УДК 004.85
ББК 32.973.26-018
М59

Микелуччи У.

М59 Прикладное глубокое обучение. Подход к пониманию глубоких нейронных сетей на основе метода кейсов: Пер. с англ. — СПб.: БХВ-Петербург, 2020. — 368 с.: ил.

ISBN 978-5-9775-4118-3

Затронуты расширенные темы глубокого обучения: оптимизационные алгоритмы, настройка гиперпараметров, отсев и анализ ошибок, стратегии решения типичных задач во время тренировки глубоких нейронных сетей. Описаны простые активационные функции с единственным нейроном (ReLu, сигмоида и Swish), линейная и логистическая регрессии, библиотека TensorFlow, выбор стоимостной функции, а также более сложные нейросетевые архитектуры с многочисленными слоями и нейронами. Показана отладка и оптимизация расширенных методов отсева и регуляризации, настройка проектов машинного обучения, ориентированных на глубокое обучение с использованием сложных наборов данных. Приведены результаты анализа ошибок нейронной сети с примерами решения проблем, возникающих из-за дисперсии, смещения, переподгонки или разрозненных наборов данных. По каждому техническому решению даны примеры решения практических задач.

Для разработчиков систем глубокого 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 978-1-4842-3789-2 (англ.)
ISBN 978-5-9775-4118-3 (рус.)

Оглавление

Об авторе

Умберто Микелуччи в настоящее время работает в области инноваций и искусственного интеллекта (ИИ) в ведущей медицинской страховой компании Швейцарии. Он возглавляет несколько стратегических инициатив, связанных с ИИ, новыми технологиями, машинным обучением и исследовательским сотрудничеством с университетами. Ранее он работал в качестве исследователя данных и ведущего моделиста для нескольких крупных проектов в области здравоохранения и приобрел большой практический опыт в программировании и разработке алгоритмов. Он руководил проектами в области бизнес-аналитики и хранения данных по реализации управляемых данными решений в сложных производственных средах. В последнее время Умберто активно работал с нейронными сетями и занимался применением глубокого обучения к нескольким задачам, связанным со страхованием, поведением клиентов (например, уходом клиентов) и сенсорной наукой. Он изучал теоретическую физику в Италии, США и Германии, где также работал исследователем. Высшее образование он получил в Великобритании.

Регулярно представляет научные результаты на конференциях, публикует научные статьи в рецензируемых журналах.

О рецензенте

Джоджо Мулайл является профессионалом в области искусственного интеллекта, глубокого обучения, машинного обучения и теории принятия решений с более чем пятилетним производственным стажем. Он автор книги "Smarter Decisions — The Intersection of IoT and Decision Science" ("Более умные решения — на стыке Интернет вещей (IoT) и теории принятия решений"). Работал с несколькими лидерами индустрии в области высокоэффективных и критически важных проектов науки о данных и машинного обучения по нескольким вертикалям. В настоящее время его имя ассоциируют с General Electric, первопроходцем и лидером в области науки о данных для промышленного IoT. Живет в Бенгалуру — Кремниевой долине Индии.

Джоджо родился и вырос в Пуне (Индия), окончил университет Пуны по специализации в области информационных технологий. Свою карьеру он начал в Mu Sigma Inc., крупнейшем в мире поставщике "чистокровной" аналитики, и работал со многими ведущими клиентами списка Fortune 50. Будучи одним из первых энтузиастов, рискнувших вложиться в IoT-аналитику, он объединил свои познания из теории принятия решений для того, чтобы привнести каркасы решения задач и свои познания из науки о данных и теории принятия решений в IoT-аналитику.

Для того чтобы укрепить свои познания науки о данных для промышленного Интернета вещей и масштабировать воздействие экспериментов с применением теории принятия решений, он присоединился к быстро растущему стартапу в области IoT-аналитики под названием Flutura, базирующемуся в Бангалоре и со штаб-квартирой в долине. После непродолжительной работы с Flutura, Джоджо перешел на работу с лидерами производственного IoT — General Electric, в Бангалоре, где и сосредоточился на работе с задачами принятия решений в промышленных IoT-приложениях. В рамках своей роли в General Electric Джоджо также фокусируется на развитии продуктов и платформ науки о данных и принятия решений для промышленного Интернета вещей.

Помимо написания книг по теории принятия решений и IoT, Джоджо также выступал техническим рецензентом различных книг по машинному обучению, глубокому обучению и бизнес-аналитике с публикациями в издательствах Apress и Packt. Он является действующим преподавателем науки о данных и ведет блог по адресу **http://www.jojomoolayil.com/web/blog/**.

Профиль:

- **http://www.jojomoolayil.com/**;
- **https://www.linkedin.com/in/jojo62000**.

"Хочу поблагодарить свою семью, друзей и наставников".

— Джоджо Мулайл

Признательности

Было бы несправедливо, если бы я не поблагодарил всех тех людей, которые помогли мне с этой книгой. Во время написания книги я обнаружил, что совершенно ничего не знаю о книгоиздании, а также что даже когда вы думаете, что знаете что-то хорошо, положить это на бумагу — совершенно другая история. Невероятно, как якобы ясный ум человека искажается при изложении мыслей на бумаге. Это предприятие было одним из самых трудных, которые я когда-либо начинал, но это событие также было одним из самых полезных в моей жизни.

Во-первых, я должен поблагодарить мою любимую жену Франческу Вентурини (Francesca Venturini), которая проводила бесчисленные часы ночью и по выходным, читая текст. Без нее книга не была бы такой ясной. Я должен также поблагодарить Селестина Суреша Джона (Celestin Suresh John), который поверил в мою идею и дал мне возможность написать эту книгу. Адите Мираши (Aditee Mirashi) — самый терпеливый редактор, каких я когда-либо встречал. Она всегда была рядом, чтобы ответить на все мои вопросы, а у меня их было немало, и не все хорошие. Я особенно хотел бы поблагодарить Мэтью Муди (Matthew Moodie), у которого хватило терпения прочитать каждую главу. Я никогда не встречал никого, кто мог бы предложить так много хороших предложений. Спасибо, Мэтт, я перед тобой в долгу. У Джоджо Мулайла (Jojo Moolayil) хватило терпения проверить каждую строчку кода и убедиться в правильности каждого объяснения. И когда я говорю "каждую", я это имею в виду. Нет, правда, я серьезно. Спасибо, Джоджо, за твои отзывы и твою поддержку. Это действительно много значило для меня.

Наконец, я бесконечно благодарен моей любимой дочери Катерине (Caterina) за ее терпение, когда я писал, и за то, что она каждый день напоминала мне, как важно следовать своим мечтам. И конечно, я должен поблагодарить родителей, которые всегда поддерживали мои решения, какими бы они ни были.

Комментарии переводчика

Данная книга послужит ценным справочным руководством по разработке нейросетевых приложений вообще и с использованием библиотеки TensorFlow в частности. В ней содержится ряд ценных сведений и советов, которые трудно найти в Интернете, среди прочих касающиеся особенностей метрического анализа, предсказания с помощью гауссовых процессов, функции обнаружения, регрессии Надарая – Ватсона и байесовой оптимизации. Подробно описаны принципы работы полносвязных, сверточных и рекуррентных сетей и применение нейросетей в сенсорных приложениях. На YouTube-канале по адресу **https://www.youtube.com/channel/UC0JFHky44E1oYJWDVJuZf7g** можно найти несколько видеороликов автора, посвященных продвинутым темам.

Кодовая база книги была протестирована в среде Windows 10. При тестировании исходного кода за основу взят Python версии 3.7.0 (время перевода — март 2019 г.). *Главы 1, 2, 7* и *8* были проверены полностью, остальные — выборочно. Следует отметить, что исходные коды книги, размещенные в хранилище GitHub по адресу **https://github.com/Apress/applied-deep-learning**, в ряде случаев слегка отличаются от исходного кода в печатном издании, и это говорит только в пользу автора, который регулярно обновляет свой код после публикации книги.

Введение

Зачем нужна еще одна книга по прикладному глубокому обучению? Именно этот вопрос я задал себе перед тем, как приступить к написанию настоящей книги. Ведь стоит только погуглить по данной теме, и вы будете ошеломлены огромным числом результатов. Однако я столкнулся с проблемой, которая состояла в том, что я нашел материал для реализации только очень простых моделей на очень простых наборах данных. Снова и снова вам предлагают одни и те же задачи, одни те же подсказки и советы. Если вы хотите научиться классифицировать набор данных MNIST[1], который состоит из десяти рукописных цифр, то вам повезло. (Почти любой, у кого есть блог, уже это сделал, в основном копируя исходный код на вебсайте TensorFlow). Ищете что-то, чтобы понять, как работает логистическая регрессия? Не так-то просто. Хотите узнать, как подготовить набор данных для выполнения интересной бинарной классификации? Еще труднее. Я чувствовал, что есть необходимость заполнить этот пробел. К примеру, я потратил часы, пытаясь отладить модели, которые не работали по таким глупым причинам, как неправильные метки. В частности, вместо меток 0 и 1 у меня были метки 1 и 2, но ни один блог меня об этом не предупредил. Во время разработки моделей важно проводить правильный метрический анализ, но никто не демонстрирует, как это делать (по крайней мере, не на легкодоступном материале). Этот пробел необходимо было заполнить. По моему мнению, охват более сложных примеров, от подготовки данных до анализа ошибок, является очень эффективным и интересным способом изучения правильных технических решений. В этой книге я все время старался охватывать полные и сложные примеры с целью объяснить понятия, которые не так легко усвоить каким-либо другим способом. Невозможно понять важность подбора правильного темпа заучивания, если вы не видите, что может произойти при выборе неправильного его значения. Поэтому я все время объясняю понятия на реальных примерах и на полноценном и протестированном исходном Python-коде, который можно использовать многократно. Обратите внимание, что цель этой книги вовсе не в том, чтобы сделать из вас эксперта в программировании на языке Python или библиотеке

[1] Модифицированный набор данных института NIST (National Institute of Standards and Technology), Национального института стандартов и технологий, США. — *Прим. пер.*

TensorFlow, или кем-то, кто может разрабатывать новые сложные алгоритмы. Python и TensorFlow — это просто инструменты, которые очень хорошо подходят для разработки моделей и быстрого получения результатов. Поэтому я ими и пользуюсь. Я мог бы использовать и другие инструменты, но эти как раз те, которые практики применяют чаще всего, и поэтому вполне резонно было выбрать именно их. Если вам еще предстоит только научиться, то лучше, когда вы работаете именно с тем, что можете использовать в собственных проектах и для своей карьеры.

Цель этой книги — дать вам увидеть более продвинутый материал новой точки зрения. Я освещаю математические основы максимально глубоко в той мере, в какой, по моим ощущениям, это необходимо для полного понимания трудностей и рассуждений, стоящих за многими понятиями. Невозможно понять, почему большой темп заучивания будет побуждать вашу модель (строго говоря, стоимостную функцию) расходиться, если вы не знаете, каким образом алгоритм градиентного спуска работает математически. Во всех реальных проектах вам не придется рассчитывать частные производные или компле́ксные суммы, но вы должны их понимать для того, чтобы уметь оценить то, что может работать, а что — нет (и в особенности, почему). По достоинству оценить, почему библиотека TensorFlow упрощает вам жизнь, можно только в том случае, если вы попытаетесь разработать с нуля тривиальную модель с одним-единственным нейроном. Это очень показательная вещь, и я покажу вам, как это сделать, в *главе 10*. Сделав это один раз, вы запомните весь ход работы навсегда и по-настоящему оцените важность таких библиотек, как TensorFlow.

Я предлагаю вам попытаться действительно разобраться в математических основах (хотя это и не является строго необходимым для того, чтобы извлечь выгоду из этой книги), потому что они позволят вам усвоить многие понятия, которые в противном случае невозможно усвоить полностью. Машинное обучение — очень сложный предмет, и утопично думать, что его можно освоить полностью без хорошего понимания математического каркаса или Python. В каждой главе я даю важные советы по эффективной разработке на Python. В этой книге нет такого утверждения, которое не подкреплялось бы конкретными примерами и воспроизводимым исходным кодом. Я не буду ничего обсуждать, не приведя соответствующих примеров из реальной жизни. Благодаря этому все сразу обретет смысл, и вы это запомните.

Потратьте время на изучение исходного кода, который вы найдете в этой книге, и пробуйте сами. Каждый хороший преподаватель знает, что усвоение учебного материала лучше всего проходит тогда, когда студенты пытаются решать задачи самостоятельно. Старайтесь, ошибайтесь и учитесь. Прочитайте главу, наберите исходный код и попробуйте его модифицировать. Например, в *главе 2* я покажу вам, как выполнять распознавание на основе бинарной классификации между двумя написанными от руки цифрами: 1 и 2. Возьмите исходный код и попробуйте две другие цифры. Играйте с кодом, экспериментируйте и получайте удовольствие.

По замыслу исходный код, который вы найдете в этой книге, написан максимально просто. Он не оптимизирован, и я знаю, что можно написать гораздо эффективнее,

но поступая так, я бы пожертвовал ясностью и удобочитаемостью. Цель этой книги не в том, чтобы научить вас писать высокооптимизированный исходный код на Python, а в том, чтобы вы вникли в фундаментальные алгоритмические понятия и их ограничения и получили прочную основу для продолжения своего обучения в этой области. Несмотря на это, я, безусловно, укажу на важные детали реализации на Python, например, что вы должны как можно чаще избегать стандартных циклов Python.

Весь код в этой книге написан в поддержку учебных целей, которые я поставил для каждой главы. Такие библиотеки, как NumPy и TensorFlow, были рекомендованы, поскольку они позволяют переводить математические формулировки непосредственно на язык Python. Я в курсе о существовании других программных библиотек, таких как TensorFlow Lite, Keras и многих других, которые могут облегчить вам жизнь, но это всего лишь инструменты. Существенная разница заключается в вашей способности понимать концепции, лежащие в основе методов. Если вы понимаете их правильно, то можете выбрать любой инструмент и сумеете добиться хорошей реализации. Если вы не понимаете того, как работают алгоритмы, то независимо от инструмента, вы не сможете выполнить надлежащую реализацию или надлежащий анализ ошибок. Я ярый противник концепции науки о данных для всех. Наука о данных и машинное обучение являются трудными и многосложными предметами, которые требуют глубокого понимания математики и стоящих за ними тонкостей.

Я надеюсь, что вы получите удовольствие от чтения этой книги (у меня точно его было вдоволь при написании этой книги) и найдете примеры и исходный код полезными. Я надеюсь, что у вас будет много моментов озарения, когда вы, наконец, поймете, почему что-то работает так, как вы и ожидаете (или почему — нет). Надеюсь, вы найдете полнофункциональные примеры интересными и полезными. И даже если я помогу вам разобраться лишь в одном термине, который раньше для вас был непонятен, то буду просто счастлив.

Несколько глав этой книги математически являются более продвинутыми. Например, в *главе 2* вычисляются частные производные. Но не переживайте. Если вы их не понимаете, то уравнения можно пропустить. Я сделал так, чтобы основные понятия легко воспринимались без большинства математических деталей. Тем не менее вы все-таки должны знать, что такое матрица, как умножать матрицы, что такое транспонирование матрицы и т. д. В принципе, у вас должно быть хорошее понимание линейной алгебры. Если у вас его нет, то предлагаю вам ознакомиться с какой-нибудь вводной книгой по линейной алгебре и только потом приступать к чтению данной книги. Если у вас есть прочный фундамент в линейной алгебре и дифференциальном исчислении, то я настоятельно рекомендую не пропускать математические разделы. Они действительно помогут понять, почему мы делаем что-то тем или иным образом. Например, это очень поможет вам понять причуды темпа заучивания или то, как работает алгоритм градиентного спуска. Вы также не должны бояться более сложных обозначений и чувствовать себя уверенно с таким сложным уравнением, как то, которое приведено ниже. (Это среднеквадратическая

ошибка, которую мы будем использовать для алгоритма линейной регрессии, и она будет подробно объяснена позже, поэтому не переживайте, если на данный момент вы не знаете, что означают эти символы.)

$$J(w_0, w_1) = \frac{1}{m}\sum_{i=1}\left(y_i - f\left(w_0, w_1, x^{(i)}\right)\right)^2 .$$

Вы должны разбираться и чувствовать себя уверенно с такими понятиями, как сумма или математический ряд. Если вы в них не уверены, то перед началом чтения книги проведите ревизию своих знаний по этим темам; в противном случае вы пропустите некоторые важные понятия, о которых должны иметь четкое представление с целью продолжения своей карьеры в сфере глубокого обучения. Цель этой книги не в том, чтобы дать вам прочную математическую основу. Полагаю, она у вас есть. Глубокое обучение и нейронные сети (в общем смысле, машинное обучение) — очень сложны, и тот, кто пытается убедить вас в обратном, обманывает или просто их не понимает.

Я не буду тратить время на обоснование или математическое выведение алгоритмов или уравнений. Вам придется мне довериться. Кроме того, я не буду обсуждать применимость конкретных уравнений. Для тех из вас, кто хорошо разбирается в дифференциальном исчислении, например, я не буду обсуждать вопрос дифференцируемости функций, для которых мы вычисляем производные. Просто будем считать, что вы можете применять формулы, которые я вам даю. Как показали многие годы их практической реализации сообществом глубокого обучения, эти методы и уравнения работают, как и ожидалось, и могут применяться на практике. Такого рода продвинутые темы, о которых упомянуто выше, потребуют отдельной книги.

В *главе 1* вы узнаете, как настроить среду Python и что такое вычислительный граф. Я расскажу о некоторых основных примерах математических расчетов, выполняемых с помощью TensorFlow. В *главе 2* мы рассмотрим, что можно делать с одним-единственным нейроном. Я расскажу, что такое активационная функция и какие их типы применяются на практике наиболее часто, в частности, такие как сигмоида, ReLU или tanh. Я покажу вам, как работает градиентный спуск и как реализовать логистическую и линейную регрессию с одним нейроном и с помощью TensorFlow. В *главе 3* мы рассмотрим полносвязную сеть. Я расскажу о размерностях матрицы, о том, что такое переподгонка, и познакомлю вас с набором данных Zalando. Затем мы построим первую реальную сеть с помощью TensorFlow и начнем рассматривать более сложные варианты алгоритмов градиентного спуска, такие как мини-пакетный градиентный спуск. Мы также рассмотрим различные способы инициализации весов и то, как сравнивать различные сетевые архитектуры. В *главе 4* мы рассмотрим алгоритмы динамического ослабления темпа заучивания, такие как ступенчатое, пошаговое или экспоненциальное ослабление, а затем обсудим передовые оптимизаторы, такие как Momentum, RMSProp и Adam. Я также дам вам несколько советов о том, как с помощью TensorFlow можно разрабатывать пользовательские оптимизаторы. В *главе 5* я расскажу о регуляризации, включая такие из-

вестные методы, как l_1, l_2, отсев и досрочная остановка. Мы рассмотрим алгоритмы этих методов и способы их реализации в TensorFlow. В *главе 6* мы рассмотрим такие понятия, как человеческая результативность и байесова ошибка. Затем я представлю рабочий процесс метрического анализа, который позволит выявлять проблемы, связанные с набором данных. Кроме того, мы рассмотрим *k*-блочную перекрестную проверку как инструмент валидации ваших результатов. В *главе 7* мы рассмотрим класс черно-ящичных задач и что собой представляет гиперпараметрическая настройка. Мы рассмотрим такие алгоритмы, как решеточный и случайный поиск, и выясним, какие из них эффективнее и почему. Затем мы рассмотрим некоторые приемы, такие как оптимизация с переходом от крупнозернистости к мелкозернистости. Я посвятил большую часть этой главы байесовой оптимизации — тому, как ее использовать и что такое функция обнаружения. Я дам несколько советов относительно того, например, как настраивать гиперпараметры на логарифмической шкале, а затем мы выполним гиперпараметрическую настройку на наборе данных Zalando с целью продемонстрировать то, как это может работать на практике. В *главе 8* мы рассмотрим сверточные и рекуррентные нейронные сети. Я покажу, что подразумевается под сверткой и сведением, и покажу базовую реализацию обеих архитектур в TensorFlow. В *главе 9* я расскажу вам о реальном исследовательском проекте, над которым работаю в Цюрихском университете прикладных наук в Винтертуре, и о том, как глубокое обучение может использоваться менее стандартным способом. Наконец, в *главе 10* я покажу вам, как выполнять логистическую регрессию с одним-единственным нейроном на Python — без использования TensorFlow — совершенно с нуля.

Очень надеюсь, что эта книга вам понравится, и вы получите от нее удовольствие.

ГЛАВА 1

Вычислительные графы и TensorFlow

Прежде чем погрузиться в изучение расширенных примеров далее в этой книге, вам потребуются среда программирования на языке Python и рабочие знания платформы машинного обучения TensorFlow. И данная глава поможет вам подготовить среду программирования на Python к выполнению исходного кода этой книги. После того как все будет готово, мы рассмотрим основы библиотеки машинного обучения TensorFlow.

Настройка среды программирования на языке Python

Весь исходный код в этой книге был разработан с использованием дистрибутива Python Anaconda и блокнотов Jupyter. Для того чтобы настроить дистрибутив Anaconda, сначала скачайте и установите его для своей операционной системы. (В книге использовалась ОС Windows 10, но указанный исходный код от этой операционной системы не зависит. Если хотите, то вы легко можете использовать его версию для Mac.) Вы можете получить дистрибутив Anaconda, обратившись по адресу **https://anaconda.org/**.

В правой части указанной веб-страницы вы найдете ссылку на скачивание дистрибутива Anaconda (рис. 1.1, справа вверху).

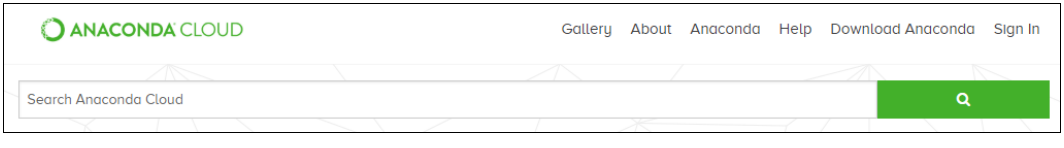

РИС. 1.1. На веб-сайте Anaconda в правом верхнем углу страницы вы найдете ссылку для скачивания необходимого программного обеспечения

Просто следуйте инструкциям по его установке. Когда после установки вы его запустите, вам будет представлен экран, показанный на рис. 1.2. В случае если этот экран вы не видите, щелкните по ссылке **Home** (Главная) на панели навигации слева.

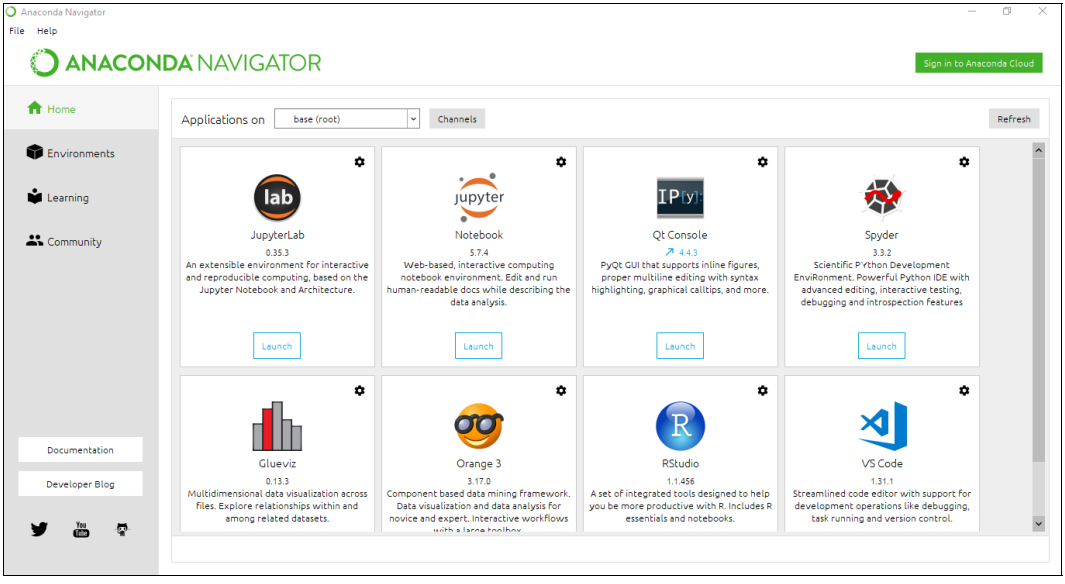

РИС. 1.2. Экран, который вы видите при запуске дистрибутива Anaconda

Программные Python-пакеты (в частности, NumPy) обновляются регулярно и очень часто. Возможно, новая версия пакета приведет к тому, что ваш код перестанет работать. Время от времени функции объявляются устаревшими, удаляются и добавляются новые. Для решения этой проблемы в Anaconda можно создавать так называемую среду. Это контейнер, который содержит определенную версию Python и конкретные версии пакетов, которые вы решили установить. Благодаря ей, например, вы можете организовать контейнер для Python 2.7 и NumPy 1.10 и еще один с Python 3.6 и NumPy 1.13. Возможно, вам придется работать с существующим кодом, который был разработан на основе Python 2.7, и, следовательно, у вас должен быть контейнер с правильной версией Python. В то же время вашим собственным проектам может потребоваться Python 3.6. С помощью контейнеров вы можете обеспечить все это одновременно. Иногда разные пакеты конфликтуют друг с другом, поэтому следует быть осторожным и избегать установки в своей среде всех тех пакетов, которые вы посчитаете интересными, в особенности, если вы разрабатываете пакеты в условиях предельного срока. Нет ничего хуже, чем обнаружить, что ваш программный код перестанет работать, и вы не знаете почему.

ПРИМЕЧАНИЕ. При определении среды старайтесь устанавливать только те программные пакеты, которые вам действительно нужны, и будьте внимательны, когда их обновляете с целью обеспечения того, чтобы никакое обновление

не нарушило ваш исходный код. (Функции объявляются устаревшими, удаляются, добавляются или часто изменяются.) Перед обновлением проверьте документацию относительно обновлений и делайте это только в том случае, если вам действительно нужен обновленный функционал.

Создать среду можно из командной строки с помощью команды `conda`, но для подготовки среды для исходного кода книги все можно проделать из графического интерфейса. Здесь будет рассмотрен именно этот метод, потому что он самый простой. Рекомендуется прочитать приведенную далее страницу из документации дистрибутива Anaconda, где подробно объясняются особенности работы внутри его среды: **https://conda.io/docs/user-guide/tasks/manage-environments.html**.

Создание среды

Давайте начнем. Сначала щелкните по ссылке **Environments** (Среды) (с маленьким значком в виде куба) на левой навигационной панели (рис. 1.3).

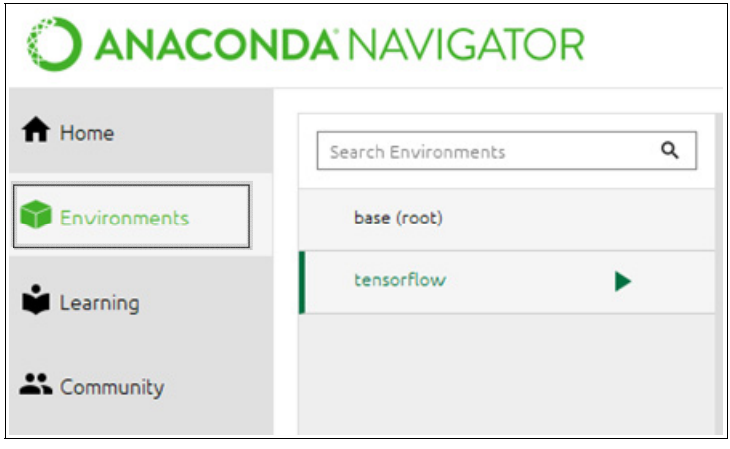

РИС. 1.3. Для того чтобы создать новую среду, сначала необходимо перейти в раздел **Environments** приложения, щелкнув по соответствующей ссылке на левой навигационной панели (обозначена на рисунке кубиком)

Затем нажмите кнопку **Create** (Создать) в средней навигационной панели (как показано на рис. 1.4).

После нажатия кнопки **Create** (Создать) откроется небольшое окно (рис. 1.5).

Вы можете выбрать любое имя. В этой книге использовалось имя tensorflow. После того как вы наберете имя, кнопка **Create** станет активной (и зеленой). Нажмите ее и подождите несколько минут до тех пор, пока не будут установлены все необходимые пакеты. Иногда может появиться всплывающее окно, сообщающее о том, что имеется новая версия дистрибутива Anaconda, и спрашивающее о том, не хотите ли вы обновить программное обеспечение. Смело нажмите кнопку **Yes**. Следуйте

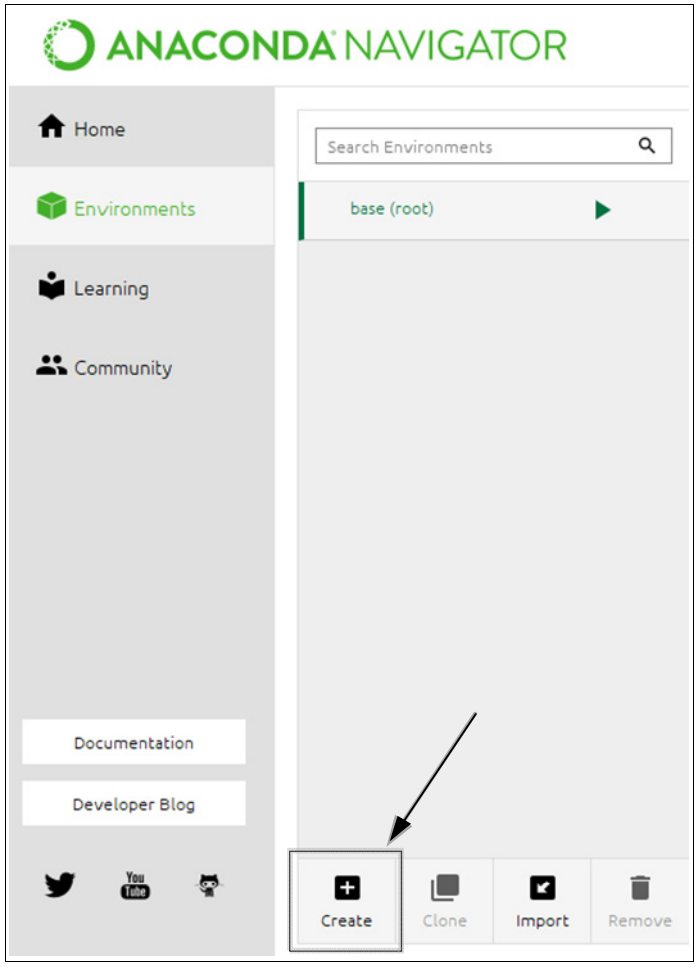

РИС. 1.4. Для того чтобы создать новую среду, необходимо нажать кнопку **Create** (обозначенную значком "плюс") на средней навигационной панели. На рисунке стрелка указывает на расположение данной кнопки

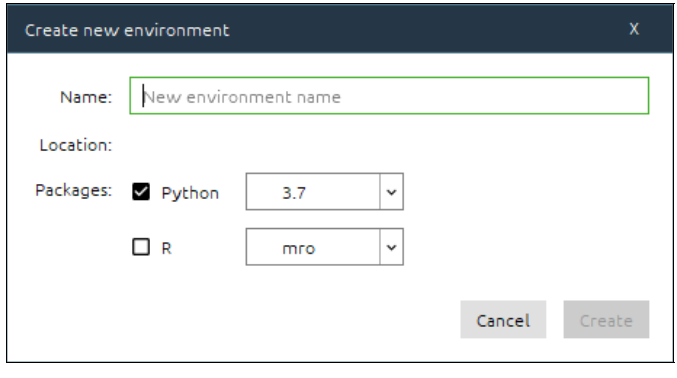

РИС. 1.5. Окно, которое вы увидите, когда нажмете кнопку **Create**, показанную на рис. 1.4

инструкциям на экране до тех пор, пока навигатор дистрибутива Anaconda не запустится снова, в случае если вы получили это сообщение и нажали кнопку **Yes**.

Мы еще не закончили. Снова щелкните по ссылке **Environments** на левой навигационной панели (см. рис. 1.3), затем щелкните на имени вновь созданной среды. Если до настоящего момента вы следовали инструкциям, то должны увидеть среду с именем "tensorflow". Через несколько секунд на правой панели вы увидите список всех установленных Python-пакетов, которые будут в вашем распоряжении в этой среде. Теперь мы должны установить несколько дополнительных пакетов: NumPy, matplotlib, TensorFlow и Jupyter. Для начала в раскрывающемся списке выберите пункт **Not installed** (Не установлены), как показано на рис. 1.6.

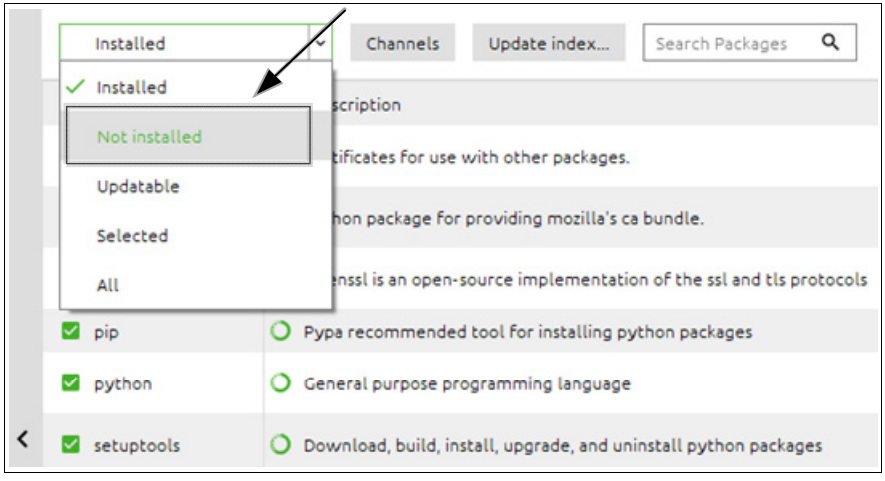

РИС. 1.6. Выбор пункта **Not installed** в раскрывающемся списке

Затем в поле **Search Packages** (Поиск пакетов) введите имя пакета, который вы хотите установить (на рис. 1.7 показано, что выбран элемент `numpy`).

Навигатор Anaconda автоматически покажет вам все пакеты, в названии или описании которых есть слово *numpy*. Щелкните на маленьком квадратике слева от имени пакета с именем *numpy*. Он станет небольшой стрелкой вниз (обозначающей, что он отмечен для установки). Затем можно нажать зеленую кнопку **Apply** (Применить) в правом нижнем углу интерфейса (рис. 1.8).

Навигатор Anaconda достаточно умен, чтобы определить, нужны ли пакету NumPy другие пакеты. Вы можете получить дополнительное окно с вопросом, можно ли установить дополнительные пакеты. Просто нажмите кнопку **Apply**. На рис. 1.9 показано, как это окно выглядит.

Для выполнения исходного кода этой книги необходимо установить следующие пакеты. (В скобках указаны версии, которые использовались для тестирования исходного кода в этой книге; никаких проблем, если это будут последующие версии.)

- numpy (1.13.3): для выполнения численных расчетов.
- matplotlib (2.1.1): для создания качественных графиков, как те, которые вы увидите в этой книге.
- scikit-learn (0.19.1): этот пакет содержит все библиотеки, связанные с машинным обучением, которые мы используем, например, для загрузки наборов данных.
- jupyter (1.0.0): позволяет использовать блокноты Jupyter.

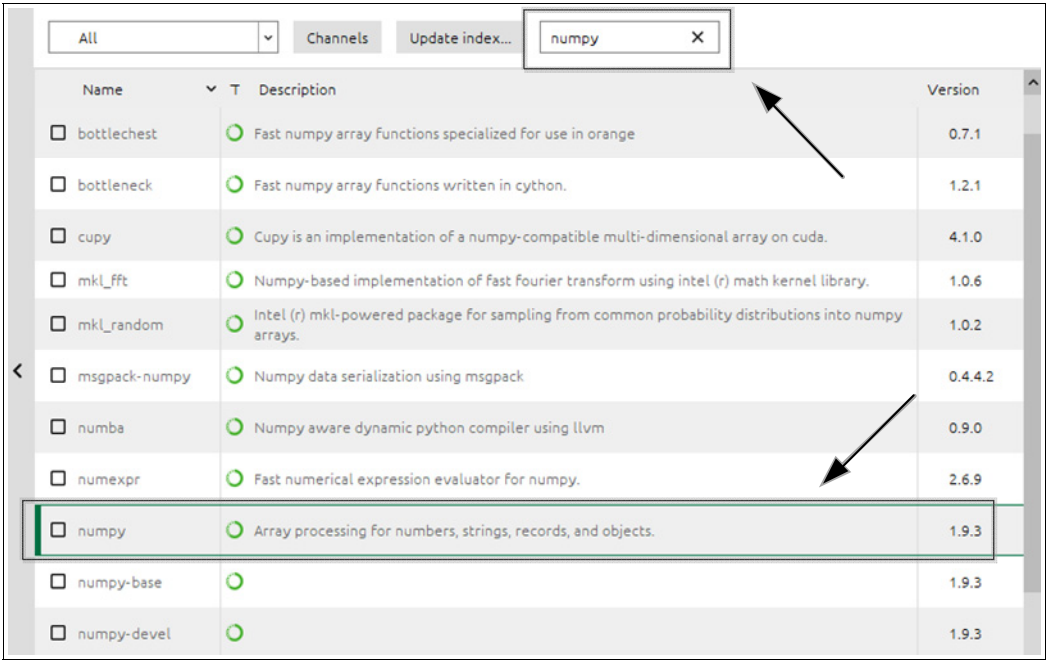

РИС. 1.7. Наберите *numpy* в поле поиска для того, чтобы включить этот пакет в репозиторий

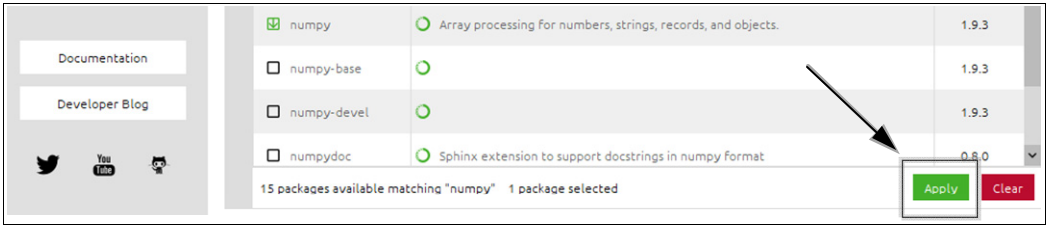

РИС. 1.8. После выбора программного пакета NumPy для установки нажмите зеленую кнопку **Apply**. Кнопка находится в правом нижнем углу интерфейса

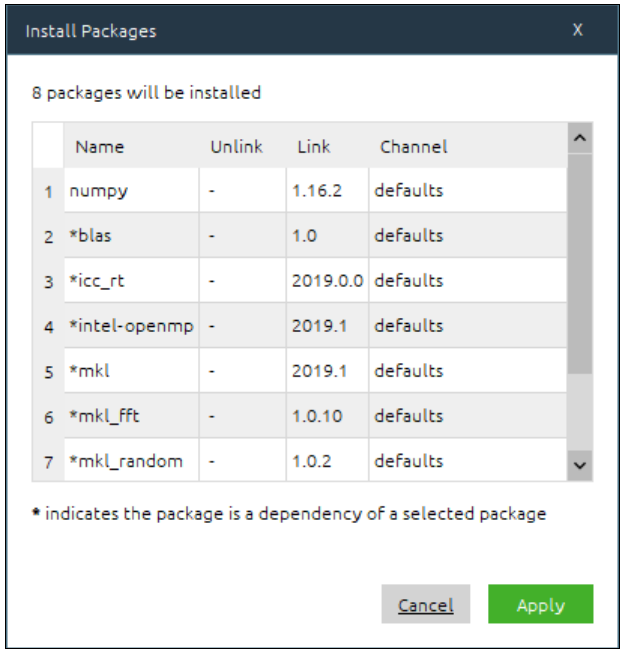

РИС. 1.9. Во время установки программного пакета навигатор Anaconda проверяет, зависит ли то, что вы хотите установить, от других еще не установленных пакетов. В таком случае будет предложено установить отсутствующие (но необходимые) пакеты из дополнительного окна. В нашем случае библиотека NumPy потребовала установки 52 дополнительных пакетов в недавно установленной системе. Просто нажмите кнопку **Apply** для того, чтобы установить их все

Установка библиотеки TensorFlow

Установка библиотеки TensorFlow проходит немного сложнее. Лучший способ сделать это — следовать инструкциям команды разработчиков TensorFlow, которые размещены по следующему адресу: **www.tensorflow.org/install/**.

На этой странице выберите вашу операционную систему, и вы получите всю необходимую информацию. Здесь будут приведены инструкции для Windows, но то же самое можно сделать, используя системы macOS или Ubuntu (Linux). Установка с Anaconda официально не поддерживается, но работает отлично (поддерживается сообществом) и представляет собой самый простой способ подготовки к работе и проверке исходного кода этой книги. В случае более продвинутых приложений можно рассмотреть другие варианты установки. (Для этого вам нужно будет обратиться на веб-сайт TensorFlow.) Для начала перейдите в меню **Пуск** в Windows и введите *anaconda*. В разделе **Лучшее соответствие** вы должны увидеть пункт **Anaconda Prompt** (Консоль Anaconda), как показано на рис. 1.10.

Запустите консоль Anaconda. В результате должен появиться интерфейс командной строки (рис. 1.11). Разница между этой и обычной консолью командной строки

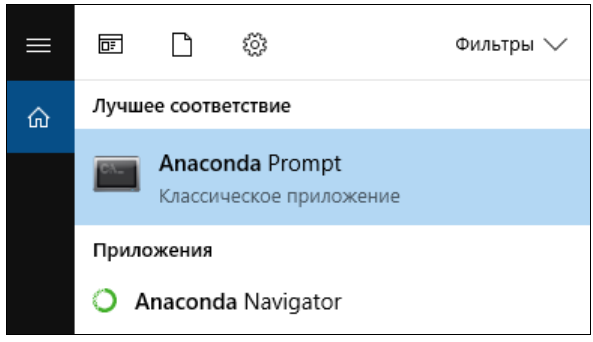

РИС. 1.10. Если в поле поиска меню **Пуск** в Windows 10 набрать *anaconda*, то можно увидеть по крайней мере две записи: **Anaconda Navigator**, где вы создали среду TensorFlow, и **Anaconda Prompt**

РИС. 1.11. Это то, что вы должны увидеть при выборе консоли **Anaconda Prompt**. Обратите внимание, что ваше пользовательское имя будет отличаться от имени на рисунке. Вы увидите не labor (мое пользовательское имя), а ваше собственное пользовательское имя

`cmd.exe` состоит в том, что здесь все команды Anaconda распознаются без настройки какой-либо переменной среды Windows.

В командной строке сначала необходимо активировать новую среду tensorflow. Это необходимо для того, чтобы дать установленной версии Python знать, в какой среде вы хотите установить TensorFlow. Для этого просто наберите следующую команду: `activate tensorflow`. Приглашение на ввод команды в вашей консоли должно измениться и выглядеть так:

```
(tensorflow) C:\Users\labor>
```

Ваше пользовательское имя будет отличаться (в приглашении на ввод команды вы увидите не labor, а ваше пользовательское имя). Будем считать, что вы установите стандартную версию TensorFlow, которая использует только CPU (а не GPU). Просто наберите следующую команду:

```
pip install --ignore-installed --upgrade tensorflow
```

Теперь дайте системе установить все необходимые пакеты. Это может занять несколько минут (в зависимости от таких факторов, как аппаратное обеспечение вашего компьютера или подключение к Интернету). Вы не должны получать никаких сообщений об ошибках. Поздравляю! Теперь у вас есть среда, в которой вы можете выполнять исходный код с использованием TensorFlow.

Блокноты Jupyter

Последний шаг, который даст возможность набирать и исполнять исходный код, состоит в запуске блокнота Jupyter. Блокнот Jupyter можно описать (согласно официальному веб-сайту) следующим образом:

> "Блокнот Jupyter — это веб-приложение с открытым исходным кодом, которое позволяет создавать и совместно использовать документы, содержащие „живой“ исходный код, уравнения, визуализации и повествовательный текст. Варианты его использования включают очистку и преобразование данных, численную симуляцию, статистическое моделирование, визуализацию данных, машинное обучение и многое другое".

В сообществе машинного обучения он применяется очень широко, и неплохо научиться его использовать. Обратитесь к веб-сайту проекта Jupyter по адресу **http://jupyter.org/**.

Этот сайт очень поучителен и содержит много примеров того, что можно делать. Весь исходный код, который вы найдете в этой книге, был разработан и протестирован с использованием блокнотов Jupyter.

Будем считать, что у вас уже есть некоторый опыт работы с этой средой веб-разработки. В случае если вам нужно освежить свои знания, рекомендуется обратиться к документации, которую можно найти на веб-сайте проекта Jupyter по следующему адресу: **http://jupyter.org/documentation.html**.

Для того чтобы запустить блокнот в вашей новой среде, необходимо вернуться в навигатор дистрибутива Anaconda в раздел **Environments** (см. рис. 1.3). Щелкните на треугольнике справа от среды "tensorflow" (если вы использовали другое имя, вам придется щелкнуть на треугольнике справа от вашей новой среды), как показано на рис. 1.12. Затем выберите пункт **Open with Jupyter Notebook** (Открыть с помощью блокнота Jupyter).

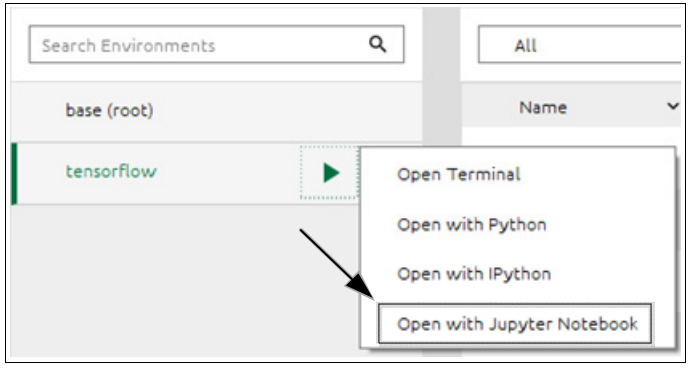

РИС. 1.12. Для того чтобы запустить записную книжку Jupyter в вашей новой среде, щелкните на треугольнике справа от имени среды "tensorflow" и выберите пункт **Open with Jupyter Notebook**

Ваш браузер запустится со списком всех папок в вашей пользовательской папке. (Если вы используете Windows, то они обычно находятся по маршруту c:\Users\ *<Имя_пользователя>*, в котором вы должны заменить *<Имя_пользователя>* на ваше собственное пользовательское имя.) Оттуда следует перейти к папке, в которой вы хотите хранить файлы ваших блокнотов и из которой вы можете создать новый блокнот, нажав кнопку **New** (Новый), как показано на рис. 1.13.

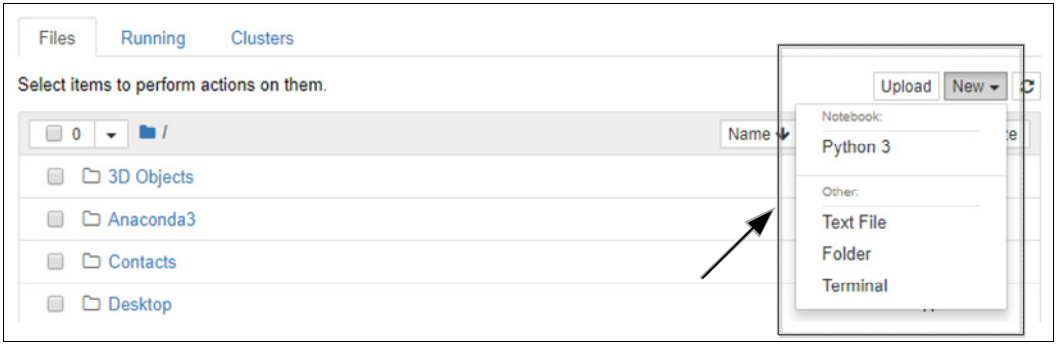

РИС. 1.13. Для того чтобы создать новый блокнот, нажмите кнопку **New** в правом верхнем углу страницы и выберите **Python 3**

Откроется новая страница, которая должна выглядеть так, как показано на рис. 1.14.

Например, вы можете набрать следующие ниже строки кода в первую "ячейку" (прямоугольное поле, в котором можно набирать текст).

```
a=1
b=2
print(a+b)
```

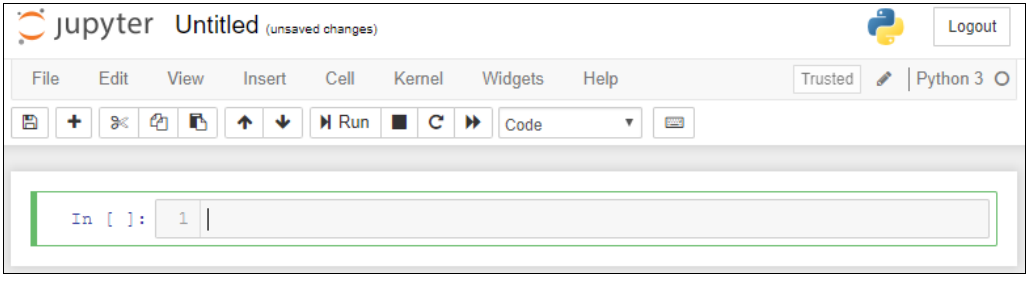

РИС. 1.14. При создании пустого блокнота откроется пустая страница, которая должна выглядеть так, как показано здесь

Для того чтобы вычислить этот исходный код, просто нажмите комбинацию клавиш <Shift>+<Enter>, и вы сразу увидите результат (3) (рис. 1.15).

Приведенный выше исходный код даст результат выражения $a+b$, т. е. 3. Новая пустая ячейка для ввода создается автоматически после получения результата. Для получения дополнительной информации о том, как добавлять комментарии, урав-

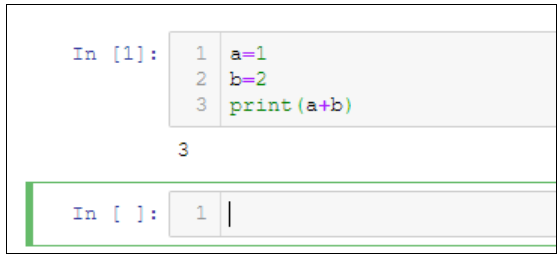

РИС. 1.15. После набора некоторого исходного кода в ячейку нажатие сочетания клавиш <Shift>+<Enter> вычислит исходный код в этой ячейке

нения, встроенные графики и многое другое, рекомендуется посетить веб-сайт Jupyter и обратиться к предоставленной там документации.

ПРИМЕЧАНИЕ. В случае если вы забыли имя папки, в которой находится блокнот, взгляните на URL-адрес страницы. Например, в моем случае, у меня **http://localhost:8888/notebooks/Documents/Data%20Science/Projects/Applied%20advanced%20deep%20learning%20(book)/chapter%201/AADL%20-%20Chapter%201%20-%20Introduction.ipynb**. Вы заметите, что URL-адрес является просто конкатенацией имен папок, в которых находится записная книжка, разделенных косой чертой. Символ %20 означает пробел. В данном случае мой блокнот находится в папке: Documents/Data Science/Projects/... и т. д. Я часто работаю с несколькими записными книжками одновременно, и полезно знать, где находится каждая из них, в случае если вы забыли ее расположение (иногда это бывает).

Элементарное введение в TensorFlow

Перед началом использования библиотеки TensorFlow вы должны понять ее философию.

Данная библиотека в значительной степени основана на концепции вычислительных графов, и если вы не понимаете, как они работают, то не сможете разобраться в том, как использовать эту библиотеку. Далее будет дано краткое введение в вычислительные графы и показано, как реализовывать простые вычисления с помощью TensorFlow. В конце следующего раздела вы должны понимать, как работает библиотека и как мы будем ее использовать в этой книге.

Вычислительные графы

Для понимания того, как работает библиотека TensorFlow, необходимо понять, что такое вычислительный граф. Вычислительный граф — это граф, в котором каждый узел соответствует операции или переменной. Переменные могут передавать свои значения в операции, а операции могут передавать свои результаты в другие операции. Обычно узлы изображают в виде круга (или многоточия) с именами перемен-

ных или операций внутри, и когда значение одного узла является входом для другого узла, стрелка проходит от одного узла к другому. Простейший граф, который только может быть, — это граф с единственным узлом, который представляет собой одну-единственную переменную. (Узел может быть переменной или операцией.) Граф на рис. 1.16 вычисляет значение переменной x.

РИС. 1.16. Простейший граф, который только можно построить, показывающий простую переменную

Не очень-то интересно. Теперь рассмотрим нечто посложнее, например сумму двух переменных x и y: $z = x + y$. Ее можно представить так, как на графе (рис. 1.17).

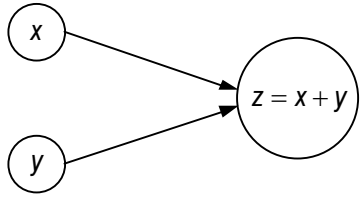

РИС. 1.17. Элементарный вычислительный граф для суммы двух переменных

Узлы слева на рис. 1.17 (узлы с x и y внутри) являются переменными, в то время как больший узел обозначает сумму двух переменных. Стрелки показывают, что две переменные, x и y, являются входами для третьего узла. Граф должен быть прочитан (и вычислен) в топологическом порядке, т. е. вы должны следовать по стрелкам, которые будут указывать, в каком порядке вы должны вычислять различные узлы. Стрелки также будут сообщать о зависимостях между узлами. Для того чтобы вычислить z, сначала необходимо вычислить x и y. Можно также сказать, что узел, выполняющий суммирование, зависит от входных узлов.

Важным для понимания аспектом является то, что такой граф определяет только те операции (в данном случае суммирование), которые нужно выполнить над двумя входными значениями (в данном случае x и y) для получения результата (в данном случае z). Таким образом определяется ответ на вопрос "как?". Вы должны закрепить значения за входами x и y, а затем выполнить суммирование и на выходе получить z. Данный граф даст результат, только когда вы вычислите все узлы.

ПРИМЕЧАНИЕ. В этой книге под словом "конструирование" графа подразумевается ситуация, когда мы будем определять то, что делает каждый узел, а под словом "оценивание" графа — ситуация, когда мы будем фактически вычислять участвующие операции.

Этот аспект очень важен для понимания. Обратите внимание, что входные переменные не обязательно должны быть вещественными числами. Они могут быть