Содержание

Об авторах

Бастиан Шарден – исследователь-аналитик и создатель с опытом работы в области искусственного интеллекта и математики. Имеет степень магистра по когнитивистике, полученную им в университете Лейдена вместе с очными курсами в Массачусетском технологическом институте (MIT). В течение прошедших 5 лет он работал над широким спектром проектов в области науки о данных и искусственного интеллекта. Часто принимает участие как непостоянный член преподавательского состава в онлайн-курсах Coursera в области анализа социальных сетей Мичиганского университета и практического машинного обучения Университета Джонса Хопкинса. Его предпочтительными языками программирования являются Python и R. В настоящее время является соучредителем компании Quandbee (http://www.quandbee.com/), предлагающей масштабируемые приложения машинного обучения и искусственного интеллекта.

Лука Массарон – исследователь-аналитик и директор по анализу рынка, специализирующийся на многомерном статистическом анализе, машинном обучении и потребительском понимании, с более чем десятилетним опытом в решении реальных задач и генерировании стоимости для заинтересованных сторон путем применения логического обоснования, статистики, анализа данных и алгоритмов. Являясь первопроходцем в сфере исследования веб-аудитории в Италии, сегодня входит в лучшую десятку аналитиков по версии профессионального аналитического веб-сайта Kaggle. Всегда очень увлекался всем, что связано с данными и их анализом, а также демонстрацией потенциала, управляемого данными обнаружения знаний для экспертов и неспециалистов. Отдавая предпочтение простоте над ненужной сложностью, он полагает, что в науке о данных можно достичь многого, всего лишь руководствуясь основами.

Хотел бы благодарить Юкико и Амелию за их постоянную поддержку, помощь и любвеобильное терпение.

Альберто Боскетти – исследователь-аналитик с профессиональным опытом в области обработки сигналов и математической статистики. Имеет научную степень доктора наук по инженерному делу в области телекоммуникаций, в настоящее время живет и работает в Лондоне. В своих рабочих проектах он занимается решением сложных задач, которые включают в себя широкий круг областей – от обработки естественного языка (NLP) вплоть до машинного обучения и распределенной обработки. Очень увлечен своей работой и всегда пытается оставаться в курсе технологических новинок в области науки о данных, участвуя во встречах профессионалов, конференциях и других событиях.

О рецензентах

Олег Окунь – эксперт по машинному обучению, является автором/редактором четырех книг, многочисленных журнальных статей и докладов на конференциях. Работает в отрасли уже более четверти века, проработав в высшей школе и промышленности на своей родине в Белоруссии и за границей (в Финляндии, Швеции и Германии). Его опыт работы включает анализ цифровых образов документов, биометрию цифрового отпечатка, биоинформатику, онлайновую и внесетевую маркетинговую аналитику и аналитику рейтинга кредитоспособности. Интересуется всеми аспектами распределенного машинного обучения и Интернетом вещей. Олег в настоящее время живет и работает в Гамбурге, Германия, и собирается приступить к новой работе в качестве главного архитектора интеллектуальных систем. Его предпочтительными языками программирования являются Python, R и Scala.

Хотел бы выразить свою самую глубокую благодарность моим родителям за все, что они для меня сделали.

Кай Лонденберг – исследователь-аналитик и эксперт в области больших данных с многолетним профессиональным опытом. В настоящее время работает исследователем-аналитиком в компании Volkswagen Data Lab. До этого имел удовольствие работать ведущим исследователем-аналитиком в компании Searchmetrics, где Лука Массарон был членом его команды. Кай любит работать с передовыми технологиями, и, являясь прагматично настроенным практиком машинного обучения и действующим разработчиком программного обеспечения, он любит всегда оставаться в курсе последних технологических и научных достижений в области машинного обучения, ИИ и смежных областях. Его можно найти в социальной сети LinkedIn по адресу https://www.linkedin.com/in/kailondenberg.

Предисловие

> «Самое приятное в мозге – то, что можно узнать, что невежество вытесняется знанием и что крохотные частички знания постепенно образуют солидные пирамиды».
>
> *– Дуглас Хофштадтер*

Машинное обучение часто называют *подобластью искусственного интеллекта, которая реально работает*. Его цель состоит в том, чтобы, основываясь на существующем подмножестве данных (тренировочном наборе), с максимально возможной точностью найти функцию для предсказания исходов подмножества ранее не наблюдавшихся данных (тестового набора). Это происходит либо в форме меток и классов (задачи классификации), либо в форме непрерывного значения (задачи регрессии). Материальные примеры машинного обучения в реальных приложениях простираются от предсказания будущих курсов акций до идентификации половой принадлежности автора, исходя из серии документов. В этой книге читатель получит объяснение самых важных принципов работы машинного обучения, а также методов, подходящих для обработки крупных наборов данных, благодаря практическим примерам на языке Python. Мы рассмотрим обучение с учителем (классификация и регрессия) и обучение без учителя (в т. ч. анализ главных компонент, кластеризацию данных и тематическое моделирование), которые оказались применимыми к большим наборам данных.

Работающие в области ИТ крупные корпорации, такие как Google, Facebook и Uber, подняли нешуточный ажиотаж, утверждая, что они успешно применили подобного рода методы машинного обучения в крупном масштабе. С появлением и распространением больших данных спрос на масштабируемые решения в области машинного обучения рос экспоненциально, и множество других компаний и отдельных специалистов устремилось на поиски зрелых плодов скрытых корреляций в больших наборах данных. К сожалению, большинство обучающихся алгоритмов не очень хорошо масштабируется, перегружая центральные процессоры и память настольного компьютера или большого вычислительного кластера. В нынешние времена, несмотря на то что *большие данные* преодолели пик ажиотажа, масштабируемых решений для машинного обучения до сих пор не так уж и много.

Откровенно говоря, нам по-прежнему приходится обходить немало узких мест даже с наборами данных, которые едва ли можно классифицировать как *большие данные* (речь идет о наборах данных объемом до 2 Гб или меньше). Главная задача настоящей книги состоит в том, чтобы предоставить способы (иногда нестандартные) для применения самых мощных методов машинного обучения с открытым исходным кодом в более крупном масштабе без привлечения дорогостоящих корпоративных решений или больших вычислительных кластеров. На протяжении всей книги мы будем использовать Python и некоторые другие легкодоступные решения, которые хорошо интегрируются в масштабируемых конвейерах машинного обучения. Чтение книги станет путешествием, которое переопределит все то,

что вы знали о машинном обучении, позволив вам сделать значительный рывок вперед в анализе по-настоящему больших данных.

О чем эта книга рассказывает

Глава 1 «*Первые шаги на пути к масштабируемости*» ставит задачу масштабируемого машинного обучения с правильной перспективой и знакомит с инструментами, которые мы будем использовать в этой книге.

Глава 2 «*Масштабируемое обучение в Scikit-learn*» обсуждает тему стратегии стохастического градиентного спуска с ограничением потребления оперативной памяти; этот прием основывается на теме *внеядерного* обучения. Мы также охватим технические приемы подготовки данных, включая хэширование признаков, способные справляться с самыми разнообразными данными.

Глава 3 «*Быстрообучающиеся реализации машин SVM*» посвящена потоковым алгоритмам, которые способны обнаруживать нелинейности в форме машин опорных векторов. Мы представим альтернативы программной библиотеке Scikit-learn, в частности программы LIBLINEAR и Vowpal Wabbit, которые, несмотря на то что выполняются как команды внешней оболочки, с легкостью обертываться в сценарии Python и ими управляются.

Глава 4 «*Нейронные сети и глубокое обучение*» охватывает полезную методику применения глубоких нейронных сетей в платформе Theano и крупномасштабные приложения на основе платформы H2O. Хотя эта тема сегодня является актуальной, их успешное применение может представлять серьезную трудность, уже не говоря о предложении масштабируемых решений. Мы также обратимся к предварительной тренировке без учителя с автокодировщиками на основе библиотеки theanets.

Глава 5 «*Глубокое обучение с библиотекой TensorFlow*» охватывает интересную методику глубокого обучения нейронных сетей и онлайновые методы. Хотя платформа TensorFlow находится в стадии становления, она предоставляет изящные решения для машинного обучения. Мы также воспользуемся возможностями библиотеки Keras по работе со сверточными нейронными сетями в среде TensorFlow.

Глава 6 «*Классификационные и регрессионные деревья в крупном масштабе*» посвящена объяснению масштабируемых решений с использованием алгоритмов случайного леса, градиентного бустинга и экстремального градиентного бустинга XGboost. Алгоритм машинного обучения CART, аббревиатура для классификационных и регрессионных деревьев, обычно применяется в рамках ансамблевых методов. Мы также предоставим примеры крупномасштабного приложения в среде H2O.

Глава 7 «*Обучение без учителя в крупном масштабе*» вплотную посвящена обучению без учителя, а именно будут рассмотрены алгоритмы PCA, кластерного анализа и тематического моделирования с использованием соответствующего подхода, позволяющего масштабировать их вертикально.

Глава 8 «*Распределенные среды – Hadoop и Spark*» научит настраивать систему Spark в среде виртуальной машины с переходом от одиночной машины к парадигме вычислительной сети. Поскольку Python способен с легкостью связывать индивидуальные наработки и подключать их к кластеру машин, привлечение возможностей кластера Hadoop становится простым делом.

Глава 9 «*Практическое машинное обучение на платформе Spark*» посвящена непосредственной работе с платформой Spark и обучает необходимым основам, для того чтобы начать напрямую управлять данными и создавать прогнозные модели на больших наборах данных.

Приложение «Введение в графические процессоры и библиотеку Theano» коснется основ работы в среде Theano и вычислений на GPU. Это пособие поможет установить и подготовить свою собственную среду для использования Theano на GPU, если, конечно, располагаемая система это позволяет сделать.

Что требуется для этой книги

Выполнение примеров кода, прилагаемых к данной книге, требует установки Python 2.7 или старших версий в Mac OS, Linux или Microsoft Windows.

В примерах по всей книге будут часто использоваться важные программные библиотеки Python для научных и статистических вычислений, в частности SciPy, NumPy, Scikit-learn, StatsModels и реже matplotlib и pandas. Мы также задействуем приложение для внеядерных облачных вычислений под названием H2O.

Эта книга в значительной мере зависит от интерактивной среды программирования Jupyter и ее так называемых блокнотов, или записных книжек, приводимых в действие ядром Python. Для этой книги мы воспользуемся ее последней версией 4.1 (на момент публикации перевода – 5.0).

Первая глава предоставит все пошаговые инструкции и некоторые полезные советы по настройке среды Python, упомянутых выше рабочих программных библиотек и всего необходимого инструментария.

Для кого эта книга

Эта книга предназначена для начинающих и действующих практиков в области науки о данных, разработчиков и всех тех, кто намеревается работать с большими и сложными наборами данных. Мы постарались сделать эту книгу максимально доступной для самой широкой аудитории. И все же, учитывая, что темы в этой книге достаточно продвинутые, читателям рекомендуется знать основные понятия из области машинного обучения, в частности классификацию и регрессию, функции минимизации ошибки и перекрестную проверку, однако это условие не является обязательным.

Мы также предполагаем, что читатель обладает некоторым опытом программирования на Python, работы с блокнотами Jupyter и выполнением команд из командной строки, а также познаниями в математике на разумном уровне, чтобы усвоить концепции, лежащие в основе разных больших решений, которые мы здесь предлагаем. Текст написан в стиле, который поймут программисты на других языках (R, Java и MATLAB). В идеальном случае он очень подходит для исследователя-аналитика (но не ограничен лишь им), знакомого с машинным обучением и заинтересованного в мобилизации возможностей языка Python, в отличие от других языков, таких как R или MATLAB, из-за его широких возможностей выполнять вычислительные задачи, работать с оперативной памятью и выполнять ввод-вывод данных.

Условные обозначения

В этой книге вы найдете несколько текстовых стилей, которые выделяют различные виды информации. Вот некоторые примеры этих стилей и объяснение их значения.

Ключевые слова в тексте, имена таблиц баз данных, имена папок, имена файлов, расширения файлов, пути файловой системы, фиктивные URL, ввод данных пользователем и дескрипторы Twitter показаны следующим образом: «Во время обследования линейной модели сначала проверьте атрибут `coef_`».

Фрагмент исходного кода оформляется следующим образом:

```
from sklearn import datasets
iris = datasets.load_iris()
```

Поскольку в большинстве примеров мы будем использовать блокноты Jupyter, фрагменты исходного кода будут оформляться в соответствии с тем, как он выглядит в ячейках блокнотов: входные данные будут всегда помечены как `In:`, а выходные данные – часто как `Out:`. На компьютере требуется только ввести исходный код после `In:` и проверить, соответствуют ли результаты содержимому `Out:`:

```
In:
clf.fit(X, y)

Out:
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
degree=3, gamma=0.0, kernel='rbf', max_iter=-1, probability=False,
random_state=None, shrinking=True, tol=0.001, verbose=False)
```

Когда команда предназначена для выполнения в командной строке терминала, такая команда будет предваряться префиксом `$>`, в противном случае, если она предназначена для интепретатора Python REPL, ей будет предшествовать приглашение `>>>`:

```
$ python
>>> import sys
>>> print sys.version_info
```

Новые термины и **важные слова** показаны полужирным шрифтом. Слова, которые выводятся на экран, например в меню или диалоговых окнах, выглядят в тексте следующим образом: «Как правило, необходимо просто ввести в ячейки после **In:** исходный код и его выполнить».


Предупреждения или важные примечания появляются в этом поле.


Подсказки и приемы появляются тут.


Дополнения к тексту оригинала книги.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф на странице с описанием соответствующей книги.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг — возможно, ошибку в тексте или в коде — мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в Интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Packt очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в Интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или вебсайта, чтобы мы могли применить санкции.

Пожалуйста, свяжитесь с нами по адресу электронной почты dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

Комментарий переводчика

Весь материал книги приведен в соответствие с последними действующими версиями программных библиотек (время перевода книги – май-июнь 2017 г.) и протестирован в среде Windows 10. При тестировании исходного кода за основу взят Python версии 2.7.13.

Большинство содержащихся в книге технических терминов и аббревиатур для удобства кратко определено в сносках, а для некоторых терминов в силу отсутствия единой терминологии приведены соответствующие варианты наименований или пояснения.

Прилагаемый к книге адаптированный и скорректированный исходный код примеров лучше всего разместить в подпапке домашней папки пользователя (`/home/Ваши_проекты_Python` или `C:\Users\[ИМЯ_ПОЛЬЗОВАТЕЛЯ]\Ваши_проекты_Python`). Ниже приведена структура папки с прилагаемыми примерами:

`Chapter 01-09`	Исходный код примеров в виде записных книжек Jupyter
`py_scripts`	Исходный код примеров в виде сценариев Python
`vowpal_wabbit_for_windows`	Исполнимые файлы программного продукта Vowpal Wabbit для 32- и 64-разрядных ОС Windows

Для просмотра исходного кода примеров лучше всего пользоваться блокнотами Jupyter. Они более читабельны, содержат графики, цветные рисунки и расширенные пояснения.

Далее приведены особенности инсталляции некоторых используемых программных библиотек Python.

Особенности программного обеспечения

В обычных условиях библиотеки Python можно скачать из каталога библиотек Python `PyPi` (https://pypi.python.org/). Однако следует учесть, что для работы библиотек SciPy и Scikit-learn в Windows 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. Ее следует скачать с репозитория whl-файлов (http://www.lfd.uci.edu/~gohlke/pythonlibs/) и установить (например, `pip install numpy-1.13.0+mkl-cp27-cp27m-win_amd64.whl` для 64-разрядной операционной системы Windows и среды Python 2.7) как whl (соответствующая процедура установки описана ниже).

Далее приведены сведения о других основных библиотеках:

- **NumPy** – основополагающая библиотека, необходимая для научных вычислений на Python;
- **Matplotlib** – библиотека для работы с двумерными графиками. Требует наличия numpy и некоторых других;
- **Pandas** – инструмент для анализа структурных данных и временных рядов. Требует наличия numpy и некоторых других;
- **Scikit-learn** – интегратор классических алгоритмов машинного обучения. Требует наличия numpy+mkl;
- **SciPy** – библиотека, используемая в математике, естественных науках и инженерном деле. Требует наличия numpy+mkl;
- **Jupyter** – интерактивная вычислительная среда.

Факультативно:

- **PyQt5** – библиотека инструментов для программирования визуального интерфейса, требуется для работы инструментальной среды программирования Spyder;
- **Spyder** – инструментальная среда программирования на Python.

Протокол установки программного обеспечения

```
python -m pip install --upgrade pip
pip install numpy
```

либо как whl:

```
pip install numpy-1.13.0+mkl-cp27-cp27m-win_amd64.whl
pip install scipy
```

либо как whl:

```
pip install scipy-0.19.0-cp27-cp27m-win_amd64.whl
pip install Scikit-learn
```

либо как whl:

```
pip3 install scikit_learn-0.18.1-cp27-cp27m-win_amd64.whl
pip install matplotlib
pip install pandas
pip install gensim
pip install jupyter
pip install neurolab
pip install theano
pip3 install tensorflow
```

Следует отметить, что библиотека TensorFlow НЕ работает с Python 2. Для работы с библиотекой TensorFlow необходимо инсталлировать Python 3. Блокнот Jupyter (гл. 5) с примерами применения библиотеки TensorFlow использует ядро Python 3.

```
pip install scikit-neuralnetwork
pip install h2o
pip install keras
```

Факультативно:

```
pip install pyqt5
pip install spyder
```

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 июнь 2017 г. версии для 64-разрядной ОС Windows и Python версии 2.7.

Работа со связкой Hadoop/Vagrant/Jupyter

Начало работы:

```
vagrant up (в папке с файлом Vagrantfile)
set PATH=%PATH%;C:\Program Files\Git\usr\bin (в командной строке cmd)
vagrant ssh
vagrant@sparkbox:~$ ./start_hadoop.sh (в консоли vagrant)
vagrant@sparkbox:~$ ./start_jupyter_yarn.sh
http://localhost:8888 (в браузере)
```

Запуск автономного режима без поддержки менеджера YARN:

```
vagrant up (в папке с файлом Vagrantfile)
set PATH=%PATH%;C:\Program Files\Git\usr\bin (в командной строке cmd)
vagrant ssh
vagrant@sparkbox:~$ ./start_jupyter_yarn.sh
http://localhost:8888 (в браузере)
```

Конец работы с Hadoop/Vagrant/Jupyter:

```
Ctrl + C (в командной строке блокнота)
vagrant@sparkbox:~$ ./stop_hadoop.sh (в консоли vagrant)
vagrant@sparkbox:~$ exit
vagrant halt (в командной строке)
```

Добавления в системную переменную Path в ОС Windows

После установки соответствующего программного обеспечения следует проверить наличие следующих значений в системной переменной Path:

```
%HADOOP_HOME%\bin
C:\HashiCorp\Vagrant\bin
C:\Program Files\Git\cmd
C:\Program Files\Git\usr\bin
C:\Program Files\mingw-w64\x86_64-7.1.0-win32-seh-rt_v5-rev0\mingw64\bin
```

Установка библиотек Python из whl-файла

Библиотеки для Python можно разрабатывать не только на чистом Python. Довольно часто библиотеки пишутся 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-0.19.0-cp27-cp27m-win_amd64.whl
```

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

```
pip install scipy-0.19.0-cp27-cp27m-win_amd64.whl
```

При выборе библиотеки важно, чтобы разрядность устанавливаемой библиотеки и разрядность интерпретатора совпадали. Пользователи Windows могут брать whl-файлы на веб-странице http://www.lfd.uci.edu/~gohlke/pythonlibs/ Кристофа Голька из Лаборатории динамики флуоресценции Калифорнийского университета в г. Ирвайн. Библиотеки там постоянно обновляются, и в архиве содержатся все, какие только могут понадобиться.

Установка и настройка инструментальной среды Spyder

Spyder – это инструментальная среда для научных вычислений для языка Python (Scientific PYthon Development EnviRonment) для Windows, Mac OS X и Linux. Это

простая, легковесная и бесплатная интерактивная среда разработки на Python, которая предлагает функционал, аналогичный среде разработки на MATLAB, включая готовые к использованию виджеты `PyQt5` и `PySide`: редактор исходного кода, редактор массивов данных `NumPy`, редактор словарей, консоли Python и IPython и многое другое.

Чтобы установить среду Spyder в Ubuntu Linux, используя официальный менеджер библиотек, нужна всего одна команда:

```
sudo apt-get install spyder
```

Чтобы установить с использованием менеджера библиотек pip:

```
sudo apt-get install python-qt5 python-sphinx
sudo pip install spyder
```

И чтобы обновить:

```
sudo pip install -U spyder
```

Установка среды Spyder в Fedora 25:

```
dnf install python-spyder
```

Установка среды Spyder в Windows:

```
pip install spyder
```

Примечание: среда Spyder требует обязательной установки библиотеки PyQt5.

Глава 1

Первые шаги к масштабируемости

Добро пожаловать в книгу по масштабируемому машинному обучению на Python.

В этой главе мы обсудим способы эффективного обучения на больших данных в среде Python и как это можно осуществить, используя всего одну машину или кластер из других машин, который, к примеру, можно получить в веб-службах облачных вычислений **Amazon Web Services** (AWS) или в веб-службах платформы Google-облако.

В настоящей книге мы будем использовать реализацию масштабируемых алгоритмов машинного обучения на языке Python. Иными словами, они смогут работать с большим объемом данных и не дадут сбоя из-за нехватки оперативной памяти. Кроме того, работа таких алгоритмов будет занимать разумное количество времени, достаточно приемлемое для прототипа в области науки о данных и для развертывания проекта в эксплуатационной среде. Главы книги организованы вокруг решений (таких как потоковая передача данных), алгоритмов (таких как нейронные сети или ансамбль деревьев) и платформ (Hadoop или Spark). Мы также предложим небольшой справочник по алгоритмам машинного обучения и объясним, как сделать их масштабируемыми и пригодными для решения задач с крупными наборами данных.

С учетом таких стартовых предпосылок вам потребуется изучить основы (чтобы уяснить перспективу, с которой эта книга была написана), а также установить и настроить все основные инструменты, которые позволят незамедлительно приступить к чтению глав.

В этой главе мы представим следующие темы:

- что в действительности означает термин «масштабируемость»;
- на какие узкие места необходимо обратить внимание во время работы с данными;
- какие задачи эта книга поможет решать;
- как использовать Python для эффективного анализа наборов данных в крупном масштабе;
- каким образом быстро настроить свою машину для выполнения представленных в этой книге примеров.

Давайте же начнем наше совместное путешествие по масштабируемым решениям в среде Python!

Подробное объяснение термина масштабируемости

Несмотря на весь нынешний ажиотаж вокруг больших данных, большие наборы данных существовали задолго до того, как был введен сам термин. Большое количество текстовых данных, последовательностей ДНК и огромное количество данных с радиотелескопов всегда представляли проблему для ученых и исследователей-аналитиков. Поскольку большинство алгоритмов машинного обучения имеет вычислительную сложность $O(n^2)$ или даже $O(n^3)$, где n – это число тренировочных прецедентов, исследователи и аналитики прежде решали поставленные крупными наборами данных сложные задачи, привлекая более эффективные алгоритмы обработки данных. Алгоритм машинного обучения считается масштабируемым, когда после соответствующей настройки он может работать в условиях больших наборов данных. Набор данных может быть большим в силу большого количества прецедентов либо переменных, либо в силу обоих причин, а масштабируемый алгоритм может справляться с ними эффективно, поскольку его время выполнения увеличивается почти линейно в соответствии с размером задачи. Следовательно, это просто вопрос обмена в соотношении 1:1 большего количества времени (или большей вычислительной мощи) на большее количество данных. Между тем обычный алгоритм машинного обучения, когда сталкивается с большими объемами данных, не масштабируется; он попросту прекращает работать либо работает со временем выполнения, которое увеличивается нелинейно, например экспоненциально, тем самым делая обучение неосуществимым.

Внедрение дешевых систем хранения данных, больших RAM и многоядерных CPU кардинально все изменило, увеличив возможности одиночных ноутбуков по анализу больших объемов данных. Появление в недавнем прошлом еще одного игрока стало переломным моментом, переориентировав внимание с одиночных мощных машин на кластеры серийных компьютеров (более дешевых и легкодоступных). Эта серьезная перемена обусловила внедрение вычислительной сетевой парадигмы **MapReduce** и платформы с открытым исходным кодом Apache Hadoop с ее **распределенной файловой системой Hadoop** HDFS (Hadoop distributed file system) и в целом параллельных вычислений в компьютерных сетях.

Чтобы выяснить, каким образом обе эти перемены глубоко и положительно повлияли на возможности решения крупномасштабных задач, прежде всего следует выяснить, что на самом деле мешало (и по-прежнему мешает, в зависимости от того, насколько крупной является решаемая задача) выполнять анализ крупных наборов данных.

Независимо от того, какую задачу вы решаете, в конечном счете вы обнаружите, что не можете выполнить анализа своих данных в силу следующих ниже ограничений:

- вычислительная емкость оказывает влияние на время, затрачиваемое на выполнение анализа;
- емкость каналов ввода-вывода данных влияет на то, сколько данных может быть передано за единицу времени из хранилища данных в оперативную память машины;
- емкость оперативной памяти влияет на то, насколько большими будут данные, которые можно обработать за один раз.

Ваш компьютер имеет ограничения, которые будут определять, сможете ли вы обучиться на данных и сколько потребуется времени, прежде чем вы упретесь в стену. Вычислительные ограничения бывают во многих вычислительно емких расчетах, проблемы, связанные с вводом-выводом, образуют узкое место для быстрого доступа к данным, и, наконец, ограничения по памяти могут вынудить принимать лишь часть данных, тем самым ограничивая возможности матричных вычислений, к которым можно было бы обратиться, либо прецизионность или даже строгость получаемых оценок.

Каждое из этих аппаратных ограничений будет также оказывать разное влияние по степени серьезности относительно анализируемых данных:

- высокие данные, отличительная особенность которых состоит в том, что они имеют большое количество прецедентов;
- широкие данные, которые характерны тем, что имеют большое количество признаков;
- высокие и широкие данные, которые имеют одновременно большое количество прецедентов и признаков;
- разреженные данные, которые отличаются тем, что имеют большое количество нулевых записей или записей, которые можно преобразовать в нули (т. е. матрица данных может быть высокой и/или широкой, но информативной, при этом не все записи в матрице имеют информационное наполнение).

И в конце дело сводится к алгоритму, который вы собираетесь использовать, чтобы обучиться на данных. Каждый алгоритм имеет свои собственные свойства и способен преобразовывать данные, используя решение, на которое по-разному воздействует смещение или дисперсия. Следовательно, относительно задачи, которую вы до сих пор решали при помощи машинного обучения, вы рассчитывали, что определенные алгоритмы могут работать лучше других, основываясь при этом на своем опыте или эмпирической проверке. В случае с крупномасштабными задачами при выборе алгоритма необходимо добавить еще несколько совсем других соображений:

- какова вычислительная сложность алгоритма, т. е. влияет ли число строк и столбцов в данных на число вычислений линейным или нелинейным образом. Большинство решений в области машинного обучения основывается на алгоритмах квадратичной или кубической сложности, тем самым строго ограничивая их применимость к большим данным;
- сколько в модели параметров; здесь дело не просто в проблеме дисперсии оценок (переподгонке), а во времени, которое может потребоваться на их вычисление;
- можно ли параллелизировать процессы оптимизации, т. е. можно ли легко разделить вычисления между многочисленными узлами или ядрами CPU, или же приходится опираться на одиночный последовательный процесс оптимизации;
- должен ли алгоритм обучаться сразу на всех данных, или же вместо этого можно использовать одиночные примеры либо небольшие пакеты данных.

Если перекрестно оценить аппаратные ограничения и свойства данных, с одной стороны, и подобного рода алгоритмы – с другой, то можно получить множество возможных проблемных сочетаний, которые могут стать препятствием для полу-

чения результатов от проведения крупномасштабного анализа. С практической точки зрения все проблемные сочетания можно решить на основе трех подходов:

- вертикального масштабирования, т. е. улучшения производительности одиночной машины путем модификации программного обеспечения и/или оборудования (больше оперативной памяти, более быстрые CPU и дисковая память, а также использование модулей GPU);
- горизонтального масштабирования, т. е. распределения вычислений (и производительности) по многочисленным машинам с привлечением внешних ресурсов, а именно другой дисковой памяти и других модулей CPU (либо GPU);
- вертикального и горизонтального масштабирования, т. е. взятия лучшего решения из вертикальных и горизонтальных решений вместе взятых.

Приведение крупномасштабных примеров

Несколько мотивирующих примеров прояснит ситуацию и сделает ее запоминающейся.

Возьмем два простых примера:

- способность предсказывать **кликабельность** (click-through-rate, CTR), т. е. отношение числа щелчков к числу показов. В наши дни, когда интернет-реклама настолько распространилась вширь и вглубь, что отъедает значительные куски у традиционных СМИ, она помогает довольно много зарабатывать;
- способность предложить правильную информацию своим клиентам, когда они ищут предлагаемые вашим сайтом продукты и услуги, может по-настоящему улучшить ваши возможности их продавать, в случае если вы сможете угадывать, что помещать во главу результатов их поискового запроса.

В обоих случаях мы располагаем довольно большими наборами данных, которые продуцируются пользователями в результате их взаимодействия в Интернете.

В зависимости от бизнеса, который мы имеем в виду (тут можно вообразить некоторых крупных игроков), в обоих указанных случаях мы, очевидно, говорим о миллионах точек данных в день. В случае рекламы данные, разумеется, являются высокими, потому что они представляют собой непрерывный поток информации с заменой старых данных на новые, в большей мере отражающих рынки и потребителей. В случае поисковой системы данные являются широкими, дополняясь компонентом, предоставляемым результатами, которые вы предложили своим клиентам: например, если вы занимаетесь туристическим бизнесом, то у вас будет довольно много признаков об отелях, местах посещений и предлагаемых услугах.

Безусловно, масштабируемость создает трудности в обеих этих задачах:

- необходимо обучаться на данных, которые растут каждый день, и причем обучаться быстро, потому что, пока вы обучаетесь, продолжают поступать новые данные. При этом вам приходится иметь дело с данными, которые, очевидно, не смогут уместиться в оперативной памяти, потому что матрица слишком высокая или слишком большая;
- необходимо часто выполнять обновления модели машинного обучения с целью размещения новых данных. Для этого потребуется алгоритм, который может обрабатывать информацию в нужные сроки. Вычислительную сложность $O(n^2)$ или $O(n^3)$ практически невозможно обработать ввиду ко-