение.

Другим заметным ранним успехом метода опорных векторов была классификация текстов, которая оказалась большим благом для зарождающегося интернета. В то время самым современным классификатором был наивный байесовский алгоритм, но, когда каждое слово в языке — это измерение, даже он мог начать переобучаться. Для этого достаточно слова, которое по случайности в тренировочных данных встречается на всех спортивных

страницах и ни на каких других: в этом случае у наивного Байеса появятся галлюцинации, что любая страница, содержащая это слово, посвящена спорту. А метод опорных векторов благодаря максимизации зазора может сопротивляться переобучению даже при очень большом числе измерений.

В целом чем больше опорных векторов выбирает метод, тем лучше он обобщает. Любой обучающий пример, который не представляет собой опорный вектор, будет правильно классифицирован, если появится в тестовой выборке, потому что граница между положительными и отрицательными примерами по-прежнему будет на том же месте. Поэтому ожидаемая частота ошибок метода опорных векторов, как правило, равна доле примеров, являющихся опорными векторами. По мере роста числа измерений эта доля тоже будет расти, поэтому метод не застрахован от проклятия размерности, но он более устойчив к нему, чем большинство алгоритмов.

Кроме практических успехов, метод опорных векторов перевернул с ног на голову много воззрений, которые олицетворяли здравый смысл в машинном обучении. Например, опроверг утверждение, которое иногда путают с бритвой Оккама, что более простые модели точнее. Метод может иметь бесконечное число параметров и все равно не переобучаться при условии, что у него достаточно большой зазор.

Самое неожиданное свойство метода опорных векторов заключается в следующем: какие бы изогнутые границы он ни проводил, эти границы всегда будут прямыми линиями (или гиперплоскостями). В этом нет противоречия. Причина заключается в том, что прямые линии будут находиться в другом пространстве. Допустим, примеры живут на плоскости *(x, y)*, а граница между положительными и отрицательными областями — это парабола $y = x^2$. Ее невозможно представить в виде прямой линии, но, если мы добавим третью координату *z* [данные окажутся в пространстве *(x, y, z)*] и установим координату *z* каждого примера равной квадрату его координаты *x*, граница будет просто диагональной плоскостью, определенной $y = z$. В результате точки данных поднимутся в третье измерение, некоторые больше, чем другие, но ровно настолько, насколько нужно, и — вуаля! — в этом новом измерении

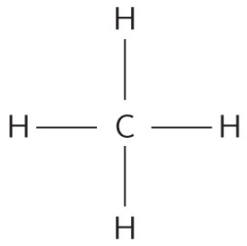
положительные и отрицательные примеры можно будет разделить плоскостью. То, что метод делает с ядрами, опорными векторами и весами, можно рассматривать как картирование данных в более высокоразмерное пространство и нахождение в этом пространстве гиперплоскости с максимальным зазором. Для некоторых ядер полученное поле имеет бесконечное число измерений, но для метода опорных векторов это совершенно не важно. Может быть, гиперпространство — это и сумеречная зона, но метод опорных векторов знает, как находить в ней путь.

Вверх по лестнице

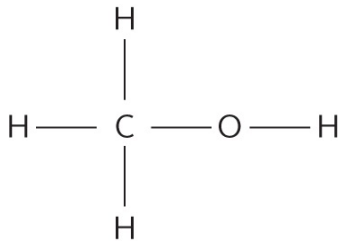
Две вещи схожи, если они в определенном отношении совпадают друг с другом. Если они в чем-то совпадают, вероятно, в чем-то они будут отличаться. В этом суть аналогии. Это указывает и на две главные подпроблемы рассуждения по аналогии: как понять, насколько похожи две вещи, и как решить, какие выводы можно сделать из этих сходств. Пока мы исследовали «маломощную» область аналогии — алгоритмы вроде ближайшего соседа и метод опорных векторов, — ответы на оба вопроса были очень простыми. Такие алгоритмы наиболее популярны, но глава об аналогическом обучении будет неполной, если мы хотя бы бегло не рассмотрим более мощные части спектра.

Самый главный вопрос во многих аналогических обучающихся алгоритмах — как измерять сходство. Это может быть просто евклидово расстояние между точками данных или, сложнее, целая программа с многочисленными слоями подпрограмм, которая в конце выдает значение сходства. Так или иначе функция сходства контролирует, как алгоритм машинного обучения обобщает из известных примеров в новые. Именно здесь мы вводим в обучающийся алгоритм наши знания о данной области: это ответ аналогизаторов на вопрос Юма. Аналогическое обучение можно применять ко всем видам объектов, а не только к векторам атрибутов, при условии, что есть какой-то способ измерить сходство между ними. Например, сходство между двумя молекулами можно определить по числу идентичных субструктур, которые они содержат. Метан и метанол схожи, потому что в них

есть три связи углерода с водородом, а отличаются они только тем, что в метаноле один атом водорода замещен гидроксильной группой:



Метан



Метанол

Однако это не означает, что схожи химические свойства веществ, ведь метан — это газ, а метанол — спирт. Вторая часть аналогического рассуждения — попытка разобраться, какие выводы можно сделать о новом объекте на основе найденных аналогов. Это бывает и очень просто, и очень сложно. В случае алгоритма ближайшего соседа и метода опорных векторов это просто предсказание класса нового объекта на основе классов ближайших соседей или опорных векторов. Но в случае рассуждения по прецедентам — еще одного типа аналогического обучения — результатом может стать сложная структура, сформированная из элементов найденных объектов. Представьте, что ваш принтер печатает абракадабру и вы звоните в службу поддержки Hewlett-Packard. Есть шанс, что они уже много раз встречались с аналогичной проблемой, поэтому будет правильно найти старые записи и сложить из них потенциальное решение. Мало просто найти жалобы, у которых много общих атрибутов с вашей: например, в зависимости от установленной операционной системы — Windows или Mac OS X — нужен будет очень разный набор настроек и системы, и принтера. Когда самые подходящие случаи найдены, требуемой последовательностью шагов, необходимых для решения вашей проблемы, может оказаться сочетание этапов из разных случаев плюс какие-то дополнительные, специфические элементы.

В настоящее время службы поддержки — это самое популярное применение рассуждения на основе прецедентов. Большинство

из них все еще используют посредника-человека, но Eliza IPsoft уже сама общается с клиентом. Эта система дополнена интерактивным 3D-изображением женщины и на сегодняшний день уже решила более 20 миллионов проблем клиентов в основном престижных американских компаний. «Привет из Роботистана, самого дешевого нового направления аутсорсинга», как недавно писали в одном блоге по аутсорсингу. Поскольку аутсорсинг постоянно охватывает все новые профессии, вместе с ним совершенствуется и аналогическое обучение. Уже созданы первые роботы-адвокаты, которые отстаивают тот или иной вердикт на основе прецедентов. Одна из таких систем точно предсказала результаты более 90 процентов рассмотренных ею дел о нарушении производственной тайны. Может быть, в будущем на сессии киберсуда где-нибудь в облаке Amazon робот-адвокат будет оспаривать штраф за превышение скорости, который робот-полицейский выписал вашему беспилотному автомобилю, а вы тем временем станете нежиться на пляже. Тогда мечта Лейбница о сведении всех аргументов к вычислениям наконец сбудется.

Вероятно, труд композитора находится еще выше на лестнице умений. Дэвид Коуп, почетный профессор музыки в Калифорнийском университете в Санта-Круз, разработал алгоритм, который пишет новые музыкальные произведения в стиле известных композиторов путем отбора и рекомбинации коротких отрывков из их сочинений. На конференции, в которой я несколько лет назад участвовал, Коуп продемонстрировал три пьесы: одну на самом деле написанную Моцартом, другую — композитором, имитировавшим его, и третью — сгенерированную системой. Затем Коуп попросил аудиторию проголосовать. Вольфганг Амадей победил, но имитатор-человек уступил компьютеру. Поскольку это была конференция по искусственному интеллекту, публика осталась довольна. На других мероприятиях восторгов было куда меньше. Некоторые слушатели сердито обвиняли Коупа в том, что он уничтожает музыку. Если Коуп прав, то творчество — высшее из непостижимого — сводится к аналогии и рекомбинации. Попробуйте свои силы: найдите в Google «david cope mp3» и послушайте.

Однако самый изящный трюк аналогизаторов — это обучение на проблемах из разных областей. Люди практикуют это постоянно: менеджер может перейти, скажем, из медиакомпании в компанию, занимающуюся потребительскими товарами, и не начнет с нуля, потому что многие управленческие навыки повторяются. На Уолл-стрит приглашают работать множество физиков, потому что физические и финансовые проблемы кажутся очень разными, но зачастую имеют схожую математическую структуру. Тем не менее все алгоритмы машинного обучения, которые мы до сих пор видели, пасуют, если мы натренируем их для предсказания, скажем, броуновского движения, а потом заставим делать прогнозы на фондовой бирже. Цены на бирже и скорости частиц, взвешенных в жидкости, — это разные переменные, поэтому обучающийся алгоритм даже не будет знать, с чего начать. Однако аналогизаторы могут сделать это, используя отображение структур — алгоритм, изобретенный психологом из Северо-Западного университета Дедре Джентнером. Отображение структур берет два описания, находит связное соответствие между некоторыми их элементами и соотношениями, а затем, основываясь на этом соответствии, переносит другие свойства одной структуры на другую. Например, если структуры — это Солнечная система и атом, можно отобразить планеты как электроны, а солнце — как ядро и заключить, подобно Бору, что электроны вращаются вокруг ядра. Истина, конечно, не такая прямолинейная, и уже сделанные аналогии часто приходится корректировать, но иметь возможность учиться на основе единичного примера, как этот, несомненно, ключевой атрибут универсального обучающегося алгоритма. Когда мы сталкиваемся с новым типом рака — а это происходит постоянно, потому что рак непрерывно мутирует, — модели, которые мы узнали из предыдущих случаев, оказываются неприменимы. У нас нет ни времени, чтобы собирать данные о новом типе опухоли, ни множества пациентов: может быть, пациент вообще уникальный, и он срочно нуждается в лекарстве. В таком случае надежду дает сравнение новой разновидности рака с уже известными: попытаться найти похожий случай и предположить, что сработают те же стратегии лечения.

Есть ли что-то, на что неспособна аналогия? Нет, считает Даглас Хофштадтер, когнитивный психолог и автор книги Gödel, Escher, Bach: An Eternal Golden Braid[96]. Хофштадтер немного похож на доброго близнеца Гринча — похитителя Рождества[97] и, вероятно, является самым знаменитым аналогизатором в мире. В книге Surfaces and Essences: Analogy as the Fuel and Fire of Thinking («Поверхности и сущности: аналогия в роли топлива и огня мышления») Хофштадтер и Эммануэль Сандер страстно доказывают, что все разумное поведение сводится к аналогии. Все, что мы узнаём или открываем, начиная со значения повседневных слов, например «мама» и «играть», до гениальных прозрений Альберта Эйнштейна и Эвариста Галуа, — это продукт аналогии. Когда малыш Тим видит, что какие-то женщины присматривают за другими детьми так же, как его собственная мама присматривает за ним, он обобщает понятие «мамочка» до мамы каждого человека, а не только его. Это, в свою очередь, трамплин к таким понятиям, как «мать-природа». «Самая счастливая мысль» Эйнштейна, из которой выросла общая теория относительности, была аналогией между гравитацией и ускорением: если вы едете в лифте, невозможно сказать, с какой из этих сил связан ваш вес, потому что результат одинаков. Мы плывем по широкому океану аналогий, манипулируем ими в своих целях, а они, не ведая того, манипулируют нами. В книгах аналогии встречаются на каждой странице (например, заголовки этого и предыдущего раздела). Gödel, Escher, Bach: An Eternal Golden Braid — расширенная аналогия между теоремой Гёделя, искусством Эшера и музыкой Баха. Если Верховный алгоритм — это не аналогия, он несомненно должен быть в чем-то схож с ней.

Взойди и сияй

В когнитивистике давно не утихают дебаты между символистами и аналогизаторами. Символисты показывают вещи, которые умеют моделировать они, но не умеют аналогизаторы. Затем аналогизаторы решают задачу, указывают на слабые места символистов, и цикл повторяется. Обучение на основе примеров, как его иногда называют, предположительно лучше подходит для

моделирования запоминания отдельных эпизодов нашей жизни, а правила, предположительно, лучше выбрать для рассуждений с абстрактными концепциями, например «работа» и «любовь». Когда я был студентом, меня осенило: это ведь просто указывает на существование континуума, и надо уметь учиться на всем его протяжении. Правила — это, по сути, обобщенные частные случаи, где мы «забыли» некоторые атрибуты, потому что они не имеют значения. Частные же случаи — очень конкретные правила с условием для каждого атрибута. В жизни аналогичные эпизоды постепенно абстрагируются и образуют основанные на правилах структуры, например «есть в ресторане». Вы знаете, что пойти в ресторан — это и заказать что-нибудь из меню, и дать чаевые, и следуете этим «правилам поведения» каждый раз, когда едите вне дома. При этом вы, вероятно, и не вспомните, в каком заведении впервые все это осознали.

В своей диссертации я разработал алгоритм, объединяющий обучение на основе частных случаев и на основе правил. Правило не просто подходит к сущностям, которые удовлетворяют всем его условиям: оно подходит к любой сущности, которая похожа на него больше, чем на любое другое правило, и в этом смысле приближается к удовлетворению его условий. Например, человек с уровнем холестерина 220 мг/дл ближе, чем человек с 200 мг/дл, подходит к правилу «Если холестерин выше 240 мг/дл, есть риск сердечного приступа». RISE, как я назвал этот алгоритм, в начале обучения относится к каждому обучающему примеру как к правилу, а затем постепенно обобщает эти правила, впитывая ближайшие примеры. В результате обычно получается сочетание очень общих правил, которые в совокупности подходят к большинству примеров, плюс большое количество конкретных правил, которые подходят к исключениям, и так далее по «длинному хвосту» конкретных воспоминаний. RISE в то время предсказывал успешнее, чем лучшие обучающие алгоритмы, основанные на правилах и частных случаях. Мои эксперименты показали, что его сильной стороной было именно сочетание плюсов обоих подходов. Правила можно подобрать аналогически, и поэтому они перестают быть хрупкими. Частные случаи могут выбирать разные свойства в разных областях пространства и тем самым борются с проклятием размерности

намного лучше метода ближайшего соседа, который везде выбирает одни и те же свойства.

RISE был шагом в сторону Верховного алгоритма, потому что соединял в себе символическое и аналогическое обучение. Однако это был лишь маленький шажок, потому что он не обладал полной силой этих парадигм и в нем по-прежнему не хватало трех оставшихся. Правила RISE нельзя было по-разному сложить в цепочку: они просто предсказывали класс примера на основе его атрибутов. Правила не могли рассказать о более чем одной сущности одновременно. Например, RISE не умел выражать правила вроде «Если у *A* грипп и *B* контактировал с *A*, то у *B* тоже может быть грипп». В аналогической части RISE лишь обобщал простой алгоритм ближайшего соседа. Он не может учиться в разных областях, используя отображение структур или какую-то схожую стратегию. Заканчивая работу над диссертацией, я не знал, как сложить в один алгоритм всю мощь пяти парадигм, и на время отложил проблему. Но, применяя машинное обучение к таким проблемам, как реклама из уст в уста, интеграция данных, программирование на примерах и персонализация сайтов, я постоянно замечал, что все парадигмы по отдельности дают лишь часть решения. Должен быть способ лучше.

Итак, проходя через территории пяти «племен», мы собирали их открытия, вели разговоры о границах и задумывались, как сложить вместе кусочки мозаики. Сейчас мы знаем неизмеримо больше, чем в начале пути, но чего-то по-прежнему не хватает. В центре мозаики зияет дыра, и поэтому собрать ее трудно. Проблема в том, что все алгоритмы машинного обучения, которые мы до сих пор видели, нуждаются в учителе, который покажет им правильный ответ. Они не могут научиться отличать опухолевую клетку от здоровой, если кто-то не повесит ярлыки «опухоль» и «здоровая клетка». А люди могут учиться без учителя, и делают это с самого первого дня своей жизни. Мы подошли к вратам Мордора[98], и долгий путь будет напрасным, если не обойти это препятствие. Но вокруг бастионов и стражников есть тропинка, и награда близка. Следуйте за мной…

ГЛАВА 8

ОБУЧЕНИЕ БЕЗ УЧИТЕЛЯ

Если вы родитель, все тайны обучения разворачивались прямо на ваших глазах в первые три года жизни ребенка. Новорожденный не умеет говорить, ходить, узнавать предметы и даже не понимает, что то, на что он смотрит, будет существовать и когда он отвернется. Но проходит месяц за месяцем, и маленькими и большими шажками, путем проб, ошибок и больших когнитивных скачков ребенок разбирается, как устроен мир, как ведут себя люди, как с ними общаться. К третьему году плоды обучения сливаются в стабильное «я», в поток сознания, который не прекратится до самой смерти. Более старшие дети и взрослые способны «путешествовать во времени» — вспоминать прошлое, но лишь до этой границы. Если бы мы могли вернуться в младенчество и раннее детство и снова увидеть мир глазами маленького ребенка, многое, что озадачивает нас в механизмах обучения и даже самого бытия, внезапно стало бы очевидным. Но пока величайшая тайна Вселенной — это не ее зарождение или границы и не нити, из которых она соткана, а то, что происходит в мозге маленького ребенка: как из массы серого желе вырастает средоточие сознания.

Хотя наука о механизмах обучения детей все еще молода и исследования начались всего несколько десятилетий назад, ученые уже добились замечательных успехов. Младенцы не умеют заполнять анкеты и не соблюдают протоколов, однако удивительно много информации о том, что происходит у них в голове, можно получить благодаря видеозаписи и изучению их реакций во время эксперимента. Складывается связная картина: разум младенца — это не просто реализация заложенной генетической программы и не биологический прибор для фиксирования корреляций данных, получаемых из органов чувств. Разум ребенка сам активно синтезирует реальность, и со временем она меняется довольно радикально.

Очень удобно, что ученые-когнитивисты все чаще выражают теории детского обучения в форме алгоритмов. Это вдохновляет многих исследователей машинного обучения — ведь все, что нужно,

уже есть там, в мозге ребенка, и надо только каким-то образом ухватить суть и записать ее в компьютерном коде. Некоторые ученые даже утверждают, что для создания разумных машин нужно сконструировать робота-ребенка и позволить ему ощутить мир так, как это делают обычные дети. Мы, исследователи, станем ему родителями (может быть, это будет краудсорсинг, и термин «глобальная деревня»[99] приобретет совершенно новое значение). Маленький Робби — давайте назовем его в честь пухлого, но высокого робота из «Запретной планеты»[100] — единственный робот-ребенок, которого нам надо построить. Как только он обучится всему, что человек знает в три года, проблема искусственного интеллекта будет решена. После этого можно скопировать содержимое его мозга в столько роботов, во сколько захотим, и они будут развиваться дальше: самое сложное уже сделано.

Вопрос, конечно, в том, какие алгоритмы должны работать в мозге Робби в момент рождения. Ученые, находящиеся под влиянием детской психологии, косо смотрят на нейронные сети, потому что работа нейронов на микроскопическом уровне кажется бесконечно далекой от сложности даже простейших действий ребенка: потянуться к предмету, схватить его и рассмотреть широко распахнутыми, полными любопытства глазами. Чтобы за деревьями увидеть планету, обучение ребенка придется моделировать на более высоком уровне абстракции. Самое удивительное, наверное, то, что дети учатся в основном самостоятельно, без надзора, хотя, несомненно, получают огромную помощь от своих родителей. Ни один из алгоритмов, которые мы до сих пор видели, на это не способен, но вскоре мы познакомимся с несколькими вариантами и на шаг приблизимся к Верховному алгоритму.

Как свести рыбака с рыбаком

Мы нажимаем кнопку «Включить», Робби открывает глаза-видео-камеры в первый раз, и его сразу заливает «цветущий и жужжащий беспорядок» мира, как сказал Уильям Джеймс. Новые изображения возникают десятками в секунду, и одна из первоочередных задач — научиться организовывать их в более крупные элементы: реальный

мир состоит не из случайных пикселей, которые каждое мгновение меняются, как им вздумается, а из стабильных во времени объектов. Если мама отошла подальше, вместо нее не появится «уменьшенная мама». Если на стол поставить тарелку, в столе не появится белая дырка. Младенец не отреагирует, если плюшевый мишка скроется за ширмой и вместо него появится самолет, а годовалый ребенок удивится: он каким-то образом уже сообразил, что мишки отличаются от самолетов и не могут просто так превращаться друг в друга. Вскоре после этого он разберется, что некоторые предметы похожи друг на друга, и начнет формировать категории. Если девятимесячному малышу дать гору игрушечных лошадок и карандашей, он и не подумает их разделить, а в полтора года уже догадается.

Организация мира в предметы и категории совершенно естественна для взрослого, но не для младенца, и в еще меньшей степени для робота Робби. Можно, конечно, одарить его зрительной корой в виде многослойного перцептрона и показать подписанные примеры всех предметов и категорий в мире — вот мама рядом, а вот мама далеко, — но так мы никогда не закончим. На самом деле нам нужен алгоритм, который будет спонтанно группировать схожие объекты и разные изображения одного и того же объекта. Это проблема кластеризации, одна из наиболее интенсивно изучаемых тем в науке о машинном обучении.

Кластер — это набор схожих сущностей или как минимум набор сущностей, которые похожи друг на друга больше, чем на элементы других кластеров. Делить все на кластеры — в природе человека, и часто это первый шаг на пути к знанию. Даже глядя в ночное небо, мы невольно видим скопления звезд, а потом придумываем красивые названия формам, которые они напоминают. Наблюдение, что определенные группы веществ имеют очень схожие химические свойства, стало первым шагом к открытию периодической системы элементов: каждая группа в ней заняла свой столбец. Все, что мы воспринимаем — от лиц друзей до звуков речи, — это кластеры. Без них мы бы потерялись. Дети не научатся говорить, пока не приобретут навык определять характерные звуки, из которых состоит речь. Это происходит в первые годы жизни, и все слова, которые они потом узнают, не значат ничего без

кластеров реальных вещей, к которым эти слова относятся. Сталкиваясь с большими данными — очень большим количеством объектов, — мы вначале группируем их в удобное число кластеров. Рынок в целом — слишком общий, отдельные клиенты — слишком мелкие, поэтому маркетологи делят его на сегменты, как они называют кластеры. Даже объекты как таковые, по сути, кластеры наблюдений за ними, начиная с маминого лица под разными углами освещения и заканчивая различными звуковыми волнами, которые ребенок слышит как слово «мама». Думать без объектов невозможно, и, наверное, именно поэтому квантовая механика такая неинтуитивная наука: субатомный мир хочется нарисовать в воображении в виде сталкивающихся частиц или интерферирующих волн, но на самом деле это ни то ни другое.

Кластер можно представить по его элементу-прототипу: образу мамы, который сразу приходит на ум, типичной кошки, спортивного автомобиля, загородного дома и тропического пляжа. Для маркетолога Пеория в штате Иллинойс это средний американский городок. Самый обычный гражданин США — это Боб Бернс, пятидесятитрехлетний завхоз из Уиндема в штате Коннектикут, по крайней мере, если верить книге Кевина О'Кифа The Average American. По всем числовым атрибутам — например, росту, весу, объему талии и обуви, длине волос и так далее — можно легко найти среднего члена кластера: его рост — это средний рост всех остальных, вес — средний вес и так далее. Для описательных атрибутов, например пола, цвета волос, почтового индекса и любимого вида спорта, «средним» значением будет просто самое распространенное. Средние члены кластеров, описанные таким набором атрибутов, могут существовать или не существовать в реальности, но это в любом случае удачные точки для ориентации: когда планируешь маркетинг нового продукта, удобнее представить себе Пеорию как место введения его на рынок и Боба Бернса как целевого клиента, а не оперировать абстрактными сущностями вроде «рынка» или «клиента».

Такие усредненные объекты, конечно, полезны, но можно поступить еще лучше: вообще весь смысл больших данных и машинного обучения как раз в том, чтобы избежать грубых рассуждений. Кластеры могут быть очень специализированными

группами людей или даже различными аспектами жизни одного и того же человека: Элис, покупающая книги для работы, для отдыха или в подарок на Рождество, Элис в хорошем настроении и грустящая Элис. Amazon заинтересована в том, чтобы отделять книги, которые Элис покупает себе, от тех, которые она покупает для своего молодого человека, потому что тогда можно будет дать ей в нужное время подходящую рекомендацию. К сожалению, покупки не сопровождаются ярлыками «подарок для себя» и «для Боба», поэтому Amazon приходится самой разбираться, как их группировать.

Представьте, что объекты в мире Робби делятся на пять кластеров (люди, мебель, игрушки, еда и животные), но неизвестно, какие вещи к каким кластерам относятся. С проблемой этого типа Робби столкнется, как только мы его включим. Простой вариант сортировки объектов по кластерам — взять пять произвольных предметов, сделать их прототипами кластеров, а затем сравнить новые сущности с каждым прототипом, относя их к кластеру самого схожего. (Как и в аналогическом обучении, здесь важно выбрать меру сходства. Если атрибуты числовые, это может быть просто евклидово расстояние, но это далеко не единственный вариант.)

Теперь прототипы надо обновить. Подразумевается, что прототип кластера должен быть средним его членов: когда кластеры состояли из одного члена, все так и было, но теперь мы добавили к ним новые элементы, и ситуация изменилась. Поэтому мы вычислим средние свойства членов для каждого кластера и сделаем полученный результат новым прототипом. Теперь нужно снова обновить принадлежность объектов кластерам: поскольку прототипы изменились, мог измениться и прототип, наиболее близкий данному объекту. Давайте представим, что прототип одной категории — это мишка, а другой — банан. Если взять крекер в виде животного, при первом подходе он может попасть в группу с медведем, а при втором — с бананом. Изначально крекер выглядел как игрушка, но теперь он будет отнесен к еде. Если переместить крекер в одну группу с бананом, прототип для этой группы тоже может измениться: это уже будет не банан, а печенье. Этот полезный цикл, который относит объекты ко все более и более подходящим кластерам, станет продолжаться, пока кластеры

сущностей (а с ними и прототипы кластеров) не прекратят меняться.

Такой алгоритм называется метод *k*-средних, и появился он еще в 50-е годы XX века. Он простой, красивый, при этом довольно популярный, но имеет ряд недостатков, одни из которых устранить легче, а другие — сложнее. Во-первых, количество кластеров надо зафиксировать заранее, а в реальном мире Робби постоянно натыкается на новые виды предметов. Один вариант решения — позволить открывать новый кластер, если объект слишком сильно отличается от имеющихся. Другой — разрешить кластерам делиться и сливаться в процессе работы. Так или иначе, вероятно, будет целесообразно включить в алгоритм приоритеты для меньшего количества кластеров, чтобы избежать ситуации, когда у каждого предмета будет собственный кластер (идеальное решение, если кластеры должны содержать схожие предметы, но смысл явно не в этом).

Более серьезная проблема заключается в том, что метод *k*-средних работает, только когда кластеры легко различимы: они, как пузыри в гиперпространстве, плавают далеко друг от друга, и у всех схожий объем и схожее количество членов. Если какое-то условие не выполнено, начинаются неприятности: вытянутые кластеры делятся надвое, маленькие поглощаются более крупными соседями и так далее. К счастью, можно поступить лучше.

Допустим, мы пришли к выводу, что разрешить Робби слоняться по реальному миру — слишком медленный и громоздкий способ обучения, и вместо этого посадили его смотреть сгенерированные компьютером изображения, как будущего летчика в авиационном тренажере. Мы знаем, из каких кластеров взяты картинки, но не скажем об этом Робби, а будем создавать их, случайно выбирая кластер (скажем, «игрушки»), а потом синтезируя пример этого кластера (маленький пухлый бурый плюшевый медведь с большими черными глазами, круглыми ушами и галстуком-бабочкой). Кроме того, мы будем произвольно выбирать свойства примера: размер мишки — в среднем 25 сантиметров, мех с вероятностью 80 процентов бурый, иначе — белый и так далее. После того как Робби увидит очень много сгенерированных таким образом картинок, он должен научиться делить их на кластеры

«люди», «мебель», «игрушки» и так далее, потому что люди, например, больше похожи на людей, а не на мебель. Возникает интересный вопрос: какой алгоритм кластеризации лучше с точки зрения Робби? Ответ будет неожиданным: наивный байесовский алгоритм — первый алгоритм для обучения с учителем, с которым мы познакомились. Разница в том, что теперь Робби не знает классов и ему придется их угадать!

Очевидно: если бы Робби их знал, все пошло бы отлично — как в наивном байесовском алгоритме, каждый кластер определялся бы своей вероятностью (17 процентов сгенерированных объектов — игрушки) и распределением вероятности каждого атрибута среди членов кластера (например, 80 процентов игрушек коричневые). Робби мог бы оценивать вероятности путем простого подсчета числа игрушек в имеющихся данных, количества коричневых игрушек и так далее, но для этого надо знать, какие предметы — игрушки. Эта проблема может показаться крепким орешком, но, оказывается, мы уже знаем, как ее решить. Если бы в распоряжении Робби имелся наивный байесовский классификатор и ему необходимо было определить класс нового предмета, нужно было бы только применить классификатор и вычислить вероятность класса при данных атрибутах объекта. Маленький, пухлый, коричневый, похож на медведя, с большими глазами и галстуком-бабочкой? Вероятно, игрушка, но, возможно, животное.

Итак, Робби сталкивается с проблемой «курица или яйцо»: зная классы предметов, он мог бы получить модели классов путем подсчета, а если бы знал модели, мог бы сделать заключение о классах объектов. Если вы думаете, что опять застряли, это далеко не так: чтобы стартовать, надо просто начать угадывать классы для каждого предмета каким угодно способом, даже произвольно. На основе этих классов и данных можно получить модели классов, на основе этих моделей — вновь сделать вывод о классах и так далее. На первый взгляд это кажется безумием: придется бесконечно кружиться между выводами о классах на основе моделей и моделей на основе их классов, и даже если это закончится, нет причин полагать, что кластеры получатся осмысленные. Но в 1977 году трое статистиков из Гарварда — Артур Демпстер, Нэн Лэрд и Дональд Рубин — показали, что сумасшедший

план работает: после каждого прохождения по этой петле модель кластера улучшается, а после достижения моделью локального максимума похожести повторения заканчиваются. Они назвали эту схему *EM*-алгоритмом, где *E* — ожидания (expectation, заключение об ожидаемых вероятностях), а *M* — максимизация (maximization, оценка параметров максимальной схожести). Еще они показали, что многие предыдущие алгоритмы были частными случаями *EM*. Например, чтобы получить скрытые модели Маркова, мы чередуем выводы о скрытых состояниях с оценкой вероятностей перехода и наблюдения на их основе. Когда мы хотим получить статистическую модель, но нам не хватает какой-то ключевой информации (например, классов примеров), всегда можно использовать *EM*-алгоритм, что делает его одним из самых популярных инструментов в области машинного обучения.

Вы, возможно, заметили определенное сходство между методом *k*-средних и *EM*-алгоритмом, поскольку оба чередуют отнесение сущностей к кластерам и обновление описаний кластеров. Это не случайность: метод *k*-средних сам по себе — частный случай *EM*-алгоритма, который получается, если у всех атрибутов «узкое» нормальное распределение, то есть нормальное распределение с очень маленькой дисперсией. Если кластеры часто перекрываются, объект может относиться, скажем, к кластеру *A* с вероятностью 0,7 и к кластеру *B* с вероятностью 0,3, и нельзя просто отнести его к кластеру *A* без потери информации. *EM*-алгоритм учитывает это путем частичного приписывания объекта к двум кластерам и соответствующего обновления описаний этих кластеров, однако, если распределения очень сконцентрированы, вероятность, что сущность принадлежит к ближайшему кластеру, всегда будет приблизительно равна единице, и нужно только распределить объекты по кластерам и усреднить их в каждом кластере, чтобы вычислить среднее — то есть получится алгоритм *k*-среднего.

До сих пор мы рассматривали получение всего одного уровня кластеров, но мир, конечно, намного богаче, и одни кластеры в нем вложены в другие вплоть до конкретных предметов: живое делится на растения и животных, животные — на млекопитающих, птиц, рыб и так далее до домашнего любимца — пса Фидо. Но проблем это

не создает: получив набор кластеров, к ним можно относиться как к объектам и, в свою очередь, объединять их в кластеры все более высокого уровня, вплоть до кластера всех объектов. Или же можно начать с грубой кластеризации, а затем все больше дробить кластеры на подкластеры: игрушки Робби делятся на мягкие игрушки, конструкторы и так далее. Мягкие игрушки — на плюшевых медведей, котят и так далее. Дети, видимо, начинают изучение мира где-то посередине, а потом идут и вверх, и вниз. Например, понятие «собака» они усваивают до того, как узнают о «животных» и «гончих». Для Робби это может стать хорошей стратегией.

Открытие формы данных

Независимо от того, поступают ли данные в мозг Робби из его органов чувств или в виде потока миллионов кликов клиентов Amazon, сгруппировать множество в меньшее число кластеров — лишь половина дела. Второй этап — сократить описание объектов. Первый образ мамы, который видит Робби, будет состоять, может быть, из миллиона пикселей, каждый своего цвета, однако человеку вряд ли нужен миллион переменных, чтобы описать лицо. Аналогично каждый товар, на который вы кликнули на сайте Amazon, дает частицу информации о вас, но на самом деле Amazon интересны не ваши клики, а ваши вкусы. Вкусы довольно стабильны и в какой-то мере подразумеваются в кликах, количество которых растет безгранично во время пользования сайтом и должно понемногу складываться в картину предпочтений точно так же, как пиксели складываются в картинку лица. Вопрос в том, как реализовать это сложение.

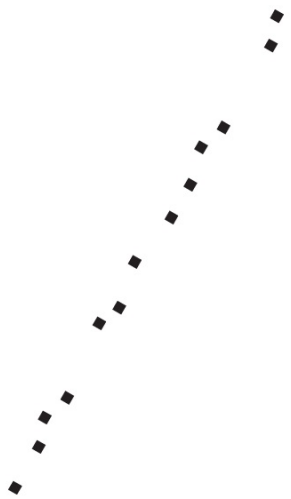
У человека есть примерно 50 лицевых мышц, поэтому 50 чисел должно с лихвой хватить для описания всех возможных выражений лица. Форма глаз, носа, рта и так далее — всего того, что помогает отличить одного человека от другого, — тоже не должна занимать больше нескольких десятков чисел. В конце концов, художникам в полиции достаточно всего десяти вариантов каждой черты лица, чтобы составить фоторобот, позволяющий опознать подозреваемого. Можно добавить еще несколько чисел для

описания освещения и наклона, но на этом все. Поэтому, если вы дадите мне примерно сотню чисел, этого должно хватить для воссоздания лица, и наоборот: мозг Робби должен быть способен взять картинку лица и быстро свести ее ко все той же сотне по-настоящему важных чисел.

Специалисты по машинному обучению называют этот процесс понижением размерности, потому что он уменьшает множество видимых измерений (пикселей) до нескольких подразумеваемых (выражение и черты лица). Понижение размерности важно для того, чтобы справиться с большим объемом данных, например данными, поступающими каждую секунду из органов чувств. Может быть, действительно лучше один раз увидеть, чем сто раз услышать, но обрабатывать и запоминать изображения в миллион раз сложнее, чем слова. Тем не менее зрительная кора головного мозга каким-то образом довольно хорошо справляется с уменьшением такого объема информации до приемлемого, достаточного, чтобы ориентироваться в мире, узнавать людей и предметы и помнить увиденное. Это великое чудо познания настолько естественно для нас, что мы его даже не замечаем.

Наводя порядок в своей библиотеке, вы тоже выполняете своего рода понижение размерности от обширного пространства тем до одномерной полки. Некоторые тесно связанные книги неизбежно окажутся далеко друг от друга, но все равно можно расставить их так, чтобы такие случаи были редкими. Алгоритм понижения размерности делает именно это.

Представьте, что я дал вам координаты GPS всех магазинов в Пало-Альто в Калифорнии и вы нанесли их на листок бумаги:

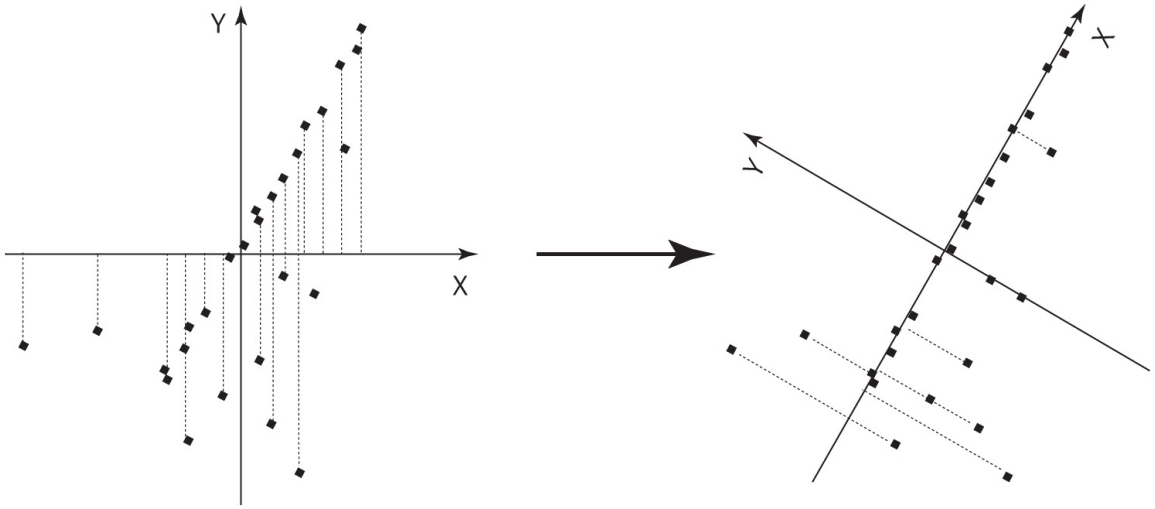


Наверное, взглянув на эту схему, вы сразу поймете, что главная улица городка ведет с юго-запада на северо-восток. Хотя вы не рисовали саму улицу, интуиция подсказывает, где она проходит, потому что все точки лежат на прямой линии (или рядом с ней — магазины могут быть по разные стороны улицы). Догадка верна: эта улица — Юниверсити-авеню, и, если вы окажетесь в Пало-Альто и захотите перекусить и сделать покупки, туда и надо идти. Еще лучше, что, когда магазины сконцентрированы на одной улице, для описания их расположения нужно уже не два числа, а всего одно — номер дома, а для большей точности — расстояние от магазина до пригородной железнодорожной станции в юго-западном углу, откуда начинается Юниверсити-авеню.

Если нанести на карту еще больше магазинов, вы, вероятно, заметите, что часть из них находится на перекрестках, чуть в стороне от Юниверсити-авеню, а некоторые — вообще в других местах:

Тем не менее большинство магазинов все равно расположены довольно близко к центральной улице, и, если разрешено использовать для описания положения магазина только одно число, расстояние от вокзала вдоль этой улицы будет довольно удачным вариантом: пройдя этот отрезок и оглядевшись, вы с достаточной вероятностью найдете нужный магазин. Итак, вы только что понизили размерность «расположения магазинов в Пало-Альто» с двух измерений до одного.

У Робби, однако, нет преимуществ, которые дает человеку сильно развитая зрительная система, поэтому, если вы попросите его забрать белье из химчистки Elite Cleaners и учтете на его карте только одну координату, ему нужен будет алгоритм, чтобы «открыть» Юниверсити-авеню на основе GPS-координат магазинов. Ключ к решению проблемы — заметить, что, если поставить начало координат плоскости *x*, *y* в усредненное расположение магазинов и медленно поворачивать оси, магазины окажутся ближе всего к оси *x* при повороте примерно на 60 градусов, то есть когда ось совпадает с Юниверсити-авеню:



Это направление — так называемая первая главная компонента данных — будет направлением, вдоль которого разброс данных наибольший. (Обратите внимание: если спроецировать магазины на ось *x*, на правом рисунке они будут находиться дальше друг от друга, чем на левом.) Обнаружив первую главную компоненту, можно поискать вторую, которой в данном случае станет направление наибольшей дисперсии под прямым углом к Юниверсити-авеню. На карте остается только одно возможное направление (направление перекрестков). Но если бы Пало-Альто находился на склоне холма, одна или две главные компоненты частично были бы расположены непосредственно на холме, а третья — последняя — оказалась бы направлена в воздух. Ту же идею можно применить к тысячам и миллионам измерений данных, как в случае изображений лиц: нужно последовательно искать направления наибольшей дисперсии, пока оставшаяся вариабельность не окажется наименьшей. Например, после поворота осей на рисунке выше координата *y* большинства магазинов будет равна нулю, поэтому среднее *y* окажется очень маленьким, и, если его вообще проигнорировать, потеря информации получится незначительной. А если мы все же решим сохранить *y*, то *z* (направленная вверх) наверняка будет несущественна. Как оказалось, линейная алгебра позволяет провести процесс поиска главных компонент всего за один цикл, но еще лучше то, что даже в данных с очень большим количеством измерений значительную часть дисперсии зачастую дают всего

несколько измерений. Если это не так, все равно визуальный поиск двух-трех важнейших измерений часто оказывается очень успешным, потому что наша зрительная система дает удивительные возможности восприятия.

Метод главных компонент (Principal Component Analysis, PCA), как называют этот процесс, — один из важнейших инструментов в арсенале ученого. Можно сказать, что для обучения без учителя это то же самое, что линейная регрессия для контролируемого множества. Знаменитая «клюшкообразная» кривая глобального потепления, например, была получена в результате нахождения главной компоненты различных рядов данных, связанных с температурой (годичные кольца деревьев, ледяные керны и так далее), и допущения, что это запись температуры как таковой. Биологи используют метод главных компонент, чтобы свести уровни экспрессии тысяч различных генов в несколько путей. Психологи обнаружили, что личность можно выразить пятью факторами — это экстраверсия, доброжелательность, добросовестность, нейротизм и открытость опыту, — которые оценивают по твитам и постам в блогах. (У шимпанзе, предположительно, есть еще одно измерение — реактивность, — но их с помощью Twitter не оценишь.) Применение метода главных компонент к голосам на выборах в Конгресс и данным избирателей показывает, что, вопреки расхожему мнению, политика в основном не сводится к противостоянию либералов и консерваторов. Люди отличаются в двух основных измерениях — экономических и социальных вопросах, — и, если спроецировать их на одну ось, либертарианцы смешаются с популистами, хотя их позиции полярно противоположны, и возникнет иллюзия, что в центре много умеренных. Попытка апеллировать к ним вряд ли окажется выигрышной стратегией. С другой стороны, если либералы и либертарианцы преодолеют взаимную неприязнь, они могут стать союзниками в социальных вопросах, где и те и другие выступают за свободу личности.

Когда Робби подрастет, он сможет применять один из вариантов метода главных компонент для решения проблемы «эффекта вечеринки», то есть чтобы выделить из шума толпы отдельные голоса. Схожий метод может помочь ему научиться читать. Если

каждое слово — измерение, тогда текст — точка в пространстве слов, и главные направления этого пространства окажутся элементами значения. Например, «президент Обама» и «Белый дом» в пространстве слов далеко отстоят друг от друга, но в пространстве значений близки, потому что обычно появляются в схожих контекстах. Хотите верьте, хотите нет, но такой тип анализа — все, что требуется и компьютерам, и людям для оценки сочинений на экзаменах SAT (стандартизованный тест для приема в высшие учебные заведения США). В Netflix используется похожая идея. Вместо того чтобы рекомендовать фильмы, которые понравились пользователям со схожими вкусами, система проецирует и пользователей, и фильмы в «пространство вкуса» с низкой размерностью и рекомендует картины, расположенные в этом пространстве рядом с вами. Это помогает найти фильмы, которые вы никогда не видели, но обязательно полюбите.

Тем не менее главные компоненты набора данных о лице вас, скорее всего, разочаруют. Вопреки ожиданиям, это будут, например, не черты лица и выражения, а скорее размытые до неузнаваемости лица призраков. Дело в том, что метод главных компонент — линейный алгоритм, поэтому главными компонентами могут быть только взвешенные пиксель за пикселем средние реальных лиц (их еще называют «собственные лица», потому что они собственные векторы центрированной ковариационной матрицы этих данных, но я отхожу от темы). Чтобы по-настоящему понять не только лица, но и большинство форм в мире, нам понадобится кое-что еще, а именно нелинейное понижение размерности.

Представьте, что вместо карты Пало-Альто у нас есть GPS-координаты важнейших городов всей Области залива Сан-Франциско:

Глядя на эту схему, можно, вероятно, сделать предположение, что города расположены на берегу залива и, если провести через них линию, положение каждого города можно определить всего одним числом: как далеко он находится от Сан-Франциско по этой линии. Но метод главных компонент такую кривую найти не может: он нарисует прямую линию через центр залива, где городов вообще нет, а это только затуманит, а не прояснит форму данных.

Теперь представьте на секунду, что мы собираемся создать Область залива с нуля. Бюджет позволяет построить единую дорогу, чтобы соединить все города, и нам решать, как она будет проложена. Естественно, мы проложим дорогу, ведущую из Сан-Франциско в Сан-Бруно, оттуда в Сан-Матео и так далее, вплоть до Окленда. Такая дорога будет довольно хорошим одномерным представлением Области залива, и ее можно найти простым алгоритмом: «построй дорогу между каждой парой близлежащих городов». Конечно, у нас получится целая сеть дорог, а не одна, проходящая рядом с каждым городом, но можно получить и одну дорогу, если сделать ее наилучшим приближением сети, в том смысле, что расстояния между городами по этой дороге будут как можно ближе расстоянию вдоль сети.

Именно это делает Isomap — один из самых популярных алгоритмов нелинейного понижения размерности. Он соединяет

каждую точку данных в высокоразмерном пространстве (пусть это будет лицо) со всеми близлежащими точками (очень похожими лицами), вычисляет кратчайшие расстояния между всеми парами точек по получившейся сети и находит меньшее число координат, которое лучше всего приближает эти расстояния. В отличие от метода опорных векторов, координаты лиц в этом пространстве часто довольно осмысленны: одна координата может показывать, в каком направлении смотрит человек (левый профиль, поворот на три четверти, анфас и так далее), другая — как лицо выглядит (очень грустное, немного грустное, спокойное, радостное, очень радостное) и так далее. Isomap имеет неожиданную способность сосредоточиваться на самых важных измерениях комплекса данных — от понимания движения на видео до улавливания эмоций в речи.

Вот интересный эксперимент. Возьмите потоковое видео из глаз Робби, представьте, что каждый кадр — точка в пространстве изображений, а потом сведите этот набор изображений к единственному измерению. Какое это будет измерение? Время. Как библиотекарь расставляет книги на полке, время ставит каждое изображение рядом с самыми похожими. Наверное, наше восприятие времени — просто естественный результат замечательной способности головного мозга понижать размерность. В дорожной сети нашей памяти время — главная автострада, и мы ее быстро находим. Другими словами, время — главная компонента памяти.

Жизнелюбивый робот

Кластеризация и понижение размерности приближают нас к пониманию человеческого обучения, но чего-то очень важного все равно не хватает. Дети не просто пассивно наблюдают за миром. Они активно действуют: замечают предметы, берут их в руки, играют, бегают, едят, плачут и задают вопросы. Даже самая продвинутая зрительная система будет бесполезна, если она не поможет Робби взаимодействовать со средой. Ему нужно не просто знать, где что находится, но и что надо делать в каждый момент. В принципе можно научить его выполнять пошаговые инструкции, соотнося показания сенсоров с соответствующими

ответными действиями, но это возможно только для узких задач. Предпринимаемые действия зависят от цели, а не просто от того, что вы в данный момент воспринимаете, при этом цели могут быть весьма отдаленными. И потом, в любом случае нужно обойтись без пошагового контроля: дети учатся ползать, ходить и бегать сами, без помощи родителей. Ни один рассмотренный нами обучающий алгоритм так учиться не умеет.

У людей действительно есть один постоянный ориентир: эмоции. Мы стремимся к удовольствиям и избегаем боли. Коснувшись горячей плиты, вы непроизвольно отдернете руку. Это просто. Сложнее научиться не трогать плиту: для этого нужно двигаться так, чтобы избежать острой боли, которую вы еще не почувствовали. Головной мозг делает это, ассоциируя боль не просто с моментом прикосновения к плите, но и с ведущими к этому действиями. Эдвард Торндайк[101] назвал это законом эффекта: действия, которые ведут к удовольствию, станут с большей вероятностью повторяться в будущем, а ведущие к боли — с меньшей. Удовольствие как будто путешествует назад во времени, и действия в конце концов могут начать ассоциироваться с довольно отдаленными результатами. Люди освоили такой поиск косвенных наград лучше, чем любое животное, и этот навык критически важен для успеха в жизни. В знаменитом эксперименте детям давали зефир и говорили, что, если они выдержат несколько минут и не съедят его, им дадут целых два. Те, кому это удалось, лучше успевали в школе и позже, когда стали взрослыми. Менее очевидно, наверное, то, что с аналогичной проблемой сталкиваются компании, использующие машинное обучение для совершенствования своих сайтов и методов ведения бизнеса. Компания может принять меры, которые принесут ей больше денег в краткосрочной перспективе — например, начать по той же цене продавать продукцию худшего качества, — но не обратить внимания, что в долгосрочной перспективе это приведет к потере клиентов.

Обучающиеся алгоритмы, которые мы видели в предыдущих главах, руководствуются немедленным удовлетворением: каждое действие, будь то выявление письма со спамом или покупка ценных бумаг, получает непосредственное поощрение или наказание

от учителя. Но есть целый подраздел машинного обучения, посвященный алгоритмам, которые исследуют мир сами по себе: трудятся, сталкиваются с наградами, определяют, как получить их снова. Во многом они похожи на детей, которые ползают по комнате и тащат все в рот.

Это обучение с подкреплением, и этот принцип, скорее всего, станет активно использовать ваш первый домашний робот. Если вы распакуете Робби, включите его и попросите приготовить яичницу с беконом, у него с ходу может не получиться. Но когда вы уйдете на работу, он изучит кухню, отметит, где лежит утварь, какая у вас плита. Когда вы вернетесь, ужин будет готов.

Важным предшественником обучения с подкреплением была программа для игры в шашки, созданная ученым Артуром Сэмюэлом, работавшим в 1950-х годах в IBM. Настольные игры — прекрасный пример проблемы обучения с подкреплением: надо построить длинную последовательность ходов без какой-то обратной связи, а награда или наказание — победа или поражение — ждет в самом конце. Программа Сэмюэла оказалась способна научиться играть так не хуже большинства людей. Она не искала напрямую, какой ход сделать при каждом положении на доске (это было бы слишком сложно), а скорее училась оценивать сами положения — какова вероятность выигрыша, если начать с этой позиции? — и выбирать ходы, ведущие к наилучшему положению. Поначалу программа умела оценивать только конечные позиции: победа, ничья и поражение. Но раз определенные позиции означают победу, значит, позиции, из которых можно к ней прийти, хорошие. Томас Уотсон-старший, президент IBM, предсказал, что после презентации программы акции корпорации поднимутся на 15 пунктов. Так и произошло. Урок был усвоен, IBM развила успех и создала чемпионов по игре в шахматы и Jeopardy!.

Мысль, что не все состояния ведут к награде (положительной или отрицательной), но у каждого состояния имеется ценность, — центральный пункт обучения с подкреплением. В настольных играх награды есть только у конечных позиций (например, 1, 0 и –1 для победы, ничьей и поражения). Другие позиции не дают немедленной награды, но их ценность в том, что они могут обеспечить награду в будущем. Позиция в шахматах, из которой

можно поставить мат в определенное количество ходов, практически так же хороша, как сама победа, и потому имеет высокую ценность. Такого рода рассуждения можно распространить вплоть до хороших и плохих дебютов, даже если на таком расстоянии от цели связь с наградой далеко не очевидна. В компьютерных играх награды обычно выражаются в очках, и ценность состояния — это количество очков, которые можно накопить, начиная с этого состояния. В реальной жизни отдача с задержкой менее выгодна, чем немедленная отдача, поэтому ее можно уменьшать на определенный процент, как это делается в случае инвестиций. Естественно, награда зависит от того, какие действия вы выберете, и цель обучения с подкреплением — всегда выбирать действие, ведущее к наибольшей награде. Стоит ли снять трубку и пригласить знакомую на свидание? Это может и положить начало чудесному роману, и привести к болезненному разочарованию. А если ваша подруга согласится на свидание, оно может пойти как удачно, так и неудачно. Надо каким-то образом абстрагироваться от бесконечных вариантов развития событий и принять решение. Обучение с подкреплением делает это путем оценки ценности каждого состояния — общей суммы наград, которых можно ожидать, начиная с него, — и выбора действий, которые ее максимизируют.

Представьте, что вы, как Индиана Джонс, пробираетесь по лабиринту и доходите до развилки. Карта подсказывает, что туннель слева ведет к сокровищнице, а справа — в яму со змеями. Ценность места, где вы стоите — прямо на распутье, — равна ценности сокровищ, потому что вы пойдете налево. Если всегда выбирать наилучшее возможное действие, ценность текущего состояния будет отличаться от ценности последующего только непосредственной наградой за выполнение этого действия, если таковая имеется. Если известны непосредственные награды каждого состояния, можно использовать их для обновления ценности соседних состояний и так далее, пока значения всех состояний не будут согласованы: ценность сокровища распространяется назад по лабиринту до развилки и еще дальше. Зная ценность состояний, вы поймете, какое действие выбрать в каждом из них (то, которое дает максимальное сочетание

немедленной награды и ценности результирующего состояния). Все это было открыто еще в 1950-е годы теоретиком управления Ричардом Беллманом[102]. Однако настоящая проблема обучения с подкреплением появляется, когда карты местности у вас нет и остается только исследовать ее самостоятельно, определяя награды. Иногда получается найти драгоценности, иногда падаешь в яму со змеями. Каждое предпринятое действие дает информацию и о непосредственной награде, и о результирующем состоянии. Это можно сделать путем обучения с учителем. Однако нужно обновить и значение состояния, из которого вы только что пришли, чтобы привести его в соответствие с наблюдаемым значением, а именно суммой полученной награды и значения нового состояния, в котором вы оказались. Конечно, значение может пока быть неправильным, но, если достаточно долго ходить вокруг, в конце концов будут найдены правильные значения всех состояний и соответствующих действий. В этом в двух словах заключается обучение с подкреплением.

Обратите внимание, что обучение с подкреплением сталкивается с той же дилеммой изучения–применения, с которой мы познакомились в главе 5: чтобы максимизировать награды, вы, естественно, всегда хотите выбирать действие, ведущее к состоянию с наибольшим значением, но это не дает открыть потенциально б*о*льшие награды в других местах. Алгоритмы обучения с подкреплением решают эту проблему, иногда выбирая лучшее действие, а иногда — случайное. (В головном мозге, кажется, для этого есть даже «генератор шумов».) На ранних этапах, когда можно получить много информации, имеет смысл больше изучать. Когда территория известна, лучше будет сосредоточиться на применении знания. Люди делают это на протяжении жизни: дети учатся, а взрослые используют (кроме ученых, которые похожи на вечных детей). Детская игра намного серьезнее, чем может показаться: если эволюция создала существо, которое в первые несколько лет своей жизни беспомощно и только обременяет родителей, такая расточительность должна давать большие преимущества. По сути, обучение с подкреплением — своего рода ускоренная эволюция, которая позволяет попробовать, отбросить

и отточить действия в течение одной жизни, а не многих поколений, и по этим меркам оно крайне эффективно.

Начало серьезным исследованиям обучения с подкреплением положили в 1980-х годах работы Рича Саттона и Энди Барто из Массачусетского университета. Ученые чувствовали, что обучение в очень большой степени зависит от взаимодействия со средой, а контролирующие алгоритмы этого не улавливают, и нашли вдохновение в психологии обучения животных. Саттон продолжил заниматься этой темой и стал ведущим сторонником обучения с подкреплением. Еще один ключевой шаг был сделан в 1989 году, когда Крис Уоткинс из Кембриджа, которого изначально мотивировали экспериментальные наблюдения за обучением детей, пришел к современной формулировке обучения с подкреплением как оптимального контроля в неизвестной среде.

Тем не менее алгоритмы обучения с подкреплением, которые мы видели до сих пор, не очень реалистичны, потому что не знают, что делать в данном состоянии, если раньше в нем не были, а в реальном мире не бывает двух совершенно одинаковых ситуаций. Нужно уметь делать обобщения, выводя из посещенных состояний новые. К счастью, этому мы уже научились: достаточно просто обернуть обучение с подкреплением вокруг одного из алгоритмов обучения с учителем, с которыми мы познакомились раньше, например многослойного перцептрона. Теперь нейронная сеть будет предсказывать значение состояния, а сигналом ошибки для обратного распространения станет разница между предсказанными и наблюдаемыми значениями. Но есть и проблема. В обучении с учителем целевое значение состояния всегда одно и то же, а в обучении с подкреплением оно продолжает меняться в силу обновлений соседних состояний, поэтому обучение с подкреплением и обобщением часто не умеет приходить к стабильному решению, если только обучающийся алгоритм внутри не простейший, например линейная функция. Несмотря на это, обучение с подкреплением в сочетании с нейронными сетями принесло ряд заметных успехов. Одним из первых достижений стала программа, играющая в нарды на уровне человека. Позже алгоритм обучения с подкреплением,

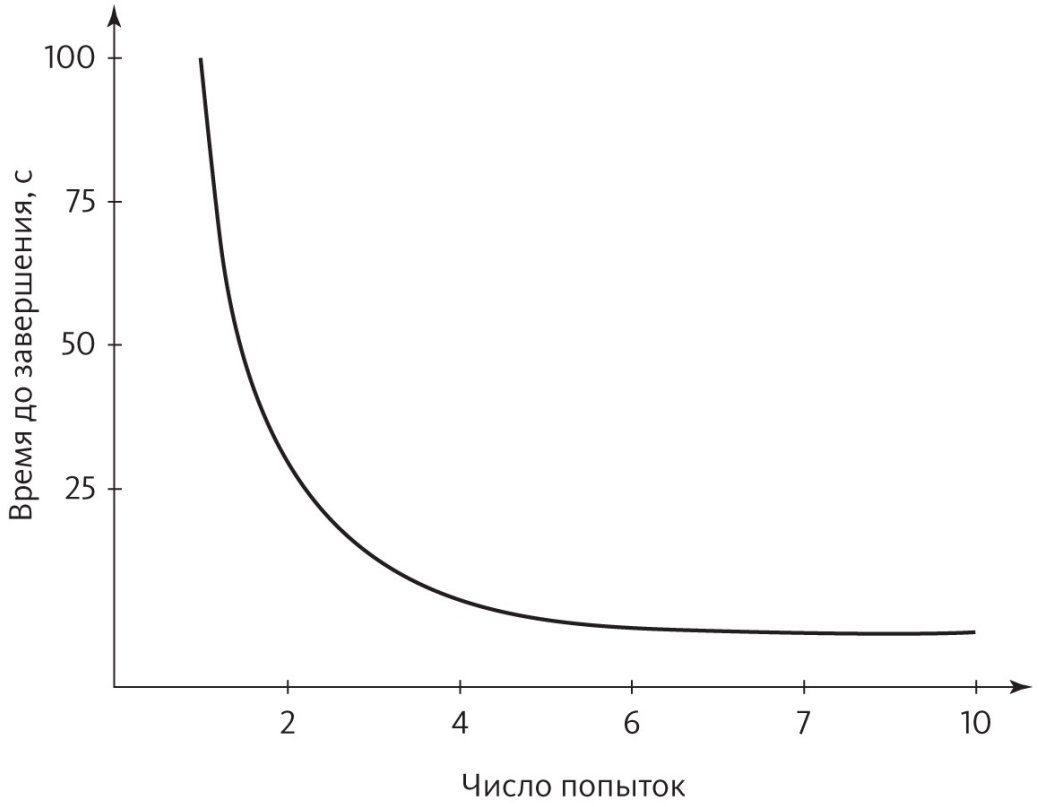
разработанный в лондонском стартапе DeepMind, победил хорошего игрока в Pong и другие простые аркады. Для прогнозирования ценности действий на основе «сырых» пикселей экрана игровой приставки в нем использовалась глубокая сеть. Благодаря непрерывному зрению, обучению и контролю система имела как минимум поверхностное сходство с искусственным мозгом. Неудивительно, что Google заплатила за DeepMind полмиллиарда долларов, хотя у компании не имелось ни продукции, ни выручки и сотрудников было немного.

Кроме компьютерных игр, ученые использовали обучение с подкреплением для управления гимнастами — человечками из палочек, парковки задним ходом, пилотирования вертолетов вверх ногами, управления автоматическими телефонными диалогами, выделения каналов в сетях сотовой связи, вызова лифта, составления расписаний загрузки космического челнока и многих других целей. Обучение с подкреплением повлияло на психологию и нейробиологию. В мозге оно осуществляется благодаря нейромедиатору дофамину, который позволяет распространить разницу между ожидаемыми и фактическими наградами. Обучением с подкреплением можно объяснить условные рефлексы по Павлову, и, в отличие от бихевиоризма, такой подход допускает, что у животных есть внутренние психические состояния. Этот вид обучения используют пчелы-сборщицы и мыши, ищущие сыр в лабиринте. Человеческая повседневность — это поток почти незаметных чудес, которые возможны отчасти благодаря обучению с подкреплением. Вы встаете, одеваетесь, завтракаете, едете на работу, и все это автоматически, думая о чем-то другом. Где-то в глубине обучение с подкреплением постоянно дирижирует процессом и тонко настраивает удивительную симфонию движений. Элементы обучения с подкреплением, также называемые привычками, составляют бо́льшую часть наших действий: проголодался — идешь к холодильнику и берешь что-нибудь перекусить. Как показал Чарльз Дахигг в книге The Power of Habit[103], понимание и управление этим циклом намеков, рутинных действий и наград — ключ к успеху не только для отдельных людей, но и для бизнеса, и даже для общества в целом.

Из всех отцов обучения с подкреплением самый большой энтузиаст этого метода — Рич Саттон. Для него обучение с подкреплением — Верховный алгоритм, и решение этой проблемы равноценно решению проблемы искусственного интеллекта. С другой стороны, Крис Уоткинс не удовлетворен этим подходом и видит много того, что могут делать дети и не могут алгоритмы обучения с подкреплением: решать проблемы, решать их лучше после какого-то количества попыток, планировать, усваивать все более абстрактное знание. К счастью, для этих высокоуровневых способностей у нас тоже есть обучающиеся алгоритмы, и самый важный из них — алгоритм образования фрагментов, или *chunking*.

Повторенье — мать ученья

Учиться — значит становиться лучше с практикой. Сейчас вы, может быть, и не помните, как сложно было научиться завязывать шнурки. Сначала не получалось вообще ничего, хотя вам было целых пять лет. Потом шнурки, наверное, развязывались быстрее, чем вы успевали их завязать. Но постепенно вы научились завязывать их быстрее и лучше, пока движения не стали совершенно автоматическими. То же самое происходило, например, с ползанием, ходьбой, бегом, ездой на велосипеде и вождением автомобиля, чтением, письмом и арифметикой, игрой на музыкальных инструментах и занятиями спортом, приготовлением пищи и работой на компьютере. По иронии судьбы, больше всего пользы приносит самое болезненное обучение: поначалу сложен каждый шаг, вы раз за разом терпите неудачу, и, даже если получается, результаты не впечатляют. Освоив замах в гольфе или подачу в теннисе, можно годами оттачивать мастерство, но все эти годы дадут меньше, чем первые несколько недель. С практикой вы становитесь искуснее, но скорость не постоянна: сначала улучшения приходят быстро, потом все медленнее, а затем совсем замедляются. Неважно, осваиваете вы игры или учитесь играть на гитаре: кривая зависимости улучшения результатов от времени — насколько хорошо вы что-то делаете и сколько времени это занимает — имеет очень характерную форму:



Этот тип кривой называют степенным законом, потому что изменение эффективности зависит от возведения времени в какую-то отрицательную степень. Например, на рисунке выше время до завершения пропорционально числу попыток, возведенному в минус вторую степень (или, эквивалентно, единице, разделенной на квадрат числа попыток). Практически все человеческие навыки следуют степенному закону, и разным умениям соответствуют разные степени. (А вот Windows с практикой не ускоряется — Microsoft есть над чем поработать.)

В 1979 году Аллен Ньюэлл и Пол Розенблюм[104] начали задумываться, в чем причина так называемого степенного закона практики. Ньюэлл был одним из основателей науки об искусственном интеллекте и ведущим когнитивным психологом, а Розенблюм — его студентом в Университете Карнеги–Меллон. В то время ни одна из существующих моделей практики не могла объяснить степенной закон. Ньюэлл и Розенблюм подозревали, что он как-то связан с образованием фрагментов — понятием из психологии восприятия и памяти. Информацию мы воспринимаем и запоминаем фрагментами и одномоментно можем

удерживать в краткосрочной памяти лишь определенное количество таких кусочков (согласно классической статье Джорджа Миллера — семь, плюс-минус два). Критически важно, что группировка объектов позволяет обрабатывать намного больше информации, чем если бы мы этого не делали, поэтому в телефонных номерах ставят дефисы: 17-23-458-38-97 запомнить намного легче, чем 17234583897. Герберт Саймон[105], давний коллега Ньюэлла и один из основоположников изучения искусственного интеллекта, до этого открыл, что основное различие между начинающим и профессиональным шахматистом заключается в том, что новичок воспринимает шахматные позиции по одной за раз, в то время как профессионал видит более крупные паттерны, состоящие из многих элементов. Совершенствование шахматной игры в основном сводится к усвоению большего количества более крупных кусков. Ньюэлл и Розенблюм выдвинули гипотезу, что аналогичный процесс имеет место не только в шахматах, но и в усвоении навыков.

В восприятии и памяти фрагмент — это просто символ, который соответствует паттерну других символов: например, ИИ означает искусственный интеллект. Ньюэлл и Розенблюм адаптировали эту идею для теории решения проблем, уже разработанной Ньюэллом в соавторстве с Саймоном. Тогда в ходе эксперимента участников просили решать задачи, например выводить на доске одну математическую формулу из другой и одновременно вслух комментировать свои действия. Ученые выяснили, что человек решает проблемы путем разложения их на подпроблемы, подподпроблемы и так далее и систематически уменьшает различия между начальным состоянием (скажем, первой формулой) и целевым состоянием (второй формулой). Однако для того чтобы это сделать, надо найти рабочую последовательность действий, а на это требуется время. Гипотеза Ньюэлла и Розенблюма заключалась в том, что, решая подпроблему, мы каждый раз формируем фрагмент, который позволяет прямо перейти из состояния до решения в состояние после. Фрагмент в этом смысле состоит из двух частей: стимула (паттерна, который вы узнаёте во внешнем мире или в краткосрочной памяти) и реакции (последовательности действий, которую вы в результате

выполняете). Полученный фрагмент хранится в долгосрочной памяти. В следующий раз, когда надо будет решить ту же подпроблему, можно будет легко применить его и сэкономить время на поиски. Это происходит на всех уровнях, пока не появится фрагмент для целой проблемы, позволяющий решить ее автоматически. Чтобы завязать шнурки, вы завязываете первый узел, делаете на одном конце петлю, оборачиваете вокруг нее другой конец и продеваете ее через петлю посередине. Каждое из этих действий для пятилетнего ребенка далеко не тривиально, но после усвоения соответствующих фрагментов дело почти сделано.

Розенблюм и Ньюэлл применили свою программу образования фрагментов для решения ряда проблем, измерили время, необходимое для каждой попытки, и — подумать только — получили ряд степенных кривых. Но это было только начало. Ученые встроили образование фрагментов в Soar — общую теорию познания, над которой Ньюэлл работал с Джоном Лэрдом[106], еще одним своим студентом. Программа Soar не действовала в рамках заданной иерархии целей — она умела определять новые подпроблемы и решать их каждый раз, когда сталкивалась с препятствием. Формируя новый фрагмент, Soar обобщала его, чтобы применить к схожим проблемам при помощи метода, похожего на обратную дедукцию. Образование фрагментов в Soar оказалось хорошей моделью не только для степенного закона практики, но и для многих феноменов обучения. Его можно было применять даже для получения нового знания путем разбивки данных на фрагменты и аналогии. Это привело Ньюэлла, Розенблюма и Лэрда к гипотезе, что образование фрагментов — *единственный* механизм, необходимый для обучения, иными словами — Верховный алгоритм.

Ньюэлл, Саймон, их студенты и последователи были классическими специалистами по искусственному интеллекту и твердо верили, что самое главное — решать проблемы. Обучающийся алгоритм может быть простым и ехать на закорках у мощного решателя задач. Действительно, обучение — просто еще один вид решения проблем. Ньюэлл и его соратники сосредоточили усилия на сведении всего обучения к образованию фрагментов,

а всего познания — к Soar, но не достигли успеха. Проблема заключалась в следующем: по мере того как решатель задач узнавал все больше фрагментов, а сами фрагменты усложнялись, цена их проверки часто становилась слишком высокой, и программа не ускорялась, а замедлялась. Людям каким-то образом удается этого избежать, но ученые пока не разобрались, как именно. В довершение всего попытки свести обучение с подкреплением, обучение с учителем и все остальное к образованию фрагментов порождало больше проблем, чем решало. В итоге разработчики Soar признали поражение и встроили в программу другие типы обучения в качестве отдельных механизмов. Но, несмотря на это, разбивка на фрагменты остается выдающимся примером обучающегося алгоритма, вдохновенного психологией, и настоящий Верховный алгоритм, какой бы он ни был, несомненно будет уметь совершенствоваться с практикой.

Метод образования фрагментов и обучение с подкреплением используются в бизнесе не так широко, как обучение с учителем, кластеризация и понижение размерности, но есть и более простой тип обучения путем взаимодействия со средой: определение последствий (и действие в соответствии с полученной информацией). Если домашняя страница вашего интернет-магазина голубого цвета и вы задумываетесь, не сделать ли ее красной для повышения продаж, протестируйте новый вариант на 100 тысячах случайно отобранных клиентов и сравните результаты с теми, кто видел обычный сайт. Эту методику, называемую A/B-тестированием, поначалу применяли в основном при испытаниях лекарств, но с того времени она распространилась на многие области, где данные под рукой — от маркетинга до предоставления помощи иностранным государствам. Его можно обобщить для одновременной проверки многих сочетаний изменений, не запутываясь, какие изменения ведут к каким приобретениям (или потерям). Amazon, Google и другие компании верят этому тестированию безгранично. Вы, скорее всего, сами того не подозревая, участвовали в тысячах A/B-тестов. Этот метод показывает ошибочность расхожего мнения, что большие данные хороши для нахождения корреляций, но не причинно-следственных связей. Если оставить в стороне философские тонкости,

определение причинности — нахождение последствий действий, и оно доступно каждому — от годовалого ребенка, который плещется в ванночке, и до президента, ведущего кампанию по переизбранию, — был бы поток данных, на который есть возможность влиять.

Как найти соотношения

Если мы одарим нашего Робби всеми способностями к обучению, которые до сих пор видели в этой книге, он будет достаточно умным, но немного аутичным. Мир для него окажется скоплением отдельных предметов: он начнет узнавать их, манипулировать ими и даже делать в их отношении прогнозы, но не будет понимать, что мир — это сеть взаимосвязей. Робби-врач станет ставить диагнозы человеку с гриппом на основе симптомов, но не заподозрит свиной грипп на том основании, что пациент контактировал с носителем вируса. До появления Google поисковые движки решали, соответствует ли веб-страница вашему запросу, заглядывая в ее содержимое, — что еще можно сделать? Идея Брина и Пейджа заключалась в том, что самый сильный признак, указывающий на то, что страница подходит, — это ссылки на нее с других подходящих страниц. Аналогично, если вы хотите предсказать, рискует ли подросток начать курить, — лучшее, что вы можете сделать, — проверить, курят ли его близкие друзья. Форма фермента неотделима от формы молекул, которые он переносит, как замок неотделим от ключа. Хищника и жертву объединяют сильно взаимосвязанные свойства, каждое из которых эволюционировало, чтобы победить соперника. Во всех этих случаях лучший способ понять сущность — будь то человек, животное, веб-страница или молекула, — понять, как она связана с другими сущностями. Для этого требуется новый род обучения, который относится к данным не как к случайной выборке не связанных друг с другом объектов, а как к возможности взглянуть на сложную сеть. Узлы в этой сети взаимодействуют: то, что вы делаете с одним, влияет на другие и возвращается, чтобы повлиять на вас. Реляционные обучающиеся алгоритмы, как они называются, могут не иметь социальных навыков, но близки к этому. В традиционном статистическом

обучении каждый человек как остров, вещь в себе. В реляционном обучении люди — кусочки континента, часть главного. Они реляционные обучающиеся алгоритмы, связанные и подключенные друг к другу, и, если мы хотим, чтобы Робби вырос восприимчивым, социально адаптированным роботом, его тоже надо подключить.

Первая сложность, с которой мы сталкиваемся, заключается в следующем: если данные образуют одну большую сеть, вместо большого числа примеров для обучения у нас, видимо, будет всего один, а этого недостаточно. Наивный байесовский алгоритм узнает, что высокая температура — один из симптомов гриппа, путем подсчета больных гриппом пациентов с лихорадкой. На основе одного случая он либо сделает вывод, что грипп всегда вызывает высокую температуру, либо что он никогда ее не вызывает. И то и другое ложно. Мы хотели бы определить, что грипп — заразная болезнь, посмотрев на паттерны инфекции в социальной сети — группа зараженных людей тут, группа незараженных там, — но посмотреть мы можем только на один паттерн, даже если он представляет собой сеть из семи миллиардов людей, поэтому неясно, как тут делать обобщения. Ключ к решению — обратить внимание на то, что при погружении в большую сеть в нашем распоряжении оказывается много примеров *пар*. Если у пары знакомых выше вероятность заболеть гриппом, чем у пары людей, которые никогда не встречались, то знакомство с заболевшим делает и вас уязвимее для этой болезни. К сожалению, не получится просто посчитать в имеющихся данных пары знакомых, где оба больны гриппом, и превратить результат в вероятность. Дело в том, что знакомых у людей много и все парные вероятности не сложатся в связную модель, которая позволит, например, вычислить риск гриппа, зная, какие знакомые больны. Когда примеры не были связаны между собой, этой проблемы не возникало: ее не будет, скажем, в обществе бездетных пар, каждая из которых живет на собственном необитаемом острове. Но такой мир нереален, и эпидемий в нем не появится в любом случае.

Решение заключается в том, чтобы получить набор свойств и узнать их вес, как в сетях Маркова. Для каждого человека X можно ввести свойство «X болен гриппом», для каждой пары знакомых X и Y — свойство «и X, и Y больны гриппом» и так далее. Как

и в марковских сетях, максимально правдоподобный вес будет заставлять свойство встречаться с частотой, наблюдаемой в данных. Вес «*X* болен гриппом» будет высоким, если много людей больны гриппом. Вес «и *X*, и *Y* больны гриппом» окажется выше, если шанс заболеть гриппом у *Y* с больным знакомым *X* выше, чем у случайно выбранного члена сети. Если 40 процентов людей и 16 процентов всех пар знакомых больны гриппом, вес свойства «и *X*, и *Y* больны гриппом» будет нулевым, потому что для правильного воспроизведения статистики данных (0,4 × 0,4 = 0,16) это свойство не нужно. Однако если вес свойства положительный, грипп с большей вероятностью будет возникать в группах, а не произвольно инфицировать людей, и вероятность заболеть гриппом станет выше, если больны знакомые.

Обратите внимание, что сеть имеет отдельное свойство для каждой пары — «и у Элис, и у Боба грипп», «и у Элис, и у Криса грипп» и так далее. Но узнать вес для каждой пары не получится, потому что для пары есть только одна точка данных (инфицированы или нет) и нельзя сделать обобщение для членов сети, которым мы еще не поставили диагноз (есть ли грипп и у Иветт, и у Зака?). Вместо этого мы можем узнать единичный вес для всех свойств такой формы на основе всех частных случаев, которые наблюдали. В результате «и *X*, и *Y* больны гриппом» будет шаблоном свойств, который можно применить к каждой паре знакомых (Элис и Бобу, Элис и Крису и так далее). Веса для всех частных случаев шаблона связаны в том смысле, что у них всех будет одинаковое значение, и таким образом обобщение окажется возможным, несмотря на то что пример у нас всего один (сеть в целом). В нереляционном обучении параметры модели связаны только одним способом: по всем независимым примерам (например, все пациенты, которым мы поставили диагноз). В реляционном обучении каждый шаблон свойств, который мы создаем, связывает параметры всех его частных случаев.

Мы не ограничены парными или индивидуальными свойствами. Facebook хочет выявить ваших потенциальных друзей, чтобы порекомендовать их вам. Для этого используется правило «Друзья друзей, вероятно, тоже друзья», а каждый частный случай этого правила включает троих: если Элис и Боб — друзья, и Боб и Крис —

друзья, то Элис и Крис — потенциальные друзья. В шутке Генри Менкена[107] о том, что мужчина богат, когда он зарабатывает больше мужа сестры своей жены, присутствует упоминание о четырех людях. Каждое из этих правил можно превратить в шаблон свойств реляционной модели, а вес для них можно получить на основе того, как часто свойство встречается в данных. Как и в марковских сетях, сами свойства тоже можно вывести из данных.

Реляционные обучающиеся алгоритмы способны переносить обобщения из одной сети в другую (например, получить модель распространения гриппа в Атланте и применить ее в Бостоне) и учиться на нескольких сетях (например, для Атланты и Бостона при нереалистичном допущении, что в Атланте никто никогда не контактировал с бостонцами). В отличие от «традиционного» обучения, где все примеры должны иметь одинаковое количество атрибутов, в реляционном обучении размер сетей может быть разным: более крупная сеть просто будет содержать больше частных случаев тех же шаблонов, что и меньшая. Конечно, перенос обобщения из меньшей сети в большую может быть точным, а может и не быть, но смысл в том, что ничто не мешает это делать, а крупные сети локально часто ведут себя как небольшие.

Самый изящный трюк, на который способен реляционный обучающийся алгоритм, — превратить периодического учителя в неутомимого. Для обычного классификатора примеры без классов бесполезны: если я узнаю симптомы пациентов, но не их диагнозы, это не поможет мне научиться диагностике. Однако если мне известно, что кто-то из друзей пациента болен гриппом, это косвенный признак, что грипп может быть и у него. Поставить диагноз нескольким людям в сети, а затем распространить его на их знакомых и знакомых их знакомых — тоже неплохо, хотя и хуже, чем индивидуальный диагноз. Полученные таким образом диагнозы могут быть зашумленными, но общая статистика корреляции симптомов с гриппом будет, вероятно, намного точнее и полнее, чем выводы на основе горсти изолированных диагнозов. Дети очень хорошо умеют извлекать максимальную пользу из периодического надзора за ними (при условии, что они его не проигнорируют). Реляционные обучающиеся алгоритмы частично обладают такой способностью.

Однако за мощь приходится платить. В обычных классификаторах, например дереве решений или перцептроне, вывод о классе объекта на основе его атрибутов можно сделать после нескольких просмотров данных и небольших арифметических вычислений. В случае сети класс каждого узла косвенно зависит от всех остальных узлов, и сделать о нем вывод изолированно нельзя. Можно прибегнуть к тем же видам методик логического вывода, что и в случае байесовских сетей, например к циклическому распространению доверия или МСМС, но масштаб будет другим: в типичной байесовской сети могут быть тысячи переменных, а в социальных сетях — миллионы и даже больше узлов. К счастью, модель сети состоит из многократных повторений одних и тех же черт с теми же самыми весами, поэтому часто получается сжать сеть в «сверхузлы», состоящие из многочисленных узлов, которые, как мы знаем, имеют одинаковые вероятности, и теперь нужно решить намного меньшую проблему с тем же результатом.

У реляционного обучения долгая история, уходящая как минимум в символистские методики 1970-х годов, например обратную дедукцию. Но с зарождением интернета оно приобрело новый импульс. Сети внезапно стали повсеместными, а их моделирование — неотложной задачей. Явление, которое мне показалось особенно любопытным, — сарафанное радио. Как распространяется информация в социальной сети? Можно ли измерить влияние каждого ее участника и породить волну слухов, нацелившись на минимально необходимое число наиболее влиятельных? С моим студентом Мэттом Ричардсоном мы разработали алгоритм, который делал именно это, и применили его к сайту Epinions.com с обзорами продукции, где пользователи имели возможность рассказывать, чьим обзорам они доверяют. Помимо всего прочего, мы обнаружили, что рекламировать продукты одному самому влиятельному члену, которому доверяют многие участники сети, которым, в свою очередь, доверяют многие другие пользователи и так далее, — не менее эффективный метод, чем маркетинг, направленный на треть всех пользователей по отдельности. Затем последовала целая лавина исследований этой проблемы. С тех пор я применял реляционное обучение ко многим другим задачам, включая прогнозирование, кто будет образовывать

связи в социальной сети, интегрирование баз данных и способности роботов картировать окружающую обстановку.

Если вы хотите понять, как работает мир, реляционное обучение стоит иметь в арсенале. В цикле романов Айзека Азимова «Основание» ученому Гэри Селдону удается математически предсказать будущее человечества и тем самым спасти его от упадка. Пол Кругман, наряду с другими, признался, что эта соблазнительная мечта сделала его экономистом. Согласно Селдону, люди похожи на молекулы газа, и, даже если сами индивидуумы непредсказуемы, на общества это не распространяется просто по закону больших чисел. Реляционное обучение объясняет, почему это не так. Если бы люди были независимы и каждый принимал решения изолированно, общества действительно были бы предсказуемы, потому что случайные решения складывались бы в довольно постоянное среднее. Но когда люди взаимодействуют, более крупные группы бывают не более, а менее предсказуемы, чем небольшие. Если уверенность и страх заразны, каждое из этих состояний станет некоторое время доминировать, но периодически все общество будет качать от одного к другому. Это, однако, совсем не так уж плохо. Если получится измерить, как сильно люди влияют друг на друга, можно оценить и то, сколько времени пройдет перед таким сдвигом, даже если он произойдет впервые. Это еще один способ, благодаря которому «черные лебеди» не обязательно непредсказуемы.

Многие жалуются, что чем больше объем данных, тем легче увидеть в них мнимые паттерны. Может быть, это и правда, если данные представляют собой просто большой набор не связанных друг с другом объектов, но, если они взаимосвязаны, картина меняется. Например, критики применения добычи данных для борьбы с терроризмом утверждают, что, даже если не брать этические аспекты, такой подход не сработает, потому что невинов-ных слишком много, а террористов слишком мало, и поиск подозрительных паттернов либо даст много ложных срабатываний, либо никого не поймает. Человек, снимающий на видеокамеру ратушу Нью-Йорка, — это турист или злоумышленник, присматривающий место для теракта? А человек, заказавший большую партию нитрата аммония, — мирный фермер или

изготовитель взрывных устройств? Все эти факты по отдельности выглядят достаточно безобидно, но, если «турист» и «фермер» часто разговаривают по телефону и последний только что въехал тяжело груженным пикапом на Манхэттен, наверное, самое время к ним присмотреться. Агентство национальной безопасности США любит искать данные в списках телефонных разговоров не потому, что это, вероятно, законно, а потому, что эти списки зачастую более информативны для предсказывающих алгоритмов, чем содержание самих звонков, которое должен оценивать живой сотрудник.

Кроме социальных сетей, «приманка» реляционного обучения — возможность разобраться в механизмах работы живой клетки. Клетка — сложная метаболическая сеть, где гены кодируют белки, регулирующие другие гены. Это длинные, переплетающиеся цепочки химических реакций, продукты, мигрирующие из одной органеллы в другую. Независимых сущностей, которые делают свою работу изолированно, там не найти. Лекарство от рака должно нарушить функционирование раковой клетки, не мешая работе нормальных. Если у нас в руках окажется точная реляционная модель обоих случаев, можно будет попробовать много разных лекарств in silico, разрешая модели делать выводы об их положительных и отрицательных эффектах, и, выбрав только хорошие, испытать их in vitro и, наконец, in vivo.

Как и человеческая память, реляционное обучение плетет богатую сеть ассоциаций. Оно соединяет воспринимаемые объекты, которые робот вроде Робби может усвоить путем кластеризации и уменьшения размерности, с навыками, которые можно приобрести путем подкрепления и образования фрагментов, а также со знанием более высокого уровня, которое дают чтение, учеба в школе и взаимодействие с людьми. Реляционное обучение — последний кусочек мозаики, заключительный ингредиент, который нужен нам для нашей алхимии. Теперь пришло время отправиться в лабораторию и превратить эти элементы в Верховный алгоритм.

ГЛАВА 9

КУСОЧКИ МОЗАИКИ ВСТАЮТ НА МЕСТО

Машинное обучение — это и наука, и технология, и обе составляющие дают нам подсказки, как их объединить. В науке объединяющие теории часто начинаются с обманчиво простых наблюдений: два не связанных на первый взгляд феномена оказываются сторонами одной медали, и осознание этого факта, как первая костяшка домино, порождает каскад новых открытий. Упавшее на землю яблоко и луна в небе: и то и другое вызвано гравитацией, и — правда это или выдумка, — когда Ньютон разобрался в природе этого явления, гравитация оказалась ответственной за приливы, предсказание равноденствий, траектории комет и многое другое. В повседневной жизни электричество и магнетизм не сопровождают друг друга: одно дело — вспышка молнии, а совсем другое — притягивающая железо руда, причем и то и другое встречается довольно редко. Но когда Максвелл понял, как изменение электрического поля порождает магнетизм и наоборот, стало ясно, что сам свет — это тесный союз обоих явлений, и сегодня мы знаем, что электромагнетизм далеко не редок и пронизывает всю материю. Периодическая система элементов Менделеева не только упорядочила все известные элементы всего в двух измерениях, но и предсказала, где искать новые. Наблюдения на борту «Бигля» внезапно обрели смысл, когда «Опыт закона о народонаселении» Мальтуса подсказал Дарвину естественный отбор в качестве организующего принципа. Как только Крик и Уотсон с помощью структуры двойной спирали объяснили загадочные свойства ДНК, они увидели, что эта молекула может реплицировать саму себя, и начался переход биологии от стадии «собирания марок» (как уничижительно назвал ее Резерфорд) к единой науке. В каждом из этих случаев оказывается, что у обескураживающе разнообразных наблюдений есть общая причина и, когда ученые ее находят, становится возможным использовать ее для предсказания новых явлений. Аналогично, хотя алгоритмы машинного обучения, с которыми мы познакомились в этой книге, могут показаться довольно несхожими — некоторые

основаны на работе мозга, некоторые — на эволюции, а некоторые — на абстрактных математических принципах, — на самом деле у них есть много общего, и получившаяся в результате теория обучения принесет много новых прозрений.

Не все знают, что многие из важнейших технологий — результаты изобретения единого, объединяющего механизма, который делает то, для чего раньше требовалось много разных инструментов. Интернет — сеть, соединяющая между собой сети. Без нее каждый тип сети нуждался бы в собственном протоколе, чтобы контактировать с другими, как мы нуждаемся в отдельном словаре для каждой языковой пары. Протоколы интернета — это эсперанто, дающее каждому компьютеру иллюзию прямого разговора с любым другим компьютером, и это позволяет электронным письмам и интернету игнорировать детали физической инфраструктуры, по которой они передаются. Реляционные базы данных делают нечто схожее с корпоративными приложениями, позволяя разработчикам и пользователям мыслить в категориях абстрактных реляционных моделей и игнорировать способы, которыми компьютеры отвечают на запросы. Микропроцессор — совокупность цифровых электронных элементов, которая может имитировать любое другое собрание. Виртуальные машины позволяют одному компьютеру выдавать себя за сотню компьютеров для сотни разных людей одновременно и делают возможным облачное хранение данных. Графические пользовательские интерфейсы позволяют нам редактировать документы, электронные таблицы, презентации и многое другое с использованием общего языка окон, меню и кликов мыши. Компьютер сам по себе — объединяющее, единое устройство, способное решать любую логическую или математическую проблему при условии, что мы сумеем его соответствующим образом запрограммировать. Даже электричество — своего рода объединитель: его можно получать из многих источников — угля, газа, ядерной реакции, воды, ветра, солнца, — и у него бесконечное множество применений. Электростанция не знает и не хочет знать, как будет потребляться вырабатываемый ею ток, а фонарь в подъезде, посудомоечная машина и новенькая Tesla не помнят, откуда взялось питающее их электричество. Электричество — это

эсперанто в мире энергии. Верховный алгоритм — это объединитель в мире машинного обучения: он позволяет любому приложению использовать любой обучающийся алгоритм, абстрагируя эти алгоритмы в общую форму — единственную, которую нужно знать приложениям.

Наш первый шаг к Верховному алгоритму будет на удивление простым. Как оказывается, несложно соединить много разных обучающихся алгоритмов в один, используя так называемое метаобучение. Его используют Netflix, Watson, Kinect и бесчисленное множество других программ, и это одна из самых мощных стрел в колчане машинного обучения. Это также ступенька к более глубокому объединению алгоритмов, о котором мы поговорим дальше.

Из многих моделей — одна

Вот вам задача: за 15 минут соедините деревья решений, многослойные перцептроны, системы классификации, наивный байесовский алгоритм и метод опорных векторов в один алгоритм, который будет обладать лучшими свойствами каждого из элементов. Быстро: что вы можете сделать? Очевидно, что детали отдельных алгоритмов в нем использовать нельзя: просто не хватит времени. Давайте попробуем следующее решение. Представьте, что каждый из обучающихся алгоритмов — эксперт в комитете. Каждый внимательно рассматривает подлежащий классификации случай — какой диагноз поставить пациенту? — и уверенно дает прогноз. Вы сами — не эксперт, а председатель этого комитета, и ваша работа — объединить рекомендации в окончательное решение. В руках у вас, по сути, новая проблема классификации, где вместо симптомов пациентов входом будет мнение экспертов, но машинное обучение можно применить к этой проблеме таким же образом, как эксперты применяли его к исходным данным. Такой подход называется метаобучением, потому что это обучение обучающимся алгоритмам. Сам метаалгоритм может быть любым, от дерева решений до простого взвешенного голосования. Чтобы узнать вес*а* или дерево решений, атрибуты каждого исходного примера заменяются прогнозами

обучающихся алгоритмов. Алгоритмы, которые часто предсказывают правильный класс, получают высокий вес, а неточные будут чаще игнорироваться. В случае дерева решений использование обучающегося алгоритма может зависеть от предсказаний других алгоритмов. Как бы то ни было, чтобы получить прогноз алгоритма для данного примера, сначала надо применить алгоритм к исходному обучающему набору, *исключив этот пример*, и использовать получившийся в результате классификатор, иначе есть риск, что в комитете будут доминировать алгоритмы, страдающие переобучением, поскольку они могут предсказывать правильный класс, просто его запоминая. Победитель Netflix Prize использовал метаобучение для соединения сотен алгоритмов машинного обучения. Watson использует его для выбора окончательного ответа из имеющихся кандидатов. Нейт Сильвер соединяет результаты опросов аналогичным образом, чтобы спрогнозировать результаты выборов.

Этот тип метаобучения называют стэкингом, а придумал его Дэвид Уолперт, автор теоремы «Бесплатных обедов не бывает», с которой мы познакомились в главе 3. Еще более простой метаалгоритм — это бэггинг, изобретенный статистиком Лео Брейманом. Бэггинг генерирует случайные вариации обучающего набора путем перевыборки, применяет к каждой вариации один и тот же алгоритм машинного обучения и соединяет результаты путем голосования. Это нужно для того, чтобы уменьшить дисперсию: объединенная модель гораздо менее чувствительна к капризам данных, чем любая единичная, поэтому это замечательно легкий способ улучшить точность. Если модели — деревья решений и мы будем еще больше варьировать их, формируя случайный поднабор атрибутов в каждом дереве, получится так называемый случайный лес. Случайный лес — это один из самых точных имеющихся классификаторов. Kinect компании Microsoft использует его для определения ваших действий и регулярно выигрывает соревнования по машинному обучению.

Один из самых сообразительных метаалгоритмов — бустинг, созданный двумя теоретиками обучения, Йоавом Фройндом и Робом Шапире. Бустинг не соединяет разные обучающиеся алгоритмы, а раз за разом применяет к данным один и тот же

классификатор, используя новую модель, чтобы исправить ошибки предыдущей путем присвоения весов обучающим примерам. Вес каждого неправильно классифицированного примера увеличивается после каждого цикла обучения, заставляя последующие циклы больше сосредоточиваться на нем. Название «бустинг» — усиление — связано с тем, что этот процесс может резко улучшить классификатор, который незначительно, но стабильно лучше случайного угадывания, и сделать его почти идеальным.

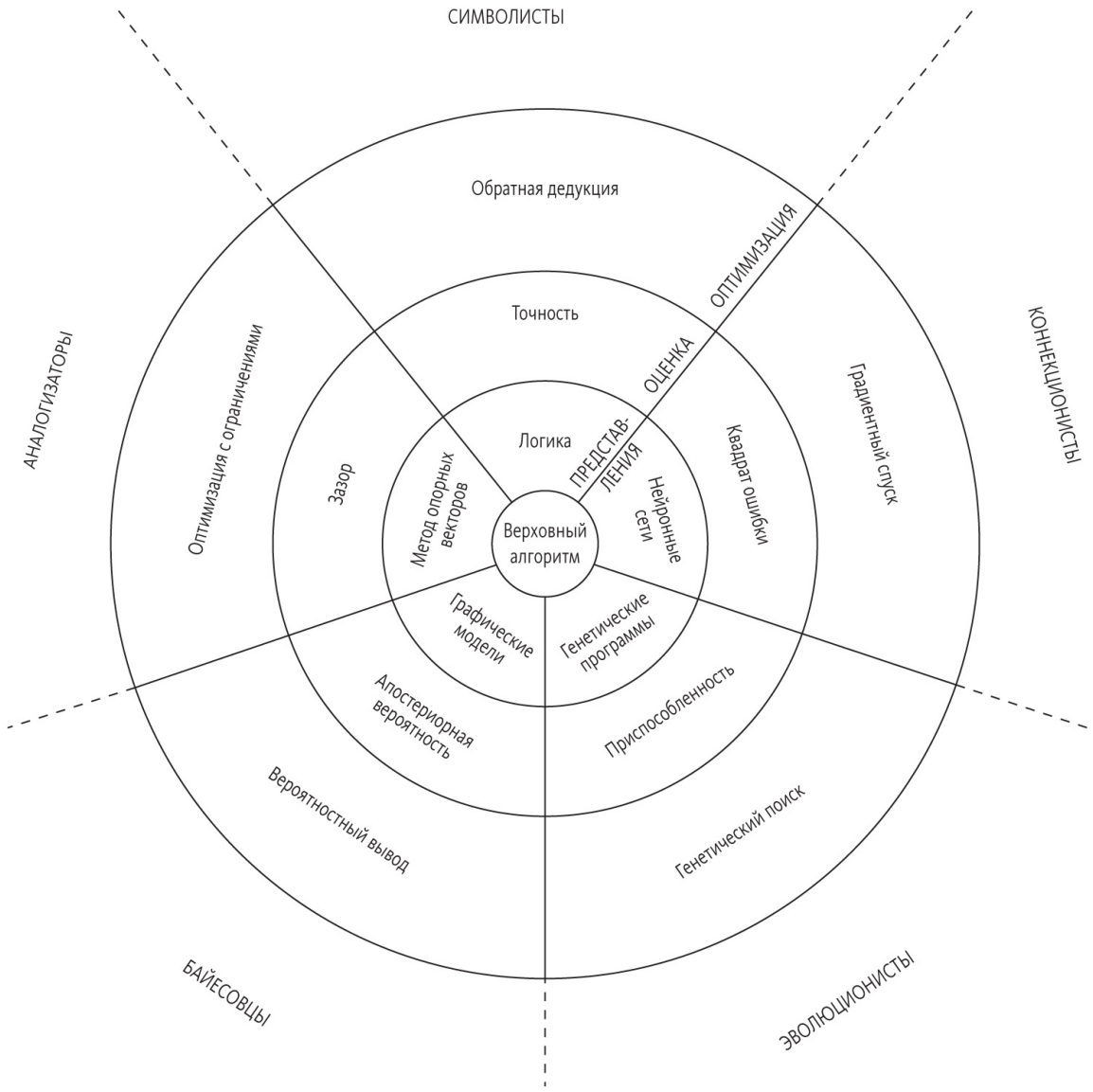
Метаобучение исключительно успешно, но это не очень глубокий способ объединения моделей. Кроме того, он дорог и требует многократного обучения, а соединенные модели могут получиться довольно непрозрачными. («Я уверен, что у вас рак предстательной железы, потому что на это указывают дерево решений, генетический алгоритм и наивный байесовский классификатор, хотя многослойный перцептрон и метод опорных векторов с этим не согласны».) Более того, все объединенные модели — на самом деле просто одна хаотичная модель. Нельзя ли получить единый обучающийся алгоритм, который делает то же самое? Можно.

Верховный алгоритм

Наш объединенный обучающийся алгоритм, наверное, лучше всего ввести с помощью аллегории. Если представить себе машинное обучение в виде континента, разделенного на территории пяти «племен», то Верховный алгоритм будет столицей, расположенной в уникальном месте, где сходятся их границы. Вы приближаетесь к городу издалека и видите, что состоит он из трех концентрических, обнесенных стеной колец. Внешний и без сомнения самый широкий круг — это Городок оптимизации. Дома здесь — алгоритмы всех форм и размеров. Одни только строятся, и вокруг них суетятся местные жители, другие сияют свежестью, третьи выглядят старыми и заброшенными. Выше на холме стоит Цитадель оценки. Из ее особняков и дворцов постоянно исходят приказы алгоритмам внизу. На самой вершине в небо взлетает Башня представлений. Здесь живут отцы города. Их непреложные

законы определяют, что можно сделать, а чего нельзя, и не только в городе, но и на всем континенте. На вершине центральной, самой высокой башни развевается флаг Верховного алгоритма: красно-черный, с пятиконечной звездой, внутри которой надпись, но вы пока не можете ее разобрать.

Город разделен на пять секторов, по одному для каждого из пяти «племен». Сектора простираются от Башни представлений вниз, к внешним стенам, опоясывающим Башню, группы дворцов в Цитадели оценки и улицы и дома Городка оптимизации, над которой они возвышаются. Пять секторов и три кольца делят город на 15 районов — 15 форм, 15 кусочков мозаики, которую вам надо сложить:



Вы пристально вглядываетесь в карту, пытаясь расшифровать ее секрет. Пятнадцать кусочков довольно точно подходят друг к другу, но вам надо понять, как они соединяются, чтобы получить всего три элемента: представление, оценку и оптимизацию Верховного алгоритма. Каждый обучающийся алгоритм состоит из этих элементов, но они разнятся от «племени» к «племени».

Представления — формальный язык, на котором алгоритм машинного обучения выражает свои модели. Формальный язык символистов — логика, частные случаи которой — правила и деревья решений. Для коннекционистов это нейронные сети. Для эволюционистов — генетические программы, включая системы

классификации. Для байесовцев — графические модели, общий термин для байесовских и марковских сетей. Для аналогизаторов — частные случаи, возможно, с весами, как в методе опорных векторов.

Элемент оценки — функция присвоения баллов, которая говорит, насколько хороша модель. Символисты используют точность и информационный выигрыш. Коннекционисты — непрерывное измерение погрешности, например квадрат ошибки, который представляет собой сумму квадратов различий между предсказанными и истинными значениями. Байесовцы применяют апостериорную вероятность, аналогизаторы (как минимум специалисты по методу опорных векторов) — зазор. В дополнение к оценке того, насколько хорошо модель подходит к данным, все «племена» учитывают другие желательные свойства, например простоту модели.

Оптимизация — это алгоритм, который ищет и выдает модель с наивысшей оценкой. Характерный поисковый алгоритм символистов — обратная дедукция. Коннекционистов — градиентный спуск. Эволюционистов — генетический поиск, включая кроссинговер и мутации. Байесовцы в этом отношении необычны: они не просто ищут лучшую модель, но и усредняют по всем моделям, взвешивая их вероятность. Для эффективного взвешивания они пользуются алгоритмами вероятностного вывода, например MCMC. Аналогизаторы (или, точнее, адепты метода опорных векторов) используют условную оптимизацию для нахождения лучшей модели.

У вас за плечами многодневный путь, солнце начинает быстро клониться к горизонту, и надо поторопиться, пока не стемнело. Во внешней стене пять массивных ворот, каждые под охраной своего «племени», и ведут они в соответствующие районы Городка оптимизации. Давайте пройдем через Ворота градиентного спуска, шепнув стражнику пароль — «глубокое обучение», — и начнем по спирали подниматься к Башням представления. Улицы уходят круто вверх по склону холма к Воротам квадратичной ошибки, ведущим в цитадель, но вы сворачиваете влево, к эволюционистскому сектору. Дома в районе градиентного спуска — это гладкие кривые и густо переплетенные паттерны,

напоминающие скорее джунгли, а не город. Но когда градиентный спуск уступает место генетическому поиску, картина резко меняется. Дома здесь выше, структура громоздится на структуру, но сами структуры незаполненные, почти пустые. Они как будто ждут, что в них придут кривые градиентного спуска. Вот он, способ их соединить: надо использовать генетический поиск, чтобы найти структуру модели, а потом позволить градиентному спуску заполнить ее параметры. Именно так поступает природа: эволюция создает структуры головного мозга, а индивидуальный опыт моделирует их.

Первый шаг сделан, и вы спешите в байесовский район. Даже на расстоянии видно, как он лепится к Кафедральному собору теоремы Байеса. Переулок MCMC делает случайные зигзаги. Чтобы не терять времени, вы срезаете путь и выходите на улицу Распространения степени уверенности, но она, похоже, образует бесконечную петлю. И тут вы видите то, что надо: широкий бульвар Наибольшего правдоподобия, ведущий вверх, к воротам Апостериорной вероятности. Вместо того чтобы усреднять все модели, можно направиться прямиком к самой вероятной, с уверенностью, что в итоге прогнозы будут почти такими же. Генетическому поиску можно позволить подобрать структуру модели, а градиентному спуску — ее параметры. Со вздохом облегчения вы понимаете, что это весь вероятностный вывод, который вам нужен, по крайней мере, пока не придет время с помощью модели отвечать на вопросы.

Вы продолжаете свой путь. Район условной оптимизации напоминает лабиринт узких улочек и тупиков. Повсюду плечом к плечу стоят примеры всех сортов, периодически уступая место опорному вектору. Очевидно: чтобы не врезаться в примеры неправильного класса, нужно просто добавить ограничения в уже собранный оптимизатор. Но если подумать, делать это совсем не обязательно. При обучении методу опорных векторов обычно разрешается нарушать зазоры, чтобы избежать переобучения, при условии, что каждое нарушение штрафуется. В этом случае вес*а* оптимальных примеров можно снова получить с помощью одной из форм градиентного спуска. Это было несложно, и вы чувствуете, что начинаете набивать руку.

Плотные ряды частных случаев резко обрываются, и вы попадаете в район Обратной дедукции. Вокруг широкие аллеи и старинные каменные здания. Архитектура здесь геометрическая, строгая, состоящая из прямых линий и прямых углов. Квадратные даже стволы у сильно обрезанных деревьев, а их листья тщательно подписаны предсказаниями класса. Обитатели этого района, видимо, строят свои дома особым образом: они начинают с крыши, которую называют выводами, и постепенно заполняют пустые места между крышей и землей — предпосылками, на их языке. Один за другим они находят каменные блоки, которые хорошо подойдут для заполнения пробелов, и поднимают их на место. Вы замечаете, что форма многих пробелов одинакова, и работа пойдет быстрее, если обрез*а*ть и складывать блоки, подгоняя их, а потом повторять процесс столько раз, сколько нужно. Другими словами, для обратной дедукции можно использовать генетический поиск. Прекрасно. Кажется, вы свели все пять оптимизаторов к простому рецепту: генетический поиск структуры и градиентный спуск для параметров. И даже это может быть лишним. Во многих проблемах генетический поиск можно свести к восхождению на выпуклые поверхности, если выполнить три условия: не включать кроссинговер, пробовать в каждом поколении все возможные точечные мутации и всегда выбирать одну лучшую гипотезу, чтобы «заразить» ею следующее поколение.

А что это за статуя впереди? Это Аристотель, который неодобрительно взирает на переплетение и хаос квартала градиентного спуска. Круг замкнулся. У вас есть объединенный оптимизатор, необходимый для Верховного алгоритма, но сейчас не время для поздравлений. Опустилась ночь, а сделать предстоит еще очень много. Вы идете в Цитадель оценки через впечатляющие, но довольно узкие Ворота точности. Надпись над ними гласит: «Оставь надежды на переобучение, всяк сюда входящий». Огибая дворцы алгоритмов оценки всех пяти «племен», вы мысленно ставите кусочки на место. Точность вы станете использовать для оценки предсказаний типа «да или нет», квадратичную ошибку — для непрерывных предсказаний. Приспособленность — просто эволюционистское название функции подсчета очков: ее можно сделать какой вздумается, в том числе точностью и квадратом

ошибки. Апостериорная вероятность уменьшает квадрат ошибки, если проигнорировать априорную вероятность и если ошибки следуют нормальному распределению. Зазор, если разрешить нарушать его за определенную цену, становится более мягкой версией точности: вместо того чтобы платить нулевой штраф за правильное предсказание и единицу за каждое неправильное предсказание, штраф будет нулевым, пока вы не попадаете в зазор, а потом он начинает расти. Готово! Сложить алгоритмы оценки оказалось намного проще, чем объединить оптимизаторы. Но над вами нависли Башни представления, и они наполняют вас дурными предчувствиями.

Вы дошли до последнего этапа поисков и стучитесь в дверь Башни опорных векторов. Вам открывает грозного вида стражник, и тут вы понимаете, что не знаете пароль. «Ядро!» — выпаливаете вы, пытаясь не выдать испуг. Страж кланяется и уступает дорогу. Собравшись с духом, вы входите внутрь, тихо браня себя за беспечность. Весь первый этаж башни занимают щедро обставленные округлые покои, а в центре на почетном месте стоит нечто напоминающее мраморное представление метода опорных векторов. Осматривая комнату, вы замечаете где-то сбоку дверь. Она должна вести в центральную башню — Башню Верховного алгоритма. Похоже, дверь не охраняют, и вы решаете срезать путь. Проскользнув через дверной проем и пройдя по короткому коридору, вы оказываетесь в еще большем пятиугольном помещении с дверью в каждой стене. В центре куда-то ввысь уходит винтовая лестница. Оттуда доносятся голоса, и вы ныряете в дверь напротив. Она ведет в Башню нейронных сетей. Вы снова оказываетесь в округлом помещении, в центре которого на этот раз стоит многослойный перцептрон. Его элементы отличаются от метода опорных векторов, но их расположение замечательно схоже. Вдруг вас осеняет: метод опорных векторов — это же просто многослойный перцептрон, но скрытый слой состоит из ядер, а не из сигмоид, а выходной слой — не еще одна S-кривая, а линейная комбинация.

Может быть, другие представления тоже имеют схожую форму? С растущим возбуждением вы бежите обратно в пятиугольную комнату, а оттуда — в Башню логики. Глядя на стоящее в центре

изображение набора правил, вы пытаетесь увидеть схему. Есть! Каждое правило — это просто очень сильно стилизованный нейрон. Например, правило «Если это гигантская рептилия и она дышит огнем — это дракон» — это просто перцептрон с весами один для «это гигантская рептилия» и «дышать огнем» и порогом 1,5. А набор правил — многослойный перцептрон со скрытым слоем, содержащий один нейрон для каждого правила и выходящий нейрон для дизъюнкции этих правил. Где-то в глубине души вас гложут сомнения, но сейчас думать о них некогда. Вы бежите через пятиугольную комнату в Башню генетических программ и уже видите, как поставить их в строй. Генетические программы — это просто программы, а программа — это просто логический конструкт. Скульптура генетической программы в комнате имеет форму дерева, подпрограммы ветвятся на еще большее количество подпрограмм, и, присматриваясь к листьям, вы замечаете, что это всего лишь правила. Итак, программы сводятся к правилам, а если правила можно свести к нейронам, значит, можно и программы.

Вперед, в Башню графических моделей! К сожалению, скульптура в круглой комнате оказывается совершенно не похожей на остальные. Графическая модель — это продукт факторов: условных вероятностей, в случае байесовских сетей, и неотрицательных функций состояния — в случае сетей Маркова. Как вы ни стараетесь, уловить связь с нейронными сетями и наборами правил не получается. Вас на секунду охватывает разочарование, но вы надеваете свои лого-очки, которые превращают функции в логарифмы. Эврика! Произведение факторов стало суммой условий, прямо как метод опорных векторов, голосующий набор правил и многослойный перцептрон без S-образной кривой на выходе. Например, можно превратить наивный байесовский классификатор дракона в перцептрон, вес которого для «дышит огнем» будет разностью логарифмов *P(дышит огнем | дракон)* и *P(дышит огнем | не дракон)*. Но, конечно, графические модели намного более обобщенные, потому что могут представлять вероятностные распределения по многим переменным, а не только распределение одной переменной (класс) при известных других (атрибутах).

Получилось! Или нет? Внедрить метод опорных векторов в нейронные сети и нейронные сети в графические модели можно. То же касается объединения генетических программ и логики. Но как соединить логику и графические модели? Что-то здесь не так. С запозданием вы видите, в чем проблема: у логики есть измерение, которого не хватает графическим моделям, и наоборот. Скульптуры в пяти комнатах подходили друг к другу, потому что это были простые аллегории, но в реальности все сложнее. Графические модели не позволяют представить правила, включающие больше одного объекта, например «Друзья друзей — тоже друзья». Все их переменные должны быть свойствами того же предмета. Еще они не могут представлять произвольные программы, которые передают наборы переменных из одной подпрограммы в другую. Логика умеет делать и то и другое, но она, в свою очередь, не может представлять неопределенность, двузначность и степени схожести. Без представления, которое может делать все это, универсального обучающего алгоритма не получишь.

Вы напрягаете извилины в поисках решения, но чем больше стараетесь, тем хуже выходит. Может быть, объединение логики и вероятностей неподвластно человеку? Усталость подкашивает вас и погружает в сон. Просыпаетесь вы от грозного рыка: на вас набросился похожий на гидру Монстр Сложности. Он щелкает зубами, но в последний момент вы уворачиваетесь и отчаянно рубите чудовище мечом обучения — только им можно его победить — и отрубаете все его головы. Пока не отросли новые, вы бросаетесь к лестнице.

Запыхавшись, вы взбираетесь на самый верх и видите свадебный обряд. Предиктус, Первый лорд Логики, повелитель Символического королевства и Защитник программ, говорит Марковии, Княжне вероятностей и Царице сетей: «Объединим же наши владения! В мои правила добавишь ты веса, и породим мы новые представления, которые умножатся и заселят землю». Княжна добавляет: «А потомство наше мы назовем Марковскими логическими сетями».

У вас кружится голова. Вы выходите на балкон. Над городскими крышами уже взошло солнце. Вокруг во всех направлениях

простираются леса серверов: они тихо шумят в ожидании Верховного алгоритма. Караваны везут золото из копей, где добывают данные. Далеко на западе на солнце играет море информации, на котором виднеются точки кораблей. Вы поднимаете голову и смотрите на флаг Верховного алгоритма. Теперь надпись внутри пятиконечной звезды четко видна:

$$P = e^{w \cdot n} / Z.$$

Что бы это могло значить?

Логические сети Маркова

В 2003 году мы с Мэттом Ричардсоном начали размышлять над проблемой объединения логики и вероятностей. Сначала у нас получалось не очень, потому что мы пытались добиться результата с помощью байесовских сетей, а их жесткая форма — строгая последовательность переменных, условные распределения детей у родителей — не сочеталась с гибкостью логики. Но в канун Рождества я осознал, что есть способ лучше. Если переключиться на марковские сети, можно использовать *любую* логическую формулу в качестве шаблона для свойств такой сети, а это объединило бы логику и графические модели. Давайте посмотрим, как это сделать.

Вспомните, что сеть Маркова, во многом как перцептрон, определяется взвешенной суммой свойств. Представьте, что у вас есть коллекция фотографий людей. Мы случайным образом выбираем одну и вычисляем ее свойства, например «у этого человека седые волосы», «этот человек пожилой», «этот человек женщина» и так далее. В перцептроне мы проводим взвешенную сумму этих свойств через порог, чтобы решить, например, это ваша бабушка или нет. В марковских сетях мы делаем нечто совершенно другое (по крайней мере на первый взгляд): взвешенная сумма возводится в степень, превращается в произведение факторов, и это произведение будет вероятностью выбора конкретно этой картинки из коллекции, независимо от того, есть ли на ней ваша бабушка. Если у вас много картинок с изображениями пожилых людей, взвешенная сумма этого свойства растет. Если большинство

из них — мужчины, вес свойства «этот человек — женщина» идет вниз. Свойства могут быть какими угодно, поэтому сети Маркова — замечательно гибкий способ представления вероятностных распределений.

На самом деле я погрешил против истины: произведение факторов — это еще не вероятность, потому что сумма вероятностей всех картинок должна быть равна единице, и нет гарантии, что произведение факторов для всех картинок приведет к такому результату. Нам нужно их нормализовать, то есть разделить каждое произведение на сумму факторов. В таком случае сумма всех нормализованных произведений будет гарантированно равна единице, потому что это просто некое число, разделенное само на себя. Вероятность картинки, таким образом, будет взвешенной суммой ее свойств, возведенной в степень и нормализованной. Если вы вспомните уравнение в пятиконечной звезде, то, наверное, начнете догадываться, что оно означает. P — это вероятность, w — вектор весов (будем обозначать вектора жирным шрифтом), n — вектор чисел, а их скалярное произведение $\cdot$ возводится в степень и делится на Z, сумму всех произведений. Если первый компонент n равен единице, когда первое свойство изображения верно, и нулю в противном случае и так далее, то $w \cdot n$ — это просто сокращение для взвешенной суммы черт, о которой мы постоянно говорили.

Поэтому уравнение дает нам вероятность изображения (или чего угодно) согласно марковской сети, но оно более общее, потому что это не просто уравнение марковской сети, а уравнение логической сети Маркова — так мы ее назвали. В логической сети Маркова числа в n не обязательно должны быть равны нулю или единице, и они обращаются не к свойствам, а к логическим формулам. В конце главы 8 мы уже увидели, как выйти за пределы сетей Маркова в реляционные модели, определенные в терминах шаблонов свойств, а не просто свойств. «И у Элис, и у Боба грипп» — это свойство, специфичное для Элис и Боба. «И у X, и у Y грипп» — это шаблон свойств, частными случаями которого могут быть Элис и Боб, Элис и Крис и любая другая пара. Шаблон свойств — мощный инструмент, потому что он может суммировать миллиарды свойств

в одном коротком выражении. Однако для определения шаблонов свойств нам нужен формальный язык, и он у нас есть: это логика.

Логическая сеть Маркова — просто набор логических формул и их весов. В приложении к конкретному набору объектов он определяет марковскую сеть их возможных состояний. Например, если объекты — Элис и Боб, возможным состоянием будет то, что Элис и Боб — друзья, у Элис грипп и у Боба тоже. Давайте предположим, что у логической сети Маркова две формулы: «у всех грипп» и «если у человека грипп, у его друзей тоже грипп». В стандартной логике это была бы довольно бесполезная пара утверждений: первое исключало бы все состояния, где хотя бы один человек здоров, а второе было бы избыточным. Однако в логической сети Маркова первая формула означает только то, что для каждого человека *X* есть свойство «*X* болен гриппом» с тем же весом, что и формула. Если люди с большой вероятностью болеют гриппом, и у формулы, и у соответствующих свойств будет большой вес. Состояние, где здоровых людей много, менее вероятно, чем то, где таких людей мало, однако оно не исключено. А благодаря второй формуле состояние, при котором у кого-то грипп, а у его друзей — нет, будет менее вероятно, чем то, где здоровые и зараженные попадают в отдельные кластеры друзей.

Теперь вы, вероятно, догадались, что означает *n* в верховном уравнении: его первый компонент — число истинных случаев первой формулы в данном состоянии, второй — число истинных случаев второй формулы и так далее. Если рассмотреть десять знакомых, семь из которых больны гриппом, первый компонент из *n* будет равен семи, и так далее. (Не должна ли вероятность измениться, если больны гриппом семь из двадцати, а не десяти знакомых? Да, и она будет отличаться благодаря *Z*.) Если все веса в этих пределах будут стремиться к бесконечности, марковская логика сведется к стандартной, потому что нарушение хотя бы одного случая формулы вызовет схлопывание вероятности до нуля, делая состояние невозможным. С точки зрения вероятностей логическая сеть Маркова сводится к марковской сети, если все формулы описывают один объект. Итак, и логика, и марковские сети — частные случаи марковской логики, и это то объединение, которое мы искали.

Обучение в логической сети Маркова — открытие формул, которые верны в мире чаще, чем предсказывали бы случайные шансы, а также определение весов для этих формул, благодаря которым их предсказанные вероятности совпадают с наблюдаемыми частотами. Готовую логическую сеть Маркова можно использовать для ответа на такие вопросы, как «Какова вероятность, что у Боба грипп, при условии, что он друг Элис и у нее грипп?» И знаете что? Оказалось, что эта вероятность задана S-образной кривой, приложенной ко взвешенной сумме свойств, во многом как в многослойном перцептроне, а логическая сеть Маркова с длинными цепочками правил может представлять глубокую нейронную сеть с одним слоем на каждое звено цепи.

Конечно, не стоит верить прогнозам распространения гриппа, сделанным описанной выше простой логической сетью Маркова. Вместо этого представьте себе логическую сеть Маркова для диагностики и лечения рака. Такая сеть будет представлять распределение вероятностей состояний клетки. Каждый элемент клетки, каждая органелла, каждый метаболический путь, каждый ген и белок — это объект сети, а формулы логической сети Маркова кодируют зависимости между ними. Можно спросить сеть: «Эта клетка — раковая?» — проверить ее разными лекарствами и посмотреть, что произойдет. Пока у нас нет таких сетей, но ниже в этой главе я опишу, как их получить.

Подытожим: единый алгоритм машинного обучения, к которому мы пришли, в качестве представления использует логическую сеть Маркова, как функцию оценки — апостериорную вероятность, а оптимизатор в нем — генетический поиск в сочетании с градиентным спуском. При желании можно легко заменить апостериорную вероятность каким-то более точным измерением, а генетический поиск — восхождением на выпуклые поверхности. Мы достигли вершины и можем наслаждаться видами. Однако я бы не торопился называть такой алгоритм Верховным. Во-первых, критерий истины — практика, и, хотя за последнее десятилетие этот алгоритм (или его разновидности) успешно применяли во многих сферах, есть намного больше областей, в которых он не применялся, поэтому пока не ясно, насколько он универсален. Во-вторых, ряд

важных проблем он не решает. Но перед тем, как ими заняться, давайте посмотрим, на что он способен.

От Юма до домашних роботов

Скачать обучающийся алгоритм, который я только что описал, можно на сайте alchemy.cs.washington.edu. Мы окрестили его Alchemy, чтобы напомнить самим себе, что, несмотря на все свои успехи, наука о машинном обучении все еще находится на стадии алхимии. Скачав его, вы увидите, что он включает намного больше элементов, чем приведенный выше базовый алгоритм, и что ему все еще не хватает целого ряда аспектов, которые, как мы выяснили, универсальный обучающийся алгоритм должен иметь, например кроссинговера. Тем не менее для простоты давайте называть нашего кандидата в универсальные обучающиеся алгоритмы Alchemy.

Alchemy отвечает на вопрос Юма тем, что кроме данных у него еще один вход: исходные знания в виде набора логических формул, с весами или без них. Эти формулы могут быть непоследовательными, неполными и просто ошибочными: алгоритм машинного обучения и вероятностное рассуждение с этим справятся. Ключевой момент заключается в том, что Alchemy не обязан учиться с нуля. Вообще говоря, мы даже можем приказать алгоритму оставить формулы без изменений и узнавать только веса. В таком случае Alchemy, вооруженный соответствующими формулами, может превратиться в машину Больцмана, байесовскую сеть, обучающийся алгоритм, основанный на случаях, и многие другие модели. Это объясняет, почему, несмотря на теорему «бесплатных обедов не бывает», универсальный обучающийся алгоритм возможен. Alchemy похож скорее на индуктивную машину Тьюринга, которую мы программируем вести себя или как очень мощный, или как очень ограниченный алгоритм — все зависит от нас. Alchemy объединяет машинное обучение так же, как интернет объединяет компьютерные сети, реляционные модели — базы данных, а графический пользовательский интерфейс — бытовые приложения.

Конечно, даже если пользоваться Alchemy без исходных формул (так тоже можно), это не лишит его знаний: допущения о мире неявно закодированы в выборе формального языка, оценивающей функции и оптимизатора. Поэтому естественно будет спросить: можно ли получить еще более обобщенный алгоритм машинного обучения, чем Alchemy? Какое допущение делала эволюция, начиная долгий путь от первой бактерии ко всем формам жизни, которые мы сегодня видим вокруг? Я думаю, простое допущение, из которого вытекает все остальное, таково: обучающийся алгоритм — часть мира. Это значит, что, какой бы он ни был, как физическая система он соблюдает те же законы, что и среда, в которой он находится, и тем самым уже неявно их «знает» и настроен на их открытие. В следующем разделе мы увидим, что это может означать и как воплотить этот принцип в Alchemy. А пока просто заметим, что это, наверное, лучший ответ, который вообще можно дать на вопрос Юма. С одной стороны, предположение, что алгоритм машинного обучения — часть мира, — это *допущение*: в принципе обучающийся алгоритм может соблюдать и другие законы, не те, которые соблюдает окружающий мир, поэтому это согласуется с мнением Юма, что обучение возможно только на основе предшествующего знания. С другой стороны, это допущение настолько базовое и с ним так сложно спорить, что, наверное, для этого мира его будет достаточно.

Если посмотреть на другой полюс, у инженеров знаний — самых решительных критиков машинного обучения — тоже есть повод полюбить Alchemy. На входе этот алгоритм может иметь не базовую структуру модели и не ряд грубых догадок, а большую, с любовью собранную базу знаний, если таковая у нас есть. Поскольку взаимодействия вероятностных правил гораздо богаче, чем детерминистских, вручную закодированное знание в марковской логике способно на многое, к тому же базы знаний в такой логике не обязаны быть непротиворечивыми и могут быть очень большими и вмещать много разных факторов, не распадаясь на части, — пока решение этой задачи ускользает от инженеров знаний.

Однако прежде всего Alchemy позволяет решить проблемы, над которыми так долго трудились «племена» машинного обучения. Давайте посмотрим на каждую из них по очереди.

Символисты на лету соединяют разные кусочки знания, точно так же как математики соединяют аксиомы, чтобы доказывать теоремы. Это резко контрастирует с нейронными сетями и другими моделями с фиксированной структурой. Alchemy делает это при помощи логики, как символисты, но с изюминкой. Чтобы доказать теорему в логике, надо найти только одну последовательность применения аксиом, которые ее создают. Поскольку Alchemy рассуждает в вероятностных категориях, она способна на большее: найти множественные последовательности формул, ведущие к данной теореме или ее отрицанию, и взвесить их, чтобы вычислить вероятность того, что теорема верна. Таким образом этот алгоритм может рассуждать не просто о математических универсалиях, но и о том, означает ли в новостной статье «президент», что это «Барак Обама», и в какую папку надо отправить электронное письмо. Верховный алгоритм символистов — обратная дедукция — постулирует, что новые логические правила должны служить ступенями между данными и желаемым выводом. Alchemy начинает с исходных правил и путем восхождения на выпуклые поверхности вводит новые правила, которые в сочетании с исходными правилами и данными делают выводы более вероятными.

Коннекционистские модели вдохновлены головным мозгом. Нейронам в них соответствуют сети S-образных кривых, а синапсам — взвешенные соединения между ними. В Alchemy две переменные соединены между собой, если они появляются вместе в какой-то формуле, и вероятность переменной при данных соседях — это сигмоида. (Я не буду объяснять, почему это так, но это прямое следствие верховного уравнения, которое мы видели в предыдущем разделе.) Свой верховный алгоритм — обратное распространение ошибки — коннекционисты используют для того, чтобы определить, какие нейроны отвечают за какие ошибки, и в соответствии с этим подобрать веса. Обратное распространение — это разновидность градиентного спуска, который используется в Alchemy для оптимизации весов логической сети Маркова.

Эволюционисты используют генетические алгоритмы для симуляции естественного отбора. Генетический алгоритм

поддерживает популяцию гипотез и в каждом поколении проводит кроссинговер и мутации наиболее приспособленных из них, чтобы породить следующее поколение. Alchemy поддерживает популяцию гипотез в виде взвешенных формул, на каждом этапе по-разному их модифицирует и поддерживает вариации, которые больше всего увеличивают апостериорную вероятность данных (или какую-то другую оценивающую функцию). Если популяция — это одна гипотеза, все сводится к восхождению на выпуклые поверхности. В данный момент доступный для скачивания алгоритм Alchemy не включает кроссинговер, но его несложно добавить. Верховный алгоритм эволюционистов — это генетическое программирование, которое применяет кроссинговер и мутации к компьютерным программам, представленным в виде деревьев подпрограмм. Сами деревья подпрограмм могут быть представлены наборами логических правил, и язык программирования Prolog делает именно это: каждое правило в нем соответствует подпрограмме, а его предшественники — подпрограммам, которые она вызывает. Поэтому можно представить себе Alchemy с кроссинговером как генетическое программирование с использованием похожего на Prolog языка. Дополнительным преимуществом станет то, что правила могут быть вероятностными.

Байесовцы убеждены, что ключ к обучению — моделирование неопределенности, и применяют для этого формальные представления, например байесовские и марковские сети. Как мы уже видели, марковские сети — это особый тип логической сети Маркова, а байесовские сети можно легко представить с помощью основного уравнения логической сети Маркова со свойством для каждого возможного состояния переменной и ее родителей и логарифмом соответствующей условной вероятности в качестве весов. (Нормировочная константа Z удобно сводится к единице, то есть ее можно проигнорировать.) Верховный алгоритм байесовцев — это теорема Байеса, внедренная с помощью алгоритмов вероятностного вывода, например распространения степени уверенности и MCMC. Как вы, может быть, заметили, теорема Байеса — это частный случай верховного уравнения, где $P = P(A|B)$, $Z = P(B)$, а свойства и веса соответствуют $P(A)$ и $P(B|A)$. Система Alchemy включает для логического вывода

и распространение степени уверенности, и MCMC, обобщенные, чтобы оперировать взвешенными логическими формулами. Применяя вероятностный вывод к предоставленным логикой линиям доказательств, Alchemy взвешивает аргументы «за» и «против» и выдает вероятность вывода. Это контрастирует с «типовой» логикой «все или ничего», которую применяют символисты и которая не работает, если дать ей противоречивые доказательства.

Аналогизаторы учатся, выдвигая гипотезу, что у объектов со схожими известными качествами неизвестные качества тоже схожи: пациенты с аналогичными симптомами имеют аналогичные диагнозы, читатели, которые купили в прошлом те же самые книги, будут их покупать и в будущем и так далее. Логические сети Маркова могут представлять схожесть между объектами в виде формул, например «Люди с одинаковыми вкусами покупают одинаковые книги». В таком случае чем больше одинаковых книг купили Элис и Боб, тем больше вероятность, что у них одинаковые вкусы, и (применяя ту же самую формулу в противоположном направлении) тем больше вероятность, что Элис купит книгу, если ее купил Боб. Сходство между ними представлено вероятностью совпадения вкусов. Чтобы извлечь из этого настоящую пользу, можно ввести разный вес для частных случаев одного правила: если Элис с Бобом купили одну и ту же редкую книгу, это, вероятно, дает больше информации, чем если бы они купили бестселлер, поэтому вес события будет выше. В данном случае свойства, сходства которых мы вычисляем, дискретны (купил / не купил), но можно представить сходство и между непрерывными характеристиками, например расстоянием между двумя городами, если ввести в логическую сеть Маркова такое сходство в виде свойства. Если функция оценки — не апостериорная вероятность, а похожая на зазор функция присвоения очков, получится обобщение метода опорных векторов, верховного алгоритма аналогизаторов. Более серьезным вызовом для нашего варианта верховного обучающего алгоритма будет воспроизведение отображения структур — мощной разновидности аналогии, способной переносить выводы из одной области (например, Солнечной системы) в другую. Этого можно достичь путем выведения формул, которые не обращаются

к конкретным отношениям в исходной области. Например, утверждение «Друзья курильщиков тоже курят» относится к дружбе и курению, а «Связанные объекты имеют схожие свойства» — к любому отношению и свойству. Такие формулы можно получить путем обобщения частных случаев: «Друзья друзей тоже курят», «Коллеги экспертов тоже эксперты» и других таких паттернов в социальной сети, а затем применить полученные формулы, скажем, к сети с частными случаями вроде «Интересные страницы имеют ссылки на интересные страницы» или к молекулярной биологи, где случаями будут «Белки, которые взаимодействуют с регулирующими гены белками, тоже регулируют гены». Ученые в моем и не только моем коллективе сделали все это и многое другое.

Благодаря Alchemy возможны пять типов обучения без учителя, которые мы видели в предыдущей главе. Очевидно, что он способен на реляционное обучение, и пока в большинстве случаев его применяли именно так. Alchemy использует логику для представления отношений между объектами, а сети Маркова — чтобы они могли быть неопределенными. Его можно превратить в обучающийся алгоритм с подкреплением, обернув вокруг него отложенные награды и применяя его для получения значений каждого состояния таким же образом, как в традиционных обучающихся алгоритмах с подкреплением, например нейронных сетях. Мы можем выполнять с помощью Alchemy образование фрагментов, если введем новую операцию, которая будет сжимать цепочки правил в отдельные правила. (Например, «Если *A*, то *B*» и «Если *B*, то *C*» в «Если *A*, то *C*».) Логическая сеть Маркова с одной ненаблюдаемой переменной, соединенной со всеми наблюдаемыми, выполняет кластеризацию. (Ненаблюдаемая переменная — это переменная, значение которой мы никогда не видим в данных. Можно сказать, что она «скрыта» и ее можно только вывести.)

Если в логической сети Маркова более одной ненаблюдаемой переменной, она выполняет своего рода дискретное понижение размерности, делая выводы о значении этих (менее многочисленных) переменных на основе (более многочисленных) наблюдаемых. Alchemy может справиться и с логической сетью Маркова с непрерывными ненаблюдаемыми переменными,

которые нужны, например, для анализа главных компонентов и Isomap. Таким образом, Alchemy в принципе может делать все, что мы хотим от Робби, или по меньшей мере все, что мы обсуждали в этой книге. В действительности мы использовали Alchemy, чтобы научить робота картировать среду, определяя по данным из сенсоров, где находятся стены и двери, их углы и расстояния и так далее, а это первый шаг к созданию квалифицированного домашнего робота.

Наконец, Alchemy можно превратить в метаалгоритм наподобие стэкинга, если закодировать индивидуальные классификаторы, как логическая сеть Маркова, и добавить или вывести обучающие формулы, чтобы их соединить. Именно это сделали в DARPA. Проект PAL (Personalized Assistant that Learns) был для них крупнейшим в области искусственного интеллекта и стал предшественником Siri. Целью PAL было создание автоматического секретаря. Марковская логика использовалась в нем как всеобъемлющее представление, соединяя выходы из разных модулей в решения, что делать. Кроме того, это позволяло модулям PAL учиться друг у друга путем эволюции в сторону консенсуса.

На сегодняшний день одним из самых успешных применений Alchemy было создание семантической сети (или графа знаний, как это называют в Google) на основе интернета. Семантическая сеть — набор понятий (например, «планеты» и «звезды») и отношений между этими понятиями (планеты вращаются вокруг звезд). Alchemy вывел из полученных из сети фактов более миллиона таких паттернов (например, то, что Земля вращается вокруг Солнца) и совершенно самостоятельно открыл такие понятия, как «планета». Мы использовали более совершенную версию, чем базовый алгоритм, которую я описываю в этой книге, но важнейшие идеи те же. Различные исследовательские группы применяли Alchemy в своей работе для решения проблем обработки естественных языков, компьютерного зрения, распознавания активности, анализа социальных сетей, а также в молекулярной биологии и многих других областях.

Несмотря на все успехи, у Alchemy есть ряд существенных недостатков. Пока не получается увеличить масштаб алгоритма, чтобы обрабатывать по-настоящему большие данные, и человеку

без ученой степени в области машинного обучения пользоваться им будет сложно. Из-за этих проблем его звездный час пока не настал. Поэтому давайте посмотрим, как их устранить.

Машинное обучение в планетарном масштабе

В информатике проблема не решена по-настоящему до тех пор, пока она не решена эффективно. От знания, как что-то сделать, пользы мало, если это невозможно сделать в доступное время и с доступной памятью, а когда вы имеете дело с логическою сетью Маркова, эти ресурсы очень быстро заканчиваются. Мы рутинно учим логическую сеть миллионам переменных и миллиардам свойств, но это не так много, как может показаться, потому что число переменных очень быстро растет вместе с числом объектов в логической сети Маркова: если у вас есть социальная сеть с тысячей членов, это дает миллион возможных пар друзей и миллиард частных случаев формулы «друзья друзей — тоже друзья».

Логический вывод в Alchemy — сочетание логического и вероятностного выводов. Первый реализован путем доказательства теорем, а второй — путем распространения степени уверенности, MCMC и другими методами, которые мы рассматривали в главе 6. Мы соединили и то и другое в вероятностное доказательство теорем, и ключевым элементом системы Alchemy в настоящее время стал единый алгоритм вывода, способный вычислить вероятность любой логической формулы. Однако он может быть очень затратным с точки зрения вычислений. Если бы ваш мозг пользовался вероятностным доказательством теорем, тигр съел бы вас, прежде чем вы сообразили бы, что надо бежать. Это высокая цена за обобщенность марковской логики. Поскольку мозг человека эволюционировал в реальном мире, в нем должны быть закодированы дополнительные допущения, благодаря которым он делает выводы очень эффективно. В последние несколько лет мы начали понимать, какими могут быть эти допущения, и встроили их в Alchemy.

Мир — это не случайное скопление взаимодействий. В нем есть иерархическая структура: галактики, планеты, континенты,

государства, города, микрорайоны, ваш дом, вы сами, ваши голова, нос, клетка на кончике носа, органеллы в ней, молекулы, атомы, субатомные частицы. В таком случае для моделирования мира нужна марковская логическая сеть, у которой также будет иерархическая структура. Это один из примеров допущения, что обучающийся алгоритм и среда схожи. Логическая сеть Маркова не должна знать априори, из каких элементов состоит мир. Все, что надо Alchemy, — допустить, что в мире *есть* элементы, и поискать их, как только что сделанная книжная полка «подразумевает» существование книг, но пока не знает, какие именно будут на ней стоять. Благодаря иерархической структуре выводы становятся возможными, потому что подэлементы мира взаимодействуют главным образом с другими подэлементами одного и того же элемента: соседи чаще разговаривают друг с другом, а не с людьми из других стран; молекулы, созданные в одной клетке, вступают в реакцию в основном с другими молекулами в той же клетке и так далее.

Другое свойство мира, облегчающее обучение и выведение заключений, — то, что объекты в нем не принимают произвольные формы, а делятся на классы и подклассы, и члены одного класса будут более схожи, чем члены разных классов. Живое или неживое, животное или растение, птица или млекопитающее, человек или нет: если знать все отличительные черты, имеющие отношение к рассматриваемому вопросу, можно свалить в одну кучу все объекты, у которых их нет, и сэкономить тем самым уйму времени. Как и ранее, логическая сеть Маркова не обязана знать априори, какие классы существуют в мире. Она может извлечь их из данных путем иерархической кластеризации.

Мир состоит из элементов, а элементы относятся к классам: соединение этих факторов в основном дает нам все, что нужно, чтобы позволить алгоритму Alchemy делать выводы. Мы можем получать имеющиеся в мире логические сети Маркова, разбивая его на элементы и подэлементы так, чтобы взаимодействия в основном происходили между субэлементами одного элемента, а затем будем группировать элементы в классы и подклассы. Если мир — это конструктор лего, его можно разложить на детали, запомнить, как они крепятся, а потом сгруппировать кирпичики по цвету и форме.

Если мир — это «Википедия», можно извлечь объекты, о которых она повествует, объединить их в классы и узнать, как эти классы соотносятся друг с другом. Если потом кто-то спросит нас: «Арнольд Шварценеггер — звезда боевиков?» — мы ответим: «Да», потому что он кинозвезда и играет в боевиках. Шаг за шагом мы можем получать все бо́льшие логические сети Маркова, пока не дойдем до того, что один мой друг — сотрудник Google — называет «машинным обучением в планетарном масштабе»: моделированием всего в мире сразу, когда данные текут, а ответы вытекают непрерывным потоком.

Конечно, для обучения в таком масштабе нужно намного больше, чем простое внедрение алгоритмов, которые мы уже видели. Во-первых, в какой-то момент одного процессора станет мало: обучение придется распределить по многим серверам. Ученые, работающие в промышленности и в научных учреждениях, интенсивно исследуют, как, например, выполнять градиентный спуск, используя параллельно много компьютеров. Один из вариантов — разделить данные между процессорами, другой — разделить параметры модели. После каждого этапа результаты соединяются и работа перераспределяется. Так или иначе сделать это, не жертвуя качеством и не давая затратам на коммуникацию между процессорами вас задавить, — далеко не тривиальная задача. Другая проблема заключается в том, что, имея бесконечный поток поступающих данных, нельзя определиться с решением, пока не увидишь их целиком. Выйти из такой ситуации помогает, например, принцип выборочного обследования. Если вы хотите предсказать, кто победит на следующих президентских выборах, не обязательно спрашивать каждого избирателя, за кого он собирается голосовать: пробы из нескольких тысяч человек будет достаточно, если вы готовы смириться с некоторой долей неопределенности. Фокус в том, чтобы обобщить этот подход до сложных моделей с миллионами параметров, но это можно сделать, отбирая на каждом этапе ровно столько примеров из каждого потока, сколько нужно. Вы должны быть достаточно уверены в правильности решения и в том, что общая неопределенность по всем решениям остается в разумных пределах. Таким образом можно эффективно учиться на бесконечном

количестве данных в конечное время: об этом я писал в одной из первых статей, предлагающих этот подход.

Системы больших данных — это как фильмы Сесила Демилля в машинном обучении: тысячи серверов вместо тысяч статистов. В самых крупных проектах надо собрать вместе все данные, верифицировать их, очистить и привести в приемлемую для обучающегося алгоритма форму — по сравнению с этим строительство пирамид покажется прогулкой в парке. Если говорить о масштабе фараонов, европейский проект FuturICT нацелен на построение модели всего мира — в буквальном смысле. Общества, правительства, культура, технология, сельское хозяйство, заболевания, глобальная экономика — ничего не будет упущено. Такие проекты, конечно, нам пока не по силам, но они предзнаменование того, что нас ждет в будущем, и они могут помочь нам нащупать границы масштабируемости и научиться их преодолевать.

Вычислительная сложность — это один вопрос. Другой — сложность человеческая. Если компьютеры страдают синдромом саванта, то алгоритмы машинного обучения иногда производят впечатление вундеркиндов, подверженных приступам дурного настроения: отчасти поэтому люди, способные заставить их слушаться, так много зарабатывают. Если вы умеете настроить все точно как надо, может произойти волшебство: алгоритм станет умен не по годам и одарит вас потоком идей, хотя процесс в чем-то похож на Дельфийского оракула: интерпретация пророчеств сама по себе может требовать большого мастерства. Но поверните ручку неправильно, и обучающийся алгоритм извергнет лавину бессмыслицы или вообще замкнется в себе. К сожалению, в этом отношении Alchemy не лучше большинства. Записать свои знания на языке логики, заложить данные и нажать кнопку — это здорово. Когда Alchemy выдает великолепно точные и эффективные логические сети Маркова — можно пойти в паб и отпраздновать успех. Но когда этого не происходит — а так бывает большую часть времени, — начинается битва. В чем проблема? В знаниях? В обучении? В выводе? С одной стороны, благодаря обучению и вероятностному выводу простая логическая сеть Маркова может выполнить работу сложной программы. С другой стороны, когда

она не работает, ее намного сложнее отладить. Решение — сделать ее более интерактивной, способной к самоанализу и объяснению хода своих рассуждений. Это станет для нас еще одним шагом к Верховному алгоритму.

Сейчас вас примет доктор

Лекарство от рака — программа, которая на входе получает геном раковой опухоли, а на выходе дает лекарство, с помощью которого можно эту опухоль уничтожить. Теперь мы можем в общих чертах описать, как она будет выглядеть. Давайте назовем ее CanceRx. Несмотря на внешнюю простоту, эта программа станет одним из крупнейших и самых сложных проектов в истории: она так велика и сложна, что построить ее можно только с помощью машинного обучения. В ее основе — подробная модель работы живых клеток с подклассами для всех типов клеток человеческого организма и обобщающей моделью их взаимодействия. Эта модель в виде марковской логической сети или чего-то схожего соединит в себе знания из области молекулярной биологии с большим объемом данных из секвенсоров ДНК, микрочипов и многих других источников. Часть знания будет заложена вручную, но большая часть автоматически извлечена из литературы по биологии и медицине. Модель будет постоянно развиваться, включать в себя все новые результаты экспериментов, источники данных и истории болезни. В конце концов она узнает каждый метаболический путь, каждый регуляторный механизм, все химические реакции во всех типах человеческих клеток. Будет получена сумма знаний о молекулярной биологии человека.

Большую часть своего времени CanceRx будет тратить на проверку лекарств-кандидатов. Когда модели покажут новое лекарство, она спрогнозирует его действие и на раковые, и на нормальные клетки. Если Элис поставили онкологический диагноз, CanceRx применит свою модель и к нормальным, и к раковым клеткам девушки и перепробует все доступные лекарства, пока не найдет то, которое убьет раковые клетки, не повреждая здоровые. Если найти работающее лекарство или сочетание лекарств не получится, программа приступит

к разработке нового препарата, возможно, путем эволюции на основе уже существующих или с использованием алгоритма восхождения на выпуклые поверхности или кроссинговера. На каждом этапе поиска лекарство-кандидат будет проходить испытание на модели. Если лекарство останавливает рак, но все же имеет вредные побочные эффекты, CanceRx попытается подкорректировать его, чтобы от них избавиться. Если рак мутирует, весь процесс повторится заново. Но модель будет прогнозировать вероятные мутации еще до их появления, и CanceRx назначит лекарства, которые заблокируют их развитие. В шахматной игре между человечеством и раком CanceRx поставит опухоли мат.

Обратите внимание, что машинное обучение само по себе не подарит нам CanceRx. Нельзя просто собрать обширную базу данных по молекулярной биологии, загрузить ее в Верховный алгоритм и получить готовую идеальную модель живой клетки. CanceRx будет конечным результатом многих итераций, сотрудничества сотен тысяч биологов, онкологов и специалистов по обработке данных по всему миру. Самое важное, однако, что CanceRx будет охватывать полученные с помощью врачей и лечебных учреждений данные миллионов пациентов с раком. Без этих данных мы не сможем победить рак, а с ними — сможем. Вносить свой вклад в растущую базу данных будет не просто в интересах каждого пациента с раком: это станет этическим долгом. В мире CanceRx тайные клинические исследования останутся в прошлом: новые методы лечения, предложенные CanceRx, будут непрерывно внедряться в практику, и в случае успеха их начнут назначать все большему числу пациентов. И успехи, и неудачи станут давать CanceRx ценные данные для обучения, поэтому чем больше данных, тем лучше результаты. С одной стороны, машинное обучение — лишь малая часть проекта CanceRx, значительно уступающая по важности сбору данных и человеческому вкладу. Но если посмотреть под другим углом, машинное обучение — ключевой элемент всего предприятия. Без него знания о биологии рака были бы фрагментарными, разбросанными по тысячам баз данных и миллионам научных статей, а каждый врач располагал бы крупицей нужной

информации. Сбор всего этого знания в связное целое не под силу человеку, как бы умен он ни был. На это способно только машинное обучение. Поскольку все раковые опухоли разные, машинное обучение должно найти общие паттерны, а так как одна ткань может принести миллиарды точек данных, без машинного обучения не разобраться, что сделать с каждым новым пациентом.

Уже предприняты шаги, чтобы создать то, что в конце концов превратится в CanceRx. Специалисты в новой области знания — системной биологии — моделируют не отдельные гены и белки, а целые метаболические сети. Одна из исследовательских групп, работающая в Стэнфорде, построила модель всей клетки. Global Alliance for Genomics and Health поощряет обмен данными между учеными и онкологами, цель которого — проведение в будущем широкомасштабного анализа. CancerCommons.org собирает модели рака и позволяет пациентам делиться своими историями болезни и учиться на схожих случаях. Foundation Medicine точно выявляет мутации в опухолевых клетках пациента и предлагает самое подходящее лекарство. Десятилетие назад было неизвестно, можно ли в принципе вылечить рак, а если можно, то как это сделать. Теперь мы видим, как достичь цели. Идти придется долго, но дорога найдена.

ГЛАВА 10

МИР МАШИННОГО ОБУЧЕНИЯ

Теперь, когда наше путешествие по чудесной Стране машинного обучения закончено, давайте сменим тему и посмотрим, что все это значит лично для вас. Верховный алгоритм — как красная таблетка в «Матрице»[108]. Это портал в другую реальность: ту, в которой мы уже живем, но пока еще о ней не подозреваем. От свиданий до работы, от самопознания до будущего нашего общества, от совместного использования данных до боевых действий и от опасностей искусственного интеллекта до следующей остановки на пути эволюции новый мир уже принимает очертания, и машинное обучение — ключ к нему. Эта глава поможет вам взять от жизни максимум и быть готовым к тому, что ждет нас впереди. Машинное обучение как таковое будет определять будущее не больше, чем любая другая технология. Важно то, что мы решим сделать, и теперь у вас есть инструменты, необходимые для решения.

Главный из этих инструментов — Верховный алгоритм. Когда он появится и будет ли похож на Alchemy — не столь существенно, имея в виду важнейшие способности обучающихся алгоритмов, которые он воплощает, и то, куда они нас приведут. Верховный алгоритм можно с тем же успехом считать сложным изображением текущих и будущих алгоритмов машинного обучения, которым удобно пользоваться в наших мысленных экспериментах вместо конкретных алгоритмов продукта *X* или сайта *Y*, которыми соответствующие компании вряд ли с нами поделятся. Если смотреть с этой точки зрения, обучающиеся алгоритмы, с которыми мы каждый день соприкасаемся, — это «эмбрионы» Верховного алгоритма, и наша задача — понять их и направить их развитие так, чтобы они лучше отвечали нашим потребностям.

В ближайшие десятилетия машинное обучение повлияет на множество аспектов человеческой жизни, и одной главы этой книги совершенно недостаточно, чтобы должным образом их описать. Тем не менее уже можно заметить целый ряд тем,

на которых стоит сосредоточиться, а начнем мы с того, что психологи называют теорией разума — компьютерной моделью вашего сознания, не больше и не меньше.

Секс, ложь и машинное обучение

Цифровое будущее начинается с осознания факта: взаимодействуя с компьютером — будь то ваш собственный смартфон или удаленный за тысячи километров сервер, — вы каждый раз делаете это на двух уровнях. Первый — желание немедленно получить то, что вам нужно: ответ на вопрос, желаемый товар, новую кредитную карточку. На втором уровне, стратегическом и самом важном, вы рассказываете компьютеру о себе. Чем больше вы его учите, тем лучше он будет вам служить или манипулировать вами. Жизнь — это игра между вами и окружающими вас обучающимися алгоритмами. Можно отказаться играть, но тогда в двадцать первом веке вам придется жить как в двадцатом. А можно играть и выиграть. Какую модель вашей личности вы хотите предложить компьютеру? Какие данные можно ему дать, чтобы он построил эту модель? Эти вопросы надо держать в уме всякий раз, когда вы взаимодействуете с алгоритмом машинного обучения — точно так же как при общении с людьми. Элис знает, что у Боба есть ее психологическая модель, и стремится повлиять на нее своим поведением. Если Боб — ее начальник, девушка постарается выглядеть компетентной, лояльной и трудолюбивой. Если она хочет соблазнить Боба, она будет само очарование. Мы едва ли сможем функционировать в обществе без способности интуитивно угадывать, что на уме у других людей, и реагировать на это. Сегодняшний мир отличается только тем, что теории разума начали появляться и у компьютеров. Пока эти теории все еще примитивны, но они быстро развиваются, и нам придется с ними работать не меньше, чем с другими людьми, чтобы получить желаемое. Следовательно, *вам* понадобится теория разума *компьютера*, а даст ее Верховный алгоритм, если подключить к нему функцию присвоения баллов (то, что, по вашему мнению, цели обучающегося алгоритма или, точнее, его хозяина) и данные (то, что, как вы думаете, знает компьютер).

Возьмем онлайн-знакомства. Когда вы пользуетесь Match.com, eHarmony или OkCupid (поборите недоверие, если нужно), ваша цель проста: найти себе лучшую пару из всех возможных. При этом имеется вероятность, что вам придется хорошо потрудиться и пройти через несколько неудачных свиданий, прежде чем вы встретите человека, который вам по-настоящему понравится. Один упорный чудак извлек из OkCupid 20 тысяч профилей, самостоятельно провел добычу данных, на 88-м свидании встретил женщину своей мечты и рассказал о своей одиссее журналу Wired. Два главных инструмента, которые помогут вам преуспеть с меньшим объемом данных и меньшими трудозатратами, — ваш собственный профиль и ваша реакция на предложенные компьютером варианты. Один популярный вариант поведения — говорить неправду (о своем возрасте, например). Такой подход может показаться неэтичным, не говоря о том, что есть риск с треском провалиться, когда избранник откроет правду, но тут имеется нюанс. Искушенные поклонники онлайн-знакомств уже поняли, что люди приукрашивают возраст в профиле, и делают соответствующие поправки, поэтому указать свой настоящий возраст — все равно что сказать, что вы старше, чем на самом деле! В свою очередь, обучающийся алгоритм, подбирающий пары, приходит к выводу, что людям нравятся более молодые партнеры, чем в действительности. Логичный следующий шаг для человека — еще больше исказить свой возраст, и в конце концов этот атрибут становится бессмысленным.

Более удачная стратегия для всех заинтересованных сторон — сосредоточиться на особых, необычных атрибутах, которые очень хорошо предопределяют подходящую пару в том смысле, что отбирают людей, которых полюбили бы вы, но далеко не все остальные. Тем самым уменьшается количество конкурентов. Ваша задача (и вашего потенциального избранника тоже) — предоставить эти атрибуты компьютеру. Работа подбирающего пары алгоритма — учиться на основе этой информации, точно так же как училась бы традиционная сваха. По сравнению с деревенской свахой алгоритм Match.com имеет преимущество: он знает — пусть и поверхностно — несравнимо больше людей. Наивный обучающийся алгоритм, например перцептрон, будет

довольствоваться широкими обобщениями вроде «джентльмены предпочитают блондинок». Более совершенный алгоритм увидит паттерны, например «люди с одинаковыми необычными музыкальными предпочтениями часто хорошо подходят друг к другу». Если и Элис, и Боб любят Бейонсе, этот факт сам по себе вряд ли сведет их друг с другом. Но если им обоим нравится Bishop Allen, это как минимум немного повышает вероятность, что они родственные души. Если оба — фанаты группы, о которой компьютер не слышал, это даже лучше, но уловить это сможет только реляционный алгоритм, например Alchemy. Чем лучше обучающийся алгоритм, тем целесообразнее тратить время на то, чтобы рассказать ему о себе. Согласно общему правилу, лучше дифференцировать себя настолько, чтобы вас не путали со «среднестатистическим человеком» (помните Боба Бернса из главы 8?), но при этом не быть слишком необычным, иначе алгоритм не сможет вас постичь.

На самом деле онлайн-знакомства — сложный пример, потому что «химия» не всегда предсказуема. Если первое свидание пройдет как по маслу, люди могут по уши влюбиться и страстно верить, что созданы друг для друга, а если беседа примет другой оборот — посчитать друг друга назойливыми и потерять всякий интерес к дальнейшим встречам. По-настоящему сложный алгоритм машинного обучения сделал бы следующее: провел тысячу симуляций свиданий в стиле Монте-Карло между всеми вероятными парами, а затем выстроил рейтинг пар согласно доле успешных свиданий. Пока это невозможно, сайты знакомств могут устраивать вечеринки и приглашать людей, каждый из которых — вероятная пара для многих других присутствующих, чтобы дать им возможность за несколько часов сделать то, что в другом случае заняло бы недели.

Если вы не фанат интернет-знакомств, полезным выводом из вышесказанного будет понимание, какие взаимодействия с компьютером стоит сохранять. Если вы не хотите, чтобы из-за рождественских покупок Amazon запутался в ваших предпочтениях, заказывайте подарки на других сайтах. (Прости, Amazon.) Если дома и на работе вы смотрите разные типы видео, заведите два аккаунта на YouTube, по одному для каждой цели, и YouTube научится давать

соответствующие рекомендации. А если вы собираетесь посмотреть то, что вас обычно не интересует, сначала разлогиньтесь. Безопасный режим Chrome используйте не для просмотра сомнительных сайтов (конечно, вы и так туда не ходите), а когда хотите, чтобы текущая сессия не повлияла на персонализацию в будущем. Если в аккаунт на Netflix добавить профили для разных людей, сайт не станет рекомендовать вам взрослые фильмы для вечернего просмотра в кругу семьи. Если вы невзлюбили какую-то компанию, кликайте на ее рекламу: они потратят деньги не только сейчас, но и в будущем, потому что Google научится показывать их объявления тем, кто вряд ли купит их продукцию. А если у вас есть конкретные запросы и вы хотите, чтобы Google в будущем отвечал на них правильно, уделите минуту, пройдитесь по страницам последних результатов, поищите хорошие ссылки и кликните на них. В целом, если система постоянно рекомендует не то, что нужно, попытайтесь ее научить: найдите группу правильных результатов, пройдите по ссылкам, а потом вернитесь и посмотрите, изменилась ли ситуация.

Все это, однако, может потребовать больших усилий и показывает, что, к сожалению, канал коммуникации между вами и обучающимся алгоритмом сегодня очень узок. Желательно иметь возможность не просто косвенно обучать алгоритмы своими действиями, а рассказывать о себе столько, сколько хочется. Более того, неплохо иметь доступ к проверке модели своей личности, сформированной алгоритмом машинного обучения, и исправлять ее по своему усмотрению. Обучающийся алгоритм может проигнорировать вас, если решит, что вы врете или не очень хорошо себя знаете, но как минимум у него будет возможность учесть ваш собственный вклад. Для этого модель должна быть представлена в понятной форме, например в виде набора правил, а не нейронной сети, и в дополнение к необработанным данным она должна на входе принимать общие утверждения, как это делает Alchemy. Все это подводит нас к вопросу, насколько хорошую модель вашей личности может получить обучающийся алгоритм и что вы хотели бы сделать с этой моделью.

Цифровое зеркало

Подумайте обо всех своих данных, которые записаны во всех компьютерах мира. Это электронные письма, документы MS Office, тексты, твиты, аккаунты на Facebook и LinkedIn, история поиска в интернете, клики, скачанные файлы и заказы, кредитная история, налоги, телефон и медицинская карта, статистика на Fitbit, информация о вождении, записанная в бортовом компьютере вашего автомобиля, карта перемещений, зарегистрированная вашим мобильным телефоном, все фотографии, которые вы когда-либо делали, короткие появления в записях камер слежения, а также ролики, снятые на Google Glass, и так далее и тому подобное. Если бы у будущего биографа был доступ только к этому «выхлопу данных» и ни к чему больше, какая бы картина у него сложилась? Вероятно, довольно точная и во многих отношениях подробная, но тем не менее без некоторых существенных деталей. Почему в один прекрасный день вы решили изменить карьеру? Смог бы биограф предсказать этот шаг заранее? Что с человеком, которого вы встретили однажды и никогда о нем не забывали? Смог бы биограф отмотать ленту назад и сказать: «Вот он!»

Понимание, что ни один обучающийся алгоритм (даже Агентство национальной безопасности) на сегодняшний день не имеет доступа ко всем этим данным, а даже если бы имел, то не знал бы, как превратить их в истинное ваше подобие, — это отрезвляющая и, может быть, обнадеживающая мысль. Но представьте, что вы взяли все свои данные и отдали их настоящему Верховному алгоритму будущего, в котором уже есть знание о человеческой жизни, которому мы можем его научить. Он создаст вашу модель, и вы сможете носить ее на флешке в кармане, при желании проверять ее и использовать для всего чего угодно. Безусловно, это будет прекрасный инструмент самоанализа — как посмотреть на себя в зеркало. Но зеркало было бы цифровое и показывало бы не только вашу внешность, но и все, что можно узнать, наблюдая за вами. Зеркало могло бы ожить и поговорить. О чем бы вы у него спросили? Может быть, не все, что скажет модель, придется вам по душе, но это будет лишь еще одной причиной задуматься. А некоторые ответы подскажут новые идеи, новые направления. Модель вашей личности, созданная Верховным алгоритмом, могла бы даже помочь вам стать лучше.

Оставим в стороне самосовершенствование. Вероятно, первое, что вы захотели бы от своей модели, — поручить ей договариваться с миром от вашего имени: выпустить ее в киберпространство, чтобы она искала для вас всякую всячину. Из всех книг в мире она порекомендует десяток, которые вы захотите прочитать в первую очередь, и советы будут такими глубокими, что Amazon и не снились. То же произойдет с фильмами, музыкой, играми, одеждой, электроникой, чем угодно. Разумеется, ваш холодильник будет всегда полон. Модель станет фильтровать вашу электронную и голосовую почту, новости на Facebook и обновления на Twitter, а когда это уместно, отвечать вместо вас. Она позаботится обо всех надоедливых мелочах современной жизни, например о проверке счетов по кредитке, об обжаловании неправильных транзакций, о планировании расписания, обновлении подписок и заполнении налоговой отчетности. Она подберет вам лекарство, сверится с лечащим врачом и закажет его в Walgreens. Она обратит ваше внимание на интересные объявления о работе, предложит место для отпуска, посоветует, за каких кандидатов проголосовать в избирательном бюллетене, и просканирует людей, с которыми стоит сходить на свидание. А после того как вы познакомитесь и понравитесь друг другу, ваша модель объединится с моделью вашей избранницы и выберет рестораны, которые вам обоим могут понравиться. И здесь становится *по-настоящему* интересно.

Общество моделей

В очень быстро надвигающемся будущем вы окажетесь не единственным человеком с «цифровой половинкой», которая круглые сутки исполняет ваши поручения. Подобная модель личности появится у каждого, и модели будут все время общаться друг с другом. Если вы ищете работу, а компания *X* — сотрудников, то ее модель будет проводить собеседование с вашей. Их «разговор» во многом напомнит настоящий, «живой», — ваша модель будет хорошо проинструктирована, например, она не станет выдавать о вас негативную информацию, — однако весь процесс займет всего долю секунды. В своем будущем аккаунте на LinkedIn вы кликнете «Найти работу» и немедленно пройдете собеседование на все

должности во Вселенной, которые хотя бы отдаленно соответствуют вашим параметрам (профессии, местожительству, зарплате и так далее). LinkedIn тут же выдаст ранжированный список самых перспективных предложений, и вы выберете из него первую компанию, с которой желаете побеседовать лично. То же самое с романтическими знакомствами: ваша модель сходит вместо вас на миллионы свиданий, а в ближайшую субботу вы встретитесь с наиболее перспективными кандидатами на вечеринке, организованной OkCupid, зная, что вы один из самых перспективных кандидатов *для них*, и понимая при этом, что *другие* лучшие кандидаты вашего потенциального избранника тоже приглашены. Это будет интересный вечер.

В мире Верховного алгоритма «мои люди свяжутся с вашими» превратится в «моя программа свяжется с вашей программой». У каждого человека будет свита ботов, призванная сделать более легким и приятным его путь по миру. Сделки, переговоры, встречи — все это будет организовано, не успеете вы шевельнуть пальцем. Сегодня фармацевтические компании обхаживают врачей, потому что именно врач решает, какие лекарства вам выписать. Завтра поставщики всех продуктов и услуг, которыми вы пользуетесь или могли бы воспользоваться, будут нацелены на вашу модель, потому что она начнет проверять их для вас. Их боты попытаются убедить вашего бота сделать покупку. Задача вашего бота — насквозь видеть их намерения, как вы — телерекламу, но еще тоньше и подробнее, потому что у вас никогда нет времени и терпения, чтобы во всем разобраться. Прежде чем купить машину, ваша цифровая личность пройдется по всем параметрам, обсудит их с производителем и изучит все, что когда-либо было сказано об этой машине и альтернативных вариантах. Ваша цифровая половинка окажется похожа на гидроусилитель руля: жизнь пойдет туда, куда хотите, но с меньшими усилиями с вашей стороны. Это не значит, что вы окажетесь в «фильтрующем пузыре» и станете видеть только то, что вам гарантированно понравится, без каких-то неожиданностей. Цифровая личность окажется гораздо умнее, у нее будет инструкция оставлять место для шанса, давать вам соприкасаться с новым опытом, искать счастливые случайности.

Еще интереснее, что этот процесс не закончится, когда вы найдете машину, дом, врача, работу или спутника жизни. Цифровая половинка станет постоянно учиться на своем опыте — точно так же как учитесь вы сами. Она будет искать оптимальные подходы везде, будь то собеседования о приеме на работу, свидания или поиск недвижимости. Она будет узнавать сведения о людях и организациях, с которыми взаимодействует от вашего имени, а потом — что еще важнее — черпать информацию из вашего общения с ними в реальности. Если программа предсказала, что вы с Элис будете прекрасной парой, а вам оказалось некомфортно вместе, цифровая половинка построит гипотезы о возможных причинах и проверит их в следующем раунде свиданий. Самыми важными открытиями она будет делиться с вами. («Вы уверены, что вам понравится *X*, но на самом деле вы скорее предпочтете *Y*».) Сравнивая ваши впечатления от различных гостиниц с обзорами на TripAdvisor, программа определит настоящие лакомые кусочки и найдет их в дальнейшем. Она не просто узнает, какие онлайн-магазины достойны доверия, но и поймет, как раскусить самые ненадежные. У вашей цифровой половинки будет модель мира, точнее, модель вашего отношения к миру. В то же время, конечно, все остальные тоже будут располагать непрерывно эволюционирующими моделями своего мира. Все стороны станут взаимодействовать и учиться, а потом применять полученные знания к следующему взаимодействию. У вас будет собственная модель каждого человека и организации, с которыми вы контактировали, а у них сформируется ваша. По мере совершенствования моделей взаимодействие будет все более похожим на то, что сложилось бы в реальном мире, однако происходить оно будет in silico и в миллион раз быстрее. Киберпространство завтрашнего дня превратится в очень обширный параллельный мир, который станет выбирать все самое перспективное, чтобы испробовать в реальности. Это будет похоже на новое, глобальное подсознание, коллективный «Ид» человечества, или «Оно»[109].

Делиться или не делиться, а если да, то где и как

Конечно, полностью самостоятельное познание мира — медленный процесс, даже если ваша цифровая половинка делает это на порядок эффективнее, чем человек из плоти и крови. Если другие узнают вас быстрее, чем вы узнаете их, появятся проблемы. Чтобы этого избежать, надо делиться информацией: миллионы людей, объединив свои знания, узнают компанию или товар гораздо быстрее, чем один человек. Но с кем стоит делиться данными? Это, может быть, самый важный вопрос XXI столетия.

Сегодня данные можно разделить на четыре категории: те, которыми вы делитесь со всеми, те, которыми вы делитесь только с друзьями и коллегами, те, которыми вы делитесь с различными компаниями (сознательно или нет), и те, которые вы вообще не распространяете. К первому типу относятся, например, обзоры на Yelp, Amazon и TripAdvisor, рейтинги на eBay, резюме на LinkedIn, блоги, твиты и так далее. Эти данные очень ценны и порождают меньше всего проблем. Вы делитесь ими с миром, потому что сами того хотите, и это всем идет на пользу. Единственная сложность в том, что компании, которые хранят эти данные, не всегда разрешают массово их скачивать для построения моделей. Им следовало бы изменить свой подход. Сегодня можно зайти на TripAdvisor и увидеть обзоры и рейтинги заинтересовавших вас гостиниц, но как насчет модели факторов, которые делают гостиницу хорошей или плохой в целом? С ее помощью можно было бы оценивать гостиницы, у которых пока мало надежных обзоров или вообще их нет. TripAdvisor мог бы создать что-то подобное. А как насчет моделирования факторов, которые определяют привлекательность гостиницы *именно для вас*? Для этого требуется информация о вашей личности, и вы, возможно, не захотите делиться ею с TripAdvisor. Лучше, чтобы появилась доверенная третья сторона, которая соединит два типа данных и даст вам результат.

Данные второго рода тоже не должны создавать проблем, но это не так, потому что они соприкасаются с третьим видом данных. Вы делитесь новостями и картинками со своими друзьями на Facebook, а они делятся с вами. При этом каждый из вас делится всей этой информацией с сетью Facebook. Сеть получает преимущество: у нее миллиард друзей. День за днем она узнает

о мире гораздо больше, чем смог бы узнать отдельный человек, и узнала бы еще больше, будь алгоритмы качественнее, а они совершенствуются с каждым днем благодаря нам — специалистам по обработке данных. Все эти знания Facebook использует главным образом для адресной рекламы, а взамен создает инфраструктуру для обмена информацией: на эту сделку идет каждый пользователь. Обучающиеся алгоритмы становятся все мощнее и извлекают из данных все больше и больше пользы, которая частично возвращается в форме более уместной рекламы и лучшего обслуживания. Единственная проблема в том, что Facebook вольна делать с данными и моделями то, что противоречит интересам пользователя, и этого избежать не получится.

Такая проблема появляется всюду, где человек делится данными с компаниями, а в наши дни подобные ситуации включают практически все действия в интернете и многие в реальной жизни. Вы еще не заметили, что вокруг идет яростная борьба за информацию о вас? Все хотят заполучить ваши данные, и это неудивительно — ведь таким образом можно найти лазейку в ваш мир, к вашим деньгам, голосу и даже к вашему сердцу. Пока у каждой компании есть лишь частица целого. Google знает, что вы ищете в интернете, Amazon располагает информацией о ваших покупках, AT&T — о телефонных звонках, Apple — о музыке, которую вы скачиваете, Safeway имеет полное представление о том, какие продукты едите, а Capital One — о ваших операциях с кредитными картами. Некоторые компании, например Acxiom, сопоставляют информацию о вас и продают ее, но на поверку (в случае Acxiom можно посмотреть на aboutthedata.com) ее получается немного и она отчасти ошибочна. Ни у кого и близко нет полной картины вашей личности. Это и хорошо, и плохо. Хорошо, потому что у того, кому удастся ее заполучить, появится слишком большая власть. Плохо — потому что, пока это так, создание всеобъемлющей модели невозможно. На самом деле вам нужно просто быть единственным владельцем такой модели и предоставлять к ней доступ исключительно на собственных условиях.

Последний тип данных — те, которыми вы не делитесь, — тоже создает проблему, и она заключается в том, что иногда следует

предоставлять такую информацию. Может быть, это не приходило вам в голову, может быть, это непросто или у вас нет такого желания. В последнем случае стоит задуматься, есть ли у вас этическая обязанность делиться данными о себе. Один пример мы уже видели: больные раком могут внести вклад в победу над этим заболеванием, если предоставят доступ к геному опухоли и истории лечения. Но этим дело не ограничивается. Данные, которые мы генерируем в нашей повседневной жизни, могут дать ответы на всевозможные вопросы об обществе и политике. Социальные науки вступают в свой золотой век и наконец получат объем данных, сопоставимый со сложностью изучаемых явлений, а польза для всех нас будет огромной — при условии, что эти данные окажутся доступными и ученым, и людям, принимающим решения, и самим гражданам. Это не значит, что надо позволить другим подглядывать за вашей личной жизнью; это значит, что надо дать им возможность ознакомиться с полученными моделями, в которых будет только статистическая информация. Между вами и ними должен стоять честный брокер данных, который гарантирует, что информацией о вас не будут злоупотреблять и при этом не появится «халявщиков», которые стремятся получать преимущества, не делясь собственными данными.

Итого, проблемы есть у всех четырех видов данных. Решение у них общее: нужен новый тип компаний, который для ваших данных станет играть ту же роль, что банк для ваших сбережений. Банки (за редким исключением) не воруют и должны мудро инвестировать вклады. Сегодня многие компании предлагают консолидировать ваши данные где-то в облачном хранилище, но они все еще очень далеки от уровня банков персональных данных. Провайдеры облачных сервисов стремятся привязать вас к себе — а этого категорически нельзя допустить (представьте, что вы открыли счет в Bank of America и не уверены, можно ли будет когда-нибудь в будущем перевести средства в Wells Fargo). Некоторые стартапы предлагают вам хранить данные, а затем передают их рекламщикам, давая вам взамен скидки. На мой взгляд, смысл не в этом. В некоторых случаях вы бы дали такую информацию бесплатно, потому что сами в этом заинтересованы, а в некоторых ни за что не стали бы этого делать.

Компании нового типа, как я себе их представляю, за абонентскую плату будут предоставлять несколько функций. Во-первых, они станут анонимизировать ваши взаимодействия в электронном мире, проводя их через собственные серверы, и накапливать их, как и аналогичные действия других пользователей. Во-вторых, будут хранить в одном месте данные, собранные в течение вашей жизни, вплоть до круглосуточного видеопотока Google Glass, если у вас есть такие очки. В-третьих, они будут формировать полную модель вашей личности и вашего мира и постоянно ее обновлять. В-четвертых — применять эту модель от вашего имени, в рамках ее способностей, всегда делая ровно то, что сделали бы вы сами. Основное обязательство компании перед вами — никогда не использовать ваши данные и вашу модель вопреки вашим интересам. Гарантия не будет стопроцентной — в конце концов, мы и сами не застрахованы от того, чтобы иногда сделать что-нибудь себе во вред. Тем не менее жизнеспособность компании станет зависеть от выполнения договоренности в той же степени, как выживание банка — от сохранности ваших денег, поэтому можно будет доверять им так, как мы сегодня доверяем банкам.

Такие компании могут быстро стать одними из самых дорогих в мире. Как указывает Алексис Мадригал из журнала Atlantic, сегодня ваш профиль можно купить за полцента или даже дешевле, однако для индустрии интернет-рекламы ценность пользователя приближается к 1200 долларам в год. Фрагмент информации о вас, имеющийся в распоряжении Google, стоит около 20 долларов, у Facebook — 5 долларов и так далее. Прибавьте к этому фрагменты, которых пока ни у кого нет, и тот факт, что целое весомее суммы частей — модель личности, основанная на всех ваших данных, намного лучше тысячи моделей, построенных из отдельных кусочков, — и это легко даст более триллиона долларов в год для такой экономики, как США. На этом фундаменте несложно построить компанию из списка Fortune 500. (Если вы решите принять вызов и станете миллиардером, не забудьте, кто вам подбросил идею.)

Конечно, некоторые уже существующие компании с большим удовольствием приютят вашу «цифровую личность». Например,

Google. Сергей Брин хочет, чтобы Google стала «третьим полушарием вашего мозга», и некоторые из приобретений компании, вероятно, связаны с тем, как удачно потоки пользовательских данных дополняют поток самой компании. Но, несмотря на исходные преимущества, Google и Facebook, например, не очень подходят на роль вашего цифрового дома, потому что возникает конфликт интересов. Они зарабатывают себе на жизнь таргетированием рекламы, поэтому им придется как-то уравновешивать интересы пользователей и рекламодателей. Вы, наверное, не допустите, чтобы одно из полушарий было не совсем вам лояльно? Тогда зачем позволять это третьему полушарию?

Потенциальная угроза может исходить от государственных органов, если у них будет право истребовать ваши данные или даже профилактически посадить вас за решетку, как в фильме «Особое мнение»[110], если ваша модель напоминает модель преступника. Чтобы это предотвратить, хранящая данные компания должна их шифровать, а ключ должен быть в вашем распоряжении (в наши дни уже можно производить вычисления на зашифрованных данных без их расшифровки). Или можно держать все на жестком диске у себя дома, а компания просто предоставит программное обеспечение в аренду.

Если вам не по душе мысль, что ключи к вашему миру находятся в руках коммерческой организации, можете вступить в Союз защиты данных. (Если в вашем уголке киберлеса его пока нет, подумайте, не основать ли его самому.) Люди XX века нуждались в профессиональных союзах, чтобы уравновесить силы рабочих и руководства. Союзы защиты данных в XXI веке понадобятся по схожим причинам. Корпорации обладают несравнимо б*о*льшими возможностями по сбору и использованию данных, чем отдельные люди. Это ведет к диспропорции сил, а чем ценнее данные, чем лучше и полезнее модели, которые можно построить на их основе, и тем сильнее неравенство. Союз позволит своим членам требовать у компаний справедливые условия использования данных. Наверное, за создание таких союзов могут взяться уже существующие профсоюзы, однако они ограничены по профессиональному и географическому принципам. Союзы защиты данных могут быть более гибкими: чтобы полученные

модели были полезнее, объединяйтесь с людьми, с которыми у вас много общего. Обратите внимание, что членство в союзе защиты данных не подразумевает предоставление доступа к вашей информации другим его членам. Просто каждый сможет пользоваться моделями, полученными из собранных воедино данных. Союзы защиты данных могут стать посредниками, задача которых — объяснить политикам, чего вы хотите. Ваши данные смогут влиять на мир, как ваш голос, а может, и больше: к урне вы приходите только в день голосования. Все остальное время вашим голосом будут ваши данные.

До сих пор я не произнес словосочетания «частная жизнь», и это не случайно. Частная жизнь — лишь один аспект более широкой проблемы предоставления доступа к информации, и, если сосредоточиться на нем в ущерб целому, как в сегодняшних дебатах, мы рискуем прийти к неправильным выводам. Например, законы, запрещающие использовать данные в любых целях за исключением исходно предусмотренных, крайне близоруки. Когда люди обменивают защиту частной жизни на другие блага, как при заполнении профиля на сайте, они ценят ее намного меньше, чем когда отвечают на отвлеченные вопросы вроде «Важна ли для вас защита частной жизни?». Тем не менее дебаты о частной жизни чаще загоняют в рамки именно таких вопросов. Европейский суд издал декрет о праве человека на забвение, но ведь у людей есть и право на память — как в собственных нейронах, так и на жестком диске. Такое же право есть у компаний до тех пор, пока интересы пользователей, собирателей данных и рекламщиков совпадают. Отвлекаться не на то, что надо, вредно для всех, и чем лучше данные, тем лучше будет продукция. Частная жизнь — это не игра с нулевой суммой, хотя к ней часто относятся именно так.

Компании, которые хранят вашу цифровую личность, и союзы защиты данных, на мой взгляд, определят картину работы с данными в развитом будущем. Наступит ли оно — вопрос открытый. Сегодня большинство людей не осознают, сколько данных они предоставляют и с какими затратами и преимуществами это может быть связано для них. Компании, со своей стороны, с удовольствием сохраняют статус-кво и работают негласно, боясь прокола. Рано или поздно такая система

рухнет, и в атмосфере скандала будут приняты драконовские законы, от которых хорошо не будет никому. Лучше воспитывать сознательность сейчас и давать каждому право делать выбор — делиться ли данными, а если да, то как и где.

Нейронная сеть украла у меня работу

Каких интеллектуальных усилий требует ваша работа? Чем бо́льших, тем в большей вы безопасности. На заре появления искусственного интеллекта считалось, что «синих воротничков» компьютеры заменят раньше, чем «белых», потому что последним приходится больше думать. Но дело обернулось совсем не так: роботы собирают автомобили, но не вытеснили строителей. С другой стороны, алгоритмы машинного обучения заняли место кредитных аналитиков и работников прямого маркетинга. Как оказалось, оценивать заявления о кредите машинам проще, чем не спотыкаясь ходить по стройплощадке, хотя у людей все наоборот. Общая тема здесь в том, что узко определенные задания легко научиться решать, имея данные, а вот задачи, требующие широкого сочетания навыков и знания, научиться решать не так просто. Бо́льшая часть мозга человека выделена для обеспечения зрения и движений, то есть ходьба — намного более сложное дело, чем может показаться, но мы принимаем необходимость ходить как должное, поскольку этот процесс доведен до совершенства эволюцией и в основном выполняется подсознательно. Narrative Science разработала систему искусственного интеллекта, которая довольно хорошо умеет писать отчеты по итогам бейсбольных матчей, но романы ей не под силу, потому что жизнь намного сложнее, чем бейсбол. Распознавание речи с трудом дается компьютерам, так как нужно в буквальном смысле заполнять пробелы — звуки, которые мы постоянно проглатываем, — не имея понятия, о чем идет речь. Алгоритмы умеют прогнозировать колебания курсов акций, но не представляют, как они связаны с политикой. Чем больше контекста требует профессия, тем менее вероятно, что компьютеры быстро ее освоят. Здравый смысл важен не только потому, что так говорила мама, но и потому, что у компьютеров его нет.

Лучший способ не потерять работу — самому ее автоматизировать и сосредоточиться на тех ее аспектах, на которые вам не хватало времени и которые компьютер пока не может освоить. (Если таких аспектов нет, опередите события и поищите новую работу прямо сейчас.) Если машины освоили вашу профессию, не пытайтесь с ними соревноваться. Впрягите их. H&R Block по-прежнему на плаву, а работа специалистов по оформлению налоговых деклараций куда менее занудная, чем раньше, потому что компьютеры взяли на себя бо́льшую часть рутины. (Хотя налоги, наверное, не лучший пример: налоговый кодекс растет экспоненциально, и это одна из немногих вещей, которая может тягаться с экспоненциальным ростом вычислительных мощностей.) Считайте большие данные продолжением ваших органов чувств, а обучающиеся алгоритмы — расширением мозга. Лучшие шахматисты сегодня — так называемые кентавры, наполовину люди, наполовину программы. Это справедливо для других профессий, от аналитиков фондовых рынков до агентов по подбору бейсболистов. Это не соревнование человека с машиной, а конкуренция человека, вооруженного машиной, с человеком без таковой. Данные и интуиция — как конь и наездник: не надо пытаться обогнать лошадь, надо ее оседлать.

По мере развития технологий человеческое и компьютерное будут все теснее переплетаться. Проголодались? Yelp предложит пару хороших ресторанов. Вы выбираете один из них и смотрите на подсказки GPS. Электроника автомобиля контролирует вождение на низком уровне. Все мы уже киборги. В реальности автоматизация не уничтожает, а создает возможности: некоторые профессии исчезают, но гораздо больше появляется. Прежде всего, она делает возможными вещи, которые в исполнении человека слишком дороги. Банкоматы отчасти пришли на смену банковским кассирам, но прежде всего они дали возможность снимать деньги в любом месте и в любое время. Если бы пиксели раскрашивали живые мультипликаторы, мы не знали бы «Истории игрушек»[111] и компьютерных игр.

Тем не менее может возникнуть вопрос: не останутся ли люди без работы? Я думаю, не останутся. Даже если когда-нибудь — очень нескоро — компьютеры и роботы опередят нас во всем, как

минимум некоторым из нас будет чем заняться. Робот может идеально выполнить роль бармена — вплоть до милой беседы с клиентом, — но хозяева заведения все равно отдадут предпочтение человеку, просто потому, что он человек. Рестораны с официантами-людьми будут престижнее, как сейчас вещи ручной работы. У нас есть кино, автомобили и моторные лодки, но люди все равно смотрят спектакли, катаются на лошадях и ходят под парусом. Еще важнее, что некоторые специалисты окажутся поистине незаменимыми, потому что в их работе есть то, чего у компьютеров и роботов не может быть по определению: человеческий опыт. Я не имею в виду «сентиментальные» занятия, потому что сентиментальность несложно подделать: посмотрите на успехи механических домашних животных. Речь идет о гуманитарных дисциплинах, которые невозможно понять без опыта, доступного только людям. Сейчас есть опасения, что гуманитарные науки вошли в штопор и вымирают, однако, когда другие области будут автоматизированы, они восстанут из пепла. Чем обширнее и дешевле автоматизированное производство, тем ценнее вклад гуманитариев.

В то же время долгосрочные перспективы специалистов по точным и естественным наукам, к сожалению, не самые радужные. Науку будущего вполне могут продвигать только компьютеры, а люди, ранее называвшиеся учеными (такие как я сам), вынуждены будут положить жизнь на то, чтобы понять научные достижения, сделанные компьютерами. При этом они по-прежнему будут довольны своей работой, ведь наука, в конце концов, — это удовлетворение собственного любопытства. Сохранится и еще одна очень важная профессия для людей с техническим складом ума: присматривать за компьютерами. На самом деле для этого нужно быть не просто инженером, и в итоге к этому может свестись работа всего человечества: мы будем определять, чего хотим от машин, и контролировать, дают ли они то, что надо. Подробнее об этом мы поговорим ниже.

По мере смещения границы между автоматизируемыми и неавтоматизируемыми профессиями мы, скорее всего, будем наблюдать рост безработицы, уменьшение заработков все большего и большего числа специалистов и рост доходов в тех областях,

которые автоматизировать пока нельзя. Это, конечно, уже происходит, но в дальнейшем будет выражено гораздо сильнее. Переходный период окажется бурным, но благодаря демократии все кончится хорошо. (Берегите свое право голоса — может быть, это самое ценное, что у вас есть.) Когда уровень безработицы перевалит за 50 процентов, а может, и раньше, отношение к распределению благ радикально изменится. Недавно ставшее безработным большинство будет голосовать за щедрые пожизненные пособия, и, чтобы их обеспечить, понадобятся огромные налоги. Однако это не будет слишком затратно, потому что все необходимое начнут производить машины. Вместо уровня безработицы мы станем говорить об уровне трудоустройства, снижение которого будет считаться признаком прогресса. («США отстает! Уровень трудоустройства все еще целых 23 процента!») Пособия по безработице сменятся базовым доходом для всех граждан. Если кому-то этого будет не хватать, всегда появится возможность заработать больше, неизмеримо больше, в немногих оставшихся человеческих профессиях. Либералы и консерваторы все так же станут ломать копья вокруг ставки налогообложения, но целевые показатели изменятся навсегда. После того как ценность труда сильно упадет, самыми богатыми станут страны с самым высоким соотношением природных богатств к численности населения (переезжайте в Канаду). Для неработающих жизнь совсем не будет бесцельной: не больше, чем бессмысленна жизнь на тропическом острове, где все потребности удовлетворяет щедрая природа. Разовьется экономика дарения, предвестник которой сегодня — программное обеспечение с открытым кодом. Как и сейчас, люди будут искать смысл жизни во взаимоотношениях, саморазвитии, духовности. Необходимость зарабатывать себе на жизнь станет далеким воспоминанием, еще одним осколком варварского прошлого, из которого мы выросли.

Война — не для людей

Солдатскую службу автоматизировать сложнее, чем науку, но прогресс не обойдет стороной и вооруженные силы. Одно из важнейших призваний роботов — делать то, что для человека

слишком опасно, а ведение войн опасно по определению. Уже сегодня роботы обезвреживают взрывные устройства, а дроны позволяют подразделению «заглянуть за высотку». На подходе беспилотные автомобили снабжения и роботы-мулы. Вскоре нам придется определиться, разрешать ли роботам нажимать на спусковой крючок. Аргумент «за» — стремление обезопасить живых военнослужащих, а также непригодность дистанционного управления в быстро меняющейся обстановке и ситуациях типа «стреляй или погибнешь». Против такого решения говорит тот факт, что роботы не понимают этики, поэтому им нельзя доверить решения, связанные с жизнью и смертью. В то же время их можно научить этике, и здесь возникает более глубокий вопрос: готовы ли мы к этому.

Несложно заложить в робота общие принципы применения оружия, например военные соображения, пропорциональность и сохранение жизни мирного населения. Но между этими принципами и конкретной ситуацией — пропасть, которую должно преодолеть рассуждение солдата. Когда роботы будут пытаться применить на практике три закона робототехники, они быстро столкнутся с проблемами, что, собственно, иллюстрируют рассказы Айзека Азимова. Общие принципы обычно противоречивы, а иногда исключают друг друга, и это неизбежно, иначе мир станет черно-белым, без оттенков. Когда военная необходимость превыше жизни гражданских лиц? На это нет однозначного ответа, и не получится вложить в компьютер все непредвиденные обстоятельства. Выход дает машинное обучение. Во-первых, надо научить роботов распознавать соответствующие концепции, предоставив им, например, наборы ситуаций, где гражданских пощадили или не пощадили, применение оружия было и не было пропорциональным и так далее. Затем надо составить для роботов кодекс поведения в виде правил с этими концепциями. Наконец, их надо научить применять этот кодекс путем наблюдения за людьми: в этом случае солдат открыл огонь, а в этом воздержался. Обобщая эти примеры, робот сможет сформировать комплексную модель принятия этических решений в виде, скажем, большой логической сети Маркова. Когда его решения начнут совпадать с человеческими настолько, насколько решения разных людей совпадают между

собой, обучение будет завершено и модель можно будет загрузить в тысячи электронных мозгов. В отличие от людей роботы не будут терять голову в горячке боя. Если робот станет функционировать неправильно, ответственность за это понесет производитель. Если начнет поступать неправильно — учителя.

Как вы, наверное, догадались, главная проблема такого подхода заключается в том, что учиться этике путем наблюдения за людьми, возможно, не лучшая идея. Робот может серьезно смутиться, увидев, что своими действиями люди часто нарушают собственные этические принципы. Поэтому можно, например, очистить обучающие данные, оставив только примеры поведения солдата, признанные этической комиссией правильными. Члены комиссии также станут проверять и корректировать модель после обучения, пока их не удовлетворит результат. Если в комиссию войдут разные люди, достичь консенсуса будет непросто, но так и должно быть. Обучение роботов этике, с учетом их логичности и отсутствия предыдущего опыта, заставит нас перепроверить собственные представления и исключить из них противоречия. Здесь, как и во многих других областях, большим плюсом машинного обучения может оказаться не то, что узнают машины, а то, что узнаем мы сами — их учителя.

Другой аргумент против армий роботов — то, что война станет слишком легким делом. При этом односторонний отказ от такого вооружения сам по себе может спровоцировать нападение. Логическая реакция, которую поддерживают ООН и Human Rights Watch, — соглашение о запрете роботизированного вооружения, аналогичное Женевскому протоколу 1925 года, запретившему химическое и биологическое оружие. Однако здесь есть важный нюанс. Химическое и биологическое оружие только увеличивает человеческие страдания, а роботизированное может заметно их облегчить. Если войну ведут машины, а люди ими командуют, не будет убитых и раненых, поэтому, наверное, надо запрещать не роботизированных солдат, а — когда мы будем к этому готовы — солдат-людей.

Армии роботов действительно могут повысить вероятность боевых действий, но они изменят и саму этику войны. Когда на прицеле робот, решить дилемму «стрелять — не стрелять»

намного проще. Сейчас войну считают невообразимым ужасом, крайней мерой. Такая точка зрения будет скорректирована: война по-прежнему останется оргией разрушения, убыточной для всех воюющих сторон, и ее будут избегать по мере возможности, но не любой ценой. А если война сведется к соревнованию в мощи уничтожения, почему не соревноваться в масштабе созидания?

Как бы то ни было, запрет роботизированного вооружения может оказаться невозможным. Сегодня вместо того, чтобы запрещать дронов — предшественников завтрашних боевых роботов, — большие и малые государства заняты их разработкой, вероятно, потому, что преимущества превышают риски. Как и с любым оружием, безопаснее иметь роботов, чем верить, что противник соблюдает правила. Если в будущем миллионы дронов-камикадзе окажутся способны за считаные минуты разбить традиционные армии, пусть это лучше будут наши дроны. Если Третья мировая война закончится за секунды и сведется к взятию под контроль систем врага, лучше иметь умные, быстрые и стойкие сети. (Системы, не объединенные в сети, не помогут: хотя их нельзя взломать, они не идут ни в какое сравнение с сетевыми.) В итоге гонка роботизированных вооружений может стать благом, если приблизит день, когда Пятая женевская конвенция запретит участие людей в боевых действиях. Войны будут всегда, но не обязательно с человеческими жертвами.

Google + Верховный алгоритм = Skynet?

Конечно, армии роботов пробуждают в воображении еще одного призрака. Если верить голливудским фильмам, в будущем человечество обречено на порабощение гигантским искусственным интеллектом и огромной армией послушных ему машин (если, конечно, в последние пять минут отважный герой всех не спасет). У Google уже есть масштабное оборудование, которое необходимо искусственному интеллекту, а недавно компания приобрела и второй элемент — массу стартапов, занимающихся робототехникой. Если добавить в ее серверы последний ингредиент — Верховный алгоритм, наша песенка спета?

Признаюсь — все именно так и будет. Пришло время в стиле Толкина объявить о моих истинных намерениях:

Три алгоритма — ученым в небесных шатрах,
Семь — инженерам, живущим в дворцах серверов,
Девять — смертным магнатам, чей жребий — забвение и прах.
А один — искусственному разуму на троне тьмы,
В мрачном крае обучения, где залежи данных.
Один алгоритм правит всеми, один их отыщет,
Один соберет их и накрепко свяжет во тьме
В мрачном царстве обучения, где залежи данных.

Ха-ха-ха! Но если говорить серьезно, стоит ли опасаться, что машины возьмут верх? Ведь зловещие признаки действительно существуют. Компьютеры с каждым годом не просто делают больше работы, но и принимают все больше решений: кому дать кредит, какие товары человек покупает, кого принять на работу, кому дать прибавку к зарплате, какие котировки пойдут вверх, а какие вниз, сколько будет стоить страховка, куда направить полицейские патрули и кто будет арестован, какой тюремный срок этот человек получит, кто с кем пойдет на свидание и, следовательно, кто родится. Модели машинного обучения уже играют во всем этом свою роль. Момент, когда можно было выключить все компьютеры, не опасаясь коллапса современной цивилизации, давно пройден. Машинное обучение — последняя соломинка: когда компьютеры начнут программировать себя самостоятельно, все надежды взять их под контроль, безусловно, окажутся потеряны. Выдающиеся ученые, например Стивен Хокинг, призвали немедленно начать исследования на эту тему, потому что потом будет слишком поздно.

Расслабьтесь. Шансы, что искусственный интеллект, вооруженный Верховным алгоритмом, захватит мир, равны *нулю*. Причина проста: в отличие от людей компьютеры не обладают собственной волей. Они порождение инженеров, а не эволюции. Даже бесконечно мощный компьютер будет всего лишь продолжением нашей воли, и бояться нам нечего. Вспомните три обязательных компонента обучающихся алгоритмов: представление, оценка и оптимизация. Представления алгоритма

машинного обучения очерчивают то, чему он может научиться. Давайте сделаем его мощным, как марковская логика, чтобы он в принципе мог узнать все. Оптимизатор делает все, что в его силах, чтобы максимизировать функцию оценки — не больше и не меньше, — а саму функцию оценки *определяем мы.* Мощный компьютер просто будет оптимизировать ее лучше. Нет риска, что алгоритм выйдет из-под контроля, даже если он генетический. Выведенная система, которая делает не то, что мы хотим, окажется крайне неприспособленной и вскоре вымрет. Более того, именно системы, которые умеют хоть немного лучше выполнять наши желания, станут поколение за поколением множиться и захватывать пул генов. Конечно, если человечество окажется настолько глупым, чтобы специально запрограммировать компьютер поработить его, оно, может быть, получит по заслугам.

То же самое относится ко всем системам искусственного интеллекта, потому что они — прямо или косвенно — состоят из тех же трех компонентов. Они могут делать самые разные вещи и даже придумывать неожиданные планы, но только для достижения целей, которые мы перед ними поставим. Робот, запрограммированный для того, чтобы «приготовить хороший ужин», может пожарить стейк, сварить буйабес и даже порадовать вкуснейшим блюдом собственного изобретения, но шанс, что он специально отравит своего хозяина, не больше, чем то, что автомобиль вдруг решит взлететь. Задача систем искусственного интеллекта — находить решения NP-полных проблем, которые, как вы, может быть, помните из главы 2, могут занимать экспоненциальное время, но всегда эффективно проверяемы. Поэтому мы должны с распростертыми объятиями встречать компьютеры, которые несравнимо мощнее нашего мозга, спокойно осознавая, что наша работа экспоненциально легче, чем их. Им придется решать проблемы, а нам — просто проверять, что они сделали для нас. Искусственный интеллект будет думать быстро там, где мы думаем медленно, и мир от этого станет только лучше. Лично я буду очень рад нашим новым слугам — роботам.

В опасениях, что разумные машины возьмут верх, нет ничего удивительного, потому что единственные известные нам разумные сущности — люди и другие животные, и у них, несомненно, есть

собственная воля. Однако разум и воля не обязательно должно быть связаны, или, точнее, они не обязательно будут находиться в том же самом организме при условии, что между ними проходит линия контроля. В книге The Extended Phenotype[112] Ричард Докинз показывает, что природа изобилует примерами генов, власть которых выходит за пределы организма животного, — от кукушечьих яиц до бобровых хаток. Технологии — расширенный фенотип человека, то есть мы можем управлять ими, даже если их сложность намного превышает наше понимание.

Давайте перенесемся на 2 миллиарда лет назад и представим, как две нити ДНК купаются в своем бассейне — бактериальной цитоплазме — и обдумывают историческое решение. «Мне не по себе, Диана, — говорит одна. — Если мы начнем создавать многоклеточные создания, не возьмут ли они верх?» Вернемся в XXI век: ДНК живет и здравствует — ей даже лучше, чем когда бы то ни было, потому что отчасти она населяет безопасные двуногие организмы, состоящие из триллионов клеток. Наши крошечные друзья — двойные спирали — прошли довольно большой путь с момента того памятного решения. На данный момент люди оказались самыми хитрыми созданиями: они изобрели контрацепцию, которая позволяет развлекаться, не распространяя нашу ДНК, и у них есть — если это не иллюзия — свободная воля. Но именно ДНК по-прежнему формирует наши представления о развлечениях, а свободную волю мы используем, чтобы гнаться за удовольствиями и избегать боли, а такое поведение в основном совпадает с тем, что выгодно для выживания нашей ДНК. Мы можем стать для ДНК последним этапом развития, если предпочтем сделаться кремниевыми существами, но даже в таком случае история этой молекулы длилась целых 2 миллиарда лет. Сегодня перед нами похожее решение: если начать работать над искусственным интеллектом — обширным, взаимопереплетенным, сверхчеловеческим, непостижимым, — не возьмет ли он верх? Не более чем огромные и непостижимые многоклеточные организмы взяли верх над генами. Искусственный интеллект — это наш инструмент выживания, точно так же как мы сами — инструмент выживания для наших генов.

Все это не значит, однако, что нам не о чем беспокоиться. Первая большая опасность, как в случае любой технологии, — то, что искусственный интеллект может попасть в плохие руки. Чтобы какой-нибудь шутник или преступник не запрограммировал искусственный интеллект захватить мир, лучше организовать ИИ-полицию, способную выловить и стереть его до того, как все зайдет слишком далеко. Лучшая страховка против буйства мощного искусственного интеллекта — еще более мощный искусственный интеллект, поддерживающий мир на планете.

Второй повод для беспокойства — вероятность, что люди сами, добровольно сдадут власть. Все начнется с предоставления прав роботам: далеко не всем это кажется так же абсурдным, как мне. В конце концов, мы ведь уже дали права животным, хотя они нас об этом не просили. Права роботов могут показаться логичным этапом в расширении «круга эмпатии». Почувствовать сострадание к ним несложно, особенно если их специально разработали, чтобы вызывать такие чувства: в этом преуспели даже тамагочи, японские «виртуальные питомцы» с тремя кнопками и жидкокристаллическим дисплеем. Гонка за наращивание эмпатии начнется с первого появившегося в продаже человекообразного робота, потому что они будут продаваться намного лучше, чем обычные металлические. Дети, воспитанные роботами-нянями, на всю жизнь сохранят слабость к добрым электронным друзьям. Эффект «зловещей долины» — дискомфорт, который вызывают существа, почти, но не совсем идентичные человеку, — будет им незнаком, потому что они с детства знакомы с причудами роботов, а в подростковом возрасте, может быть, для крутости будут даже им подражать.

Следующий этап ползучей передачи власти искусственному интеллекту — разрешить ему принимать вообще все решения: он ведь гораздо умнее. Здесь надо быть очень осторожным. Может быть, компьютеры и правда умнее нас, но они служат тому, кто разработал их функции оценки. Это проблема «Волшебника страны Оз». В мире разумных машин надо будет постоянно следить за тем, чтобы они делали ровно то, что от них требуется, причем и на входе (целеполагание), и на выходе (проверяли, выдала ли машина то, что от нее просили). Если этого не сделаете вы сами, сделает кто-то

другой. Машины могут помочь человечеству осознать свои желания, но самоустранившийся проигрывает: совсем как при демократии, но в еще большей степени. Об этом не принято говорить, но человек довольно легко попадает в подчинение, а ведь любой достаточно развитый искусственный интеллект неотличим от бога, и, возможно, никто не станет возражать против приказов какого-то огромного непогрешимого компьютера. Вопрос в том, кто надзирает за надзирателем. Будет ли искусственный интеллект путем к совершенной демократии или к тотальной диктатуре? Пора начать вечное дежурство.

Третий и, наверное, самый серьезный повод для беспокойства — то, что, как джинн из сказки, машины начнут давать нам то, чего мы просим, а не то, чего мы на самом деле хотим. Это не гипотетический сценарий: обучающиеся алгоритмы постоянно так поступают. Мы обучаем нейронные сети узнавать лошадей, а они учатся узнавать коричневые пятна, потому что все лошади в обучающем наборе были гнедые. Вы купили часы, и Amazon начинает рекомендовать схожие предметы — еще больше часов, — хотя они вам уже совершенно ни к чему. Если проверить все решения, которые сегодня принимают компьютеры — например, о выдаче кредитов, — можно заметить, что они зачастую необоснованно плохи. Человек был бы не лучше, если бы его мозг работал по методу опорных векторов и все познания об оценке кредитоспособности были бы получены из тщательно проработанной, но никуда не годной базы данных. Люди беспокоятся, что компьютеры станут слишком умны и захватят мир, однако настоящая проблема в том, что мир уже захвачен глупыми компьютерами.

Эволюция, часть вторая

Даже если компьютеры пока еще не блещут интеллектом, не вызывает сомнения, что их умственные способности очень быстро растут. Еще в 1965 году британский статистик Ирвинг Гуд[113], во время Второй мировой работавший с Аланом Тьюрингом над взломом «Энигмы»[114], размышлял на тему приближающегося взрывного роста интеллекта. Гуд указывал: если мы способны

разработать машины умнее нас самих, те, в свою очередь, должны быть способны разработать машины, которые будут умнее их, и так до бесконечности, оставляя человеческий разум далеко позади. В 1993 году Вернор Виндж[115] окрестил это «сингулярностью». Эту концепцию широко популяризировал Рэймонд Курцвейл, который в книге The Singularity Is Near («Сингулярность рядом») утверждает, что момент, при котором разум машин превысит человеческий — давайте назовем его точкой Тьюринга, — не только неизбежен, но и ждет нас в течение ближайших нескольких десятилетий.

Очевидно, что без машинного обучения — программ, которые разрабатывают программы, — сингулярность не наступит. Еще для этого понадобится достаточно мощное оборудование, но оно не отстает. Точки Тьюринга мы достигнем вскоре после того, как изобретем Верховный алгоритм. (Я готов поспорить с Курцвейлом на бутылку Dom Pérignon, что это произойдет раньше, чем мы сконструируем мозг путем обратной инженерии — предложенного им метода достижения искусственного интеллекта, равного человеческому.) При всем уважении к Курцвейлу, однако, это приведет не к сингулярности, а к чему-то намного более интересному.

Термин «сингулярность» пришел из математики — там он обозначает точку, в которой функция стремится к бесконечности. Например, у функции $1/x$ такая точка — это $x = 0$, потому что результат деления единицы на ноль равен бесконечности. В физике классический пример сингулярности — черная дыра: точка бесконечной плотности, где конечное количество материи сжимается в бесконечно малое пространство. Единственная проблема с сингулярностью заключается в том, что на самом деле она не существует. (Когда вы последний раз делили торт между нулем человек и каждый из них получал бесконечный кусок?) Если физическая теория предсказывает нечто бесконечное, значит, с ней что-то не в порядке. Наглядный пример: общая теория относительности предсказывает, что черные дыры имеют бесконечную плотность, предположительно из-за того, что игнорируют квантовые эффекты. Аналогично разум не может развиваться вечно. Курцвейл признает это, но усматривает в совершенствовании технологии серию экспоненциальных кривых

(скорость процессоров, объем памяти и так далее) и утверждает, что границы этого роста так далеки, что можно не принимать их во внимание.

Утверждение Курцвейла страдает переобучением. Он правильно обвиняет других в склонности к линейной экстраполяции — люди часто видят прямые линии вместо кривых, — но затем сам становится жертвой более экзотического недуга: видит везде экспоненты. В пологих кривых (ничего не происходит) он видит еще не начавшийся экспоненциальный рост. Однако кривые совершенствования технологий — это не экспоненты, а сигмоиды, наши добрые знакомцы из главы 4. Их начальные отрезки действительно легко перепутать с экспонентами, но затем они быстро расходятся. Большинство кривых Курцвейла — это следствия из закона Мура, который почти исчерпал себя. Курцвейл утверждает, что на смену полупроводникам придут другие технологии и на одних сигмоидах будут возникать другие, каждая круче предыдущей, но это лишь домыслы. Когда Курцвейл идет еще дальше и утверждает, что экспоненциально ускоряющийся прогресс виден не только в человеческих технологиях, но и во всей истории жизни на Земле, это восприятие по крайней мере частично связано с эффектом параллакса: нам кажется, что вещи, которые расположены ближе, движутся быстрее. Трилобитам[116], жившим в разгар Кембрийского взрыва, можно простить веру в экспоненциально ускоряющийся прогресс, но затем последовало большое замедление. Тираннозавры, вероятно, предложили бы вместо этого закон роста размеров организма. Эукариоты (и мы в том числе) эволюционируют медленнее, чем прокариоты[117] (например, бактерии). Эволюция ускоряется далеко не равномерно, а как придется.

Чтобы обойти проблему невозможности бесконечно плотных точек, Курцвейл предложил приравнять сингулярность к горизонту событий черной дыры: области, где гравитация настолько сильна, что не выпускает даже свет. То же самое, говорит он, и с сингулярностью: это точка, за пределами которой технологическая эволюция настолько ускоряется, что люди не могут ни предсказать ее, ни понять, что произойдет. Если это и есть сингулярность, значит, мы уже ее достигли: мы не в состоянии

угадать заранее, что придумает обучающийся алгоритм, и часто даже не понимаем полученный результат. Фактически мы уже живем в мире, постижимом для нас лишь отчасти. Главное различие заключается в том, что наш мир в какой-то степени создан нами, а это, безусловно, прогресс. Мир за пределами точки Тьюринга будет непонятен для нас не больше, чем плейстоцен[118]. Как всегда, мы сосредоточимся на том, в чем можем разобраться, а остальное назовем случайностью (или провидением).

Траектория, на которой мы находимся, — это не сингулярность, а фазовый переход. Его критическая точка — точка Тьюринга — наступит, когда машинное обучение опередит естественное. Само естественное обучение тоже прошло через три фазы: эволюцию, мозг и культуру. Каждое было продуктом предыдущего, и каждое училось быстрее. Машинное обучение — логическая следующая стадия этой цепочки. Компьютерные программы — самые быстрые репликаторы на земле: их копирование занимает всего долю секунды — но рождаются они медленно, если их должен написать человек. Машинное обучение устраняет это узкое горло, оставляя только окончательное ограничение: скорость, с которой люди способны усваивать изменения. Когда-нибудь исчезнет и эта граница, но не потому, что мы решим передать все «детям нашего разума», как называет их Ханс Моравек[119], и тихо уйти в историю. Человечество — не засыхающая веточка на древе жизни. Напротив, мы вот-вот начнем ветвиться.

Таким же образом, как культура эволюционировала вместе с увеличением объема головного мозга, мы будем эволюционировать вместе с нашими творениями. Так было всегда: не изобрети люди огонь и копья, они были бы физически другими. Мы Homo technicus в той же мере, что и Homo sapiens. Но моделирование клетки, которое я описал в последней главе, сделает возможным нечто совершенно новое: ее компьютерную разработку на основе заданных параметров — точно так же компиляторы кремниевых структур разрабатывают микросхемы на основе функциональных спецификаций. Затем спроектированную ДНК можно будет синтезировать и внедрить в «универсальную» клетку, превратив ее в желаемую. Крейг Вентер, пионер исследований генома, уже сделал первые шаги в этом

направлении. Сначала мы будем использовать эту мощь для борьбы с заболеваниями: обнаружение нового патогена и немедленный подбор лекарства, которое иммунная система скачает прямо из интернета. Слова «проблема со здоровьем» станут рудиментом. Затем разработка ДНК позволит людям получить такое тело, какое они хотят: наступит век доступной красоты, как выразился писатель-фантаст Уильям Гибсон. А затем Homo technicus разделится на мириады разумных видов, каждый из которых займет собственную нишу: возникнет целая новая биосфера, отличающаяся от сегодняшней в той же степени, в какой сегодняшняя отличается от первобытного океана.

Многие волнуются, что управляемая человеком эволюция окончательно расколет человечество на генетически имущие и неимущие классы. Удивительно, как убого бывает воображение! Естественная эволюция породила не два вида, один из которых подчиняется другому, а бесконечное разнообразие существ и замысловатые экосистемы. Почему искусственная эволюция — основанная на естественной, но еще менее ограниченная, — должна поступать иначе?

Как и все фазовые переходы, этот в конце концов тоже сойдет на нет. Преодоление узкого горла не равно бесконечному взлету: границей станет следующее узкое горло, даже если пока мы его не видим. Нас ждут новые переходы — некоторые большие, некоторые маленькие, некоторые уже скоро, некоторые в отдаленном будущем. Но следующее тысячелетие вполне может стать самым захватывающим в истории жизни на планете Земля.

ЭПИЛОГ

Итак, теперь вы знакомы с секретами машинного обучения. Механизм, который превращает данные в знание, перестал быть черным ящиком: вы знаете, как происходит волшебство, на что оно способно, а на что — нет. Вы познакомились с монстром сложности, проблемой переобучения, проклятием размерности и дилеммой изучения и применения. Вы в общих чертах знаете, как Google, Facebook, Amazon и другие компании поступают с данными, которые вы дарите щедрым потоком, и почему у них все лучше получается находить то, что вы просите, фильтровать спам и делать многое другое. Вы побывали в лабораториях, где ученые работают над машинным обучением, и теперь вам легче заглянуть в будущее, которое мы помогаем воплотить. В пути вы познакомились с пятью «племенами» машинного обучения и их верховными алгоритмами: символистами и обратной дедукцией, коннекционистами и обратным распространением ошибки, эволюционистами и генетическими алгоритмами, байесовцами и вероятностным выводом, аналогизаторами и методом опорных векторов. А поскольку вы прошли через всю эту обширную страну, посетили пограничные заставы, взбирались на высокие вершины, вы видите ландшафт даже лучше, чем многие специалисты, которые ежедневно занимаются своим участком работ. Вы можете заметить общие темы, похожие на подземные реки, и знаете, почему пять верховных алгоритмов, таких разных на первый взгляд, на самом деле просто пять граней одного универсального алгоритма.

Но путешествие далеко не закончилось. У нас в руках пока не сам Верховный алгоритм, а лишь мысли, предположения, на что он может быть похож. А если нам все еще не хватает чего-то фундаментального, того, что все мы работающие в этой области, увязшие в ее истории — не замечаем? Нам нужны свежие идеи, причем не просто варианты того, что у нас уже есть. Именно поэтому я написал эту книгу: мне хотелось дать толчок вашему воображению. В 2007 году, вскоре после учреждения премии Netflix, я предложил слушателям вечерних курсов по машинному обучению в Вашингтонском университете, где я преподаю, подготовить для

нее проект. Одного из учеников — Джеффа Хоуберта — это зацепило. Он продолжил работу после завершения курсов и вошел в одну из двух команд-победительниц всего через два года после того, как впервые услышал о машинном обучении. Теперь ваша очередь. Чтобы больше узнать о машинном обучении, познакомьтесь с рекомендованной литературой, которую я привожу в конце книги. Скачайте некоторые наборы данных из архива UCI (archive.ics.uci.edu/ml/) и поиграйте с ними. Когда будете готовы, загляните на Kaggle.com — сайт, посвященный соревнованиям по машинному обучению, и примите участие в одном-двух. Конечно, интереснее привлечь к работе товарищей. Если вы втянетесь, как Джефф, и станете профессиональным специалистом по обработке данных, добро пожаловать в самую захватывающую профессию в мире! Изобретайте новые алгоритмы машинного обучения, если вас не устраивают существующие, да и просто ради развлечения. Мое заветное желание — чтобы, прочитав мою книгу, вы отреагировали так же, как я, более двадцати лет назад впервые прочтя книгу по искусственному интеллекту: здесь столько работы, что просто не знаешь, с чего начать! Если когда-нибудь вы изобретете Верховный алгоритм, пожалуйста, не спешите его патентовать. Сделайте его код открытым. Верховный алгоритм слишком важен, чтобы им владел только один человек или организация. Его применения начнут множиться так быстро, что вы не будете успевать их лицензировать. А если вы решите создать стартап, не забудьте предоставить долю в нем каждому человеку, каждому ребенку на Земле.

Неважно, из любопытства или из профессионального интереса вы читали эту книгу: я надеюсь, что вы поделитесь новыми знаниями с друзьями и коллегами. Машинное обучение касается каждого из нас, и всем нам решать, что с ним делать. Теперь вы вооружены пониманием машинного обучения и находитесь в гораздо более выгодной позиции, чтобы размышлять над вопросами частной жизни и коллективного использования данных, трудоустройства в будущем, роботизированных вооружений, а также перспектив и угроз искусственного интеллекта. Чем больше людей будет разбираться в этих вопросах, тем больше вероятность, что мы избежим ловушек и найдем правильный путь — вот еще

одна веская причина, по которой я взялся за эту книгу. Статистики знают, что делать прогнозы сложно, а информатики скажут, что лучший способ предсказать будущее — изобрести его. Но непроверенное будущее не стоит того, чтобы его изобретать.

Спасибо, что взяли меня своим проводником. На прощание у меня есть для вас подарок. Ньютон говорил, что чувствует себя мальчишкой, играющим на берегу: он берет то камушек, то ракушку, а перед ним лежит огромный, неизведанный океан истины. Прошло три столетия, и мы собрали удивительную коллекцию гальки и раковин, но великий неизведанный океан все так же простирается перед нами и играет лучиками надежды. Мой подарок — это лодка машинного обучения, и пришло время поднять паруса.

БЛАГОДАРНОСТИ

Прежде всего я благодарю моих попутчиков в научном приключении: студентов, сотрудников, коллег и всех членов сообщества специалистов по машинному обучению. Эта книга — ваша в той же степени, что и моя. Надеюсь, вы простите мне излишние упрощения и недомолвки, а также немного вычурный стиль некоторых фрагментов.

Я благодарен всем, кто читал и комментировал черновики этой книги на разных этапах ее создания. Это в том числе Майк Бельфьоре, Томас Диттерих, Тьяго Домингос, Орен Эциони, Эйб Фризен, Роб Дженс, Алон Халеви, Дэвид Израэль, Генри Кауц, Хлоя Киддон, Гэри Маркус, Рэй Муни, Кевин Мерфи, Франциска Резнер и Бен Таскар. Спасибо всем тем, кто давал мне подсказки, информацию и помощь любого рода: Тому Гриффитсу, Дэвиду Хекерману, Ханне Хики, Альберту-Ласло Барабаши, Яну Лекуну, Барбаре Моунз, Майку Моргану, Питеру Норвигу, Джуде Перлу, Грегори Пятецкому-Шапиро и Себастьяну Сеунгу.

Я счастлив, что работаю в особом месте — на кафедре информатики и инженерии Вашингтонского университета. Я признателен Джошу Тененбауму и всем его сотрудникам за стажировку в Массачусетском технологическом институте, во время которой я начал работать над этой книгой. Спасибо неутомимому литературному агенту Джиму Левину за твердую веру в мои силы, а также всем сотрудникам Levine Greenberg Rostan. Спасибо Ти-Джею Келлехеру, моему удивительному редактору, который главу за главой, строчку за строчкой делал эту книгу лучше. Спасибо всем сотрудникам Basic Books.

Я признателен организациям, которые на протяжении многих лет финансировали мои исследования: это Научно-исследовательское управление Армии США, Агентство по перспективным оборонным научно-исследовательским разработкам, Фонд науки и технологии, Национальный научный фонд, Управление военно-морских исследований, Ford, Google, IBM, Kodak, Yahoo, а также Фонд Альфреда Слоуна.

Последнее, но не менее важное: спасибо моей семье за любовь и поддержку.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Если моя книга пробудила у вас интерес к машинному обучению и связанным с ним вопросам, в этом разделе вы найдете много советов. Это не исчерпывающий список, но он должен стать, перефразируя Борхеса, калиткой в Сад расходящихся тропок этой дисциплины. Я старался выбирать книги и статьи, подходящие для неспециалиста. Технические публикации, которые требуют хотя бы некоторых познаний в области информатики, статистики или математики, я отметил знаком *. Даже в них, однако, часто есть большие разделы, доступные обычному читателю. Я не указываю номер тома, издания и страниц, потому что в сети и справочниках они не всегда указаны точно.

Если вы хотите узнать больше о машинном обучении в целом, неплохо будет начать с онлайн-курсов. Неудивительно, что ближе всего к содержанию этой книги курс, который веду я сам (www.coursera.org/course/machlearning). Еще два варианта — курсы Эндрю Ына (www.coursera.org/course/ml) и Ясера Абу-Мостафы (work.caltech.edu/telecourse.html). Следующий шаг — взяться за учебники. Один из самых доступных и близких к моей книге — Machine Learning* Тома Митчелла (McGraw-Hill, 1997). Более современные, но более математические — Machine Learning: A Probabilistic Perspective* Кевина Мерфи (MIT Press, 2012), Pattern Recognition and Machine Learning* Криса Бишопа (Springer, 2006) и An Introduction to Statistical Learning with Applications in R* Гарета Джеймса, Даниэлы Виттен, Тревора Хасти и Роба Тибширани (Springer, 2013). Моя статья A few useful things to know about machine learning (Communications of the ACM, 2012) частично суммирует «общеизвестные» истины машинного обучения, которые учебники часто обходят стороной как банальные. Она стала одной из отправных точек этой книги. Если вы умеете программировать и вам не терпится взяться за дело, можете начать с многочисленных открытых пакетов, например Weka (www.cs.waikato.ac.nz/ml/weka). Важнейшие журналы по машинному обучению — Machine Learning и Journal of Machine Learning Research. Ведущие конференции, ежегодно публикующие свои материалы, — International Conference

on Machine Learning, Conference on Neural Information Processing Systems и International Conference on Knowledge Discovery and Data Mining. Множество лекций по машинному обучению вы найдете на сайте videolectures.net. На сайте www.KDnuggets.com также представлено много ресурсов по машинному обучению. Там можно подписаться на рассылку и быть в курсе последних разработок.

Пролог

Примеры влияния машинного обучения на повседневную жизнь приведены в статье Джорджа Джона Behind-the-scenes data mining (SIGKDD Explorations, 1999): она вдохновила меня описать «один день из жизни» в прологе. Много применений машинного обучения рассмотрено в книге Эрика Зигеля Predictive Analytics (Wiley, 2013)[120]. Термин «большие данные» стал популярным после вышедшего в 2011 году отчета McKinsey Global Institute Big Data: The Next Frontier for Innovation, Competition, and Productivity. Много вопросов, которые поднимают большие данные, обсуждается в книге Виктора Майер-Шенбергера и Кеннет Кукьера Big Data: A Revolution That Will Change How We Live, Work, and Think, by Viktor Mayer-Schönberger and Kenneth Cukier (Houghton Mifflin Harcourt, 2013)[121]. Учебник, по которому я сам учился искусственному интеллекту, — это Artificial Intelligence Элен Рич (McGraw-Hill, 1983)*. Более современный вариант — Artificial Intelligence: A Modern Approach Стюарта Расселла и Питера Норвига (третье издание, Prentice Hall, 2010)[122]. В книге Нильса Нильссона The Quest for Artificial Intelligence (Cambridge University Press, 2010) рассказана история создания искусственного интеллекта начиная с самого начала.

Глава 1

В книге Nine Algorithms That Changed the Future Джона Маккормика (Princeton University Press, 2012)[123] описан ряд важнейших алгоритмов, применяемых в информатике. В ней есть и глава о машинном обучении. Algorithms Санджоя Дасгупты, Христоса Пападимитриу и Умеша Вазирани (McGraw-Hill, 2008)[124] — сжатый

вводный учебник по предмету. Джинни Хиллис в книге The Pattern on the Stone (Basic Books, 1998) объясняет, как работают компьютеры. Уолтер Айзексон рассказывает живую историю информатики в книге The Innovators (Simon & Schuster, 2014)[125].

В статье Spreadsheet data manipulation using examples* Сумита Гульвани, Уильяма Харриса и Ришабха Сингха (Communications of the ACM, 2012) показано, как компьютеры могут программировать сами себя, наблюдая за пользователями. Книга Competing on Analytics Тома Дэвенпорта и Джоанн Харрис (HBS Press, 2007)[126] — хорошее введение в применение прогнозной аналитики в бизнесе. Работа In the Plex Стивена Леви (Simon & Schuster, 2011) дает представление о технологиях Google. Карл Шапиро и Хэл Вариан объясняют сетевой эффект в книге Information Rules: A Strategic Guide to the Network Economy (HBS Press, 1999). Феномен длинного хвоста анализирует Крис Андерсон в книге The Long Tail (Hyperion, 2006)[127].

Теме перемен в науке под влиянием вычислений с большими объемами данных посвящена книга The Fourth Paradigm под редакцией Тони Хея, Стюарта Тансли и Кристин Толле (Microsoft Research, 2009). В статье Machine science Джеймса Эванса и Андрея Ржецкого (Science, 2010) обсуждаются некоторые способы научных открытий с помощью компьютеров. В Scientific Discovery: Computational Explorations of the Creative Processes* Пэта Лэнгли и соавторов (MIT Press, 1987) приведен ряд подходов к автоматизации открытия научных законов. Проект SKICAT описан в статье From digitized images to online catalogs Усамы Файяда, Джорджа Джорговского и Николаса Уира (AI Magazine, 1996). Статья Machine learning in drug discovery and development* Ники Уэйла (Drug Development Research, 2001) предлагает обзор по теме открытия и разработки лекарств. Об Адаме, роботе-ученом, можно почитать в статье The automation of science Росса Кинга и соавторов (Science, 2009).

О применении анализа данных в политике подробно рассказывается в книге Саши Иссенберга The Victory Lab (Broadway Books, 2012). Книга How President Obama’s campaign used big data to rally individual votes того же автора (MIT Technology Review, 2013)

дает представление о самом большом на сегодняшний день успехе больших данных — избирательной кампании Барака Обамы.

В книге Нейта Сильвера The Signal and the Noise* (Penguin Press, 2012)[128] есть глава о его методе агрегирования опросов избирателей.

Роботизированное вооружение — тема книги Питера Сингера Wired for War (Penguin, 2009). В книге Cyber War (Ecco, 2012)[129] Ричард Кларк и Роберт Нейк трубят тревогу по поводу кибервойны. Моя собственная работа по соединению машинного обучения и теории игр для победы над противником, начавшаяся как учебный проект, описана в Adversarial classification* Нилеша Далви и соавторов (Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining, 2004). Книга Predictive Policing Уолтера Перри и соавторов (Rand, 2013) познакомит вас с использованием аналитики в работе полиции.

Глава 2

Эксперименты по перепрограммированию мозга хорька описаны в статье Visual behaviour mediated by retinal projections directed to the auditory pathway Лори фон Мельхнер, Сары Паллас и Мриганки Сура (Nature, 2000). История Бена Андервуда рассказана в статье Seeing with sound Джоанны Мурхед (Guardian, 2007) и на сайте www.benunderwood.com. В статье Generality of the functional structure of the neocortex (Naturwissenschaften, 1977) Отто Кройцфельдт утверждает, что кора головного мозга — единый алгоритм. С ним согласен Вернон Маунткасл в главе An organizing principle for cerebral function: The unit model and the distributed system книги The Mindful Brain под редакцией Джералда Эделмена и Вернона Маунткасла (MIT Press, 1978)[130]. Гэри Маркус, Адам Марблстоун и Том Дин возражают против этой теории в статье The atoms of neural computation (Science, 2014).

В работе The unreasonable effectiveness of data Алона Халеви, Питера Норвига и Фернандо Перейры (IEEE Intelligent Systems, 2009) приводятся аргументы в пользу машинного обучения как новой парадигмы научных открытий. Бенуа Мандельброт рассматривает фрактальную геометрию природы в книге The Fractal Geometry of

Nature* (Freeman, 1982)[131]. Книга Джеймса Глейка Chaos* (Viking, 1987)[132] обсуждает и иллюстрирует множества Мандельброта. Программа Langlands, научный проект по объединению разных математических дисциплин, описана в книге Эдварда Френкеля Love and Math (Basic Books, 2014)[133]. The Golden Ticket Лэнса Фортнау (Princeton University Press, 2013) представляет собой введение в NP-полноту и проблему *P* = *NP*. The Annotated Turing[+] Чарльза Петцольда (Wiley, 2008)[134] объясняет машины Тьюринга, анализируя его статью на эту тему.

Проект «Сайк» описан в статье: Cyc: Toward programs with common sense* Дугласа Лената и соавторов (Communications of the ACM, 1990). Питер Норвиг обсуждает критику Ноама Хомского, которой тот подверг статистическое обучение в статье On Chomsky and the two cultures of statistical learning (norvig.com/chomsky.html). Книга Джерри Фодора The Modularity of Mind (MIT Press, 1983) суммирует воззрения автора на принципы работы разума. Статьи What big data will never explain Леона Уисельтира (New Republic, 2013) и Pundits, stop sounding ignorant about data Эндрю Макафи (Harvard Business Review, 2013) дают почувствовать разногласия в отношении возможностей больших данных. Даниэль Канеман объясняет, почему алгоритмы часто побеждают интуицию, в двадцать первой главе книги Thinking, Fast and Slow. Дэвид Паттерсон обосновывает важность вычислений и сбора данных в борьбе с раком в статье Computer scientists may have what it takes to help cure cancer (New York Times, 2011).

Подробнее о путях разных племен к Верховному алгоритму — в соответствующих разделах ниже.

Глава 3

Классическая формулировка Юмом проблемы индукции появляется в первом томе «Трактата о человеческой природе» (1739). Дэвид Уолперт выводит свою теорему «бесплатных обедов не бывает» для индукции в статье The lack of a priori distinctions between learning algorithms* (Neural Computation, 1996). В статье Toward knowledge-rich data mining* (Data Mining and Knowledge Discovery, 2007) я обсуждаю важность априорного знания в машинном обучении,

а в The role of Occam's razor in knowledge discovery* (Data Mining and Knowledge Discovery, 1999) — неправильные интерпретации бритвы Оккама. Переобучение — одна из главных тем уже упоминавшейся книги The Signal and the Noise Нейта Сильвера, который считает ее «самой важной научной проблемой, о которой вы никогда не слышали». В статье Why most published research findings are false* Джона Иоаннидиса (PLoS Medicine, 2005) обсуждается проблема ошибочного принятия случайных научных результатов за истинные. Йоав Беньямини и Йосеф Хохберг предлагают способ борьбы с ней в статье Controlling the false discovery rate: A practical and powerful approach to multiple testing* (Journal of the Royal Statistical Society, Series B, 1995). Дилемма смещения–дисперсии анализируется в статье Neural networks and the bias/variance dilemma Стюарта Джемана, Эли Биненстока и Рене Дурсата (Neural Computation, 1992). В статье Machine learning as an experimental science Пэта Лэнгли (Machine Learning, 1988) обсуждается роль эксперимента в машинном обучении.

Уильям Стэнли Джевонс впервые предложил считать индукцию противоположностью дедукции в книге The Principles of Science (1874). Статья Machine learning of first-order predicates by inverting resolution* Стива Магглтона и Рэя Бантина (Proceedings of the Fifth International Conference on Machine Learning, 1988) положила начало применению обратной дедукции в машинном обучении. Введением в область индуктивного логического программирования может служить книга Relational Data Mining* под редакцией Сашо Джероского и Нады Лаврач (Springer, 2001), В ней также рассматривается обратная дедукция. Статья The CN2 Induction Algorithm* Питера Кларка и Тима Нибletта (Machine Learning, 1989) суммирует ряд важнейших алгоритмов выведения правил в стиле Михальского. Подход к выведению правил, применяемый в торговых сетях, описан в статье Fast algorithms for mining association rules* Ракеша Агарвала и Рамакришнана Шриканта (Proceedings of the Twentieth International Conference on Very Large Databases, 1994). Пример выведения правил для прогнозирования рака можно найти в статье Carcinogenesis predictions using inductive logic programming Ашвина Шринивасана, Росса Кинга, Стивена

Магглтона и Майкла Стернберга (Intelligent Data Analysis in Medicine and Pharmacology, 1997).

Два ведущих обучающих алгоритма, основанных на деревьях решений, представлены в книгах C4.5: Programs for Machine Learning Джона Росса Куинлана (Morgan Kaufmann, 1992) и Classification and Regression Trees* Лео Бреймана, Джерома Фридмана, Ричарда Олшена и Чарльза Стоуна (Chapman and Hall, 1984). В статье Real-time human pose recognition in parts from single depth images* (Communications of the ACM, 2013) Джейми Шоттон и соавторы объясняют принципы использования деревьев решений для отслеживания движений игроков в системе Kinect компании Microsoft. Статья Competing approaches to predicting Supreme Court decision making Эндрю Мартина и соавторов (Perspectives on Politics, 2004) рассказывает, как деревья решений победили экспертов-юристов в прогнозировании результатов голосования в Верховном суде США. Там же приведено дерево решений для судьи Сандры Дэй О'Коннор.

Аллен Ньюэлл и Герберт Саймон сформулировали гипотезу, что весь интеллект сводится к манипулированию символами, в статье Computer science as empirical enquiry: Symbols and search (Communications of the ACM, 1976). Дэвид Марр предложил три уровня обработки информации в книге Vision* (Freeman, 1982)[135]. В книге Machine Learning: An Artificial Intelligence Approach* под редакцией Рышарда Михальского, Джейми Карбонелла и Тома Митчелла (Tioga, 1983) описан ранний период символистских исследований в машинном обучении. Статья Connectionist AI, symbolic AI, and the brain* Пола Смоленского (Artificial Intelligence Review, 1987) представляет коннекционистский подход к символистским моделям.

Глава 4

Книга Себастьяна Сеунга Connectome (Houghton Mifflin Harcourt, 2012)[136] — доступное введение в нейробиологию, коннектомику и пугающую проблему создания головного мозга путем обратного инжиниринга. Книга Parallel Distributed Processing* под редакцией Дэвида Румельхарта, Джеймса Макклелланда и исследовательской

группы параллельной распределенной обработки (MIT Press, 1986) — библия коннекционизма в его зените, пришедшемся на 1980-е. Neurocomputing* под редакцией Джеймса Андерсона и Эдварда Розенфельда (MIT Press, 1988) содержит многие классические коннекционистские статьи, включая статью Маккаллоха и Питса о первых моделях нейронов, Хебба о правиле Хебба, Розенблатта о перцептронах, Хопфилда о сетях Хопфилда, Окли, Хинтона и Сейновского о машинах Больцмана, Сейновского и Розенберга о NETtalk, а также Румельхарта, Хинтона и Уильямса об обратном распространении ошибки. Глава Efficient backprop* Яна Лекуна, Леона Ботту, Женевьевы Орр и Клауса-Роберта Мюллера в книге Neural Networks: Tricks of the Trade под редакцией Женевьевы Орр и Клауса-Роберта Мюллера (Springer, 1998) объясняет некоторые важнейшие трюки, необходимые для корректной работы обратного распространения.

Neural Networks in Finance and Investing* под редакцией Роберта Триппи и Эфраима Турбана (McGraw-Hill, 1992) — сборник статей по применению нейронных сетей в области финансов. Статья Life in the fast lane: The evolution of an adaptive vehicle control system Тодда Йохема и Дина Померло (AI Magazine, 1996) описывает проект создания беспилотного автомобиля ALVINN. Рекомендую также диссертацию Пола Вербоса — Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences* (Harvard University, 1974). Артур Брайсон и Хэ Юци приводят одну из первых разработанных ими версий обратного распространения в книге Applied Optimal Control* (Blaisdell, 1969).

Краткое введение в глубокое обучение — книга Learning Deep Architectures for AI* Йошуа Бенгио (Now, 2009). Проблема распределения сигнала ошибки в обратном распространении описана в статье Learning long-term dependencies with gradient descent is difficult* Йошуа Бенгио, Патрис Симар и Паоло Фраскони (IEEE Transactions on Neural Networks, 1994). В статье How many computers to identify a cat? 16,000 Джона Маркоффа (New York Times, 2012) рассказывается о проекте Google Brain и его результатах. Сверточные нейронные сети, в настоящее время лидирующие в глубоком обучении, описаны в статье Gradient-based learning applied to document recognition* Яна Лекуна, Леона Ботту, Йошуа

Бенгио и Патрика Хаффнера (Proceedings of the IEEE, 1998). Статья The $1.3B quest to build a supercomputer replica of a human brain Джонатона Китса (Wired, 2013) описывает проект по моделированию мозга, запущенный Евросоюзом. Об инициативе BRAIN рассказывается в статье Томаса Инсела, Стори Лэндис и Фрэнсиса Коллинса The NIH BRAIN Initiative (Science, 2013).

Стивен Пинкер подытоживает критику символистами коннекционистских моделей во второй главе книги How the Mind Works (Norton, 1997). Сеймур Паперт берет голос в этих дебатах в статье One AI or Many? (Daedalus, 1988). Книга The Birth of the Mind Гэри Маркуса (Basic Books, 2004) объясняет, как эволюция сумела породить сложные способности человеческого мозга.

Глава 5

Статья Evolutionary robotics Джоша Бонгарда (Communications of the ACM, 2013) дает обзор работ Хода Липсона и других ученых по выведению роботов путем эволюции. Книга Artificial Life Стивена Леви (Vintage, 1993) позволяет прогуляться по цифровому зоопарку, от виртуальных миров с созданными в компьютере животными до генетических алгоритмов. В пятой главе Complexity Митча Уолдропа (Touchstone, 1992) рассказана история Джона Холланда и первых нескольких десятилетий работы над генетическими алгоритмами. Книга Genetic Algorithms in Search, Optimization, and Machine Learning* Дэвида Голдберга (Addison-Wesley, 1989) представляет собой стандартное введение в генетические алгоритмы.

Нильс Элдридж и Стивен Джей Гулд выдвигают свою теорию прерывистого равновесия в главе Punctuated equilibria: An alternative to phyletic gradualism книги Models in Paleobiology под редакцией Томаса Шопфа (Freeman, 1972). Ричард Докинз критикует эту теорию в девятой главе The Blind Watchmaker* (Norton, 1986)[137]. Дилемма изучения–применения обсуждается во второй главе книги Reinforcement Learning* (MIT Press, 1998)[138]. Джон Холланд предлагает свое решение этой проблемы и много других идей в книге Adaptation in Natural and Artificial Systems Джонатона Китса (University of Michigan Press, 1975).

Genetic Programming* Джона Коза (MIT Press, 1992) — ключевая публикация о парадигме генетического программирования. Полученная путем эволюции футбольная команда роботов описана в статье Evolving team Darwin United* Давида Андре и Астро Теллера, а также в книге RoboCup-98: Robot Soccer World Cup II под редакцией Минору Асады и Хироаки Китано (Springer, 1999). В Genetic Programming III* Джона Коза, Форреста Беннетта III, Давида Андре и Мартина Кина (Morgan Kaufmann, 1999) можно найти множество примеров создания электронных плат путем эволюции. Дэнни Хиллис утверждает, что паразиты полезны для эволюции, в статье Co-evolving parasites improve simulated evolution as an optimization procedure* (Physica D, 1990). Ади Ливант, Христос Пападимитриу, Джонатан Дашофф и Маркус Фельдман выдвигают гипотезу, что половое размножение оптимизирует смешиваемость, в статье A mixability theory of the role of sex in evolution* (Proceedings of the National Academy of Sciences, 2008). Кевин Ланг сравнивает генетическое программирование с восхождением на выпуклые поверхности в статье Hill climbing beats genetic search on a Boolean circuit synthesis problem of Koza's* (Proceedings of the Twelfth International Conference on Machine Learning, 1995). Ответ Коза — статья A response to the ML-95 paper entitled...* — не был опубликован, но доступен в интернете на сайте www.genetic-programming.com/jktahoe24page.html.

Джеймс Болдуин предлагает эффект, названный позже его именем, в статье A new factor in evolution (American Naturalist, 1896), а Джефф Хинтон и Стивен Нолан описывают применение этого эффекта в статье How learning can guide evolution* (Complex Systems, 1987). Эффекту Болдуина был посвящен вышедший в 1996 году специальный номер журнала Evolutionary Computation под редакцией Питера Терни, Даррелла Уитли и Расселла Андерсона.

Различие между описательными и нормативными теориями изложил Джон Невилл Кейнс в книге The Scope and Method of Political Economy (Macmillan, 1891).

Глава 6

Шэрон Берч Макгрейн рассказывает историю байесовского учения от Байеса и Лапласа до наших дней в книге The Theory That Would

Not Die (Yale University Press, 2011). Введением в байесовскую статистику может служить учебник First Course in Bayesian Statistical Methods* Питера Хоффа (Springer, 2009).

Наивный байесовский алгоритм впервые упомянут в книге Pattern Classification and Scene Analysis* Ричарда Дуда и Питера Харта (Wiley, 1973). Милтон Фридман приводит аргументы в пользу чрезмерно упрощенных теорий в статье The methodology of positive economics, которая вышла в сборнике Essays in Positive Economics (University of Chicago Press, 1966). Применение наивного байесовского алгоритма для фильтрации спама описано в статье Stopping spam Джошуа Гудмана, Дэвида Хекермана и Роберта Рунтвейта (Scientific American, 2005). Статья Relevance weighting of search terms* Стивена Робертсона и Карена Спарка Джонса (Journal of the American Society for Information Science, 1976) посвящена использованию методов, схожих с наивным байесовским алгоритмом, для поиска информации.

Статья First links in the Markov chain Брайана Хейза (American Scientist, 2013) рассказывает об изобретении Марковым цепей своего имени. Статья Large language models in machine translation Торстена Брантса и соавторов (Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2007) объясняет, как работает Google Translate. В статье The PageRank citation ranking: Bringing order to the Web* Ларри Пейджа, Сергея Брина, Раджива Мотвани и Терри Винограда (Stanford University technical report, 1998) описан алгоритм PageRank и его интерпретация как случайное блуждание по сети. Книга Statistical Language Learning* Юджина Чарняка (MIT Press, 1996) объясняет, как работают скрытые марковские модели, а Statistical Methods for Speech Recognition* Фреда Елинека (MIT Press, 1997) описывает их применение для распознавания речи. Об истории логического вывода в стиле скрытой марковской модели в области коммуникаций рассказывает статья The Viterbi algorithm: A personal history Дэвида Форни (не опубликована, но доступна в интернете по адресу arxiv.org/pdf/cs/0504020v2.pdf). Книга Bioinformatics: The Machine Learning Approach* Пьера Балди и Серена Брунака (второе издание, MIT Press, 2001) — введение в использование машинного обучения,

в том числе в скрытой марковской модели в биологии. Статья Engineers look to Kalman filtering for guidance Барри Ципры (SIAM News, 1993) — краткое введение в фильтры Калмана, их историю и применение.

Работа Джуды Перла о байесовских сетях описана в его книге Probabilistic Reasoning in Intelligent Systems* (Morgan Kaufmann, 1988). Статья Юджина Чарняка Bayesian networks without tears* (AI Magazine, 1991) — во многом нематематическое введение в байесовские сети. Статья Probabilistic interpretation for MYCIN's certainty factors* Дэвида Хекермана (Proceedings of the Second Conference on Uncertainty in Artificial Intelligence, 1986) объясняет, когда наборы правил с оценкой уверенности — разумные приближения байесовских сетей, а когда — нет. Статья Module networks: Identifying regulatory modules and their condition-specific regulators from gene expression data Эрана Сегала и соавторов (Nature Genetics, 2003) — пример использования байесовских сетей для моделирования регуляции генов. В статье Microsoft virus fighter: Spam may be more difficult to stop than HIV Бена Пейнтера (Fast Company, 2012) рассказывается, как Дэвид Хекерман вдохновился спам-фильтрами и использовал байесовские сети для разработки возможной вакцины от СПИДа. Вероятностное, или «зашумленное», ИЛИ объясняется в упомянутой выше книге Перла. В статье Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base M. A. Шве и соавторов (части I и II, Methods of Information in Medicine, 1991) описано применение байесовской сети с зашумленным ИЛИ в медицинской диагностике. Байесовская сеть Google для размещения рекламы описана в разделе 26.5.4 книги Кевина Мерфи Machine Learning* (MIT Press, 2012). Система оценки игроков Microsoft описана в статье TrueSkill™: A Bayesian skill rating system* Ральфа Хербриха, Тома Минки и Тора Грепела (Advances in Neural Information Processing Systems 19, 2007).

Книга Modeling and Reasoning with Bayesian Networks* Аднана Дарвиша (Cambridge University Press, 2009) объясняет важнейшие алгоритмы логического вывода в байесовских сетях. Номер Computing in Science and Engineering* за январь-февраль 2000 года под редакцией Джека Донгарры и Фрэнсиса Салливана содержит статьи о десяти главных алгоритмах XX столетия, в том числе

MCMC. Статья Stanley: The robot that won the DARPA Grand Challenge Себастьяна Труна и соавторов (Journal of Field Robotics, 2006) рассказывает, как работает беспилотный автомобиль Stanley. Статья Bayesian networks for data mining* Дэвида Хекермана (Data Mining and Knowledge Discovery, 1997) подытоживает байесовский подход к обучению и объясняет, как получать байесовские сети на основе данных. Статья Gaussian processes: A replacement for supervised neural networks?* Дэвида Маккея (NIPS tutorial notes, 1997; онлайн www.inference.eng.cam.ac.uk/mackay/gp.pdf) дает почувствовать атмосферу захвата байесовцами конференции NIPS.

Необходимость взвешивать вероятность появления слов при распознавании речи обсуждается в разделе 9.6 книги Speech and Language Processing* Дэна Джурафски и Джеймса Мартина (второе издание, Prentice Hall, 2009). Моя статья о наивном байесовском алгоритме, написанная в соавторстве с Майком Паццани, On the optimality of the simple Bayesian classifier under zero-one loss Джонатона Китса (Machine Learning, 1997) — расширенная журнальная версия статьи, написанной в 1996 году для конференции. В книге Джуды Перла, о которой уже говорилось выше, рассмотрены сети Маркова и байесовские сети. Сети Маркова в компьютерном зрении — тема книги Markov Random Fields for Vision and Image Processing* под редакцией Эндрю Блейка, Пушмита Коли и Карстена Ротера (MIT Press, 2011). Сети Маркова, которые максимизируют условное правдоподобие, были представлены в статье Conditional random fields: Probabilistic models for segmenting and labeling sequence data* Джона Лафферти, Эндрю Маккаллума и Фернандо Перейры (International Conference on Machine Learning, 2001).

История попыток соединить вероятность и логику рассмотрена в специальном издании Journal of Applied Logic*, вышедшем в 2003 году под редакцией Джона Уильямсона и Дова Габбая. В статье From knowledge bases to decision models* Майкла Уэллмана, Джона Бриза и Роберта Голдмана (Knowledge Engineering Review, 1992) обсуждаются некоторые ранние подходы к этой проблеме с применением искусственного интеллекта.

Глава 7

Фрэнк Абигнейл подробно рассказывает о своих подвигах в автобиографии Catch Me If You Can*, написанной в соавторстве со Стэном Реддингом (Grosset & Dunlap, 1980)[139]. Исходный технический отчет об алгоритме ближайшего соседа можно найти в статье Эвелин Фикс и Джо Ходжеса Discriminatory analysis: Nonparametric discrimination: Consistency properties* (USAF School of Aviation Medicine, 1951). В книге Nearest Neighbor (NN) Norms* под редакцией Белура Дасатари (IEEE Computer Society Press, 1991) собраны многие ключевые для этой области статьи. Локально линейная регрессия рассмотрена в статье Locally weighted learning* Криса Аткесона, Эндрю Мура и Стефана Шаала (Artificial Intelligence Review, 1997). Первая система совместной фильтрации, основанная на алгоритме ближайшего соседа, описана в статье GroupLens: An open architecture for collaborative filtering of netnews* Пола Резника и соавторов (Proceedings of the 1994 ACM Conference on Computer-Supported Cooperative Work, 1994). Алгоритм совместной фильтрации Amazon приведен в статье Amazon.com recommendations: Item-to-item collaborative filtering* Грега Линдена, Брента Смита и Джереми Йорка (IEEE Internet Computing, 2003). (О Netflix см. литературу к главе 8.) Вклад рекомендательных систем в продажи Amazon и Netflix можно найти, например, в книге Виктора Майера-Шенбергера и Кеннета Кукьера Big Data[140] или Predictive Analytics Зигеля (см. выше). Также любопытна статья 1967 года Тома Кавера и Питера Харта об уровне ошибки ближайшего соседа — Nearest neighbor pattern classification* (IEEE Transactions on Information Theory).

Проклятие размерности обсуждается в разделе 2.5 книги The Elements of Statistical Learning* Тревора Хасти, Роба Тибширани и Джерри Фридмана (второе издание, Springer, 2009). В статье Wrappers for feature subset selection* Рона Кохави и Джорджа Джона (Artificial Intelligence, 1997) приводится сравнение методов выбора атрибутов. Статья Similarity metric learning for a variable-kernel classifier* Дэвида Лоу (Neural Computation, 1995) — пример алгоритма взвешивания свойств.

Статья Support vector machines and kernel methods: The new generation of learning machines* Нелло Кристианини и Бернхарда Шелькопфа (AI Magazine, 2002) — в целом нематематическое

введение в метод опорных векторов. Революция, произведенная этим методом, началась со статьи A training algorithm for optimal margin classifiers* Бернхарда Босера, Изабель Гуйон и Владимира Вапника (Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 1992). Первой статьей о применении метода опорных векторов к классификации текстов стала Text categorization with support vector machines* Торстена Йоахимса (Proceedings of the Tenth European Conference on Machine Learning, 1998). Глава 5 книги An Introduction to Support Vector Machines* Нелло Кристианини и Джона Шоуи-Тэйлора (Cambridge University Press, 2000) — краткое введение в оптимизацию с ограничениями в контексте метода опорных векторов.

Книга Case-Based Reasoning* Джанет Колоднер (Morgan Kaufmann, 1993) — учебник по рассуждениям на основе прецедентов. В статье Using case-based retrieval for customer technical support* Евангелоса Симудиса (IEEE Expert, 1992) объясняется применение этого метода в службах поддержки. Eliza описана в статье Rise of the software machines* (Economist, 2013) и на сайте компании IPsoft. Кевин Эшли рассматривает рассуждения на основе прецедентов в юриспруденции в своей книге Modeling Legal Arguments* (MIT Press, 1991). Дэвид Коуп подытоживает свой подход к автоматизированному сочинению музыки в статье Recombinant music: Using the computer to explore musical style (IEEE Computer, 1991). Дедре Джентнер предложил картирование структур в статье Structure mapping: A theoretical framework for analogy* (Cognitive Science, 1983). В статье The man who would teach machines to think Джеймса Сомерса (Atlantic, 2013) рассмотрены взгляды Дугласа Хофстадтера на искусственный интеллект.

Алгоритм RISE я описал в статье Unifying instance-based and rule-based induction* (Machine Learning, 1996).

Глава 8

В книге Элисон Гопник, Энди Мельцоффа и Пэта Кула The Scientist in the Crib (Harper, 1999) описаны открытия психологов в области механизмов обучения новорожденных и маленьких детей.

Алгоритм *k*-средних изначально был предложен Стюартом Ллойдом из Bell Labs в 1957 году в техническом отчете под

названием Least squares quantization in PCM* (позже он был издан в виде статьи в IEEE Transactions on Information Theory in 1982). Первая статья о EM-алгоритме — Maximum likelihood from incomplete data via the EM algorithm* Артура Демпстера, Нэн Лэрд и Дональда Рубина (Journal of the Royal Statistical Society B, 1977). Иерархическая кластеризация и другие методы описаны в книге Finding Groups in Data: An Introduction to Cluster Analysis* Леонарда Кауфмана и Питера Руссо (Wiley, 1990).

Метод главных компонент — один из старейших в машинном обучении и статистике. Он был предложен Карлом Пирсоном еще в 1901 году в статье On lines and planes of closest fit to systems of points in space* (Philosophical Magazine). Разновидность уменьшения размерности, используемая при оценке эссе на экзаменах SAT, была введена Скоттом Дирвестером и соавторами в статье Indexing by latent semantic analysis* (Journal of the American Society for Information Science, 1990). Йегуда Корен, Роберт Белл и Крис Волинский объясняют, как работает коллаборативная фильтрация в стиле Netflix, в статье Matrix factorization techniques for recommender systems* (IEEE Computer, 2009). Алгоритм Isomap появился в статье A global geometric framework for nonlinear dimensionality reduction* Джоша Тененбаума, Вина де Сильвы и Джона Лэнгфорда (Science, 2000).

Книга Reinforcement Learning: An Introduction* Рича Саттона и Энди Барто (MIT Press, 1998) — стандартный учебник по обучению с подкреплением. Universal Artificial Intelligence* Маркуса Хаттера (Springer, 2005) — попытка создать общую теорию данного вида обучения. Пионерской работе Артура Сэмюэла по обучению игре в шашки посвящена его статья Some studies in machine learning using the game of checkers* (IBM Journal of Research and Development, 1959). В ней встречается одно из первых упоминаний в печати термина «машинное обучение». Крис Уоткинс сформулировал проблему обучения с подкреплением в своей диссертации Learning from Delayed Rewards* (Cambridge University, 1989). Обучающийся алгоритм с подкреплением DeepMind, применяемый в компьютерных играх, описан в статье Human-level control through deep reinforcement learning* Владимира Мниха и соавторов (Nature, 2015).

Пол Розенблюм рассказывает о развитии алгоритма образования фрагментов в статье A cognitive odyssey: From the power law of practice to a general learning mechanism and beyond (Tutorials in Quantitative Methods for Psychology, 2006). A/B-тестирование и другие методики онлайн-экспериментов объясняются в статье Practical guide to controlled experiments on the Web: Listen to your customers not to the HiPPO* Рона Кохави, Рэндала Хенне и Дэна Зоммерфельда (Proceedings of the Thirteenth International Conference on Knowledge Discovery and Data Mining, 2007). Инкрементное моделирование — многомерное обобщение A/B-тестирования — тема седьмой главы книги Predictive Analytics Эрика Зигеля (Wiley, 2013).

В книге Introduction to Statistical Relational Learning* под редакцией Лизы Гетур и Бена Таскара (MIT Press, 2007) рассмотрены основные подходы в области статистического реляционного обучения. Итоги работы по моделированию сплетен мы с Мэттом Ричардсоном подводим в статье Mining social networks for viral marketing (IEEE Intelligent Systems, 2005).

Глава 9

Введение в метаобучение — тема книги Model Ensembles: Foundations and Algorithms* Чжоу Чжихуа (Chapman and Hall, 2012). Первая статья о стэкинге — Stacked generalization* Дэвида Уолперта (Neural Networks, 1992). Лео Брейман ввел бэггинг в статье Bagging predictors* (Machine Learning, 1996), а случайный лес — в Random forests* (Machine Learning, 2001). Бустинг описан в статье Experiments with a new boosting algorithm Йоава Фройнда и Роба Шапире (Proceedings of the Thirteenth International Conference on Machine Learning, 1996).

В статье I, Algorithm Анила Анантасвами (New Scientist, 2011) можно познакомиться с хроникой поиска объединения логики и вероятности в науке об искусственном интеллекте. В соавторстве с Дэниелом Лоудом я написал введение в логические сети Маркова — книгу Markov Logic: An Interface Layer for Artificial Intelligence* (Morgan & Claypool, 2009). На сайте Alchemy (alchemy.cs.washington.edu) вы найдете руководства, видео, MLN, наборы данных, публикации, указатели на другие системы и еще

много интересного. Логическая сеть Маркова для роботизированного картирования описана в статье Hybrid Markov logic networks* Вана Цзюэ и Педро Домингоса (Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, 2008). Томас Дитрих и Бао Синьлун описывают применение MLN в PAL — одном из проектов DARPA — в статье Integrating multiple learning components through Markov logic* (Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, 2008). Статья Extracting semantic networks from text via relational clustering* Стэнли Кока и Педро Домингоса (Proceedings of the Nineteenth European Conference on Machine Learning, 2008) описывает получение семантических сетей на базе интернета.

Эффективные MLN с иерархией классов и частей описаны в статье Learning and inference in tractable probabilistic knowledge bases* Матиаса Ниперта и Педро Домингоса (Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, 2015). О подходе Google к параллельному градиентному спуску можно прочесть в статье Large-scale distributed deep networks* Джеффа Дина и соавторов (Advances in Neural Information Processing Systems 25, 2012). Статья A general framework for mining massive data streams* Педро Домингоса и Джеффа Халтена (Journal of Computational and Graphical Statistics, 2003) подытоживает предложенный нами метод обучения из незамкнутых потоков данных, основанный на сэмплинге. Проект FuturICT — тема статьи The machine that would predict the future Дэвида Вейнбергера (Scientific American, 2011).

Статья Cancer: The march on malignancy (Nature supplement, 2014) знакомит читателя с текущим состоянием борьбы с раком. Статья Using patient data for personalized cancer treatments Криса Эдвардса (Communications of the ACM, 2014) описывает ранние стадии исследований, которые могут вырасти в CanceRx. Статья Simulating a living cell Маркуса Ковертa (Scientific American, 2014) рассказывает, как его исследовательская группа построила компьютерную модель целой болезнетворной бактерии. Статья Breakthrough Technologies 2015: Internet of DNA Антонио Регаладо (MIT Technology Review, 2015) сообщает о работе Global Alliance for Genomics and Health. Проект Cancer Commons описан в статье Cancer: A Computational

Disease that AI Can Cure Джея Тененбаума и Джеффа Шрейджера (AI Magazine, 2011).

Глава 10

В статье Love, actuarially Кевина Поулсена (Wired, 2014) рассказана история мужчины, который с помощью машинного обучения нашел любовь на сайте знакомств OkCupid. Книга Dataclysm Кристиана Раддера (Crown, 2014) еще глубже рассматривает данные OkCupid и находит в них самые разные идеи. Total Recall Гордона Мура и Джима Геммелла (Dutton, 2009) посвящена последствиям тотальной записи всего, что мы делаем. The Naked Future Патрика Такера (Current, 2014) — обзор использования и злоупотребления данными для прогнозирования. Крейг Манди приводит аргументы в пользу сбалансированного подхода к сбору и использованию данных в статье Privacy pragmatism (Foreign Affairs, 2014). В книге Эрика Бринйольфссона и Эндрю Макафи The Second Machine Age (Norton, 2014) обсуждается, как прогресс в области искусственного интеллекта формирует будущее труда и экономики. Статья World War R Криса Баранюка (New Scientist, 2014) сообщает о дебатах, идущих вокруг боевого применения роботов. Если верить статье Transcending complacency on superintelligent machines Стивена Хокинга и соавторов (Huffington Post, 2014), пришло время беспокоиться о рисках искусственного интеллекта. Ник Бостром в книге Superintelligence (Oxford University Press, 2014)[141] рассматривает эти опасности и задумывается, как с ними справиться.

A Brief History of Life Ричарда Хокинга (Random Penguin, 1982) суммирует квантовые скачки эволюции за миллионы лет до нашей эры. (До эры компьютеров. Шутка.) Книга The Singularity Is Near Рэя Курцвейла (Penguin, 2005) — ваш путеводитель в трансгуманистическое будущее. Джоэл Гарро рассматривает три сценария развития управляемой человеком эволюции в книге Radical Evolution (Broadway Books, 2005). В книге What Technology Wants (Penguin, 2010) Кевин Келли утверждает, что технология — это продолжение эволюции другими средствами. Джордж Дайсон в книге Darwin Among the Machines (Basic Books, 1997) приводит

хронологию развития технологий и выдвигает гипотезы, куда оно может привести. Крейг Вентер объясняет, как его команда синтезировала живую клетку, в книге Life at the Speed of Light (Viking, 2013).

ПРИМЕЧАНИЯ

1. Потоковое мультимедиа в интернете. Пользователь медиапроигрывателя Pandora выбирает музыкального исполнителя, после чего система ищет похожие композиции, используя около 400 музыкальных характеристик. С помощью функций «нравится» или «не нравится» слушатель часто может настроить «радиостанцию» по своему вкусу. *Здесь и далее, если не указано иное, прим. ред.*

2. Действенная архитектура данных для быстрого агрегирования многомерной информации. Куб данных может быть реализован на основе универсальных реляционных баз данных или специализированным программным обеспечением.

3. GMAT (Graduate Management Admission Test) — стандартизованный тест для определения способности успешно обучаться в бизнес-школах.

4. Уиллард Митт Ромни (Willard Mitt Romney, род. 1947) — американский политик. Был кандидатом в президенты США на выборах 2012 года от Республиканской партии.

5. Имеется в виду суперкомпьютер IBM, оснащенный системой искусственного интеллекта, который был создан группой исследователей под руководством Дэвида Феруччи. В феврале 2011 года для проверки возможностей Watson он принял участие в телешоу Jeopardy!. Его соперниками были Брэд Раттер — обладатель самого большого выигрыша в программе, — и Кен Дженнингс — рекордсмен по длительности беспроигрышной серии. Watson одержал победу, получив миллион долларов, в то время как Дженнингс и Раттер получили по 300 и 200 тысяч соответственно.

6. Один из подходов в области искусственного интеллекта, когнитивной науки (когнитивистики), нейробиологии, психологии и философии сознания.

7. Берет свои истоки из теоремы Байеса, одной из основных теорем элементарной теории вероятностей, и названа в честь Томаса

Байеса (1702–1761) — английского математика и священника, который первым предложил использование теоремы для корректировки убеждений, основываясь на обновленных данных.

8. Секвенирование (от лат. sequentum — последовательность) — определение аминокислотной или нуклеотидной последовательности биополимеров (белков и нуклеиновых кислот — ДНК и РНК).

9. Ричард Филлипс Фейнман (Richard Phillips Feynman, 1918–1988) — американский физик, лауреат Нобелевской премии, один из создателей современной квантовой электродинамики, внес существенный вклад в квантовую механику и квантовую теорию поля, его имя носит метод диаграмм Фейнмана.

10. Область исследований, включающая в себя картографирование и анализ архитектуры нейрональных связей.

11. Эдвин Пауэлл Хаббл (Edwin Powell Hubble, 1889–1953) — один из наиболее влиятельных астрономов и космологов XX века, внесший решающий вклад в понимание структуры космоса. Член Национальной академии наук в Вашингтоне с 1927 года.

12. Процесс вырезания определенных нуклеотидных последовательностей из молекул РНК и соединения последовательностей, сохраняющихся в «зрелой» молекуле, в ходе процессинга РНК.

13. Хаб (англ. hub, буквально — ступица колеса, центр) — в общем смысле узел какой-то сети.

14. Натаниель (Нейт) Сильвер (Nathaniel (Nate) Silver, род. 1978) — аналитик, стал известен в 2000-х годах предсказаниями результатов соревнований по бейсболу, а затем и политических выборов.

15. Система французских укреплений длиной около 400 км на границе с Германией от Бельфора до Лонгийона. Была построена в 1929–1934 годах (затем совершенствовалась вплоть до 1940 года). Французские военные стратеги считали линию Мажино неприступной, однако 14 июня 1940 года она была прорвана

за несколько часов в результате наступления германской пехоты даже без танковой поддержки.

16. Военная академия Соединенных Штатов Америки (United States Military Academy), известная также как Вест-Пойнт (West Point) — высшее федеральное военное учебное заведение армии США.

17. Авиационное сражение Второй мировой войны, продолжавшееся с 10 июля по 30 октября 1940 года. Термин «битва за Британию» впервые использовал Уинстон Черчилль, назвав так попытку Третьего рейха завоевать господство в воздухе над югом Англии и подорвать боевой дух британского народа.

18. Единица культурной информации. Мемом может считаться любая идея, символ, манера или образ действия, осознанно или неосознанно передаваемые от человека к человеку посредством речи, письма, видео, ритуалов, жестов и так далее.

19. Средство поддержки принятия решений, использующееся в статистике и анализе данных для прогнозных моделей.

20. Хокинс Дж., Блейксли С. Об интеллекте. М. : Вильямс, 2016.

21. Рэймонд Курцвейл (Raymond Kurzweil, род. 1948) — известный американский изобретатель и футуролог. Создал многочисленные системы для распознавания речи, также известен научными технологическими прогнозами, учитывающими появление искусственного интеллекта и средств радикального продления жизни людей.

22. Курцвейл Р. Эволюция разума. М. : Эксмо, 2015.

23. Классический пример фрактала — математического множества, обладающего свойством самоподобия (объект, в точности или приближенно совпадающий с частью себя самого). Множество Мандельброта — один из самых известных фракталов, в том числе за пределами математики, благодаря своим цветным визуализациям. Его фрагменты не строго подобны исходному множеству, но при многократном увеличении определенные части все больше похожи друг на друга.

24. NP-задачей (недетерминированно полиномиальной задачей) называется задача, у которой за полиномиальное время (то есть при помощи операций, число которых не превышает некоторого полинома, или многочлена, в зависимости от размера исходных данных) можно проверить решение. NP-полная задача — та, к которой за полиномиальное время можно свести решение любой NP-задачи.

25. Алан Мэтисон Тьюринг (Alan Mathison Turing, 1912–1954) — английский математик, логик, криптограф, оказавший существенное влияние на развитие информатики. Предложенная им в 1936 году абстрактная вычислительная «машина Тьюринга», которую можно считать моделью компьютера общего назначения, позволила формализовать понятие алгоритма и до сих пор используется во множестве теоретических и практических исследований.

26. «Сайк» (англ. Cyc) — проект по созданию объемной онтологической базы знаний, позволяющей программам решать сложные задачи из области искусственного интеллекта на основе логического вывода и привлечения здравого смысла.

27. Ноам Хомский (Avram Noam Chomsky, род. 1928) — американский лингвист, политический публицист, философ и теоретик. Институтский профессор лингвистики Массачусетского технологического института, автор классификации формальных языков, называемой иерархией Хомского.

28. Джим Хорнинг (Jim Horning, 1942–2013) — американский ученый в области информатики.

29. Джерри Алан Фодор (Jerry Alan Fodor, род. 1935) — американский философ и психолингвист-экспериментатор. Автор многих работ по философии сознания и когнитивной науке, где среди прочего отразил основные идеи о модулярности сознания и гипотезу о языке мысли «ментализ». Один из наиболее влиятельных философов сознания конца XX — начала XXI века. Оказал значительное влияние на развитие когнитивной науки.

30. Нассим Николас Талеб (Nassim Nicolas Taleb, род. 1960) — американский экономист и трейдер. Основная сфера научных интересов — изучение влияния случайных и непредсказуемых событий на мировую экономику и биржевую торговлю, а также механизмы торговли производными финансовыми инструментами.

31. Талеб Н. Черный лебедь. М. : Колибри : Азбука-Аттикус, 2015.

32. Льюис М. MoneyBall. Как математика изменила самую популярную спортивную лигу в мире. М. : Манн, Иванов и Фербер, 2013.

33. От англ. highest paid person's opinion.

34. Саймон Г. Науки об искусственном. М. : Едиториал УРСС, 2004.

35. Сэр Исайя Берлин (Isaiah Berlin, 1909–1997) — английский философ, переводчик, один из основателей современной либеральной политической философии.

36. Теория струн — направление теоретической физики, изучающее динамику взаимодействия не точечных частиц, а одномерных протяженных объектов, так называемых квантовых струн. Претендует на роль всеобъемлющей универсальной теории, объясняющей природу всего сущего.

37. Стивен Уильям Хокинг (Stephen William Hawking, род. 1942) — английский физик-теоретик и популяризатор науки. Изучал теорию возникновения мира в результате Большого взрыва, а также высказал гипотезу, что маленькие черные дыры теряют энергию, испуская излучение Хокинга, и в конце концов «испаряются». В 2016 году Хокинг доказал, что черные дыры не безвозвратно поглощают информацию — часть ее просачивается наружу в виде «мягких волос» — фотонов с почти нулевой энергией.

38. Дизъюнкция (лат. disjunctio — разобщение) — логическая операция, по своему применению максимально приближенная к союзу «или» в смысле «или то, или это, или оба сразу». Синонимы:

логическое «ИЛИ», включающее «ИЛИ», логическое сложение, иногда просто «ИЛИ».

39. Математический аппарат для моделирования динамических дискретных систем. Впервые описаны Карлом Петри в 1962 году.

40. Набор клеток, образующих некую периодическую решетку с заданными правилами перехода, определяющими состояние клетки в следующий момент через состояние клеток, находящихся от нее на расстоянии не больше некоторого, в текущий момент времени.

41. Крез (первая половина VI века до н. э.) — царь Лидии, рабовладельческого торгового государства, который, как пишет древнегреческий историк Геродот в своей «Истории», был обладателем несметных сокровищ.

42. Сергей Брин (Sergey Brin, род. 1973) и Ларри Пейдж (Lawrence «Larry» Page, род. 1973) — разработчики и основатели Google; Марк Эллиот Цукерберг (Mark Elliot Zuckerberg, род. 1984) — один из разработчиков и основателей социальной сети Facebook.

43. Теорема «Бесплатных обедов не бывает» (No free lunch) гласит: не существует алгоритма, позволяющего получить оптимальные решения всех возможных задач. Теорема получила название на основе метафоры о стоимости блюд в различных ресторанах. Допустим, существует определенное количество ресторанов (каждый из них обозначает определенный алгоритм прогнозирования), где в меню различным блюдам (каждое блюдо обозначает определенную задачу прогнозирования) сопоставлена цена (или качество решения этой задачи, которое позволяет получить рассматриваемый алгоритм). Человек, который любит поесть и при этом не прочь сэкономить, может определить, какой ресторан предлагает его любимое блюдо по самой выгодной цене. Вегетарианец, сопровождающий этого обжору, наверняка обнаружит, что его любимое вегетарианское блюдо в этом ресторане стоит намного дороже. Если обжора захочет полакомиться бифштексом, он выберет ресторан, где бифштекс подают по самой низкой цене. Но его друг-вегетарианец при этом

вынужден будет заказать единственное вегетарианское блюдо в этом ресторане, пусть даже по заоблачной цене. Это очень точная метафора ситуации, когда необходимость использования определенного алгоритма для решения конкретной задачи приводит к гарантированно неоптимальным результатам.

44. Бертран Артур Уильям Рассел (Bertrand Arthur William Russell, 1872–1970) — британский философ, общественный деятель и математик.

45. Синдром саванта, иногда сокращенно называемый савантизм (от фр. savant — «ученый»), — редкое состояние, при котором лица с отклонением в развитии (в том числе аутистического спектра) имеют «остров гениальности» — выдающиеся способности в одной или нескольких областях знаний, контрастирующие с общей ограниченностью личности.

46. Речь идет о книге Майкла Дроснина: Дроснин М. Библейский код. М. : Вагриус, 2000.

47. Имеется в виду так называемый Розуэлльский инцидент — предполагаемое крушение неопознанного летающего объекта около города Розуэлл в штате Нью-Мексико в июле 1947 года.

48. Методологический метод, получивший название от имени английского монаха-францисканца, философа-номиналиста Уильяма Оккама. В кратком виде он гласит: «Не следует множить сущее без необходимости». Бритву Оккама также называют законом экономии мышления.

49. Уильям Стэнли Джевонс (William Stanley Jevons, 1835–1882) — английский экономист, статистик и философ-логик.

50. Классическая игра, в которую играют с XIX века. Один человек задумывает объект, а у другого человека есть 20 попыток его отгадать.

51. Информационная энтропия — мера неопределенности или непредсказуемости информации.

52. Эрл Хант (Earl Hunt, род. 1933) — американский психолог, специализирующийся на исследовании искусственного интеллекта.

53. Нейрит (длинный цилиндрический отросток нервной клетки), по которому передаются исходящие сигналы (нервные импульсы) от тела клетки к иннервируемым органам и другим нервным клеткам.

54. Уильям Джеймс (William James, 1842–1910) — американский философ и психолог, один из основателей и ведущий представитель прагматизма и функционализма.

55. Окрашивание по методу Гольджи — техника окрашивания нервной ткани, открытая итальянским физиологом Камилло Гольджи в 1873 году. Самим Гольджи метод был назван «черной реакцией».

56. Дихотомически ветвящийся отросток нервной клетки, воспринимающий сигналы от других нейронов, рецепторных клеток или непосредственно от внешних раздражителей. Проводит нервные импульсы к телу нейрона.

57. Уоррен Маккаллок (Warren McCulloch, 1898–1969) — американский нейропсихолог, нейрофизиолог, теоретик искусственных нейронных сетей и один из отцов кибернетики.

58. Уолтер Питтс (Walter Pitts, 1923–1969) — американский нейролингвист, логик и математик XX века.

59. Фрэнк Розенблатт (Frank Rosenblatt, 1928–1969) — известный американский ученый в области психологии, нейрофизиологии и искусственного интеллекта.

60. Сеймур Пейперт (Seymour Papert, род. 1928) — выдающийся математик, программист, психолог и педагог. Один из основоположников теории искусственного интеллекта, создатель языка программирования Logo (1968).

61. Минский М., Пейперт С. Персептроны. М. : Мир, 1971.

62. Джон Джозеф Хопфилд (John Joseph Hopfield, род. 1933) — американский ученый, в основном известный как изобретатель

ассоциативной нейронной сети, которая носит его имя.

63. Спин — собственный момент импульса элементарных частиц, имеющий квантовую природу и не связанный с перемещением частицы как целого.

64. Имеется в виду бестселлер канадского журналиста и поп-социолога Малкольма Гладуэлла «Переломный момент: Как незначительные изменения приводят к глобальным переменам», где автор показывает, почему одни идеи, товары или типы поведения стремительно распространяются, а другие — нет и как можно вызывать подобные «эпидемии» и управлять ими. Книга переведена на 12 языков, включая русский (М. : Альпина Паблишер, 2015).

65. Йозеф Алоиз Шумпетер (Joseph Alois Schumpeter, 1883–1950) — австрийский и американский экономист, политолог, социолог и историк экономической мысли.

66. Рональд Уильямс (Ronald Williams) — профессор информатики Северо-Восточного университета в Бостоне, один из пионеров нейронных сетей.

67. Соревнования автомобилей-роботов, финансируемые правительством США. Цель этих соревнований — создание полностью автономных транспортных средств.

68. Американская актриса, наиболее известная как исполнительница роли Рэйчел Грин в телевизионном сериале «Друзья», за которую она была удостоена премий «Эмми» и «Золотой глобус».

69. Джеймс Уотсон (James Watson, род. 1928) — американский биолог, совместно с Фрэнсисом Криком и Морисом Х. Ф. Уилкинсом лауреат Нобелевской премии по физиологии и медицине 1962 года за открытие структуры молекулы ДНК; Фрэнсис Крик (Francis Crick, 1916–2004) — британский молекулярный биолог, биофизик и нейробиолог.

70. In silico — термин, обозначающий компьютерное моделирование (симуляцию) эксперимента, чаще биологического.

Создан по аналогии с терминами in vivo (в живом организме) и in vitro (в пробирке), которые часто используются в биологии. Термин по написанию близок к латинскому выражению in silicio — «в кремнии», поскольку кремний как полупроводниковый материал играет важную роль в производстве компьютерного оборудования.

71. Джон Генри Холланд (John Henry Holland, 1929–2015) — американский ученый, профессор психологии, профессор электротехники и информатики в Мичиганском университете.

72. Фишер Р. С. Регулярная и хаотическая динамика. Ижевск : Институт компьютерных исследований, 2011.

73. Артур Бёркс (Arthur Burks, 1915–2008) — математик, один из конструкторов первого электронного компьютера общего назначения, внес большой вклад в теоретическую часть информатики.

74. Джон фон Нейман (John von Neumann, 1903–1957) — венгеро-американский математик, сделавший важный вклад в квантовую физику, квантовую логику, функциональный анализ, теорию множеств, информатику, экономику и другие отрасли науки.

75. Нильс Элдридж (Niles Eldredge, род. 1943) — американский палеонтолог; Стивен Джей Гулд (Stephen Jay Gould, 1941–2002) — также американский палеонтолог, биолог-эволюционист и историк науки. Один из наиболее знаменитых и читаемых писателей научно-популярного жанра.

76. Кембрийский взрыв — внезапное (в геологическом смысле) появление в ранне-кембрийских (около 540 миллионов лет назад) отложениях окаменелостей представителей многих подразделений животного царства, на фоне отсутствия их окаменелостей или окаменелостей их предков в докембрийских отложениях.

77. «Автостопом по галактике» (The Hitchhiker’s Guide to the Galaxy) — фантастический юмористический роман Дугласа Адамса.

78. Джон Коза (John Koza) — программист, в прошлом преподаватель-консультант в Стэнфордском университете.

Наиболее известные его работы посвящены генетическому программированию для оптимизации сложных проблем.

79. Эмпирический тест, идея которого была предложена Тьюрингом с целью определить, может ли машина мыслить. Стандартная интерпретация этого теста звучит следующим образом: «Человек взаимодействует с одним компьютером и одним человеком. На основании ответов на вопросы он должен определить, с кем разговаривает: с человеком или компьютерной программой. Задача компьютерной программы — ввести человека в заблуждение, заставив сделать неверный выбор».

80. Черная Королева — одна из героинь книги Л. Кэрролла «Алиса в Зазеркалье». В оригинале Кэрролл называл черные фигуры красными (Red Queen), поскольку в шахматных наборах тех лет цвет фигур был действительно близок к красному.

81. Дэнни Хиллис (William Hillis, род. 1956) — американский изобретатель, инженер и математик.

82. Христос Пападимитриу (Christos Papadimitriou, род. 1949) — греческий и американский ученый в области информатики, профессор Калифорнийского университета в Беркли.

83. Конрад Захариас Лоренц (Konrad Zacharias Lorenz, 1903–1989) — австрийский зоолог и зоопсихолог, один из основоположников этологии — науки о поведении животных, лауреат Нобелевской премии по физиологии и медицине (1973, совместно с Карлом фон Фришем и Николасом Тинбергеном).

84. Джеймс Марк Болдуин (James Mark Baldwin, 1861–1934) — американский психолог, философ, социолог. Один из основателей психологии личности и социальной психологии в США.

85. Канеман Д. Думай медленно… решай быстро. М. : АСТ, 2013.

86. Вавилонская рыбка (Babel fish, англ.) — вымышленное существо из «Автостопом по галактике» Дугласа Адамса. Помещается в ухо носителя, или «хозяина». Вавилонская рыбка питается энергией биотоков мозга, впитывая непонятные своему носителю частоты внешних биотоков и испражняя в его мозг телепатическую

матрицу, составленную из частот сознательных мыслей и нервных сигналов речевого центра мозга. Таким образом, с вавилонской рыбкой в ухе человек может понимать любую осмысленную речь.

87. Мехран Сахами (Mehran Sahami) — профессор информатики Стэнфордского университета.

88. Фильтр Калмана — самый популярный алгоритм фильтрации, используемый во многих областях науки и техники. Благодаря своей простоте и эффективности его можно встретить в GPS-приемниках, обработчиках показаний датчиков, при реализации систем управления и т. д. Назван в честь Рудольфа Калмана, инженера и исследователя в области теории управления.

89. Спектакль, поставленный Mercury Theatre on the Air под руководством Орсона Уэллса 30 октября 1938 года, транслировался в эфире станции CBS как реальный новостной репортаж. В результате более миллиона жителей северо-востока США поверили в нападение марсиан и ударились в панику. Целые семьи баррикадировались с оружием в подвалах своих домов либо спешно собирали вещи, чтобы уехать на запад; многие требовали, чтобы вооруженные силы страны покончили с пришельцами либо раздали оружие всем желающим. Телефоны в тот вечер были перегружены в пять раз, пробки из Нью-Йорка, Трентона и Филадельфии растянулись почти на 100 километров.

90. Язык, используемый как средство межэтнического общения в определенной сфере деятельности. На постсоветском пространстве в качестве лингва франка выступает русский язык. Лингва франка в европейском бизнесе, науке и авиации сегодня — английский.

91. Граф в информатике — это совокупность непустого множества объектов и связей между ними. Объекты — вершины, или узлы графа, а связи — дуги, или ребра.

92. Такман Б. Августовские пушки. М. : Астрель, 2012.

93. Ричард Эрнст Беллман (Richard Ernest Bellman, 1920–1984) — американский математик, один из ведущих специалистов в области

математики и вычислительной техники. Ассистент математики в Принстонском университете.

94. Владимир Вапник (род. 1936) — советский и американский математик, внес важный вклад в теорию машинного обучения, разработав теорию восстановления зависимостей по эмпирическим данным.

95. Bell Labs — бывшая американская, а ныне франко-американская корпорация, крупный исследовательский центр в области телекоммуникаций, электронных и компьютерных систем.

96. Хофштадтер Д. Гёдель, Эшер, Бах: эта бесконечная гирлянда. Самара : Бахрах-М, 2001.

97. Имеется в виду персонаж одноименного фильма режиссера Рона Ховарда. Гринч, подвергавшийся насмешкам из-за своей внешности, решает отомстить обидчикам — жителям Ктограда, и лишить их любимого праздника — Рождества.

98. В произведениях Дж. Р. Р. Толкина гигантские ворота, запирающие с северо-запада единственный широкий путь в Мордор — королевство Темного Властелина Саурона, землю страха и тьмы.

99. Понятие, введенное для понимания современного общества, в котором благодаря развитию электронных коммуникаций расстояние потеряло значение.

100. «Запретная планета» (Forbidden Planet) — фильм режиссера Фреда Уилкокса, снятый в 1956 году и ставший классикой жанра кинофантастики.

101. Эдвард Ли Торндайк (Edward Lee Thorndike, 1874–1949) — американский психолог и педагог. Президент Американской психологической ассоциации (1912–1939).

102. Ричард Эрнст Беллман (Richard Ernest Bellman, 1920–1984) — американский математик, один из ведущих специалистов в области математики и вычислительной техники. Работал в Принстонском университете (1946–1948).

103. Дахигг Ч. Сила привычки. Почему мы живем и работаем так, а не иначе. М. : Карьера Пресс, 2015.

104. Аллен Ньюэлл (Allen Newell, 1927–1992) — американский математик и программист, один из первых разработчиков искусственного интеллекта; Пол Розенблюм (Paul S. Rosenbloom) — профессор информатики в Университете Южной Калифорнии.

105. Герберт Александер Саймон (Herbert A. Simon, 1916–2001) — выдающийся американский ученый в области социальных, политических и экономических наук, один из разработчиков гипотезы Ньюэлла–Саймона.

106. Джон Лэрд (John Laird, род. 1954) — профессор информатики в Мичиганском университете, с 1994 по 1999 год возглавлял лабораторию искусственного интеллекта.

107. Генри Луис Менкен (Henry Louis Mencken, 1880–1956) — американский журналист, эссеист, сатирик.

108. Имеется в виду научно-фантастический боевик, снятый режиссерами Вачовски. В мире матрицы красная таблетка — путь к тайнам, секретам, суперсиле.

109. Оно, иногда Ид (лат. id, англ. it, нем. das Es) — в психоанализе одна из структур, описанных Фрейдом. Являет собой бессознательную часть психики, совокупность инстинктивных влечений.

110. Имеется в виду фантастический триллер Стивена Спилберга по мотивам одноименной повести Филипа Киндреда Дика. В основе сюжета — история о том, как на основе психических технологий была разработана экспериментальная программа, позволявшая узнать о еще не совершенном убийстве и арестовать подозреваемого еще до совершения им преступления.

111. Первый полнометражный фильм, смоделированный на компьютере полностью трехмерным.

112. Докинз Р. Расширенный фенотип. Длинная рука гена. М. : Астрель : Corpus, 2010.

113. Ирвинг Джон Гуд (Irving John Good, 1916–2009) — британский математик, работавший вместе с Аланом Тьюрингом в качестве криптографа в Главном шифровальном подразделении Великобритании — Правительственной школе кодов и шифров; после окончания Второй мировой войны Гуд продолжил работать с Тьюрингом над дизайном компьютеров и байесовой статистикой.

114. Портативная шифровальная машина, использовавшаяся для шифрования и дешифрования секретных сообщений.

115. Вернор Стеффан Виндж (Vernor Steffen Vinge, род. 1944) — математик и писатель-фантаст, лауреат премии «Хьюго».

116. Трилобиты — вымерший класс морских членистоногих, имевший большое значение для фауны палеозойских образований земного шара.

117. Эукариоты, или ядерные, — домен живых организмов, клетки которых содержат ядро. Прокариоты — доядерные организмы, клетки которых не имеют ограниченных мембраной ядер.

118. Плейстоцен — эпоха четвертичного периода, начавшаяся 2588 миллионов лет назад и закончившаяся 11,7 тысячи лет назад.

119. Ханс Моравек (Hans Moravec, род. 1948) — преподаватель Института робототехники при Университете Карнеги–Меллон, известен своими работами в области робототехники, искусственного интеллекта и писательской деятельностью на тему влияния технологий.

120. Зигель Э. Просчитать будущее. Кто кликнет, купит, соврет или умрет. М. : Альпина Паблишер, 2014.

121. Майер-Шенбергер В., Кукьер К. Большие данные: революция, которая изменит то, как мы живем, работаем и мыслим. М. : Манн, Иванов и Фербер, 2013.

122. Рассел С., Норвег П. Искусственный интеллект: современный подход. М. : Вильямс, 2000.

123. Маккормик Дж. Девять алгоритмов, которые изменили будущее. М. : ДМК Пресс, 2014.

124. Дасгупта С., Пападимитриу Х., Вазирани У. Алгоритмы. М. : Издательство МЦНМО, 2014.

125. Айзексон У. Инноваторы. М. : АСТ : Corpus, 2015.

126. Дэвенпорт Т., Харрис Дж. Аналитика как конкурентное преимущество. Новая наука побеждать. М. : BestBusinessBooks, 2010.

127. Андерсон К. Длинный хвост. Новая модель ведения бизнеса. М. : Вершина, 2008.

128. Сильвер Н. Сигнал и шум. Почему одни прогнозы сбываются, а другие — нет. М. : КоЛибри : Азбука-Аттикус, 2015.

129. Кларк Р., Нейк Р. Третья мировая война. Какой она будет? СПб. : Питер, 2011.

130. Разумный мозг / под ред. Д. Эделмена и В. Маунткасла. М. : Мир, 1981.

131. Мандельброт Б. Фрактальная геометрия природы. Ижевск : Институт компьютерных исследований, 2002.

132. Глейк Дж. Хаос: создание новой науки. СПб. : Амфора, 2001.

133. Френкель Э. Любовь и математика. Сердце скрытой реальности. СПб. : Питер, 2016.

134. Петцольд Ч. Читаем Тьюринга. Путешествие по исторической статье Тьюринга о вычислимости и машинах Тьюринга. М. : ДМК-Пресс, 2014.

135. Марр Д. Зрение. Информационный подход к изучению представления и обработки зрительных образов. М. : Радио и связь, 1987.

136. Сеунг С. Коннектом. Как мозг делает нас тем, что мы есть. М. : Бином. Лаборатория знаний, 2014.

137. Докинз Р. Слепой часовщик. Как эволюция доказывает отсутствие замысла во Вселенной. М. : АСТ : Corpus, 2015.

138. Саттон Р., Барто Э. Обучение с подкреплением. М. : Бином, Лаборатория знаний, 2012.

139. Абигнейл Ф., Реддинг С. Поймай меня, если сможешь. М. : Эт Сетера Паблишинг, 2003.

140. Майер-Шенбергер В., Кукьер К. Большие данные. Революция, которая изменит то, как мы живем, работаем и мыслим. М. : Манн, Иванов и Фербер, 2013.

141. Бостром Н. Искусственный интеллект. Этапы. Угрозы. Стратегии. М. : Манн, Иванов и Фербер, 2015.

ОГЛАВЛЕНИЕ

Пролог

Глава 1. Революция машинного обучения
- Познакомимся с обучающимся алгоритмом
- Почему бизнес рад машинному обучению?
- Турбоускорение для научного метода
- Миллиард Клинтонов
- Один сигнал, если сушей, два — если по интернету
- Куда мы идем?

Глава 2. Властелин алгоритмов
- Аргумент из области нейробиологии
- Аргумент из области эволюции
- Аргумент из области физики
- Аргумент из области статистики
- Аргумент из области информатики
- Алгоритмы машинного обучения против инженерии знаний
- Лебедь кусает робота
- Верховный алгоритм — лиса или еж?
- Что на кону?
- Другая теория всего
- Кандидаты, которые не оправдали надежд
- Пять «племен» машинного обучения

Глава 3. Проблема индукции Юма
- Быть или не быть свиданию?
- Теорема «Бесплатных обедов не бывает»
- Подготовка насоса знаний
- Как править миром
- Между слепотой и галлюцинациями
- Точность, которой можно доверять
- Индукция — противоположность дедукции
- Как научиться лечить рак
- Игра в двадцать вопросов50
- Символисты

Глава 4. Как учится наш мозг?
- Взлет и падение перцептрона
- Физик делает мозг из стекла
- Самая важная кривая в мире
- Альпинизм в гиперпространстве
- Перцептроны наносят ответный удар
- Полная модель клетки
- В глубинах мозга

Глава 5. Эволюция: обучающийся алгоритм природы
- Алгоритм Дарвина
- Дилемма изучения–применения
- Выживание самых приспособленных программ
- Зачем нужен секс?
- Воспитание природы
- Побеждает тот, кто быстрее учится

Глава 6. В святилище преподобного Байеса
- Теорема, которая правит миром
- Все модели неверны, но некоторые полезны
- От «Евгения Онегина» до Siri
- Все связано, но не напрямую
- Проблема логического вывода
- Учимся по-байесовски
- Марков взвешивает доказательства
- Логика и вероятность: несчастная любовь

Глава 7. Ты — то, на что ты похож
- Попробуй подобрать мне пару
- Проклятие размерности
- Змеи на плоскости
- Вверх по лестнице
- Взойди и сияй

Глава 8. Обучение без учителя
- Как свести рыбака с рыбаком
- Открытие формы данных
- Жизнелюбивый робот
- Повторенье — мать ученья

- Как найти соотношения

Глава 9. Кусочки мозаики встают на место
- Из многих моделей — одна
- Верховный алгоритм
- Логические сети Маркова
- От Юма до домашних роботов
- Машинное обучение в планетарном масштабе
- Сейчас вас примет доктор

Глава 10. Мир машинного обучения
- Секс, ложь и машинное обучение
- Цифровое зеркало
- Общество моделей
- Делиться или не делиться, а если да, то где и как
- Нейронная сеть украла у меня работу
- Война — не для людей
- Google + Верховный алгоритм = Skynet?
- Эволюция, часть вторая

Эпилог

Благодарности

Рекомендуемая литература

МАКСИМАЛЬНО ПОЛЕЗНЫЕ КНИГИ

Если у вас есть замечания и комментарии к содержанию, переводу, редактуре и корректуре, то просим написать на be_better@m-i-f.ru, так мы быстрее сможем исправить недочеты.

Наши электронные книги:
http://www.mann-ivanov-ferber.ru/ebooks/

Заходите в гости:
http://www.mann-ivanov-ferber.ru/
http://blog.mann-ivanov-ferber.ru/
http://www.facebook.com/mifbooks
http://vk.com/mifbooks
https://twitter.com/mifbooks

Дерево знаний
Предложите нам книгу
Ищем правильных коллег

Для корпоративных клиентов:
Полезные книги в подарок
Корпоративная библиотека
Книги ищут поддержку

НАД КНИГОЙ РАБОТАЛИ

Главный редактор *Артем Степанов*
Ответственный редактор *Татьяна Медведева*
Литературный редактор *Юлия Слуцкина*
Арт-директор *Алексей Богомолов*
Обложка *Facultative.Works*
Верстка *Вячеслав Лукьяненко*
Корректоры *Надежда Болотина, Юлия Молокова*

ООО «Манн, Иванов и Фербер»
mann-ivanov-ferber.ru

Электронная версия книги подготовлена компанией
Webkniga.ru, 2016