

ОГЛАВЛЕНИЕ

Отзывы о книге «Глубокое обучение в картинках»	13
Предисловие	16
Вступление	18
Как пользоваться этой книгой.....	20
Благодарности	22
Об авторах	23
От издательства	24
ЧАСТЬ I. ВВЕДЕНИЕ В ГЛУБОКОЕ ОБУЧЕНИЕ	25
Глава 1. Биологическое и компьютерное зрение	26
Биологическое зрение	26
Компьютерное зрение.....	31
Неокогнитрон	32
LeNet-5.....	33
Традиционное машинное обучение	35
ImageNet и ILSVRC.....	37
AlexNet	38
Интерактивная среда TensorFlow	41
Quick, Draw!.....	43
Итоги	43
Глава 2. Языки людей и машин	44
Глубокое обучение для обработки естественного языка.....	45
Сети глубокого обучения автоматически изучают варианты представления	45
Обработка естественного языка.....	46

Краткая история глубокого обучения для NLP	48
Вычислительное представление языка	49
Прямое кодирование слов.....	50
Векторы слов	51
Арифметика с векторами слов.....	54
word2viz	55
Локальные и распределенные представления	57
Элементы естественного языка.....	59
Google Duplex	62
Итоги	64
Глава 3. Машинное искусство.....	65
Ночная пьянка.....	65
Арифметика изображений несуществующих людей.....	68
Передача стиля: преобразование фотографий в изображения в стиле Моне (и наоборот)	71
Придание фотореалистичности простым рисункам.....	72
Создание фотореалистичных изображений из текста	73
Обработка изображений с использованием технологий глубокого обучения	73
Итоги	75
Глава 4. Машины-игроки.....	77
Глубокое обучение, искусственный интеллект и другие	77
Искусственный интеллект	77
Машинное обучение.....	79
Обучение представлению	79
Искусственные нейронные сети	79
Глубокое обучение	80
Компьютерное зрение.....	81
Обработка естественного языка.....	82
Три категории задач машинного обучения.....	82
Обучение с учителем.....	82
Обучение без учителя.....	83
Обучение с подкреплением	83
Глубокое обучение с подкреплением	86
Видеоигры.....	87
Настольные игры	90
AlphaGo.....	90

AlphaGo Zero.....	93
AlphaZero.....	95
Манипулирование объектами	97
Популярные окружения для глубокого обучения с подкреплением.....	99
OpenAI Gym.....	99
DeepMind Lab.....	100
Unity ML-Agents.....	102
Три категории ИИ	103
Ограниченный искусственный интеллект	103
Универсальный искусственный интеллект	103
Искусственный суперинтеллект.....	103
Итоги	104
ЧАСТЬ II. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ В КАРТИНКАХ	105
Глава 5. Телега (код) впереди лошади (теории)	106
Подготовка	106
Установка.....	107
Неглубокая сеть в Keras.....	108
Коллекция изображений рукописных цифр MNIST	108
Схема сети.....	109
Загрузка данных	111
Переформатирование данных.....	113
Проектирование архитектуры нейронной сети.....	115
Обучение модели глубокого обучения	115
Итоги	116
Глава 6. Искусственные нейроны, определяющие хот-доги	117
Введение в биологическую нейроанатомию.....	117
Перцептрон.....	118
Детектор хот-догов	119
Самое важное уравнение в этой книге.....	122
Современные нейроны и функции активации	124
Нейроны sigmoid	125
Нейрон типа tanh	127
ReLU: Rectified Linear Unit.....	128
Выбор типа нейронов.....	129
Итоги	130

Глава 7. Искусственные нейронные сети	132
Входной слой.....	132
Полносвязанный слой.....	133
Полносвязанная сеть, определяющая хот-доги.....	134
Прямое распространение через первый скрытый слой	135
Прямое распространение через последующие слои	137
Слой softmax для сети классификации фастфуда	139
Повторный обзор неглубокой сети	142
Итоги	144
Глава 8. Обучение глубоких сетей	145
Функции стоимости.....	145
Квадратичная функция стоимости	146
Насыщенные нейроны.....	147
Перекрестная энтропия.....	148
Оптимизация: обучение методом минимизации стоимости.....	150
Градиентный спуск.....	150
Скорость обучения	152
Размер пакета и стохастический градиентный спуск.....	154
Как избежать ловушки локального минимума.....	158
Обратное распространение.....	160
Настройка числа скрытых слоев и нейронов	161
Сеть промежуточной глубины на основе Keras.....	163
Итоги	166
Глава 9. Совершенствование глубоких сетей	168
Инициализация весов.....	168
Распределения Ксавье Глоро	172
Нестабильность градиентов	175
Исчезающие градиенты.....	175
Взрывные градиенты.....	176
Пакетная нормализация	176
Обобщающая способность модели (предотвращение переобучения)	178
Регуляризация L1 и L2.....	180
Прореживание	181
Обогащение данных	184
Необычные оптимизаторы.....	184
Метод моментов.....	185

Метод Нестерова	185
AdaGrad	185
AdaDelta и RMSProp.....	186
Adam.....	187
Глубокая нейронная сеть на основе Keras.....	188
Регрессия.....	189
TensorBoard	192
Итоги	195
ЧАСТЬ III. ИНТЕРАКТИВНЫЕ ПРИЛОЖЕНИЯ ГЛУБОКОГО ОБУЧЕНИЯ	197
Глава 10. Компьютерное зрение	198
Сверточные нейронные сети.....	198
Двумерная структура визуальных изображений.....	199
Вычислительная сложность	199
Сверточные слои	200
Множество фильтров.....	202
Пример сверточной сети.....	203
Гиперпараметры сверточных фильтров	207
Слои субдискретизации.....	209
LeNet-5 в Keras.....	211
AlexNet и VGGNet в Keras	217
Остаточные сети	220
Затухание градиентов: ахиллесова пята глубоких сверточных сетей	220
Остаточные связи.....	221
ResNet.....	223
Применения компьютерного зрения.....	224
Обнаружение объектов	225
Сегментация изображений	229
Перенос обучения	231
Капсульные сети	235
Итоги	236
Глава 11. Обработка естественного языка	238
Предварительная обработка данных на естественном языке.....	238
Лексемизация	241
Преобразование всех символов в нижний регистр	244
Удаление стоп-слов и знаков препинания	244

Стемминг.....	245
Обработка n-грамм	246
Предварительная обработка полного корпуса.....	248
Создание векторных представлений с помощью алгоритма word2vec.....	251
Теоретические основы алгоритма word2vec.....	251
Вычисление векторов слов	254
Запуск word2vec	254
Отображение векторов слов на графике.....	259
Площадь под кривой ROC.....	262
Матрица ошибок.....	264
Вычисление метрики ROC AUC	265
Классификация естественного языка с использованием уже знакомых сетей.....	268
Загрузка отзывов к фильмам из IMDb.....	268
Исследование данных из IMDb	272
Стандартизация длин отзывов	275
Полносвязанная сеть.....	276
Сверточные сети	283
Сети, специализирующиеся на изучении последовательных данных	288
Рекуррентные нейронные сети.....	288
Реализация RNN с помощью Keras.....	290
Долгая краткосрочная память	293
Реализация LSTM с помощью Keras.....	295
Двунаправленные LSTM.....	297
Многослойные рекуррентные модели	297
Seq2seq и механизм внимания	299
Перенос обучения в NLP	300
Непоследовательные архитектуры: функциональный API в библиотеке Keras.....	302
Итоги	307
Глава 12. Генеративно-сопоставительные сети	309
Базовая теория GAN	309
Набор данных Quick, Draw!	314
Сеть дискриминатора	317
Сеть генератора.....	320
Сопоставительная сеть	323
Обучение генеративно-сопоставительной сети.....	326
Итоги	332

Глава 13. Глубокое обучение с подкреплением	334
Теоретические основы глубокого обучения с подкреплением	334
Игра Cart-Pole	335
Марковский процесс принятия решений	338
Оптимальная стратегия	340
Базовая теория сетей глубокого Q-обучения	342
Функции ценности	343
Функции Q-ценности	343
Оценка оптимальной Q-ценности	344
Определение агента DQN	345
Инициализация параметров	347
Создание модели нейронной сети агента	349
Запоминание игрового процесса	350
Обучение посредством воспроизведения воспоминаний	351
Выбор действия	352
Сохранение и загрузка параметров модели	353
Взаимодействие с окружением из OpenAI Gym	353
Оптимизация гиперпараметров с помощью SLM Lab	357
Другие агенты, отличные от DQN	360
Градиенты стратегий и алгоритм REINFORCE	361
Алгоритм Actor-Critic	362
Итоги	363
 ЧАСТЬ IV. ВЫ И ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ	365
 Глава 14. Вперед, к собственным проектам глубокого обучения	366
Идеи для проектов глубокого обучения	366
Компьютерное зрение и генеративно-сопоставительные сети	366
Обработка естественного языка	368
Глубокое обучение с подкреплением	369
Преобразование имеющегося проекта машинного обучения	370
Ресурсы для будущих проектов	371
Социально значимые проекты	371
Процесс моделирования, включая настройку гиперпараметров	372
Автоматизация поиска гиперпараметров	375
Библиотеки глубокого обучения	376
Keras и TensorFlow	376

PyTorch	378
MXNet, CNTK, Caffe и другие	379
Программное обеспечение 2.0	379
На пути к универсальному искусственному интеллекту.....	381
Итоги	384
ЧАСТЬ V. ПРИЛОЖЕНИЯ	385
Приложение А. Формальная нотация нейронных сетей.....	386
Приложение Б. Обратное распространение	388
Приложение В. PyTorch.....	392
Особенности PyTorch.....	392
Система Autograd.....	392
Динамическая инфраструктура	392
PyTorch и TensorFlow.....	393
Практическое использование PyTorch.....	394
Установка PyTorch.....	395
Основные компоненты PyTorch	395
Конструирование глубоких нейронных сетей в PyTorch	397

СОВЕРШЕНСТВОВАНИЕ ГЛУБОКИХ СЕТЕЙ

В главе 6 мы детально исследовали отдельные искусственные нейроны. В главе 7 объединили их в сеть, организовав прямое распространение некоторых входных данных x через сеть для получения некоторого прогноза \hat{y} . А совсем недавно, в главе 8, мы узнали, как количественно оценить ошибки сети (сравнить оценку \hat{y} с истинным значением y с помощью функции стоимости), а также как минимизировать эти ошибки (настроить параметры сети w и b с применением оптимизаторов — стохастического градиентного спуска и обратного распространения).

Теперь мы рассмотрим типичные препятствия, встречающиеся на пути создания высококачественных нейронных сетей, и методы их преодоления. Мы применим эти идеи при разработке нашей первой глубокой нейронной сети¹. Объединив увеличенную глубину сети с новыми приемами, мы посмотрим, можно ли превзойти более простые и мелкие архитектуры из предыдущих глав в точности классификации рукописных цифр.

ИНИЦИАЛИЗАЦИЯ ВЕСОВ

В главе 8 мы познакомились с эффектом насыщения нейронов (см. рис. 8.1), когда очень низкие или очень высокие значения z ухудшают способность нейрона к обучению. Там же было предложено решение в виде функции стоимости на основе перекрестной энтропии. Перекрестная энтропия существенно ослабляет влияние эффекта насыщения нейронов, но, объединив ее с вдумчивой *инициализацией весов*, можно еще больше уменьшить вероятность появления эффекта насыщения. Как отмечалось в сноске в главе 1, применение усовершенствованных методов

¹ Как рассказывалось в главе 4, *глубокой* называется нейронная сеть, состоящая по крайней мере из трех скрытых слоев.

инициализации весов обеспечило существенный скачок в развитии сферы глубокого обучения: это одно из знаковых теоретических достижений, сделанных между LeNet-5 (см. рис. 1.11) и AlexNet (см. рис. 1.17), которые значительно расширили диапазон задач, решаемых с помощью искусственных нейронных сетей. В этом разделе мы поэкспериментируем с несколькими подходами к инициализации весов, чтобы вы смогли понять, насколько они эффективны.

Описывая порядок обучения нейронной сети в главе 8, мы упоминали, что параметры w и b инициализируются случайными значениями, из-за чего начальная аппроксимация сети оказывается далекой от цели и, соответственно, имеет высокую начальную стоимость C . Вообще говоря, нет большой нужды подробно рассматривать эту проблему, потому что за кулисами библиотека Keras создает модели TensorFlow, которые инициализируются вполне разумными значениями w и b . Тем не менее мы обсудим ее, чтобы вы не только знали о существовании еще одного метода предотвращения насыщения нейронов, но и восполнили пробел в вашем понимании особенностей обучения нейронных сетей. Библиотека Keras берет на себя всю тяжелую работу по выбору начальных значений, и это главное ее преимущество; но вы должны понимать, что возможно, а иногда необходимо изменить эти значения по умолчанию, чтобы добиться лучшего соответствия вашей задаче.

Чтобы чтение этого раздела проходило более увлекательно, мы советуем использовать наш блокнот Jupyter *weight_initialization.ipynb*. Как показано в следующем фрагменте кода, блокнот зависит от NumPy (библиотека числовых операций), matplotlib (библиотека для построения графиков) и нескольких методов Keras, которые мы подробно рассмотрим в этом разделе.

```
import numpy as np
import matplotlib.pyplot as plt
from keras import Sequential
from keras.layers import Dense, Activation
from keras.initializers import Zeros, RandomNormal
from keras.initializers import glorot_normal, glorot_uniform
```

В этом блокноте мы моделируем 784-пиксельные значения, которые служат входными данными для одного полносвязанного слоя искусственных нейронов. Такая размерность входных данных, конечно же, была выбрана по образу и подобию наших любимых цифр из коллекции MNIST (рис. 5.3). Для полносвязанного слоя мы выбрали достаточно большое число нейронов (256), чтобы потом, когда мы будем строить графики, у нас имелось достаточно данных:

```
n_input = 784
n_dense = 256
```

Основной темой этого раздела является инициализация параметров сети w и b . Прежде чем начать передавать обучающие данные в сеть, желательно определить разумные начальные значения параметров. Тому есть две причины.

1. Большие значения w и b , как правило, приводят к большим значениям z и, соответственно, к появлению эффекта насыщения нейронов (см. график на рис. 8.1).
2. Большие значения параметров означают, что сеть имеет четкое представление о том, как входные данные x связаны с истинными значениями y , но было бы странно строить любые предположения до обучения на фактических данных.

Нулевые значения параметров, с другой стороны, предполагают весьма расплывчатое представление о взаимосвязях между x и y . Проводя аналогию со сказкой «Три медведя», наша цель — уподобиться Машеньке и начать обучение со сбалансированными и *пригодными к обучению* исходными значениями параметров. По этой причине, проектируя архитектуру нейронной сети, мы использовали метод `Zeros()` для инициализации нейронов полносвязанного слоя $b = 0$:

```
b_init = Zeros()
```

Продолжая рассуждения, начатые в предыдущем абзаце, можно подумать, что веса w сети тоже следует инициализировать нулями. Но в действительности это привело бы к катастрофе: при одинаковых значениях весов и смещений многие нейроны в сети будут одинаково интерпретировать одни и те же входные данные x , мешая стохастическому градиентному спуску (SGD) определять индивидуальные настройки параметров, способствующие снижению стоимости C . Продуктивнее было бы инициализировать веса разными значениями из некоторого диапазона, чтобы каждый нейрон обрабатывал входные данные x уникально и давал алгоритму SGD широкий выбор начальных точек для аппроксимации y . По теории вероятностей, начальные результаты некоторых нейронов могут внести верный вклад в разумное отображение x в y . Сначала этот вклад будет слабым, но у SGD появится шанс поэкспериментировать с ним и определить, может ли он способствовать снижению стоимости C между прогнозируемым \hat{y} и целевым значением y .

Как отмечалось ранее (например, при обсуждении рис. 7.5 и 8.9), подавляющее большинство параметров в типичной сети составляют веса, а на смещения приходится относительно небольшая их часть. Поэтому допустимо (в действительности это наиболее распространенная практика) инициализировать смещения нулями, а веса — случайными значениями, *близкими* к нулю. Один из самых простых способов сгенерировать случайные значения, близкие к нулю, — выбрать их из стандартного нормального распределения¹, как в листинге 9.1.

Листинг 9.1. Инициализация весов значениями, выбираемыми из стандартного нормального распределения

```
w_init = RandomNormal(stddev=1.0)
```

¹ Нормальное распределение также называют распределением Гаусса или, в разговорной речи, «колоколообразной кривой» из-за соответствующей формы графика. *Стандартное нормальное* распределение — это нормальное распределение со средним значением 0 и стандартным отклонением 1.

Чтобы оценить влияние выбранного способа инициализации весов, в листинге 9.2 создается архитектура нейронной сети с единственным полносвязанным слоем сигмоидных нейронов.

Листинг 9.2. Архитектура с единственным полносвязанным слоем сигмоидных нейронов

```
model = Sequential()
model.add(Dense(n_dense,
                input_dim=n_input,
                kernel_initializer=w_init,
                bias_initializer=b_init))
model.add(Activation('sigmoid'))
```

Как и во всех предыдущих примерах, модель создается с помощью `Sequential()`. Затем используется метод `add()` для создания одного полносвязанного слоя со следующими параметрами:

- 256 нейронов (`n_dense`);
- 784 входа (`n_input`);
- в `kernel_initializer` передается массив `w_init` для инициализации весов сети желаемыми значениями — в данном случае выбранными из стандартного нормального распределения;
- в `bias_initializer` передается массив `b_init` для инициализации смещений нулевыми значениями.

Чтобы упростить корректировку параметров далее в этом разделе, отдельно добавляем в слой сигмоидную функцию активации вызовом `Activation('sigmoid')`.

После настройки сети вызывается метод `random()` из библиотеки NumPy, чтобы сгенерировать 784 «значения пикселей» — случайные числа с плавающей точкой из диапазона `[0.0, 1.0)`:

```
x = np.random.random((1,n_input))
```

Далее вызывается метод `predict()` для прямого распространения `x` через один слой и получения активаций `a`:

```
a = model.predict(x)
```

В заключение генерируется гистограмма, иллюстрирующая распределение активаций¹:

```
_ = plt.hist(np.transpose(a))
```

¹ Для тех, кому интересно, что означает символ подчеркивания (`_ =`), поясняем, что этот прием помогает сохранить блокнот Jupyter более аккуратным и просто вывести график, без создания объекта, хранящего этот график.

Полученные нами результаты показаны на рис. 9.1. Ваши результаты будут немного отличаться от наших из-за метода `random()`, который использовался для получения входных значений, но в целом они должны выглядеть примерно похожими.

Как и ожидалось (см. рис. 6.9), активации a на выходе слоя сигмоидных нейронов ограничены диапазоном от 0 до 1. Однако в их распределении отмечается нежелательная закономерность — большая их часть сосредоточена по краям диапазона: они примыкают либо к 0, либо к 1. Это указывает на то, что при использовании нормального распределения для инициализации весов w слоя наши искусственные нейроны склонны производить большие значения z . Это нежелательно по двум причинам, упомянутым выше в этом разделе:

1. Подавляющее большинство нейронов в слое подвержено эффекту насыщения.
2. Нейроны выражают твердое мнение о том, как x влияет на y еще до обучения на данных.

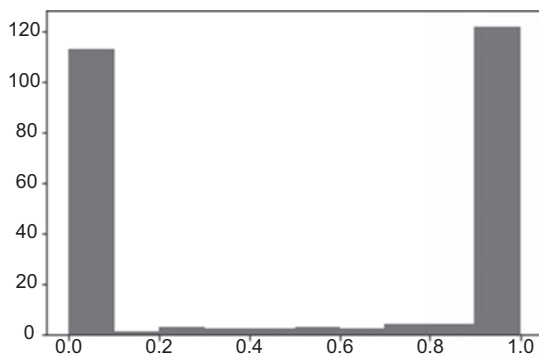


Рис. 9.1. Гистограмма распределения активаций на выходе слоя сигмоидных нейронов с весами, инициализированными с использованием стандартного нормального распределения

К счастью, эту проблему легко устранить, инициализировав веса сети значениями, выбранными из альтернативных распределений.

РАСПРЕДЕЛЕНИЯ КСАВЬЕ ГЛОРО

В сфере глубокого обучения большой популярностью для выборки начальных значений весов пользуются распределения, разработанные Ксавье Глоро (Xavier Glorot) и Йошуа Бенжио¹, портрет которого представлен на рис. 1.10.

¹ Glorot X. & Bengio Y. (2010). «Understanding the difficulty of training deep feedforward neural networks». Proceedings of Machine Learning Research, 9, 249–56.