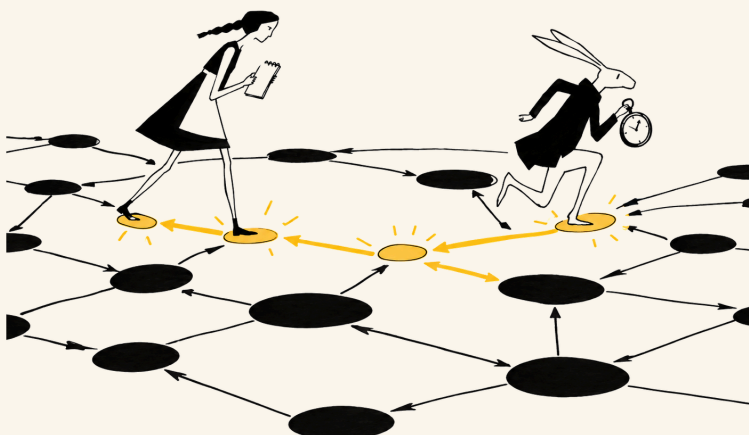


Приключения Алисы в Стране дифференцируемых чудес



Simone Scardapane · 2024

Иллюстрированные аудиолекции

Lecturized

Приключения Алисы в Стране дифференцируемых чудес

Иллюстрированные аудиолекции

Simone Scardapane · 2024

Пересказано в иллюстрированных лекциях Lecturized

lecturized.com

По оригиналу: Alice's Adventures in a Differentiable Wonderland, Volume I: A Tour of
the Land

Содержание

Один простой рецепт, увеличенный в масштабе

Стрелки, которые узнают друг друга

Как растут затраты и где здесь вверх

Спуск с горы в тумане

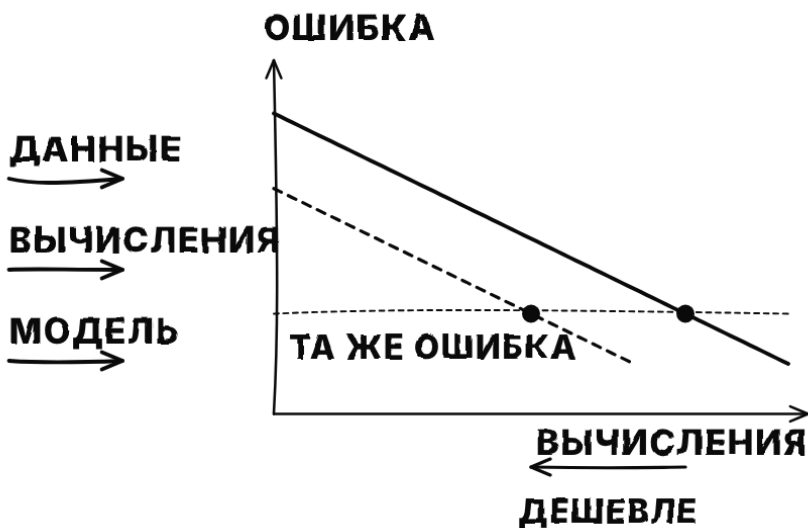
Момент, и откуда берутся примеры

Счёт, который нужно понизить, и тест, которого вы не видели

Много возможных моделей, и самая простая

Один простой рецепт, увеличенный в масштабе

Откройте чат-ассистент на телефоне и спросите его о чём-то, чего он наверняка никогда не видел слово в слово, скажем, лимерик про кошку вашего соседа и налоговую декларацию. Вы, наверное, ждёте неловкой мозаики из предложений, откуда-то скопированных. А приходит нечто новое, уместное и часто смешное. Что-то внутри этой программы выучило куда больше, чем список готовых ответов, и книга открывается вопросом: как такое вообще строится? Ответ неожиданный: из очень небольшого набора принципов и деталей, применяемых снова и снова, а затем раздутых до огромных размеров.



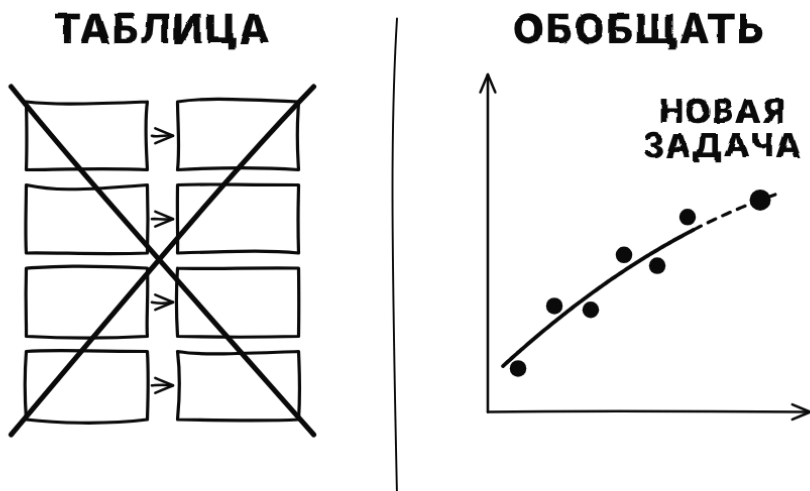
ГОРЬКИЙ УРОК

Масштаб снижает ошибку

Книга называет это силой масштабирования. Исследователи обнаружили, что почти для любой задачи добавление сразу большего объёма данных, большей вычислительной мощности и более крупной модели повышает точность предсказуемым образом. Если сказать наоборот, вычисления, нужные для достижения заданной точности, сокращаются год от года. Учёный Ричард Саттон назвал это горьким уроком: простые универсальные инструменты плюс всё больше вычислений побеждают хитрые частные приёмы.

Вот загадка, которая делает это почти волшебством. Эти сети настолько велики, что могли бы просто запомнить всё, что им показывают, и стать гигантской справочной таблицей. Вместо

этого обученные модели обобщают: они справляются с задачами, которых в обучающих данных не было в явном виде, и чем больше данных, тем меньше они опираются на чистое запоминание. Именно поэтому ваш ассистент вообще способен написать лимерик про кошку.

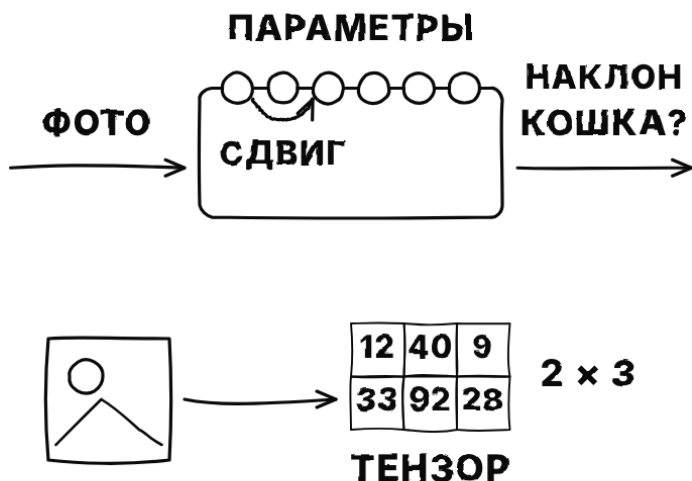


Запомнить или обобщить

Книга также отмечает, как изменился сам наш способ пользоваться этими моделями. Огромные предобученные модели, называемые фундаментальными моделями, можно направлять, просто подавая им запрос текстом или картинками, а их внутренности при этом остаются чёрным ящиком. Лишь несколько команд в мире способны построить самые крупные из них, подобно тому как лишь несколько компаний способны

создавать огромные бизнес-системы. И всё же на обычном оборудовании и с некоторой изобретательностью вы можете дообучать, объединять, сжимать или тестировать такие модели, а для этого нужно заглянуть под капот.

Взгляд под капот начинается с одной идеи: обучение — это поиск. Вы пишете программу с огромным числом регулируемых ручек, называемых параметрами, и крутите их, пока программа не начнёт вести себя хорошо. Фотоприложение, отличающее кошек от собак, обучают именно так: показывают ему тысячи размеченных картинок и после каждой ошибки чуть подкручивают ручки. Чтобы искать среди миллионов или миллиардов ручек, книга ограничивается деталями, эффект которых можно измерить производными, а они говорят, что произойдёт, если ручку слегка повернуть. Поэтому она предпочитает название дифференцируемые модели вместо нейронных сетей и поэтому строит всё из автоматических способов вычисления производных и шагов вниз по склону.



Обучение — это поиск

Дифференцируемые модели не могут напрямую работать с буквами или целыми числами, которые скачут, а не меняются плавно, поэтому все данные превращаются в большие массивы чисел, которые книга называет тензорами. Изображение, звук или предложение становятся такими массивами, и каждая часть модели классифицируется по тому, какой формы массив она принимает и выдаёт. Дальше книга проходит по свёрткам для изображений, звука и текста, вниманию и трансформерам, графам и рекуррентным моделям.

Ни одна из этих идей не нова. Обучение простых моделей градиентным спуском восходит к девятнадцатому веку, а в 1958 году психолог Фрэнк Розенблатт уже вызывал газетную шумиху

своими перцептронами. Изменились данные и мощности: конкурсы по распознаванию изображений с 2012 по 2017 год и появление трансформеров в 2017 году показали, как хорошо эти старые идеи работают в масштабе. Здесь опущены план книги по главам и её обозначения для тензоров и их форм; они находятся в разделах «Об этой книге» и «Обозначения и символы».

Запомним

Почти всё в современном ИИ построено из нескольких простых принципов и деталей, ставших мощными благодаря совместному масштабированию данных, вычислений и размера модели. Большие модели не просто запоминают; они обобщают на задачи, которых им никогда не показывали, и это то волшебство, которое книга хочет объяснить. Обучение — это поиск: программу с огромным числом регулируемых параметров настраивают, пока она не начнёт вести себя хорошо. Поиск работает потому, что каждая деталь дифференцируема, так что производные подсказывают, как подкрутить каждый параметр, и поэтому книга говорит о дифференцируемых моделях. Все данные, от фотографий до предложений, обрабатываются как большие массивы чисел, называемые тензорами.

Вопросы

1. Что, согласно законам масштабирования, обычно происходит, когда данные, вычислительную мощность и

размер модели увеличивают одновременно?

- a. Точность растёт предсказуемым образом почти для любой задачи
 - b. Точность растёт только для задач с изображениями, но не для языка
 - c. Модель начинает запоминать и хуже справляется с новыми задачами
 - d. Ничего не меняется, пока устройство модели не переделают вручную
2. Почему удивительно, что большие обученные модели справляются с задачами, которых им никогда не показывали?
- a. Потому что они слишком малы, чтобы удержать хоть какие-то примеры
 - b. Потому что они достаточно велики, чтобы просто запомнить обучающие данные, как справочная таблица, но вместо этого обобщают
 - c. Потому что их обучали только на числах и никогда на тексте или картинках
 - d. Потому что после обучения никто не может подать им запрос
3. Почему книга строит свои модели только из дифференцируемых деталей?
- a. Потому что дифференцируемые детали дешевле хранить в памяти
 - b. Потому что только дифференцируемым деталям можно подать текстовый запрос

- c. Потому что производные подсказывают, как подкрутить каждый из миллионов параметров, и это делает поиск хороших настроек осуществимым
 - d. Потому что дифференцируемым деталям для обучения вообще не нужны данные
- 4. Почему предложение или фотографию нужно превратить в большие массивы чисел, прежде чем дифференцируемая модель сможет с ними работать?
 - a. Потому что буквы и целые числа скачут, а не меняются плавно, так что производные не могут работать с ними напрямую
 - b. Потому что массивы чисел занимают меньше места, чем буквы
 - c. Потому что обозначения книги запрещают буквы
 - d. Потому что только фотографии можно хранить в виде букв

Ответы

1. **a)** Исследователи обнаружили, что добавление сразу большего объёма данных, большей вычислительной мощности и более крупной модели повышает точность предсказуемым образом почти для любой задачи.
2. **b)** Сети настолько велики, что могли бы запомнить всё, что видят, и стать гигантской справочной таблицей; вместо этого они обобщают и с ростом данных всё меньше опираются на запоминание.
3. **c)** Обучение — это поиск среди огромного числа параметров, а производные говорят, что произойдёт, если ручку слегка повернуть, и так автоматически направляют поиск.

4. а) Дифференцируемые модели не могут напрямую работать с буквами или целыми числами, которые скачут, а не меняются плавно, поэтому все данные становятся массивами чисел, называемыми тензорами.

Задачи

1. Представьте фотоприложение, которое учится отличать кошек от собак. Объясните своими словами, что значит утверждение «обучение — это поиск» и что в этом примере является параметрами.

Подсказка: Представьте приложение как машину с огромной панелью ручек и спросите себя, что делает с ними обучение.

Ответ: Приложение — это программа с огромным числом регулируемых ручек, её параметров. При обучении ему показывают размеченные картинки и после каждой ошибки чуть поворачивают ручки, ища среди всех возможных положений ручек то, при котором приложение хорошо разделяет кошек и собак.

2. Идеи, лежащие в основе сегодняшнего ИИ, такие как градиентный спуск и перцептрон, стары. Объясните, как это делает данная часть, почему они стали настолько мощными лишь за последние пятнадцать лет или около того.

Подсказка: Спросите себя, чего не хватало в 1958 году и чего в избытке сегодня.

Ответ: Методы были известны давно, но не хватало данных и вычислительной мощности, чтобы показать, насколько хорошо они работают. Крупные конкурсы по распознаванию изображений с 2012 года и трансформеры с 2017 года показали, что при достаточном объёме данных и параметров эти старые идеи работают замечательно.

3. Многие сейчас пользуются фундаментальной моделью только через запросы, как чёрным ящиком. Объясните, зачем кому-то всё же может понадобиться понимать, что происходит у неё внутри.

Подсказка: Подумайте, что можно сделать с открытой программой, помимо того чтобы просто её запускать.

Ответ: Для использования модели запросов достаточно, но чтобы её приспособить, нужно больше: дообучить под свою задачу, объединить модели, сжать их для слабого оборудования или проверить их устойчивость — всё это требует заглянуть под капот и понять, как модель обрабатывает данные внутри.

Стрелки, которые узнают друг друга

Возьмите два карандаша и положите их на стол так, чтобы оба начинались в одной точке. Направьте их в одну сторону, затем под прямым углом, затем в противоположные стороны и каждый раз спрашивайте себя, насколько похожи эти два направления. Ответ чувствуется без всякой арифметики: в одну сторону — очень похожи, под прямым углом — никак не связаны, в противоположные — непохожи настолько, насколько возможно. Это простое ощущение «похоже или непохоже» оказывается главным инструментом главы о математике, и современный ИИ пользуется им миллионы раз в секунду.

СКАЛЯРНОЕ



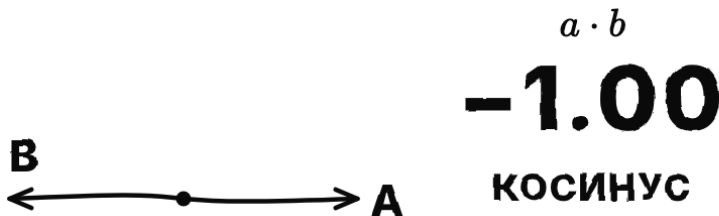
Одно направление, прямой угол, противоположные

Прежде чем перейти к математике, книга говорит, откуда она взялась и чем она не является. Она выросла из курса по нейронным сетям, который автор читает в магистерской программе по науке о данных в Римском университете Сапиенца, и стремится добавить собственный взгляд к другим хорошим книгам по этой теме. Она подчёркивает, что теория и код идут рука об руку и что бум нейронных сетей примерно с 2012 года во многом обязан программным библиотекам, таким как TensorFlow, PyTorch и JAX. Это не руководство по программированию, она не пытается охватить всё и мало говорит об оборудовании или о ремесле заставить огромные модели работать на практике. Кроме того, она выпущена под свободной лицензией, которая позволяет любому переделывать её и строить

на её основе, а именно это и делает возможным курс вроде нашего.

Затем глава о математике задаёт язык для всего, что последует. Как мы видели, все данные живут в тензорах, массивах чисел, описываемых своей формой, и они хорошо подходят графическим чипам, которые выполняют много простых вычислений параллельно. Одна строка чисел — это вектор, который можно представить как стрелку из начала координат, как ваши карандаши. Две стрелки можно растягивать и складывать, получая третью, и у каждой стрелки есть длина.

$$a \cdot b = a_1 b_1 + a_2 b_2$$



Скалярное произведение как шкала

Ключевая операция — скалярное произведение: перемножить два вектора число за числом и сложить результаты, что даёт одно число. Это число велико, когда стрелки смотрят в одну сторону, равно нулю, когда они под прямым углом, и отрицательно, когда они смотрят в противоположные стороны, ровно как показали ваши карандаши. Для стрелок длиной один оно измеряет в точности, насколько похожи их направления, поэтому его называют косинусным сходством. Книга выводит самый важный урок: скалярное произведение максимально, когда вторая стрелка совпадает с первой. В следующих главах одной стрелкой будет вход, а другой — настраиваемые параметры модели, так что модель откликается сильнее всего, когда вход резонирует с тем, что выучили её параметры, своего рода сопоставление с шаблоном. Когда ваше фотоприложение находит все снимки, похожие на вашу собаку, именно такое сравнение измеряет, насколько похожи две картинки.

Опыт

Возьмите два карандаша и держите их с общей начальной точкой. Медленно поворачивайте один из них от направления вдоль другого до противоположного направления. Вы должны почувствовать, как совпадение падает от полного, через ничто под прямым углом, до полного несовпадения в конце, и тот же путь проходит скалярное произведение.

Таблицу чисел, матрицу, можно читать как стопку таких векторов, по одному на строку, и книга называет это батчем данных. Умножение матриц тогда не более чем вычисление

многих скалярных произведений сразу, каждой строки одной матрицы с каждым столбцом другой, за одну операцию. Когда чат-ассистент обрабатывает сразу целый батч слов, он делает именно такое массовое сравнение. Есть и более простое, поэлементное произведение, используемое для маскирования и масштабирования, а функции вроде экспоненты обычно применяются к каждому числу таблицы отдельно, что отличается от особых матричных версий из учебников. Книга предупреждает о распространённой ловушке в коде: библиотеки незаметно растягивают массивы несовпадающих форм, чтобы те подошли друг к другу, и два вектора могут случайно превратиться в целую таблицу.

БАТЧ

1	2	0
0	1	3

×

1	0
1	1
0	2

=

3	2
1	7

--	--	--

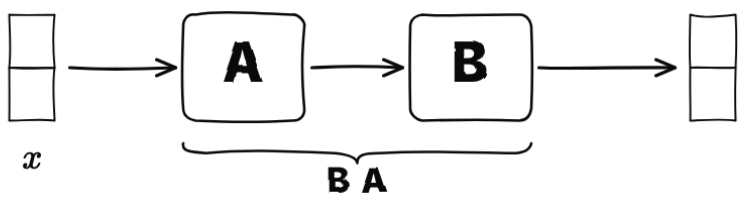
+

=

BROADCAST!

Много скалярных произведений сразу

Наконец, книга объясняет, почему умножение матриц определено именно так. Каждую функцию, которая уважает сложение и растяжение, называемую линейной, можно записать как матрицу, так что умножить матрицу на вектор значит применить функцию, а умножить две матрицы значит применить одну функцию за другой. Здесь опущены точные формулы для длин, скалярных произведений, матричных произведений и индексации; они находятся в разделах «Основные операции с векторами» и «Основные операции с матрицами».



СТОПКА СТРОК

Матрицы как линейные функции

Запомним

Книга выросла из университетского курса и считает теорию и код партнёрами, держа в уме фреймворки вроде PyTorch и JAX, но оставляя в стороне оборудование и инженерию.

Данные живут в тензорах; вектор — это стрелка, а матрица — стопка векторов, часто батч примеров. Скалярное произведение превращает две стрелки в одно число, которое говорит, насколько они похожи, и оно максимально, когда они совпадают, так модель и узнаёт выученные ею образы. Умножение матриц — это много скалярных произведений сразу, и это же способ сцеплять линейные функции одну за другой.

Вопросы

1. Две стрелки длиной один направлены под прямым углом друг к другу. Что говорит их скалярное произведение?
 - a. Что они похожи настолько, насколько возможно
 - b. Что они не связаны: скалярное произведение равно нулю
 - c. Что они непохожи настолько, насколько возможно
 - d. Что одна из них должна быть длиннее другой
2. Почему модель откликается сильнее всего, когда вход похож на то, что выучили её параметры?

- a. Потому что скалярное произведение входа и параметров максимально, когда они совпадают
 - b. Потому что параметры — всегда более длинные стрелки, чем входы
 - c. Потому что модель копирует вход в свои параметры
 - d. Потому что матрицы можно умножать только на совпадающие матрицы
3. Что на самом деле вычисляет умножение двух матриц?
- a. Длины всех строк одну за другой
 - b. Каждое число одной таблицы, умноженное на число на том же месте в другой
 - c. Много скалярных произведений сразу, каждой строки первой матрицы с каждым столбцом второй
 - d. Экспоненту каждого числа в таблице
4. Если смотреть на линейные функции как на матрицы, как это делает книга, чему соответствует умножение двух матриц?
- a. Сложению двух функций
 - b. Применению одной функции за другой
 - c. Отмене первой функции второй
 - d. Применению обеих функций к каждому числу отдельно

Ответы

1. b) Скалярное произведение велико, когда стрелки смотрят в одну сторону, равно нулю под прямым углом и отрицательно, когда они смотрят в противоположные стороны.

2. а) Скалярное произведение максимально, когда вторая стрелка совпадает с первой, так что когда вход резонирует с параметрами, отклик достигает пика: сопоставление с шаблоном.

3. с) Умножение матриц — не более чем много скалярных произведений за одну операцию, каждой строки одной таблицы с каждым столбцом другой.

4. б) Матрица, умноженная на вектор, применяет функцию, а матрица, умноженная на матрицу, применяет одну функцию за другой.

Разобранные примеры

Скалярное произведение вручную Найдите скалярное произведение двух векторов $0.1, 0, -0.3$ и $-4.0, 0.05, 0.1$ — пример, который разбирает сама книга.

Перемножьте числа на одинаковых местах: 0.1 умножить на -4.0 даёт -0.4 .

Следующая пара: 0 умножить на 0.05 даёт 0 .

Последняя пара: -0.3 умножить на 0.1 даёт -0.03 .

Сложите три результата: $-0.4 + 0 - 0.03 = -0.43$, одно число; оно отрицательно, значит, стрелки направлены скорее в разные стороны, чем в одну.

Ответ: -0.43

Задачи

1. Вычислите скалярное произведение векторов $1, 2$ и $3, 4$. Затем вычислите его для $1, 0$ и $0, 1$ и скажите, что второй ответ говорит вам о двух стрелках.

Подсказка: Перемножьте числа на одинаковых местах и сложите результаты.

Ответ: $1 \times 3 + 2 \times 4 = 11$. Для 1, 0 и 0, 1 результат равен 0, так что две стрелки находятся под прямым углом: направления не связаны.

2. Книга предупреждает, что библиотеки незаметно растягивают массивы несовпадающих форм. В её примере столбец из 4 чисел складывается с обычным списком из 4 чисел. Какая форма получается на выходе и почему это ловушка?

Подсказка: Подумайте, что произойдёт, если каждое число столбца объединить с каждым числом списка.

Ответ: Получается таблица 4 на 4 вместо списка из 4 чисел, потому что библиотека растягивает оба массива, чтобы они подошли друг к другу. Ничего не ломается, поэтому ошибка может остаться незамеченной.

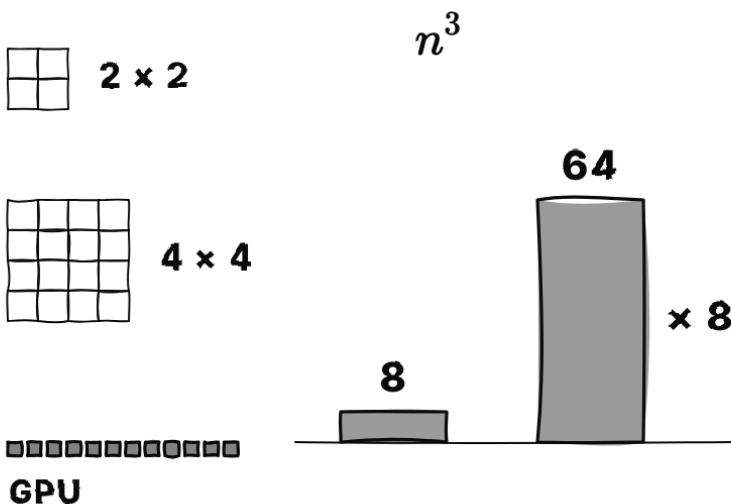
3. Объясните своими словами, почему фотоприложение могло бы использовать скалярные произведения, чтобы находить снимки, похожие на вашу собаку.

Подсказка: Представьте, что каждый снимок превращён в стрелку из чисел.

Ответ: Если каждый снимок превратить в стрелку из чисел, похожие снимки дадут стрелки, направленные в близкие стороны. Их скалярное произведение тогда велико, так что приложение может ранжировать снимки по тому, насколько велико их скалярное произведение со стрелкой вашей собаки.

Как растут затраты и где здесь вверх

Перемножьте вручную две маленькие таблицы чисел, по две строки на два столбца каждая: вам понадобится восемь умножений. Теперь удвойте размер до четырёх на четыре, и вы, возможно, ждёте шестнадцати. Посчитайте, и вы найдёте шестьдесят четыре, в восемь раз больше, потому что удвоение каждой стороны задачи умножает работу трижды. Этот сюрприз — тот путь, которым книга подходит к вопросу, который задаёт каждый инженер ИИ: как растёт стоимость операции с ростом данных?

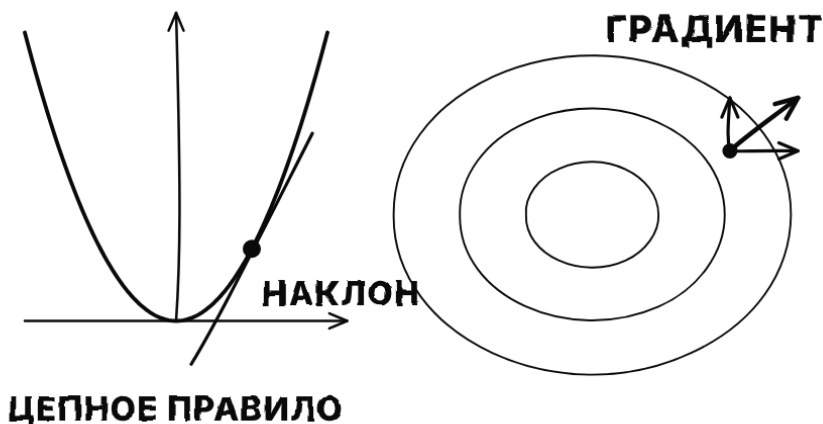


Удвоить размер — в восемь раз больше работы

Книга отделяет время, число шагов, от памяти, которая нужна, и описывает их рост сокращённой записью, принятой у специалистов по информатике, которая оставляет только то, что доминирует при больших размерах. По этой мере обычное умножение матриц растёт как куб размера, и это тот восьмикратный скачок, который вы только что насчитали. Существуют более хитрые методы с более медленным ростом, но их редко используют, потому что их трудно распределить между тысячами маленьких работников в графическом чипе. Много работников лишь делят время на фиксированное число, а перемещение данных может стоить дороже самой арифметики, поэтому на практике измеряют реальный код, а не доверяют одной лишь формуле. Вот почему обучение модели, стоящей за чат-ассистентом, планируют вокруг памяти и чипов не меньше, чем вокруг математики.

Помимо таблиц, книге нужны лишь два расширения: одно и то же матричное произведение сразу для целого батча таблиц и скалярное произведение для массивов любой формы. Она также показывает необязательную сокращённую запись `einsum`, где короткая строка букв называет, какие оси объединяются, а какие суммируются, так что одна и та же строка работает в разных библиотеках.

Затем идёт сердце главы — производная. Для функции одного числа производная — это наклон касательной в точке: она говорит, растёт функция или падает, если шагнуть немного вправо. Несколько правил позволяют находить производные сумм, произведений и, что важнее всего, одной функции, применённой после другой, где наклоны просто перемножаются; это цепное правило вынесет на себе большую часть книги.

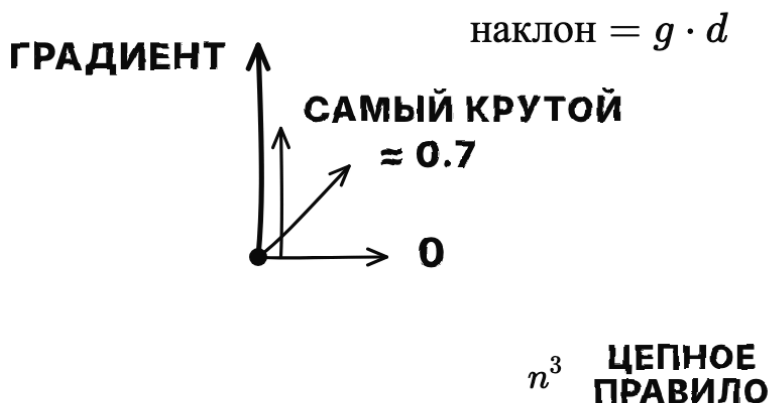


Наклон, затем градиент

Функция многих чисел похожа на ландшафт, и единого наклона нет, потому что шагнуть можно в бесконечно многих направлениях. Шаг вдоль одной оси за раз даёт по одному наклону на каждое число, называемому частной производной, а если сложить их все в стопку, получится градиент. Ключевой результат книги в том, что наклон в любом другом направлении — это просто скалярное произведение градиента с этим направлением, так что знать градиент значит знать сразу все наклоны. Когда фотоприложение обучают, градиент говорит для каждого из его миллионов параметров, как изменится ошибка, если этот параметр подкрутить.

Опыт

Встаньте на наклонную лужайку или пандус и сделайте по одному маленькому шагу в нескольких направлениях: прямо вверх, вбок, по диагонали. Вы должны почувствовать самый крутой подъём в одном направлении, никакого подъёма под прямым углом к нему и частичные подъёмы между ними. Градиент — это стрелка того самого крутого направления, а наклон любого другого шага следует из того, насколько точно он выстраивается вдоль этой стрелки.



Все наклоны из одной стрелки

Здесь опущены формальное определение записи роста, синтаксис `einsum` и точные правила дифференцирования; они находятся в разделах «Вычислительная сложность», «Нотация Эйнштейна» и «Производные скалярных функций».

Запомним

О стоимости операции судят по тому, как она растёт с размером; обычное умножение матриц растёт как куб размера, так что удвоение таблиц даёт в восемь раз больше работы. Реальная скорость зависит также от оборудования, от параллельной работы и от перемещения данных, поэтому код измеряют, а не только анализируют. Производная — это наклон функции в точке, а цепное правило перемножает наклоны, когда одна функция следует за другой. Градиент собирает наклон вдоль каждой оси, а наклон в любом направлении — это его скалярное произведение с градиентом, так что один градиент говорит, как функция меняется во всех направлениях.

Вопросы

1. Почему удвоение размера двух квадратных таблиц делает обычное умножение матриц примерно в восемь раз медленнее?
 - а. Потому что работа растёт как куб размера, а два в кубе — это восемь

- b. Потому что каждое число приходится копировать восемь раз
 - c. Потому что графический чип замедляется на больших таблицах
 - d. Потому что таблица результата становится в восемь раз шире
2. Почему более хитрые методы умножения с более медленным ростом редко используются на практике?
- a. Потому что они дают неверные ответы для больших таблиц
 - b. Потому что их трудно распределить между множеством маленьких работников графического чипа
 - c. Потому что им нужны таблицы особой формы
 - d. Потому что их ещё никто не записал
3. У функции многих чисел есть наклон в бесконечно многих направлениях. Чего достаточно, чтобы знать их все?
- a. Только наклона вдоль первой оси
 - b. Значения функции во многих случайных точках
 - c. Градиента: наклон в любом направлении — это его скалярное произведение с этим направлением
 - d. Второй производной функции
4. На наклонной лужайке в каком направлении маленький шаг не даёт никакого подъёма?
- a. Прямо вдоль самого крутого направления
 - b. Под прямым углом к самому крутому направлению
 - c. Ровно противоположно самому крутому направлению

d. По диагонали между вверх и вбок

Ответы

1. a) Обычное умножение матриц растёт как куб размера, так что удвоение каждой стороны умножает работу на два трижды: восемь.
2. b) Методы с более медленным ростом существуют, но их трудно распределить между тысячами маленьких работников в графическом чипе, поэтому в реальном применении они проигрывают.
3. c) Наклон в любом направлении — это скалярное произведение градиента с этим направлением, так что знать градиент значит знать сразу все наклоны.
4. b) Градиент указывает вдоль самого крутого подъёма, а шаг под прямым углом к нему никак с ним не выстраивается, поэтому его наклон равен нулю.

Разобранные примеры

Подсчёт работы матричного произведения Таблица из 3 строк и 4 столбцов умножается на таблицу из 4 строк и 5 столбцов. Сколько отдельных умножений нужно обычному методу?

У результата 3 строки и 5 столбцов, так что нужно вычислить 15 чисел.

Каждое из них — скалярное произведение строки длиной 4 на столбец длиной 4, что требует 4 умножений.

Итого 15 умножить на 4, то есть 60, произведение трёх размеров 3, 4 и 5.

Ответ: 60 умножений

Задачи

1. В примере кода из книги батч из 4 таблиц, каждая 5 на 2, умножается на батч из 4 таблиц, каждая 2 на 3. Какова форма результата?

Подсказка: Каждая из 4 пар умножается отдельно, как обычные таблицы.

Ответ: Батч из 4 таблиц, каждая 5 на 3: в каждой паре 5 на 2 умножить на 2 на 3 даёт 5 на 3.

2. Книга строит график функции x в квадрате минус $1.5x$. Её наклон в точке x равен $2x$ минус 1.5 . Функция растёт или падает при $x = 1$ и при $x = 0$?

Подсказка: Положительный наклон означает рост вправо, отрицательный — падение.

Ответ: При $x = 1$ наклон равен $2 - 1.5 = 0.5$, положительный, так что функция растёт; при $x = 0$ наклон равен -1.5 , отрицательный, так что она падает.

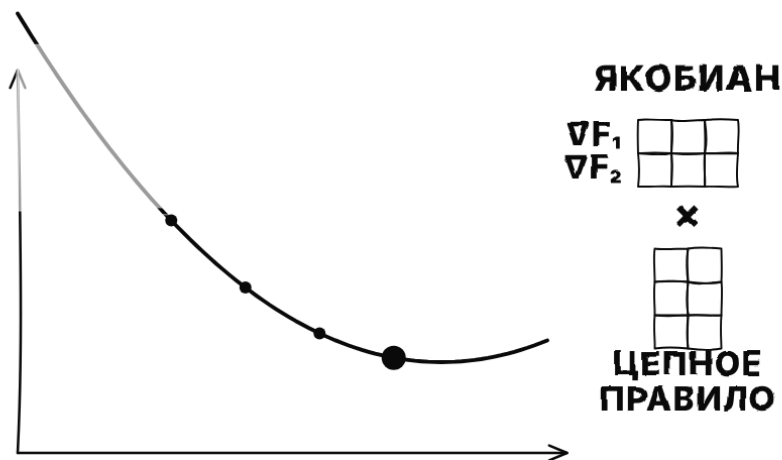
3. В некоторой точке функция двух чисел имеет градиент $2, -1$. Каков наклон, если шагнуть в направлении $1, 1$?

Подсказка: Возьмите скалярное произведение градиента с направлением.

Ответ: $2 \times 1 + (-1) \times 1 = 1$, так что в этом направлении функция растёт с наклоном 1.

Спуск с горы в тумане

Представьте, что вы стоите на склоне холма в густом тумане, не видите ничего дальше собственных ног и хотите добраться до самой низкой точки долины. Долины вы не видите, но чувствуете, куда наклонена земля под вашими подошвами. Поэтому вы делаете маленький шаг в ту сторону, где она падает круче всего, снова прислушиваетесь к ощущениям и снова шагаете. На удивление часто этот слепой рецепт приводит вас вниз, и именно так обучают почти каждую модель ИИ.



Чувствуя наклон в тумане

Сначала книга ещё раз расширяет идею наклона. Когда функция возвращает несколько чисел вместо одного, у каждого выхода есть собственный градиент, и если сложить их в стопку, получится таблица, называемая якобианом. Цепное правило переносится: для одной функции, применённой после другой, их якобианы просто перемножаются, как таблицы. Как и наклон одиночной кривой, якобиан — это лучшее прямолинейное предположение о том, как функция ведёт себя вблизи точки. Книга добавляет предупреждение на будущее: для простейшего слоя модели эти таблицы могут быть огромными и полными повторяющихся чисел, поэтому на практике их никогда не хранят целиком, и этот трюк объясняет глава об автоматическом дифференцировании.

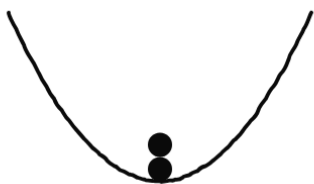
Теперь туман. Параметры модели — это ваше положение, а число, измеряющее, насколько плохо модель работает, — ваша высота; обучение означает найти положение, где это число наименьшее. За редкими исключениями нет формулы, которая прыгает прямо к ответу, поэтому вы двигаетесь шаг за шагом, и у каждого шага есть длина и направление. В машинном обучении длина называется скоростью обучения. Любое направление, ведущее вниз, считается направлением спуска, а поскольку наклон в направлении — это его скалярное произведение с градиентом, самый крутой путь вниз — ровно противоположно градиенту. Шагать снова и снова против градиента, достаточно маленькими шагами, — это градиентный спуск. Когда обучают чат-ассистент, каждый из его бесчётных шагов обучения — это один такой маленький шаг вниз сразу для всех его параметров.



Шаг против градиента

Где заканчивается прогулка? Она останавливается там, где земля плоская, в стационарной точке, но плоская земля может быть дном долины, вершиной холма или седлом вроде горного перевала, который падает в одну сторону и поднимается в другую. В общем случае градиентный спуск обещает лишь достичь какой-нибудь плоской точки, а какой именно, зависит от того, откуда вы стартовали. Для ландшафта в форме чаши, называемого выпуклым, каждый склон ведёт к одной и той же низшей точке, так что глобальный минимум достигается откуда угодно. Найти наверняка низшую точку бугристого ландшафта означало бы стартовать из каждого возможного места, что безнадежно.

ВЫПУКЛАЯ



ЛОТОК ДЛЯ ЯИЦ



Чаша против лотка для яиц

Опыт

Бросьте шарик в круглую салатницу с любой стороны: он всегда окажется в одной и той же точке дна. Теперь бросьте его на лоток для яиц: он уляжется в ту ямку, которая ближе всего к месту падения. Салатница — это выпуклая задача; лоток для яиц — это то, как выглядит обучение настоящих моделей.

Книга заканчивает поворотом: каждая полезная нейронная сеть — это лоток для яиц, а не чаша, в отличие от некоторых старых методов обучения, которые держали свои задачи выпуклыми. И

всё же, стартовав из разумного места, градиентный спуск на этих сетях приходит в точки, которые на практике работают очень хорошо. Здесь опущены формальные определения якобиана, минимумов и выпуклости, а также доказательство того, что самый крутой спуск противоположен градиенту; они находятся в разделах «Якобианы», «Градиентный спуск» и «Сходимость градиентного спуска».

Запомним

Для функции со многими выходами градиенты складываются в якобиан, а якобианы сцепленных функций перемножаются как таблицы. Обучение означает найти параметры, при которых мера ошибки наименьшая, маленькими повторяющимися шагами. Самый крутой путь вниз противоположен градиенту, и идти туда с маленькой скоростью обучения — это градиентный спуск. На чашеобразной, выпуклой задаче он достигает единственной низшей точки откуда угодно; на бугристых ландшафтах нейронных сетей он достигает лишь какой-то плоской точки, что на практике работает на удивление хорошо.

Вопросы

1. Почему самый крутой путь вниз по ландшафту ровно противоположен градиенту?

- a. Потому что градиент всегда указывает на глобальный минимум
 - b. Потому что наклон в направлении — это его скалярное произведение с градиентом, а оно наиболее отрицательно, когда направление противоположно ему
 - c. Потому что скорость обучения автоматически обращает градиент
 - d. Потому что градиент равен нулю везде, кроме дна
2. Градиентный спуск останавливается на плоской земле. Почему это не гарантирует самую низкую точку?
- a. Потому что плоская земля может быть и вершиной холма, и седлом, а какая плоская точка будет достигнута, зависит от старта
 - b. Потому что градиентный спуск никогда не останавливается на плоской земле
 - c. Потому что скорость обучения растёт вблизи дна
 - d. Потому что плоская земля бывает только в выпуклых задачах
3. Что особенного в выпуклой, чашеобразной задаче?
- a. У неё много ямок, поэтому ответ определяется стартом
 - b. К ней нельзя применить градиентный спуск
 - c. Каждый склон ведёт к одной и той же низшей точке, так что глобальный минимум достигается откуда угодно
 - d. Её градиент везде одинаков
4. Нейронные сети не выпуклы. Что книга говорит об их обучении, несмотря на это?

- a. Оно никогда не работает без формулы в замкнутом виде
- b. Стартовав из разумного места, градиентный спуск приходит в точки, которые на практике работают очень хорошо
- c. Оно работает, только если сеть сначала превратить в выпуклую задачу
- d. Оно всегда достигает глобального минимума после достаточного числа шагов

Ответы

- 1. **b)** Наклон в направлении — это скалярное произведение этого направления с градиентом, и оно наименьшее, когда они направлены в противоположные стороны.
- 2. **a)** Плоские точки могут быть дном долины, вершиной холма или седлом, и в общем случае градиентный спуск достигает лишь какой-то плоской точки, в зависимости от того, откуда стартует.
- 3. **c)** Для чашеобразного, выпуклого ландшафта каждый склон ведёт к одной и той же низшей точке, так что градиентный спуск достигает глобального минимума из любого старта.
- 4. **b)** Каждая полезная нейронная сеть — это лоток для яиц, а не чаша, и всё же из разумного старта градиентный спуск приходит в точки, которые работают очень хорошо.

Разобранные примеры

Прямолинейное предположение вблизи точки Книга берёт кривую x в квадрате минус $1.5x$, наклон которой равен $2x$ минус 1.5 , и заменяет её вблизи $x = 0.5$ касательной. Сравните кривую и прямую при $x = 0.51$.

При $x = 0.5$ значение кривой равно $0.25 - 0.75 = -0.5$, а её наклон равен $1 - 1.5 = -0.5$.

Касательная при 0.51 — это значение плюс наклон, умноженный на маленький шаг: $-0.5 + (-0.5 \times 0.01) = -0.505$.

Сама кривая при 0.51 даёт $0.2601 - 0.765 = -0.5049$.

Они различаются всего на одну десятитысячную: совсем рядом с точкой прямая — отличное предположение.

Ответ: Кривая -0.5049 , прямая -0.5050

Задачи

1. Выполните два шага градиентного спуска для функции x в квадрате, наклон которой равен $2x$, начав с $x = 3$ при скорости обучения 0.1 .

Подсказка: Каждый шаг: новый x равен старому x минус скорость обучения, умноженная на наклон.

Ответ: Шаг 1: наклон 6 , поэтому $x = 3 - 0.6 = 2.4$. Шаг 2: наклон 4.8 , поэтому $x = 2.4 - 0.48 = 1.92$. Каждый шаг приближает к дну в точке 0 .

2. Функция принимает 5 чисел и выдаёт 3 числа. Какой формы её якобиан и что содержит каждая строка?

Подсказка: У каждого выхода есть собственный градиент.

Ответ: Это таблица из 3 строк и 5 столбцов: каждая строка — градиент одного выхода по всем 5 входам.

3. Используя картинку с шариком, объясните, почему начальная точка важна при обучении нейронной сети, но не при минимизации чашеобразной функции.

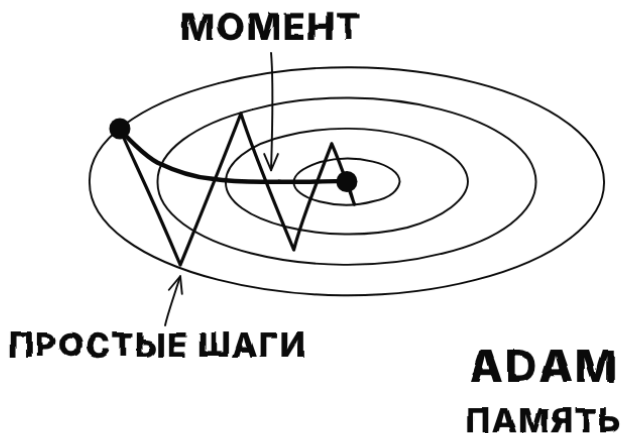
Подсказка: Сравните салатницу с лотком для яиц.

Ответ: В чаше каждый склон ведёт к одному дну, так что шарик оказывается там откуда угодно. Ландшафт ошибки нейронной сети похож на лоток для яиц со множеством ямок и плоских точек, поэтому градиентный спуск укладывается в ту, что ближе к его старту.

Момент, и откуда берутся примеры

Скатите шарик для пинг-понга, а затем тяжёлый стеклянный шарик по бугристому, рифлёному склону, например по смятому листу картона, приподнятому с одного края. Лёгкий шарик дёргается, цепляется за каждую складку и меняет направление на каждом бугорке. Тяжёлый шарик сохраняет большую часть прежнего движения, перекачивается через мелкие складки и достигает низа по более гладкому пути. Эта разница — первая идея этого отрезка книги, а вторая касается самой почвы: примеров, на которых учится модель.

ВИД СВЕРХУ



Дёргающиеся шаги против момента

Как мы видели, градиентный спуск шагает против градиента, но градиент говорит самый крутой путь вниз только там, где вы стоите, и для больших моделей он может быть очень шумным. Момент придаёт прогулке вес: каждый шаг сохраняет часть направления предыдущего шага, а более старые шаги постепенно затухают. Путь становится более гладким, и обучение обычно приходит к цели быстрее. Другой приём даёт каждому параметру собственный размер шага в зависимости от того, насколько велики были его градиенты, а популярный метод под названием Adam объединяет такие идеи; это выбор по умолчанию в большинстве фреймворков, и он довольно терпим к своим настройкам. Цена — память, потому что метод должен помнить предыдущие шаги. Ассистенты и фотомодели, которыми вы пользуетесь, почти наверняка обучались с Adam или одним из его близких родственников.

Затем книга делает паузу для практики вместо упражнений. Она советует освоиться с NumPy, базовой библиотекой для массивов, а потом с JAX и PyTorch, которые выглядят очень похоже на NumPy, но ещё и работают на графических чипах и вычисляют градиенты за вас. Первое задание — написать градиентный спуск вручную для волнистого ландшафта со множеством холмов и впадин, с моментом и без, и посмотреть на пути из разных начальных точек. Книга с улыбкой замечает, что эту волнистую функцию попросили придумать ChatGPT и что больше ничего в книге машиной не сгенерировано.

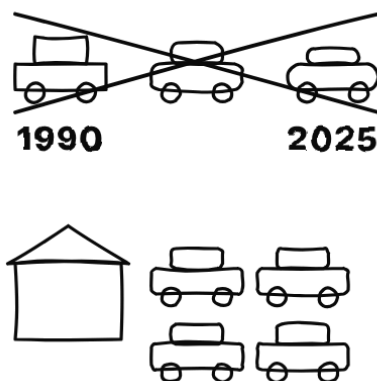
Затем глава 3 спрашивает, с чего начинается обучение: с набора данных. Некоторые правила невозможно написать вручную, например распознавание объектов на фотографиях, но примеры желаемого поведения собрать легко, так что набор данных — это

просто много пар из входа и правильного выхода. Книга подчёркивает одно допущение, которое незаметно решает успех: примеры должны приходить из одного стабильного источника и собираться без смещения. Фотографии автомобилей, собранные за несколько десятилетий, смешивают модели, которых больше не существует, а фотографии, снятые рядом с салоном одной марки, завышают долю этой марки; набор данных из Италии может хорошо служить Риму и плохо — Токио. Даже когда модель уже работает, мир уплывает от её данных, а рекомендательная система меняет само поведение, на котором учится, так что проверка никогда не заканчивается.

ОДИН ИСТОЧНИК ПАРЫ



ПРОВЕРЯЙТЕ ↻



Набор данных должен быть честным

Не у каждой задачи есть приложенные правильные ответы. Без них, в обучении без учителя, группируют похожие вещи или ищут ближайшие совпадения, и помогает модель, уже обученная на другой задаче, потому что её внутренние числа помещают похожие вещи рядом. Самообучение, или обучение с самоконтролем, придумывает собственные ответы из сырых данных, например предсказывая, как продолжается кусок текста, и именно так сначала обучают большие языковые модели. Такой фундаментальной модели затем можно напрямую задать задачу, показать несколько примеров в запросе или дообучить градиентным спуском на маленьком наборе данных, иногда меняя лишь малую долю её параметров. Когда вы вставляете в чат-ассистент два образца писем и просите третье в том же стиле, вы используете его с несколькими примерами, не меняя ни одного из его параметров. Здесь опущены формула момента, подробности упражнения по программированию и длинный список других постановок обучения; они находятся в разделах «Ускорение градиентного спуска», «От теории к практике» и «Варианты обучения с учителем».

САМООБУЧЕНИЕ

НАШ КОТ СЕЛ НА

ВОПРОС

В

МОДЕЛЬ

ПРИМЕРЫ

ПР. ПР. В

МОДЕЛЬ

ДООБУЧЕНИЕ



Без меток, затем три применения

Запомним

Момент позволяет каждому шагу градиентного спуска сохранять часть предыдущего направления, что сглаживает шумный путь и ускоряет обучение ценой некоторой памяти; Adam, обычный выбор по умолчанию, добавляет размер шага для каждого параметра. Набор данных — это коллекция пар вход-выход, используемая, когда само правило записать нельзя. Он должен приходить из одного стабильного источника и собираться без смещения, и это нужно проверять всё время, пока модель используется. Без размеченных ответов модели могут группировать, искать или учиться на задачах, которые придумывают сами, и так начинаются большие языковые модели, а позже их можно использовать напрямую, подсказывать примерами или дообучать.

Вопросы

1. Почему момент делает путь градиентного спуска более гладким?
 - a. Потому что он удлиняет каждый шаг
 - b. Потому что каждый шаг сохраняет часть направления предыдущего шага, так что отдельные шумные градиенты значат меньше
 - c. Потому что он убирает градиент из обновления

- d. Потому что он двигается, только когда градиент равен нулю
2. Какова цена использования момента или Adam?
- a. Больше памяти, потому что нужно помнить предыдущие шаги
 - b. Им нужны вторые производные функции
 - c. Их можно применять только к выпуклым задачам
 - d. Им нужен новый набор данных на каждом шаге
3. Набор данных для распознавания машин собран только из фотографий, снятых у автосалона одной марки. Какое допущение это нарушает?
- a. Что набор данных достаточно мал, чтобы уместиться в памяти
 - b. Что все фотографии одного размера
 - c. Что примеры собраны без смещения из всего источника, с которым встретится модель
 - d. Что модель использует момент
4. Как, согласно этой части, сначала обучают большие языковые модели?
- a. На задаче самообучения, придуманной из сырого текста, например предсказании того, как продолжается кусок текста
 - b. Только на парах вопросов и ответов, написанных вручную
 - c. Группируя предложения в кластеры без всякой задачи

d. Дообучая несколько параметров на маленьком наборе данных

Ответы

1. **b)** Момент придаёт прогулке вес: каждый шаг сохраняет часть предыдущего направления, а старые шаги затухают, поэтому путь становится более гладким.
2. **a)** Метод должен помнить предыдущие шаги, поэтому ему нужно больше памяти.
3. **c)** Примеры должны приходить из одного стабильного источника и собираться без смещения; фотографии у одного автосалона завышают долю этой марки.
4. **a)** Самообучение придумывает собственные ответы из сырых данных, например предсказывая, как продолжается текст, и так начинаются большие языковые модели.

Разобранные примеры

Как быстро затухают старые шаги С моментом каждый шаг сохраняет долю предыдущего; на рисунке в книге используется доля 0.3. Какой вес несут в текущем шаге градиенты с одного, двух и трёх шагов назад?

Самый новый градиент входит в шаг целиком: вес 1.

Предыдущий шаг сохраняется с долей 0.3: вес 0.3.

Тот, что перед ним, сохранялся дважды: 0.3 умножить на 0.3, то есть 0.09.

Более старые градиенты продолжают сжиматься в 0.3 раза каждый раз, так что прогулка помнит только своё недавнее направление.

Ответ: 1, 0.3 и 0.09

Задачи

1. Модель, распознающая марки машин, обучена на фотографиях, собранных в Италии. Объясните, как это делает книга, почему она может работать в Милане, но провалиться в Токио, и почему проверка никогда не заканчивается.

Подсказка: Спросите себя, приходят ли примеры из того же источника, с которым встретится модель.

Ответ: Набор данных отражает машины, распространённые в Италии, поэтому в Токио модель встречает другой набор машин, которые она редко видела. Даже в Италии машины на дорогах меняются с годами, так что данные уплывают от того, что выучила модель, и её качество нужно постоянно проверять.

2. Вы даёте чат-ассистенту два образца писем в запросе и просите третье в том же стиле. Это дообучение? Объясните разницу.

Подсказка: Спросите себя, меняется ли хоть один параметр модели.

Ответ: Нет, это запрос с несколькими примерами: примеры находятся в запросе, и ни один параметр не меняется. Дообучение запустило бы градиентный спуск на маленьком наборе данных и действительно изменило бы параметры модели.

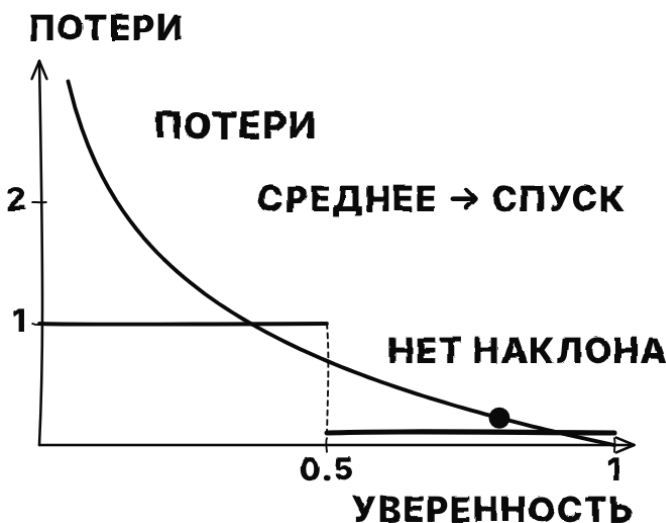
3. Почему модель, обученная классифицировать изображения, может помочь вам искать похожие снимки в фотоколлекции, хотя никто их не размечал?

Подсказка: Подумайте, что её внутренние числа делают с похожими изображениями.

Ответ: Внутренние числа обученной модели помещают похожие изображения рядом, поэтому сравнение этих чисел находит самые похожие снимки без всяких меток.

Счёт, который нужно понизить, и тест, которого вы не видели

Дайте другу прошлогодний тест по теории вождения с ответами и позвольте ему изучать его сколько угодно. Затем дайте ему тот же самый тест: он сдаст его идеально. Теперь дайте ему свежий тест с новыми вопросами, и его результат может рухнуть, потому что он выучил ответы, а не правила дорожного движения. В этом маленьком эксперименте — весь этот отрезок книги: как обучение превращают в число, которое нужно уменьшить, и почему уменьшить его на собственных примерах — не настоящая цель.



Оценка, которую можно понизить

Это число — функция потерь: она сравнивает предсказание модели с правильным ответом и даёт одну оценку, где меньше значит лучше. Она должна быть дифференцируемой, потому что план такой: усреднить потери по всему набору данных и понизить это среднее градиентным спуском по параметрам, и этот рецепт книга называет минимизацией эмпирического риска. Функция потерь — то место, где вы записываете, что для вас важно, и при этом она должна годиться для оптимизации. Простая оценка «верно или неверно» здесь не подходит, потому что она почти везде плоская, так что её градиент равен нулю и градиентный спуск никогда не сдвинется; вместо неё используют более гладкие оценки, которые ещё и вознаграждают уверенно правильный ответ.

Затем ловушка. Средние потери на обучающих примерах всегда можно сделать идеальными жульничеством: справочной таблицей, которая хранит каждый пример и отвечает им, а для всего остального выдаёт что-нибудь по умолчанию. Эта таблица всё запомнила и ничему не научилась, ровно как ваш друг с прошлого года тестом. Что нам по-настоящему нужно — это низкие потери на будущих входах, которых модель никогда не видела, называемые ожидаемым риском, а его никогда нельзя вычислить, потому что собрать все возможные примеры невозможно. Разница между этими двумя — разрыв обобщения, и о модели с большим разрывом говорят, что она переобучилась. На практике держат отдельную тестовую выборку, никогда не используемую в обучении, и сравнивают потери на ней с потерями на обучении. Вот почему создатели фотоприложения оценивают его на снимках, на которых оно никогда не обучалось, а не на тех, по которым училось.

ТАБЛИЦА

x_1	y_1
x_2	y_2
x_3	y_3

ЗАПОМНЕНО

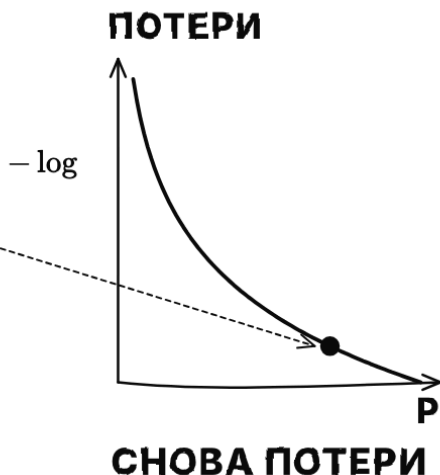
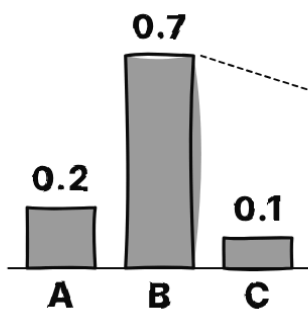
ПЕРЕОБУЧЕНИЕ



Запомнено против выучено

Как выбрать хорошую функцию потерь? Книга предлагает второй, более принципиальный взгляд: пусть модель предсказывает не один ответ, а вероятность для каждого возможного ответа. Для выбора среди трёх классов модель даёт три вероятности, в сумме равные единице, а для числа она может дать наиболее вероятное значение вместе с тем, насколько она в нём не уверена. Когда чат-ассистент пишет, он делает ровно это для каждого следующего слова: вероятность для каждого слова, которое он мог бы использовать. Обучение тогда следует принципу максимального правдоподобия: выбрать модель, при которой набор данных, который вы реально наблюдали, настолько вероятен, насколько возможно.

**СУММА = 1
МАКС. ПРАВДОП.**



Вероятности, затем $-\log$

Два взгляда аккуратно сходятся. Взятие логарифмов и смена знака превращает максимизацию этой вероятности в минимизацию суммы отрицательных логарифмов вероятностей, а это снова просто средние потери. Так что всякую функцию потерь, используемую на практике, можно рассматривать как следствие выбора вероятностей, и книга просит держать в уме оба взгляда. Здесь опущены формулы рисков, кусочно-линейная функция потерь *hinge* и гауссовская и категориальная модели; они находятся в разделах «3.2 Функции потерь», «Ожидаемый риск и переобучение» и «Максимальное правдоподобие».

Запомним

Функция потерь — это дифференцируемая оценка того, насколько ошибочно предсказание, и обучение понижает её среднее по набору данных градиентным спуском.

Запоминание обучающих примеров может сделать это среднее идеальным при полной бесполезности, поэтому настоящая цель — низкие потери на невиданных данных. Отдельная тестовая выборка измеряет разрыв между ними, и большой разрыв означает переобучение. Если позволить модели предсказывать вероятности и выбрать модель, при которой наблюдаемые данные наиболее вероятны, то есть максимальное правдоподобие, получится то же, что минимизация отрицательного логарифма вероятности — функции потерь.

Вопросы

1. Почему простая оценка «верно или неверно» бесполезна для обучения градиентным спуском?
 - a. Потому что её слишком дорого вычислять
 - b. Потому что она почти везде плоская, так что её градиент равен нулю и параметры никогда не сдвигаются
 - c. Потому что её можно применять только к тестовым выборкам
 - d. Потому что она даёт отрицательные значения для правильных ответов

2. Справочная таблица, хранящая каждый обучающий пример, получает на них идеальные средние потери. Почему она всё равно бесполезна?
- a. Потому что она запоминает вместо того, чтобы учиться, и ничего полезного не говорит о входах, которых не видела
 - b. Потому что справочные таблицы не могут хранить числа
 - c. Потому что её потери недифференцируемы
 - d. Потому что она увеличивает набор данных
3. Как на практике понять, переобучилась ли модель?
- a. Обучить её вдвое дольше
 - b. Проверить, что её потери на обучении ровно нулевые
 - c. Сравнить её потери на отдельной тестовой выборке, никогда не использованной в обучении, с потерями на обучении
 - d. Посчитать её параметры
4. Что выбирает принцип максимального правдоподобия?
- a. Модель с наименьшим числом параметров
 - b. Модель, при которой реально наблюдаемый набор данных настолько вероятен, насколько возможно
 - c. Модель, чьи предсказания — всегда самый вероятный класс
 - d. Модель с самой большой тестовой выборкой

Ответы

1. b) Оценка «верно или неверно» почти везде плоская, так что её градиент равен нулю и градиентный спуск никогда не сдвигается.

2. а) Таблица всё запомнила и ничему не научилась; настоящая цель — низкие потери на будущих входах, которых модель никогда не видела.
3. с) Держат отдельную тестовую выборку, никогда не используемую в обучении, и сравнивают потери на ней с потерями на обучении; большой разрыв означает переобучение.
4. б) Максимальное правдоподобие выбирает модель, при которой наблюдаемый набор данных настолько вероятен, насколько возможно.

Разобранные примеры

Функция потерь hinge в действии Для ответов, равных -1 или $+1$, книга упоминает функцию потерь hinge: большее из 0 и 1 минус ответ, умноженный на предсказание. Вычислите её, когда правильный ответ $+1$, а модель предсказывает 0.3, затем 2, и когда правильный ответ -1 , а модель предсказывает 0.5.

Ответ $+1$, предсказание 0.3: $1 - 0.3 = 0.7$, больше 0, так что потери равны 0.7; верная сторона, но недостаточно уверенно.

Ответ $+1$, предсказание 2: $1 - 2 = -1$, так что потери равны 0; верно и уверенно, исправлять нечего.

Ответ -1 , предсказание 0.5: ответ, умноженный на предсказание, равен -0.5 , так что $1 + 0.5 = 1.5$; неверная сторона, потери больше.

В отличие от оценки «верно или неверно», эти потери меняются плавно вместе с предсказанием, так что их наклон может направлять обучение.

Ответ: 0.7, 0 и 1.5

Задачи

1. Модель даёт правильным ответам трёх обучающих примеров вероятности 0.9, 0.8 и 0.5. Каково правдоподобие этого

маленького набора данных и какова сумма отрицательных натуральных логарифмов трёх вероятностей?

Подсказка: Примеры независимы, поэтому их вероятности перемножаются; логарифм превращает произведение в сумму.

Ответ: Правдоподобие равно $0.9 \times 0.8 \times 0.5 = 0.36$. Отрицательные логарифмы примерно 0.105, 0.223 и 0.693, в сумме около 1.02, то же, что минус логарифм 0.36. Повысить правдоподобие значит понизить эту сумму.

2. На примере теста по теории вождения объясните разницу между эмпирическим риском и ожидаемым риском.

Подсказка: Какой тест — обучающая выборка, а какой — настоящая дорога?

Ответ: Эмпирический риск — это результат на прошлогоднем тесте, на примерах, на которых вы учились; ожидаемый риск — результат по всем вопросам, которые вам вообще могут задать. Запоминание понижает первый, не понижая второго, а важен именно второй.

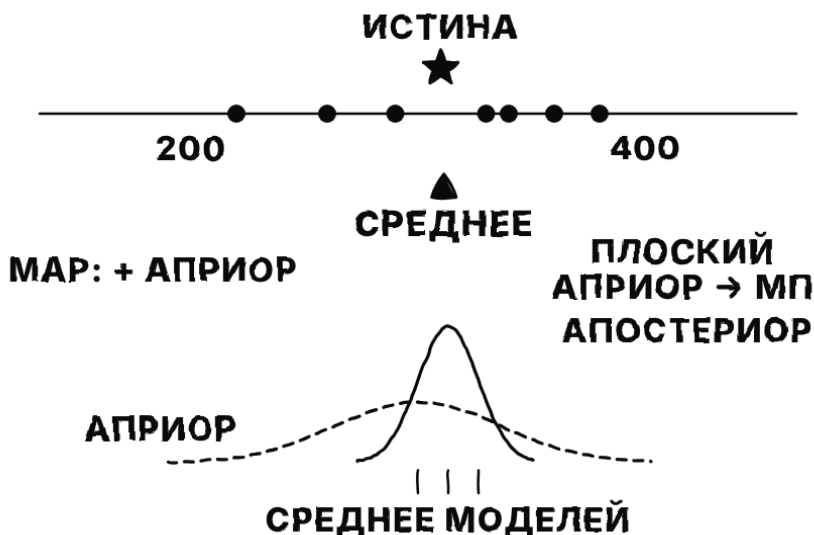
3. Почему полезно, что модель может предсказывать наиболее вероятное значение вместе с тем, насколько она в нём не уверена, а не одно число?

Подсказка: Подумайте, какую дополнительную информацию даёт второй выход.

Ответ: Второй выход говорит, насколько уверенно предсказание, так что модель может выражать неуверенность, а выбрать такую вероятностную модель бывает легче, чем придумывать функцию потерь вручную; максимальное правдоподобие затем даёт функцию потерь.

Много возможных моделей, и самая простая

Наполните банку конфетами и попросите нескольких друзей угадать, сколько их там. Каждая догадка окажется неточной, одни завьются, другие занижат. И всё же среднее всех догадок часто ближе к истине, чем большинство отдельных догадок, потому что их разные ошибки частично взаимно уничтожаются. Книга использует ту же мысль, чтобы сделать шаг за пределы максимального правдоподобия, а затем строит свою самую первую модель.



Усреднение многих догадок

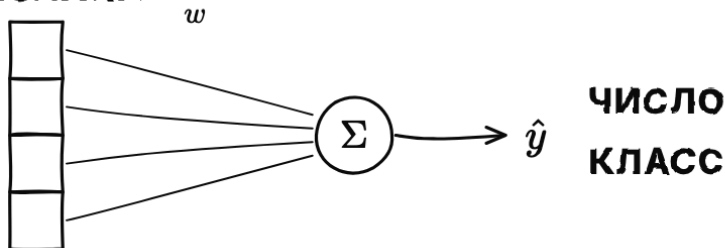
Максимальное правдоподобие выбирает одну лучшую модель, но часто много разных настроек параметров объясняют данные примерно одинаково хорошо. Байесовский взгляд сохраняет их все: прежде чем увидеть какие-либо данные, вы задаёте априорное распределение, убеждение о том, какие модели правдоподобны, например что модели с меньшими параметрами устойчивее и потому предпочтительнее. Увидев данные, теорема Байеса объединяет это априорное убеждение с правдоподобием в апостериорное, обновлённое убеждение обо всех моделях. Предсказание тогда усредняет то, что говорят все модели, каждая с весом по её нынешней правдоподобности, ровно как догадки друзей. Загвоздка в том, что апостериорное распределение почти никогда нельзя вычислить точно, поэтому на практике используют приближения.

Два побочных результата очень важны. Если оставить только одну самую правдоподобную модель, получится правдоподобие плюс слагаемое от априорного распределения, и это дополнительное слагаемое мягко тянет решение к тому, что предпочитает априор; это называется регуляризацией и помогает против переобучения. С априорным распределением, которое ничему не отдаёт предпочтения, это сводится ровно к максимальному правдоподобию. А когда приходят новые данные, вчерашнее апостериорное просто становится сегодняшним априорным, и это помогает модели учиться дальше, не забывая то, что она уже знала.

Теперь глава 4 строит самый простой блок из всех, линейную модель, о которой книга говорит, что всякая дифференцируемая модель — её хитрая вариация. Вход — это список признаков, скажем, сведения о клиенте банка, а выход — либо число, задача

регрессии, либо один из нескольких классов, задача классификации. Для предсказания числа естественная функция потерь — квадратичная ошибка, и книга показывает, что она не произвольна: это ровно то, что даёт максимальное правдоподобие, если считать, что ошибки следуют колоколообразной кривой. Поскольку возведение в квадрат позволяет нескольким сильно ошибочным примерам доминировать, есть более мягкие варианты, такие как абсолютная ошибка и смесь двух, а единственный острый угол абсолютной ошибки — не настоящая помеха для градиентного спуска.

ПРИЗНАКИ

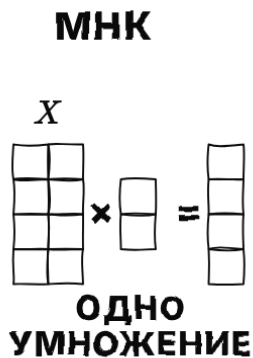



КВАДРАТ


МОДУЛЬ

Самый простой блок

Сама линейная модель даёт каждому признаку вес, складывает все взвешенные признаки и добавляет смещение, предсказание при всех признаках, равных нулю. С одним признаком она рисует прямую линию, с двумя — плоскость. Простую оценку того, сколько продержится батарея вашего телефона, можно построить ровно так: столько-то минут теряется за час экранного времени, столько-то за час навигации, плюс фиксированная величина. Соединение линейной модели, квадратичной ошибки и среднего по набору данных даёт задачу наименьших квадратов. Затем книга складывает весь набор данных в одну таблицу, по одному примеру на строку, так что все предсказания получаются из одного матричного произведения, а это ровно то, что графические чипы делают лучше всего; перестановка строк лишь так же переставляет предсказания. Здесь опущены формула Байеса и её приближения, формулы функций потерь и векторизованное выражение наименьших квадратов; они находятся в разделах «3.3 Байесовское обучение», «Квадратичная функция потерь и варианты» и «Модель наименьших квадратов».



Прямая со смещением

Запомним

Байесовский взгляд сохраняет убеждение обо всех моделях, обновляет его данными через теорему Байеса и предсказывает усреднением моделей, хотя на практике это приходится приближать. Если оставить только самую правдоподобную модель, к максимальному правдоподобию добавляется слагаемое регуляризации, которое тянет к априорному и борется с переобучением. Линейная модель взвешивает каждый признак, складывает их со смещением и является базовым блоком всех дифференцируемых моделей. Квадратичная ошибка следует из максимального правдоподобия с колоколообразными ошибками; вместе с линейной моделью она даёт наименьшие квадраты, вычисляемые для всего набора данных сразу матричными произведениями.

Вопросы

1. Как байесовский взгляд делает предсказание по сравнению с максимальным правдоподобием?
 - a. Он использует только одну модель с наименьшими потерями
 - b. Он усредняет то, что говорят все правдоподобные модели, каждая с весом по её правдоподобности после данных

- c. Он игнорирует данные и использует только априорное распределение
 - d. Он выбирает модель случайно
- 2. Что добавляет априорное распределение, если оставить только одну самую правдоподобную модель?
 - a. Слагаемое регуляризации, которое тянет решение к тому, что предпочитает априор, и помогает против переобучения
 - b. Второй набор данных для обучения
 - c. Большую скорость обучения
 - d. Вообще ничего, во всех случаях
- 3. Почему квадратичная ошибка — естественная функция потерь для предсказания чисел?
 - a. Потому что она игнорирует большие ошибки
 - b. Потому что это единственная дифференцируемая функция потерь
 - c. Потому что это то, что даёт максимальное правдоподобие, когда ошибки следуют колоколообразной кривой
 - d. Потому что у неё острый угол в нуле
- 4. Почему книга складывает весь набор данных в одну таблицу?
 - a. Чтобы все предсказания получались из одного матричного произведения, которое графические чипы выполняют очень быстро
 - b. Потому что порядок примеров меняет предсказания

- с. Потому что линейные модели не справляются с
одиночными примерами
- д. Чтобы убрать смещение из модели

Ответы

1. **б)** Предсказание усредняет то, что говорят все модели, каждая с весом по её правдоподобности после данных, как усреднение догадок друзей.
2. **а)** Если оставить только самую правдоподобную модель, получится правдоподобие плюс априорное слагаемое, которое тянет к априору: регуляризация, полезная против переобучения.
3. **с)** Квадратичная ошибка — ровно то, что даёт максимальное правдоподобие, если считать, что ошибки следуют колоколообразной кривой.
4. **а)** При одном примере на строку все предсказания получаются из одного матричного произведения, а это ровно то, что графические чипы делают лучше всего.

Разобранные примеры

Три функции потерь, один выброс Сравните квадратичную функцию потерь, абсолютную и смешанную функцию потерь Хьюбера для маленькой ошибки 0.5 и большой ошибки 3. Функция потерь Хьюбера — это половина квадратичной ошибки, когда ошибка не больше 1, и абсолютная ошибка минус одна вторая в остальных случаях.

Ошибка 0.5: квадратичная 0.25, абсолютная 0.5, Хьюбер — половина от 0.25, то есть 0.125.

Ошибка 3: квадратичная 9, абсолютная 3, Хьюбер $3 - 0.5 = 2.5$.

Квадратичные потери от большой ошибки вырастают в девять раз, так что один плохо размеченный пример может доминировать в среднем; абсолютная функция и функция Хьюбера растут лишь вместе с ошибкой.

Ответ: 0.25, 0.5, 0.125 для маленькой ошибки; 9, 3, 2.5 для большой

Задачи

1. У линейной модели веса 2 и -1 и смещение 0.5. Что она предсказывает для признаков 3 и 4 и какова её квадратичная ошибка, если правильный ответ 3?

Подсказка: Взвесьте каждый признак, сложите, затем добавьте смещение.

Ответ: $2 \times 3 + (-1) \times 4 + 0.5 = 2.5$. Ошибка равна $2.5 - 3 = -0.5$, так что квадратичная ошибка равна 0.25.

2. В наборе данных 100 примеров по 3 признака каждый, сложенных в одну таблицу. Какой формы таблица и сколько предсказаний даёт одно матричное произведение с 3 весами?

Подсказка: Один пример на строку.

Ответ: В таблице 100 строк и 3 столбца; умножение её на 3 веса даёт 100 предсказаний сразу, к которым добавляется одно и то же смещение.

3. Объясните, как байесовский взгляд помогает модели учиться на новых данных, не забывая то, что она уже знала.

Подсказка: Какую роль сегодня играет вчерашнее апостериорное распределение?

Ответ: После первых данных апостериорное распределение хранит то, что выучила модель. Когда приходят новые данные, это апостериорное используется как новое априорное и снова

обновляется, так что старое знание переносится дальше, а не перезаписывается.