

УДК 004.85
ББК 32.973.26-018
P21

Рамсундар, Б.

P21 TensorFlow для глубокого обучения: Пер. с англ. / Б. Рамсундар, Р. Б. Заде. — СПб.: БХВ-Петербург, 2019. — 256 с.: ил.
ISBN 978-5-9775-4014-8

Книга знакомит с основами программной библиотеки TensorFlow и принципами глубокого обучения, начиная с нулевого уровня. В книге рассмотрены базовые вычисления в библиотеке TensorFlow, простые обучающиеся системы и их построение, полносвязные глубокие сети, прототипы и превращение прототипов в высококачественные модели, сверточные нейронные сети и обработка изображений, рекуррентные нейронные сети и наборы естественно-языковых данных, способы обучения с максимизацией подкрепления на примере известных игр, приемы тренинга глубоких сетей с помощью графических и тензорных процессоров.

Для разработчиков систем машинного обучения

УДК 004.85
ББК 32.973.26-018

Группа подготовки издания:

Руководитель проекта	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Екатерина Сависте</i>
Перевод с английского	<i>Андрея Логунова</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Оформление обложки	<i>Карины Соловьевой</i>

© 2019 BHV

Authorized Russian translation of the English edition of *TensorFlow for Deep Learning* ISBN 978-1-491-98045-3

© 2018 Reza Zadeh, Bharath Ramsundar.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

Авторизованный русский перевод английской редакции книги *TensorFlow for Deep Learning*

ISBN 978-1-491-98045-3 © 2018 Reza Zadeh, Bharath Ramsundar.

Перевод опубликован и продается с разрешения O'Reilly Media, Inc., собственника всех прав на публикацию и продажу издания.

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-1-491-98045-3 (англ.)

ISBN 978-5-9775-4014-8 (рус.)

© 2018 Reza Zadeh, Bharath Ramsundar

© Перевод на русский язык, оформление. ООО "БХВ-Петербург",
ООО "БХВ", 2019

Оглавление

Об авторах.....	11
Предисловие	13
Условные обозначения, принятые в этой книге	13
Использование примеров программ	14
Признательности	14
Комментарии переводчика	15
Исходный код.....	16
Протокол установки библиотек.....	17
Установка библиотек Python из whl-файлов	17
Глава 1. Введение в глубокое самообучение	19
Машинное самообучение "питается" информатикой.....	19
Глубоко обучающиеся примитивы	21
Полносвязный слой	21
Сверточный слой	22
Слои рекуррентной нейронной сети	22
Ячейки долгой краткосрочной памяти	23
Глубоко обучающиеся архитектуры	24
LeNet	24
AlexNet.....	24
ResNet	25
Нейронная модель титрования изображений.....	26
Нейронный машинный перевод Google.....	27
Однократные модели.....	27
AlphaGo	29
Генеративно-сопоставительные сети.....	31
Нейронные машины Тьюринга.....	31
Вычислительные каркасы для глубокого самообучения	32
Ограничения вычислительной среды TensorFlow	33
Резюме	34
Глава 2. Введение в примитивы TensorFlow	35
Введение в тензоры	35
Скаляры, векторы и матрицы	36
Матричная математика.....	39
Тензоры.....	41
Тензоры в физике.....	42
Математические ремарки.....	44

Базовые вычисления в TensorFlow.....	44
Установка TensorFlow и начало работы	45
Инициализация константных тензоров	45
Отбор случайных значений для тензоров.....	47
Сложение и шкалирование тензоров	48
Матричные операции	49
Типы тензоров.....	50
Обработка форм тензоров.....	51
Введение в операцию транслирования	52
Императивное и декларативное программирование	53
Графы TensorFlow.....	55
Сеансы TensorFlow	55
Переменные TensorFlow.....	56
Резюме	58
Глава 3. Линейная и логистическая регрессия с помощью TensorFlow.....	59
Математический обзор.....	59
Функции и дифференцируемость.....	59
Функции потерь	61
Классификация и регрессия.....	62
L^2 -потеря.....	63
Режимы сбоя функции L^2 -потери.....	63
Распределения вероятностей	64
Перекрестно-энтропийная потеря	65
Градиентный спуск.....	66
Системы автоматического дифференцирования.....	69
Самообучение с помощью TensorFlow	70
Создание игрушечных наборов данных	71
Чрезвычайно краткое введение в NumPy	71
Почему важны игрушечные наборы данных?.....	71
Добавление шума с помощью гауссиан.....	72
Игрушечные наборы регрессионных данных	73
Игрушечные наборы классификационных данных	74
Новые понятия TensorFlow	76
Заполнители	76
Словари передачи данных и выборки.....	76
Области имен	77
Оптимизаторы	77
Взятие градиентов с помощью TensorFlow	78
Сводки и пишущие объекты для TensorBoard	79
Тренировка моделей с помощью TensorFlow.....	80
Тренировка линейной и логистической моделей в TensorFlow	80
Линейная регрессия в TensorFlow	80
Определение и тренировка линейной регрессии в TensorFlow	81
Визуализация линейных регрессионных моделей с помощью TensorBoard.....	83
Метрические показатели для оценивания регрессионных моделей	86
Логистическая регрессия в TensorFlow	89
Визуализация логистических регрессионных моделей с помощью TensorBoard.....	90
Метрические показатели для оценивания классификационных моделей	93
Резюме	94

Глава 4. Полносвязные глубокие сети	95
Что такое полносвязная глубокая сеть?.....	95
"Нейроны" в полносвязных сетях	97
Обучающиеся полносвязные сети с обратным распространением	99
Теорема об универсальной сходимости	100
Почему именно глубокие сети?.....	102
Тренировка полносвязных нейронных сетей.....	102
Заучиваемые представления	102
Активации	103
Полносвязные сети запоминают	104
Регуляризация	104
Отсев	105
Ранняя остановка	106
Регуляризация весов.....	107
Тренировка полносвязных сетей.....	108
Мини-пакетирование.....	108
Скорости заучивания.....	108
Реализация в TensorFlow.....	109
Инсталляция DeepChem	109
Набор данных Tox21	109
Принятие мини-пакетов заполнителей	110
Реализация скрытого слоя.....	111
Добавление отсева в скрытый слой.....	112
Реализация мини-пакетирования	113
Оценивание точности модели.....	113
Использование пакета TensorBoard для отслеживания схождения модели	114
Резюме	116
Глава 5. Гиперпараметрическая оптимизация	117
Оценивание модели и гиперпараметрическая оптимизация	118
Метрики, метрики, метрики	120
Бинарно-классификационные показатели	120
Метрические показатели мультиклассовой классификации.....	123
Регрессионные метрические показатели	124
Алгоритмы оптимизации гиперпараметров.....	125
Установление ориентира.....	126
Спуск студента магистратуры	128
Решеточный поиск гиперпараметров.....	129
Случайный поиск гиперпараметров.....	130
Задание для читателя.....	132
Резюме	132
Глава 6. Сверточные нейронные сети.....	133
Введение в сверточные архитектуры.....	134
Локальные рецептивные поля	134
Сверточные ядра	136
Редуцирующие слои	138
Конструирование сверточных сетей.....	139
Растянутые свертки	139

Применения сверточных сетей.....	140
Обнаружение и локализация объектов	140
Сегментация изображений.....	141
Графовые свертки.....	142
Генерирование изображений с помощью вариационных автокодировщиков	144
Состязательные модели.....	146
Тренировка сверточной сети в TensorFlow	147
Набор данных MNIST	147
Скачивание набора данных MNIST	148
Сверточные примитивы TensorFlow	152
Сверточная архитектура.....	153
Оценивание натренированных моделей	158
Задание для читателя.....	160
Резюме	160
Глава 7. Рекуррентные нейронные сети	161
Обзор рекуррентных архитектур.....	162
Рекуррентные ячейки	164
Долгая краткосрочная память.....	164
Вентильные рекуррентные блоки.....	166
Применение рекуррентных моделей.....	166
Получение образцов из рекуррентных сетей	167
Модели Seq2seq	167
Нейронные машины Тьюринга	169
Работа с рекуррентными нейронными сетями на практике.....	171
Обработка корпуса Penn Treebank.....	171
Программный код для предобработки	173
Загрузка данных в TensorFlow.....	175
Базовая рекуррентная архитектура	177
Задание для читателя.....	179
Резюме	179
Глава 8. Самообучение с максимизацией подкрепления	181
Марковские процессы принятия решений	185
Алгоритмы для самообучения с максимизацией подкрепления	187
Q-заучивание.....	188
Заучивание стратегии.....	189
Асинхронная тренировка	191
Ограничения самообучения с максимизацией подкрепления	191
Игра крестики-нолики.....	193
Объектная ориентированность	194
Абстрактная среда	194
Среда игры крестики-нолики.....	195
Слоевая абстракция	198
Определение графа слоев.....	201
Алгоритм A3C.....	206
Функция потерь	211
Определение рабочих процессов.....	213
Разворачивание игровой ситуации в рабочих процессах	214
Тренировка стратегии	217

Задание для читателя.....	218
Резюме	218
Глава 9. Тренировка крупных глубоких сетей.....	219
Специальное аппаратное обеспечение для глубоких сетей.....	219
Тренировка на CPU	220
Тренировка на GPU	221
Тензорные процессоры.....	222
Программируемые пользователем вентиляемые матрицы	224
Нейроморфные чипы	224
Распределенная тренировка глубоких сетей.....	225
Параллелизм данных	226
Параллелизм моделей.....	227
Параллельная тренировка на данных Cifar10 с использованием многочисленных GPU.....	228
Скачивание и загрузка данных	229
Глубокое погружение в архитектуру	231
Тренировка на многочисленных GPU.....	234
Задание для читателя.....	237
Резюме	237
Глава 10. Будущее глубокого самообучения	239
Глубокое самообучение вне технологической индустрии.....	240
Глубокое самообучение в фармацевтической промышленности.....	241
Глубокое самообучение в юстиции.....	241
Глубокое самообучение для робототехники	242
Глубокое самообучение в сельском хозяйстве	242
Этическое использование глубокого самообучения	243
Действительно ли универсальный искусственный интеллект неизбежен?.....	245
Куда направиться дальше?	246
Предметный указатель.....	247

Бхарат Рамсундар (Bharath Ramsundar) получил степень бакалавра гуманитарных и естественных наук по электротехнике, компьютерным наукам и математике в институте Беркли Калифорнийского университета, с отличием закончил выпускной класс по математике. В настоящее время он является аспирантом в области вычислительной техники в Стэнфордском университете в лаборатории Панде. Его исследовательская работа лежит в сфере применения глубокого обучения при создании медицинских препаратов. В частности, Бхарат является ведущим разработчиком DeepChem.io — пакета с открытым исходным кодом, основанного на библиотеке TensorFlow, который ориентирован на демократизацию использования глубокого обучения при создании медицинских препаратов. Он стипендиат аспирантуры Герца, самой избирательной аспирантуры в данной области наук.

Реза Босаг Заде (Reza Bosagh Zadeh) — основатель и генеральный директор компании Matroid и адъюнкт-профессор в Стэнфордском университете. Его профессиональная деятельность лежит в области машинного обучения, вычислительной и дискретной прикладной математики. Реза получил докторскую степень по вычислительной математике под руководством Гуннара Карлссона в Стэнфордском университете. Его награды включают премию KDD Best Paper Award в области открытия знаний в базах данных и премию Gene Golub Outstanding Thesis Award за выдающуюся дипломную работу. Он работал в технических консультативных советах Microsoft и Databricks.

В рамках своего исследования Реза занимался построением машинно-обучающихся алгоритмов, лежащих в основе системы Twitter "кого-читать", которая стала первым продуктом с использованием машинного обучения в этой социальной сети. Реза является инициатором создания линейно-алгебраического пакета в Apache Spark, и его работа была включена в промышленные и академические кластерные вычислительные среды. В дополнение к исследованиям, Реза разработал и преподает два класса уровня докторантуры в Стэнфордском университете: распределенные алгоритмы и оптимизация (CME 323) и дискретная математика и алгоритмы (CME 305).

Предисловие

Эта книга познакомит вас с фундаментальными принципами машинного обучения с использованием TensorFlow — новой программной библиотеки компании Google для глубокого обучения, которая значительно облегчает инженерам работу по проектированию и внедрению сложных глубоко обучающихся архитектур. Вы познакомитесь с тем, как использовать библиотеку TensorFlow для построения систем, способных обнаруживать объекты на изображениях, понимать человеческий текст и предсказывать свойства потенциальных лекарств. Кроме того, вы получите интуитивное понимание возможностей TensorFlow как системы для выполнения тензорного исчисления и узнаете, как использовать ее для задач, лежащих за пределами традиционной сферы машинного обучения.

Важно отметить, что эта книга является одной из первых книг по глубокому обучению, написанных для практикующих специалистов. Она обучает фундаментальным понятиям на практических примерах и развивает понимание основ машинного обучения, начиная с нулевого уровня. Целевая аудитория этой книги — практикующие разработчики, которые не испытывают проблем с проектированием программных систем, но отнюдь не всегда справляются с созданием обучающихся систем. По ходу изложения мы лишь изредка будем использовать элементарную линейную алгебру и математический анализ, но при этом рассмотрим все необходимые фундаментальные принципы. Мы также ожидаем, что наша книга окажется полезной для ученых и других специалистов, которые не испытывают проблем с написанием сценариев, но отнюдь не всегда — с проектированием обучающихся алгоритмов.

Условные обозначения, принятые в этой книге

В книге используются следующие типографические условные обозначения:

- ◆ *курсивный шрифт* указывает новые термины;
- ◆ **полужирный шрифт** служит для выделения интернет-адресов, элементов интерфейса программных продуктов;
- ◆ моноширинный шрифт используется для листингов программ, а также внутри абзацев для отсылки на элементы программ, такие как переменные или имена функций, базы данных, типы данных, переменные среды, инструкции и ключевые слова;
- ◆ **полужирный моноширинный шрифт** показывает команды либо другой текст, который должен быть напечатан непосредственно пользователем;

- ◆ *курсивный моноширинный шрифт* показывает текст, который должен быть заменен на предоставленные пользователем значения либо на значения, определяемые контекстом.



Данный элемент обозначает общее замечание.



Данный элемент обозначает подсказку или совет.



Данный элемент обозначает предупреждение или предостережение.

Использование примеров программ

Дополнительные материалы (примеры кода, упражнения и т. д.) доступны для скачивания по адресу <https://github.com/matroid/dlwithtf>.

У нас есть веб-страница для этой книги, где мы размещаем опечатки, примеры и любую дополнительную информацию. Вы можете получить доступ к этой странице по адресу <http://bit.ly/tensorflowForDeepLearning>.

Признательности

Бхарат благодарен своему научному руководителю по докторской диссертации за то, что он дал ему возможность работать над этой книгой по ночам и в выходные дни, и особо благодарен своей семье за их неизменную поддержку в течение всего процесса.

Реза благодарен сообществам разработчиков открытого программного обеспечения, на которых основывается подавляющая часть программного обеспечения и исследований в области информатики. Программное обеспечение с открытым исходным кодом является одной из крупнейших концентраций человеческих знаний, когда-либо созданных, и эта книга была бы невозможна без всего сообщества.

Комментарии переводчика

В центре внимания машинного самообучения и глубокого самообучения как под-областей информатики находятся алгоритмы, модели и системы, способные обучаться. Обучающаяся система — это система, способная с течением времени улучшать свою работу, используя поступающую информацию¹. В зарубежной специализированной литературе по данной теме в отличие от русского языка для передачи знаний и приобретения знаний существуют отдельные термины — *train* (*обучить*, *натренировать*) и *learn* (*обучиться*). Приведенное ниже предложение из *главы 7* настоящей книги четко это демонстрирует.

Neural Turing machine can be *trained* end-to-end *to learn* to perform arbitrary computations.

Нейронная машина Тьюринга может быть натренирована в сквозном порядке, чтобы обучиться производить любые вычисления.

Тренировка (training) — это работа, которую выполняет исследователь-проектировщик для получения работающей самообучающейся модели или системы. В основе такой модели или системы лежит обучающийся алгоритм, который представляет собой "не что иное, как искателя минимумов (или максимумов) для надлежащим образом сформулированных функций". *Самообучение* (learning) — это работа, которую выполняет алгоритм-ученик, это процесс приобретения новых или изменения и закрепления существующих знаний, поведения, навыков, ценностей или предпочтений².

Итак, русский термин "обучение" несет в себе двусмысленность, потому что под ним может подразумеваться и передача знаний, и получение знаний. С другой стороны, появление термина learning в любом виде подразумевает исключительно второе — работу, выполняемую обучающимся алгоритмом, т. е. *самообучение*, *заучивание* алгоритмом весов и параметров. Поэтому в отличие от русского термина "машинное обучение", который может означать и тренировку машин, и способность машин обучаться, в своей основе английский термин machine learning обозначает именно *машинное приобретение знаний*.

По этому поводу следует особо отметить следующий момент. С начала 1960-х до середины 1980-х годов в ходу был термин "обучающиеся машины". Проблематика обучающихся и самопроизводящихся машин изучалась в работах А. Тьюринга "Может ли машина мыслить?", 1960 г. (обучающиеся машины), К. Шеннона "Работы по теории информации и кибернетике" (самовоспроизводящиеся машины), Н. Винера, "Кибернетика, или управление и связь в животном и машине" (1961), Н. Нильсона "Обучающиеся машины" (1974) и Я. З. Цыпкина "Основы теории обучающихся систем" (1970).

¹ См. https://ru.wikipedia.org/wiki/Обучающаяся_система, а также <https://bigenc.ru/mathematics/text/1810335>. — *Прим. пер.*

² См. <http://www.basicknowledge101.com/subjects/learningstyles.html#diy>. — *Прим. пер.*

В сухом остатке, английский термин *machine learning* более точно передается термином "*машинное самообучение*" или "*технология обучающихся машин*". Интересный факт: в испанском языке он переводится как *aprendizaje automático*, т. е. автоматическое самообучение.

В настоящем переводе далее за основу принят зарубежный подход, который неизбежно заставил подкорректировать терминологию. Вот несколько таких корректировок. Соответствующие области информатики переведены как *машинное самообучение* и *глубокое самообучение*. Применяемые в этих областях алгоритмы, модели и системы переведены как *обучающиеся*, *машинно-обучающиеся* и *глубоко обучающиеся*. То есть акцент делается не на классификации алгоритма в соответствующей иерархии, а на его характерном свойстве. Далее, методы, которые реализуются в обучающихся алгоритмах, переведены как *методы самообучения* (ср. методы обучения). Как известно, эти методы делятся на три широкие категории. Следуя принципу бритвы Оккама, они переведены как методы контролируемого самообучения (ср. обучение с учителем), методы неконтролируемого самообучения (ср. обучение без учителя) и методы самообучения с максимизацией подкрепления (ср. обучение с подкреплением). Последний термин обусловлен особенностью лежащего в его основе алгоритма — "учиться максимизировать некое понятие вознаграждения", получаемого за правильное действие³. Среди многих гиперпараметров, которые позволяют настроить работу обучающегося алгоритма, имеется *rate of learning*, который переведен как *скорость заучивания* (ср. темп обучения).

Исходный код

Перевод книги снабжен пояснениями и ссылками, размещенными в сносках. Данная книга действительно является полезным ресурсом из разряда "все, что вам нужно знать о глубоком самообучении" на профессиональном уровне.

Элементарная кодовая база книги протестирована в среде Windows 10. При тестировании исходного кода за основу взят Python версии 3.6.4 (время перевода — апрель 2018 г.).

В этой книге используются библиотеки *tensorflow*, *numpy*, *matplotlib*, *scikit-learn* и *deepchem* (<https://deepchem.io>). В обычных условиях библиотеки Python можно скачать и установить из каталога библиотек Python PyPi (<https://pypi.python.org/>) при помощи менеджера пакетов *pip*. Однако следует учесть, что в ОС Windows для работы некоторых библиотек, в частности *scipy*, *scikit-learn* и *scikit-image*, требуется, чтобы в системе была установлена библиотека *Numpy+MKL*. Библиотека *Numpy+MKL* привязана к библиотеке Intel® Math Kernel Library и включает в свой состав необходимые динамические библиотеки (DLL) в каталоге *numpy.core*. Библиотеку *Numpy+MKL* следует скачать из хранилища *whl*-файлов на веб-странице Кристофа Голька из Лаборатории динамики флуоресценции Калифорнийского университета в г. Ирвайн (<http://www.lfd.uci.edu/~gohlke/pythonlibs/>) и установить при

³ См. https://en.wikipedia.org/wiki/Reinforcement_learning. — Прим. пер.

помощи менеджера пакетов `pip` как `whl` (соответствующая процедура установки пакетов в формате `WHL` описана далее). Например, для 64-разрядной операционной системы `Windows` и среды `Python 3.6` команда будет такой:

```
pip install numpy-1.14.2+mkl-cp36-cp36m-win_amd64.whl
```

Стоит также отметить, что эти особенности установки не относятся к ОС `Linux` и `Mac OS X`.

Протокол установки библиотек

Далее предлагается список команд установки библиотек, скачанных с хранилища `whl`-файлов.

```
python -m pip install --upgrade pip
pip install numpy-1.14.2+mkl-cp36-cp36m-win_amd64.whl
pip install scikit_learn-0.19.1-cp36-cp36m-win_amd64.whl
```

Библиотеки `tensorflow` и `matplotlib` устанавливаются стандартным образом:

```
pip install tensorflow
pip install matplotlib
```



Примечание

В зависимости от базовой ОС, версий языка `Python` и версий программных библиотек устанавливаемые вами версии `whl`-файлов могут отличаться от приведенных выше, где показаны последние на май 2018 г. версии для 64-разрядной ОС `Windows` и `Python 3.6.4`.

Установка библиотек Python из whl-файлов

Библиотеки для `Python` можно разрабатывать не только на чистом `Python`. Довольно часто библиотеки программируются на `C` (динамические библиотеки), и для них пишется обертка `Python`, или же библиотека пишется на `Python`, но для оптимизации узких мест часть кода пишется на `C`. Такие библиотеки получаются очень быстрыми, однако библиотеки с вкраплениями кода на `C` программисту на `Python` тяжелее установить ввиду банального отсутствия соответствующих знаний либо необходимых компонентов и настроек в рабочей среде (в особенности в `Windows`). Для решения описанных проблем разработан специальный формат (файлы с расширением `whl`) для распространения библиотек, который содержит заранее скомпилированную версию библиотеки со всеми ее зависимостями. Формат `WHL` поддерживается всеми основными платформами (`Mac OS X`, `Linux`, `Windows`).

Установка производится с помощью менеджера библиотек `pip`. В отличие от обычной установки командой `pip install <имя библиотеки>`, вместо имени библиотеки указывается путь к `whl`-файлу: `pip install <путь_к_whl-файлу>`. Например,

```
pip install C:\temp\scipy-1.0.1-cp36-cp36m-win_amd64.whl
```

Откройте окно командой строки и при помощи команды `cd` перейдите в каталог, где размещен ваш `whl`-файл. Просто скопируйте туда ваш `whl`-файл. В этом случае полный путь указывать не понадобится. Например,

```
pip install scipy-1.0.1-cp36-cp36m-win_amd64.whl
```

При выборе библиотеки важно, чтобы разрядность устанавливаемой библиотеки и разрядность интерпретатора совпадали. Пользователи Windows могут брать `whl`-файлы с веб-сайта Кристофа Голька. Библиотеки там постоянно обновляются, и в архиве содержатся все, какие только могут понадобиться.

Введение в глубокое самообучение

Глубокое самообучение произвело революцию в технологической индустрии. Современный машинный перевод, поисковые системы и компьютерные помощники работают на основе глубокого самообучения. Эта тенденция будет только продолжаться по мере того, как глубокое самообучение будет распространяться на робототехнику, фармацевтические препараты, энергетику и другие области современных технологий. Для специалиста в области разработки программного обеспечения приобретение практических навыков в сфере глубокого самообучения быстро становится первостепенной необходимостью.

В этой главе мы познакомим вас с историей глубокого самообучения и с его более широким влиянием на исследовательские и деловые сообщества. Далее мы рассмотрим несколько самых знаменитых приложений глубокого самообучения. Они будут включать как выдающиеся машинно-обучающиеся архитектуры, так и фундаментальные глубоко обучающиеся примитивы. В конце мы представим краткий обзор направлений, по которым пойдет развитие глубокого самообучения в течение последующих нескольких лет, после чего в следующих нескольких главах мы погрузимся в TensorFlow.

Машинное самообучение "питается" информатикой

До недавнего времени будущие инженеры-программисты посещали лекции в специализированных учебных заведениях, чтобы изучить ряд базовых алгоритмов (поиск в графах, сортировку, запросы к базам данных и т. д.). По окончании учебы эти инженеры применяли полученные знания в конкретной работе. Подавляющая часть современной цифровой экономики построена на сложных цепочках базовых алгоритмов, кропотливо склеенных поколениями инженеров. И большая часть этих систем не приспособлена к адаптации. Все конфигурации и перенастройки должны выполняться высококвалифицированными инженерами, что делает такие системы хрупкими.

Машинное самообучение обещает изменить область разработки программного обеспечения, позволяя системам динамически адаптироваться. Развернутые машинно-обучающиеся системы способны обучаться желаемому поведению из баз данных с примерами. Кроме того, такие системы могут регулярно тренироваться по мере поступления новых данных. Очень сложные программные системы, приводи-

мые в движение машинным самообучением, способны кардинально изменить свое поведение без существенных модификаций программного кода (только в тренировочных данных). Эта тенденция, скорее всего, лишь ускорится по мере упрощения и внедрения средств машинного самообучения.

В ходе изменения линий поведения программных систем меняются и роли инженеров-программистов. В некотором роде эта трансформация будет аналогична трансформации, последовавшей за развитием языков программирования. Первые компьютеры тщательно программировались. Провода объединялись в сложную взаимосвязанную сеть. Затем были придуманы перфокарты, позволившие создавать новые программы без аппаратных изменений в компьютерах. После эпохи перфокарт появились первые языки ассемблера. Потом языки более высокого уровня, такие как Fortran или Lisp. Последующие уровни разработки создавали языки очень высокого уровня, такие как Python, со сложными экосистемами предварительно запрограммированных алгоритмов. Современная информатика во многом уже опирается на автогенерируемый программный код. Современные разработчики приложений используют инструменты, например Android Studio, для автогенерации большей части программного кода, который они хотели бы создать. Таким образом, каждая последующая волна упрощения расширяла сферу информатики, снижая барьеры для доступа.

Машинное самообучение обещает снизить барьеры еще больше; программисты скоро смогут менять поведение систем путем изменения тренировочных данных, возможно, без написания кода. На стороне пользователя системы, построенные на понимании естественного языка, такие как Alexa и Siri, дадут непрограммистам возможность выполнять сложные вычисления. Более того, системы со встроенным машинным самообучением, вероятно, станут более *устойчивыми* к ошибкам. Способность многократно тренировать модели будет означать, что кодовые базы могут сократиться, а удобство сопровождения повысится. Иными словами, машинное самообучение, судя по всему, полностью перевесит роль инженеров-программистов. Сегодняшние программисты должны понимать, каким образом машинно-обучающиеся системы учатся, и знать классы ошибок, которые возникают в обычных машинно-обучающихся системах. Кроме того, им нужно будет понять шаблоны проектирования, лежащие в основе машинно-обучающихся систем (сильно различающихся по стилю и форме от классических шаблонов проектирования программного обеспечения). И им нужно будет в достаточной мере разбираться в тензорном исчислении, чтобы понимать, почему сложная глубокая архитектура может вести себя ненадлежащим образом во время самообучения. Не будет преуменьшением сказать, что понимание (теории и практики) машинного самообучения станет фундаментальным навыком, который каждый компьютерный ученый и инженер-программист должен будет приобрести в течение следующего десятилетия.

Далее в этой главе мы проведем экскурсию по основам современного глубокого самообучения. В остальных главах этой книги мы более подробно рассмотрим темы, которые здесь только затронем.

Глубоко обучающиеся примитивы

Большинство глубоких архитектур строятся путем группирования и перегруппирования ограниченного набора архитектурных примитивов. Такие примитивы, обычно называемые слоями нейронных сетей, являются основополагающими строительными блоками глубоких сетей. В остальной части этой книги мы предоставим подробные вводные сведения о таких слоях. Однако в этом разделе мы дадим краткий обзор общих модулей, которые находятся во многих глубоких сетях. Данный раздел не предназначен для подробного ознакомления с этими модулями. Скорее, мы стремимся акцентировать ваше внимание на строительных блоках сложных глубоких архитектур, чтобы подогреть ваш интерес. Искусство глубокого самообучения заключается в группировании и перегруппировании таких модулей, и мы хотим направить вас по правильному пути к специальным знаниям в области глубокого самообучения.

Полносвязный слой

Полносвязная сеть преобразует список входов в список выходов. Такое преобразование называется *полносвязным*, поскольку любое входное значение может повлиять на любое выходное значение. Такие слои будут иметь много заучиваемых параметров, даже для относительно небольших входных данных, но эти слои обладают большим преимуществом, т. к. они допускают отсутствие структуры во входных данных. Эта концепция проиллюстрирована на рис. 1.1.

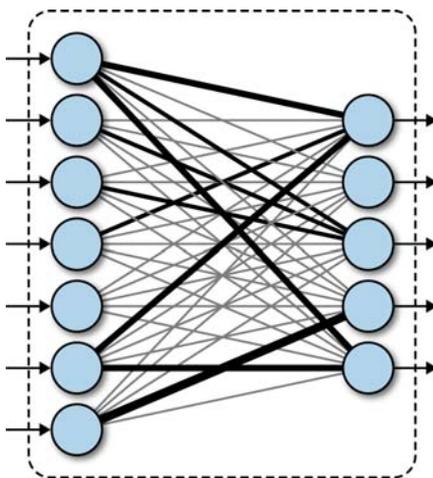


Рис. 1.1. Полносвязный слой. Входящие стрелки представляют входные данные, а исходящие стрелки — выходные данные. Толщина взаимосвязанных линий представляет собой величину заученных весов. Полносвязный слой преобразует входные данные в выходные посредством заученного правила