



Оглавление

Предисловие	12
Благодарности	14
О книге	16
Кому адресована книга	16
Структура	16
Соглашения об оформлении кода и его загрузке	18
Форум книги	18
Об авторе	19
От издательства	19
1 Введение в глубокое обучение: зачем его изучать	20
Добро пожаловать в «Грокаем глубокое обучение»!	21
Почему вам стоит изучать глубокое обучение	21
Этому трудно учиться?	22
Почему вы должны прочитать эту книгу	23
Что нужно для начала	25
Возможно, вам потребуется знание Python	26
Итоги	26

2 Основные понятия: как учатся машины?	27
Что такое глубокое обучение?	28
Что такое машинное обучение?	29
Машинное обучение с учителем	30
Машинное обучение без учителя	31
Параметрическое и непараметрическое обучение	32
Параметрическое обучение с учителем	33
Параметрическое обучение без учителя	35
Непараметрическое обучение	37
Итоги	38
3 Введение в нейронное прогнозирование: прямое распространение	39
Шаг 1: прогнозирование	40
Простая нейронная сеть, делающая прогноз	42
Что такое нейронная сеть?	43
Что делает эта нейронная сеть?	44
Прогнозирование с несколькими входами	47
Несколько входов: что делает эта нейронная сеть?	49
Несколько входов: полный выполняемый код	54
Прогнозирование с несколькими выходами	56
Прогнозирование с несколькими входами и выходами	58
Несколько входов и выходов: как это работает?	60
Прогнозирование на основе прогнозов	62
Короткий пример использования NumPy	64
Итоги	67
4 Введение в нейронное обучение: градиентный спуск	69
Предсказание, сравнение и обучение	70
Сравнение	70
Обучение	71
Сравнение: способны ли нейронные сети делать точные прогнозы?	71
Зачем измерять ошибку?	72
Как выглядит простейшая форма нейронного обучения?	74
Обучение методом «холодно/горячо»	76
Особенности обучения методом «холодно/горячо»	77
Вычисление направления и величины из ошибки	79
Одна итерация градиентного спуска	81
Обучение просто уменьшает ошибку	83
Рассмотрим несколько циклов обучения	86
Как это работает? Что такое <code>weight_delta</code> на самом деле?	88
Узкий взгляд на одно понятие	90

Коробка со стержнями	91
Производные: второй пример	92
Что действительно необходимо знать	94
Что знать необязательно	94
Как использовать производные для обучения	95
Выглядит знакомо?	97
Ломаем градиентный спуск	98
Визуальное представление избыточной коррекции	99
Расхождение	100
Знакомьтесь: альфа-коэффициент	101
Альфа-коэффициент в коде	102
Запоминание	103

5 Корректировка сразу нескольких весов: обобщение градиентного спуска 104

Обучение методом градиентного спуска с несколькими входами	105
Градиентный спуск с несколькими входами, описание	107
Рассмотрим несколько шагов обучения	113
Замораживание одного веса: для чего?	115
Обучение методом градиентного спуска с несколькими выходами	117
Обучение методом градиентного спуска с несколькими входами и выходами	120
Чему обучаются эти веса?	121
Визуализация значений весов	124
Визуализация скалярных произведений (сумм весов)	125
Итоги	126

6 Создание первой глубокой нейронной сети: введение в обратное распространение 127

Задача о светофоре	128
Подготовка данных	130
Матрицы и матричные отношения	131
Создание матриц в Python	134
Создание нейронной сети	135
Обучение на полном наборе данных	137
Полный, пакетный и стохастический градиентный спуск	138
Нейронные сети изучают корреляцию	139
Повышающее и понижающее давление	140
Пограничный случай: переобучение	142
Пограничный случай: конфликт давлений	143
Определение косвенной корреляции	145
Создание корреляции	146
Объединение нейронных сетей в стек: обзор	147

8 Оглавление

Обратное распространение: определение причин ошибок на расстоянии	148
Обратное распространение: как это работает?	150
Линейность и нелинейность	151
Почему составная нейронная сеть не работает	152
Тайна эпизодической корреляции	153
Короткий перерыв	154
Ваша первая глубокая нейронная сеть	155
Обратное распространение в коде	156
Одна итерация обратного распространения	159
Объединяем все вместе	161
Почему глубокие сети важны для нас?	162
7 Как изобразить нейронную сеть: в голове и на бумаге	164
Время упрощать	165
Обобщение корреляции	166
Прежняя усложненная визуализация	167
Упрощенная визуализация	169
Еще более упрощенная визуализация	170
Посмотрим, как эта сеть получает прогноз	171
Визуализация с использованием букв вместо картинок	173
Связывание переменных	174
Сравнение разных способов визуализации	175
Важность инструментов визуализации	175
8 Усиление сигнала и игнорирование шума: введение в регуляризацию и группировку	177
Трехслойная сеть для классификации набора данных MNIST	178
Это было просто	180
Запоминание и обобщение	181
Переобучение нейронных сетей	182
Причины переобучения	184
Простейшая регуляризация: ранняя остановка	185
Стандартный способ регуляризации: прореживание (дропаут)	186
Как работает прореживание: в работе участвуют ансамбли	187
Прореживание в коде	188
Влияние прореживания на модель MNIST	191
Пакетный градиентный спуск	192
Итоги	194
9 Моделирование случайности и нелинейности: функции активации . . .	195
Что такое функция активации?	196
Стандартные функции активации для скрытых слоев	200

Стандартные функции активации для выходного слоя	201
Главная проблема: входные данные могут быть схожи между собой	204
Вычисление softmax	205
Инструкции по внедрению функций активации	207
Умножение разности на производную	209
Преобразование выхода в наклон (производную)	211
Усовершенствование сети MNIST	212

10 Края и углы нейронного обучения: введение в сверточные нейронные сети 215

Повторное использование весов в нескольких местах	216
Сверточный слой	217
Простая реализация в NumPy	220
Итоги	224

11 Нейронные сети, понимающие человеческий язык: король – мужчина + женщина == ? 226

Что значит понимать человеческий язык?	227
Обработка естественного языка (NLP)	228
Обработка естественного языка с учителем	229
Набор данных IMDB с обзорами фильмов	230
Выявление корреляции слов во входных данных	231
Прогнозирование обзоров фильмов	232
Введение в слой с векторным представлением	234
Интерпретация результата	236
Нейронная архитектура	237
Сравнение векторных представлений слов	240
В чем заключается смысл нейрона?	241
Подстановка пропущенных слов	242
Смысл определяется потерями	244
Король – мужчина + женщина \sim королева	248
Словесные аналогии	249
Итоги	251

12 Нейронные сети, которые пишут как Шекспир: рекуррентные слои для данных переменной длины 252

Проблема произвольной длины	253
Действительно ли сравнение имеет значение?	254
Удивительная мощь усредненных векторов слов	255
Как векторные представления хранят информацию?	257
Как нейронная сеть использует векторные представления?	258
Ограничение векторов в модели «мешок слов»	259

Объединение векторных представлений слов с использованием единичной матрицы	261
Матрицы, которые ничего не меняют	262
Определение переходных матриц	264
Обучение созданию векторов предложений	265
Прямое распространение на Python	266
Как добавить сюда обратное распространение?	267
Обучим ее!	268
Подготовка	269
Прямое распространение с данными произвольной длины	271
Обратное распространение с данными произвольной длины	272
Корректировка весов с данными произвольной длины	273
Запуск и анализ результатов	274
Итоги	277

13 Введение в автоматическую оптимизацию: создание фреймворка глубокого обучения 278

Что такое фреймворк глубокого обучения?	279
Введение в тензоры	280
Введение в автоматическое вычисление градиента (autograd)	281
Контрольная точка	283
Тензоры, используемые многократно	284
Добавление поддержки тензоров многократного использования в реализацию autograd	286
Как работает сложение в обратном распространении?	288
Добавление поддержки отрицания	289
Добавление поддержки других операций	290
Использование autograd в обучении нейронной сети	295
Добавление автоматической оптимизации	297
Добавление поддержки слоев разных типов	298
Слои, содержащие другие слои	299
Слои с функцией потерь	300
Как научиться пользоваться фреймворком	301
Нелинейные слои	302
Слой с векторным представлением	304
Добавление индексирования в autograd	305
Слой с векторным представлением (повтор)	306
Слой с перекрестной энтропией	307
Рекуррентный слой	309
Итоги	313

14	Обучаем сеть писать как Шекспир: долгая краткосрочная память	314
	Моделирование языка символов	315
	Необходимо усеченное обратное распространение	316
	Усеченное обратное распространение	317
	Образец вывода	321
	Затухающие и взрывные градиенты	322
	Упрощенный пример обратного распространения в RNN	323
	Ячейки долгой краткосрочной памяти (LSTM)	324
	Аналогия, помогающая понять идею вентилях LSTM	325
	Слой долгой краткосрочной памяти	326
	Усовершенствование модели языка символов	328
	Обучение LSTM-модели языка символов	329
	Настройка LSTM-модели языка символов	330
	Итоги	331
15	Глубокое обучение на конфиденциальных данных: введение в федеративное обучение	332
	Проблема конфиденциальности в глубоком обучении	333
	Федеративное обучение	334
	Обучаем выявлять спам	335
	Сделаем модель федеративной	337
	Взламываем федеративную модель	338
	Безопасное агрегирование	340
	Гомоморфное шифрование	341
	Федеративное обучение с гомоморфным шифрованием	342
	Итоги	343
16	Куда пойти дальше: краткий путеводитель	345
	Поздравляю!	346
	Шаг 1: начните изучать PyTorch	346
	Шаг 2: начните изучать следующий курс по глубокому обучению	347
	Шаг 3: купите учебник по математике глубокого обучения	347
	Шаг 4: заведите блог и рассказывайте в нем о глубоком обучении	348
	Шаг 5: Twitter	349
	Шаг 6: напишите руководство на основе академической статьи	350
	Шаг 7: получите доступ к GPU	350
	Шаг 8: найдите оплачиваемую работу, связанную с глубоким обучением	351
	Шаг 9: присоединитесь к открытому проекту	351
	Шаг 10: ищите единомышленников	352

4

Введение в нейронное обучение: градиентный спуск



В этой главе

- ✓ Способны ли нейронные сети делать точные прогнозы?
- ✓ Зачем измерять ошибку?
- ✓ Обучение методом «холодно/горячо».
- ✓ Вычисление направления и величины из ошибки.
- ✓ Градиентный спуск.
- ✓ Обучение просто уменьшает ошибку.
- ✓ Производные и как их использовать для обучения.
- ✓ Расхождение и альфа-коэффициент.

Единственный надежный способ проверить гипотезу —
сравнить ее прогноз с экспериментальными данными.

*Милтон Фридман (Milton Friedman),
Essays in Positive Economics
(издательство Чикагского университета, 1953)*

Предсказание, сравнение и обучение

В главе 3 вы познакомились с парадигмой «предсказание, сравнение, обучение» и углубились в изучение первого шага: *предсказания*. Попутно вы узнали много нового, включая основные компоненты нейронных сетей (узлы и веса), как организуется соответствие наборов данных и сети (количество точек данных, одновременно подаваемых на вход) и как нейронная сеть получает прогноз.

Возможно, в процессе чтения у вас возник вопрос: «Как выбрать значения весов так, чтобы сеть получала точные прогнозы?» Ответ на этот вопрос является главной темой этой главы, и здесь мы рассмотрим следующие два шага парадигмы: *сравнение* и *обучение*.

Сравнение

Сравнение позволяет оценить, насколько прогноз «промахнулся»

Следующий шаг после получения прогноза — оценка его качества. Это может показаться простым делом, но, как вы убедитесь сами, выбор хорошего способа измерения ошибки — одна из самых сложных и важных задач в глубоком обучении.

Есть много разных подходов к измерению ошибок, которые вы наверняка использовали в своей жизни, даже не подозревая об этом. Например, вы (или ваш знакомый) могли преувеличивать большие ошибки и игнорировать мелкие. В этой главе вы познакомитесь с математическим аппаратом, который поможет научить сеть это делать. Вы также узнаете, что ошибка всегда положительна! В качестве аналогии возьмем стрельбу из лука по мишени: если стрела попала в мишень на дюйм выше или ниже, в обоих случаях ошибка составит 1 дюйм. На этапе *сравнения* результатов, полученных от нейронной сети, необходимо учитывать это при оценке ошибки.

Сразу отмечу, что в этой главе мы будем оценивать только один простой способ измерения ошибки: вычисление *среднеквадратической ошибки*. Это лишь один из способов оценки точности нейронной сети.

Этот шаг поможет вам получить представление, насколько вы промахнулись, однако этого недостаточно для обучения. Результатом логики *сравнения* является сигнал «горячо/холодно». Мера ошибки, вычисленная по результатам прогноза, сообщит вам, «насколько сильно» вы промахнулись, но она ничего

не скажет, почему случился промах, в какую сторону вы промахнулись или что нужно сделать, чтобы исправить ошибку, — она лишь скажет «сильно промахнулись», «мало промахнулись» или «попали точно в цель». Исправление ошибки — это уже задача следующего этапа: *обучения*.

Обучение

Процесс обучения определяет, как изменить каждый вес, чтобы уменьшить ошибку

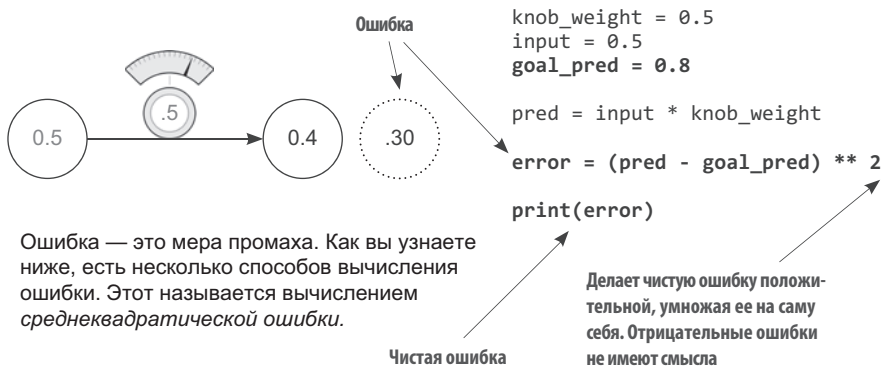
Обучение — это процесс *определения причин ошибок*, или искусство выяснения вклада каждого веса в общую ошибку. Это главная задача глубокого обучения. В этой главе мы много времени уделим изучению одного из самых популярных способов решения этой задачи: *градиентного спуска*.

Этот способ позволяет вычислить для каждого веса некоторое число, определяющее, насколько этот вес должен быть выше или ниже, чтобы уменьшить ошибку. После этого вам останется только изменить вес на это число, и дело в шляпе.

Сравнение: способны ли нейронные сети делать точные прогнозы?

Измерим ошибку и узнаем!

Выполните следующий код в своем блокноте Jupyter Notebook. Он должен вывести 0.3025:



ЧТО ЭТО ЗА ПЕРЕМЕННАЯ GOAL_PRED?

Переменная `goal_pred`, так же как `input`, хранит число, полученное в реальном мире путем наблюдений, иногда очень сложных, как, например, «процент людей, *надевших* теплую одежду» при данной температуре воздуха; или «*попал ли* отбивающий в хоум-ран»¹.

ПОЧЕМУ ОШИБКА ВОЗВОДИТСЯ В КВАДРАТ?

Представьте лучника, стреляющего в мишень. Допустим, стрела попала в мишень на 2 дюйма выше центра. Насколько промахнулся лучник? А если на 2 дюйма ниже? В обоих случаях лучник промахнулся на 2 дюйма. Основная причина *возведения в квадрат* «величины промаха» заключается в получении *положительного* числа. Выражение $(pred - goal_pred)$ может дать отрицательный результат, *в отличие от фактической ошибки*.

РАЗВЕ ВОЗВЕДЕНИЕ В КВАДРАТ НЕ УВЕЛИЧИВАЕТ БОЛЬШИЕ ОШИБКИ (>1) И НЕ УМЕНЬШАЕТ МАЛЕНЬКИЕ (<1)?

Да... Это немного странный способ измерения ошибки, но, как оказывается, *преувеличение* больших ошибок и *преуменьшение* маленьких — это нормально. Позднее вы будете использовать эту ошибку для обучения сети, поэтому лучше сосредоточить внимание на больших ошибках и игнорировать маленькие. Так же поступают хорошие родители: они не замечают мелких ошибок своих детей (например, сломанный грифель карандаша), но могут взорваться в случае большой ошибки (например, если сын или дочь разбились автомобиль). Теперь понимаете, почему возведение в квадрат может быть полезным?

Зачем измерять ошибку?

Измерение ошибки упрощает задачу

Цель обучения нейронной сети — получение достоверных прогнозов. Это наше желание. И в нашем прагматичном мире (как отмечалось в предыдущей главе) хотелось бы иметь сеть, принимающую входные данные, которые легко получить (например, сегодняшние цены на акции), и предсказывающую что-то, что трудно вычислить (завтрашние цены на акции). Это то, что делает нейронные сети полезными.

¹ https://ru.wikipedia.org/wiki/Бейсбольная_терминология.