УДК 004.85
ББК 32.973.26-018
Р21

Рамсундар, Б.

Р21 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, сверхточные 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	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Сависте*
Перевод с английского	*Андрея Логунова*
Компьютерная верстка	*Ольги Сергиенко*
Оформление обложки	*Карины Соловьевой*

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

ISBN 978-1-491-98045-3 (англ.)
ISBN 978-5-9775-4014-8 (рус.)

Оглавление

Об авторах

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 подобластей информатики находятся алгоритмы, модели и системы, способные обучаться. Обучающаяся система — это система, способная с течением времени улучшать свою работу, используя поступающую информацию[1]. 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 предпочтений[2].

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 действие[3]. Среди многих гиперпараметров, которые позволяют настроить работу обучающегося алгоритма, имеется rate of learing, 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 бóльшая 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 разнящихся 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 акцентировавать 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.

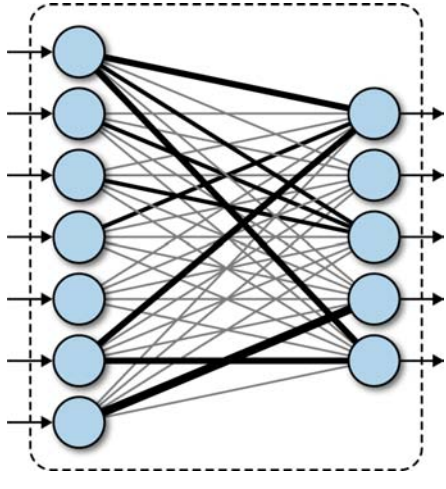

Рис. 1.1. Полносвязный слой. Входящие стрелки представляют входные данные, а исходящие стрелки — выходные данные. Толщина взаимосвязанных линий представляет собой величину заученных весов. Полносвязный слой преобразует входные данные в выходные посредством заученного правила