Содержание

Об авторе

Юси (Хэйден) Лю – опытный специалист по обработке данных, специализирующийся на разработке моделей и систем машинного и глубокого обучения. Он работал в различных предметных областях, применяя свои познания в обучении с подкреплением. С удовольствием преподает и является автором ряда книг по машинному обучению. Его первая книга «Python Machine Learning By Example» была бестселлером Amazon в Индии в 2017 и 2018 годах. Его перу принадлежат также книги «R Deep Learning Projects» и «Hands-On Deep Learning Architectures with Python», опубликованные издательством Packt. Во время работы над магистерской диссертацией в Торонтском университете написал пять работ, опубликованных в изданиях IEEE и сборниках докладов на конференциях.

О рецензентах

Грег Уолтерс занимается компьютерами и программированием с 1972 года. Отлично владеет языками Visual Basic, Visual Basic .NET, Python и SQL (диалектами MySQL, SQLite, Microsoft SQL Server, Oracle), C++, Delphi, Modula-2, Pascal, C, ассемблером 80x86, COBOL и Fortran. Обучает программированию, через его руки прошло множество людей, которых он учил таким продуктам, как MySQL, Open Database Connectivity, Quattro Pro, Corel Draw!, Paradox, Microsoft Word, Excel, DOS, Windows 3.11, Windows for Workgroups, Windows 95, Windows NT, Windows 2000, Windows XP и Linux. Сейчас на пенсии и в свободное время музицирует и обожает готовить, но всегда готов поработать фрилансером над разными проектами.

Роберт Мони работает над докторской диссертацией в Будапештском университете технологии и экономики (BME), а также является экспертом по глубокому обучению в Континентальном центре компетенций по глубокому обучению в Будапеште. Руководит проектом, направленным на поддержку студенческих исследований в области глубокого обучения и разработки беспилотных автомобилей. Тема его исследований – глубокое обучение с подкреплением в сложных окружающих средах, а конечная цель – применение этой технологии к беспилотным транспортным средствам.

Предисловие

Всплеск интереса к обучению с подкреплением (ОП) объясняется тем, что это революционный подход к автоматизации посредством обучения тому, какие действия следует предпринимать в окружающей среде, чтобы максимизировать полное вознаграждение.

Эта книга представляет собой введение в важные концепции обучения с подкреплением и реализации его алгоритмов с применением библиотеки PyTorch. В каждой главе рассматривается какой-то один метод ОП и его применения в промышленности. Рецепты, содержащие практические примеры, помогут вам обогатить свои знания и навыки в области ОП, в том числе динамическое программирование, методы Монте-Карло, методы на основе временных различий, Q-обучение, решение задачи о многоруком бандите, аппроксимация функций, глубокие Q-сети, методы градиента стратегии. Интересные и легкие для усвоения примеры – игры Atari, блэкджек, сеточный мир, реклама в интернете, машина на горе, игра Flappy Bird – не позволят вам заскучать.

Прочитав книгу, вы будете уверенно владеть распространенными алгоритмами обучения с подкреплением и научитесь применять их к решению различных практических задач.

Предполагаемая аудитория

Специалисты по машинному обучению, по обработке данных и искусственному интеллекту, которым нужна помощь в решении задач ОП. Предполагается предварительное знакомство с концепциями машинного обучения, опыт работы с библиотекой PyTorch необязателен, но желателен.

Структура книги

Глава 1 «Приступаем к обучению с подкреплением и PyTorch» – отправная точка, с которой начинается путешествие в мир обучения с подкреплением и PyTorch. Мы настроим рабочую среду и OpenAI Gym и познакомимся с окружающими средами для экспериментов с ОП, включая CartPole и игры Atari. Здесь же будет рассмотрена реализация таких базовых алгоритмов, как случайный поиск, восхождение на вершину и градиент стратегии. В конце главы будет дан краткий обзор PyTorch.

Глава 2 «Марковский процесс принятия решений и динамическое программирование» начинается с создания марковской цепи и марковского процесса принятия решений (МППР) – понятия, которое лежит в основе большинства алгоритмов обучения с подкреплением. Затем мы рассмотрим два подхода

к решению МППР – итерация по ценности и итерация по стратегиям. Мы ближе познакомимся с МППР и уравнением Беллмана, попрактиковавшись в оценивании стратегии. Также будет продемонстрировано решение интересной игры с подбрасыванием монеты. И в конце мы покажем, как с помощью динамического программирования масштабировать обучение.

Глава 3 «Применение методов Монте-Карло для численного оценивания» посвящена методам Монте-Карло. Для начала мы оценим, чему равно число π. Затем рассмотрим алгоритм с единой стратегией – управление методом Монте-Карло первого посещения – и несколько алгоритмов с разделенной стратегией на основе методов Монте-Карло. Также будут рассмотрены ε-жадная стратегия и взвешенная выборка по значимости.

Глава 4 «TD-обучение и Q-обучение» начинается с подготовки двух окружающих сред: блуждание на краю обрыва и ветреный сеточный мир, которые понадобятся для исследования обучения на основе временных различий (TD-обучения) и Q-обучения. Мы научимся выполнять предсказания с помощью TD-обучения и обсудим Q-обучение как пример алгоритма с разделенной стратегией и SARSA как пример алгоритма с единой стратегией. Мы также сформулируем задачу о такси и покажем, как ее решать с помощью алгоритмов Q-обучения и SARSA. И наконец, будет рассмотрен алгоритм двойного Q-обучения.

В главе 5 «Решение задачи о многоруком бандите» рассматривается алгоритм многорукого бандита – пожалуй, один из самых популярных в обучении с подкреплением. Мы покажем четыре подхода к решению этой задачи: ε-жадная стратегия, исследование с помощью функции softmax, алгоритм верхней доверительной границы и алгоритм на основе выборки Томпсона. Мы также поговорим о рекламе в интернете и продемонстрируем ее решение с помощью алгоритма многорукого бандита. Напоследок разработаем более сложный алгоритм контекстуального бандита и применим его к решению задачи об оптимизации показа рекламных объявлений.

Глава 6 «Масштабирование с помощью аппроксимации функций» посвящена аппроксимации. Мы начнем с подготовки окружающей среды Mountain Car. Объясним, чем аппроксимация функций лучше табличного поиска, и научимся включать аппроксимацию в уже известные алгоритмы Q-обучения и SARSA. Также будет рассмотрена техника пакетного обучения с использованием буфера воспроизведения опыта. И наконец, мы покажем, как, воспользовавшись полученными знаниями, решить задачу о балансировании стержня на тележке.

В главе 7 «Глубокие Q-сети в действии» рассматривается алгоритм глубокой Q-сети (DQN), который считается одним из наиболее передовых методов обучения с подкреплением. Мы разработаем модель DQN и объясним два принципа, лежащих в основе ее работы: буфер воспроизведения и целевая сеть. Для решения игр Atari мы покажем, как интегрировать в DQN сверточную нейронную сеть. Будут рассмотрены два варианта DQN: Double DQN и Dueling DQN. Мы также опишем точную настройку алгоритма Q-обучения, взяв в качестве примера Double DQN.

Глава 8 «Реализация методов градиента стратегии и оптимизация стратегии» посвящена методам градиента стратегии и начинается с реализации алгоритма REINFORCE. Затем мы разработаем алгоритм REINFORCE с базой для

решения задачи о блуждании на краю обрыва. Мы также реализуем алгоритм исполнитель–критик и применим его к решению той же задачи. Чтобы масштабировать детерминированный алгоритм градиента стратегии, воспользуемся приемами, заимствованными из DQN, и разработаем алгоритм глубокого детерминированного градиента стратегии. Ради интереса мы применим метод перекрестной энтропии, чтобы обучить агента балансированию стержня. И наконец, поговорим о том, как масштабировать алгоритм градиента стратегии с помощью асинхронного метода исполнитель–критик и нейронных сетей.

В главе 9 «Кульминационный проект – применение DQN к игре Flappy Bird» мы рассмотрим, как методами обучения с подкреплением можно воспользоваться в игре Flappy Bird. Мы применим все полученные знания, чтобы создать интеллектуального бота. Затем настроим параметры модели и развернем ее. И посмотрим, как долго птица сможет продержаться в воздухе.

Графические выделения

В этой книге для выделения семантически различной информации применяются различные стили. Ниже приведены примеры стилей с пояснениями.

`Код в тексте`: фрагменты кода, имена таблиц базы данных, папок и файлов, URL-адреса, данные, введенные пользователем, адреса в Twitter, например: «Слово `пустая` не означает, что значения всех элементов равны `Null`».

Отдельно стоящие фрагменты кода набраны так:

```
>>> def random_policy():
...     action = torch.multinomial(torch.ones(n_action), 1).item()
...     return action
```

Текст, который вводится на консоли или выводится на консоль, напечатан следующим образом:

```
conda install pytorch torchvision -c pytorch
```

Новые термины, важные слова и слова на экране набраны **полужирным шрифтом**. Так же выделяются элементы интерфейса, например пункты меню и поля в диалоговых окнах. Например: «Этот подход называется **случайным поиском**, потому что вес в каждом испытании выбирается случайно в надежде, что при большом числе испытаний будет найден наилучший вес».

Предупреждения и важные замечания оформлены так.

Советы и рекомендации выглядят так.

Разделы

В этой книге повторяются одни и те же заголовки разделов: *Подготовка*, *Как это делается*, *Как это работает*, *Это еще не все* и *См. также*.

Опишем их назначение.

Подготовка

В этом разделе объясняется, чего ожидать от рецепта, как подготовить программную среду и выполнить все прочие предварительные условия.

Как это делается

Выполнение рецепта по шагам.

Как это работает

Подробное объяснение того, что происходило на каждом шаге, описанном в предыдущем разделе.

Это еще не все

Дополнительная информация, относящаяся к рецепту.

См. также

Ссылки на другую полезную информацию.

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 http://dmkpress.com/authors/publish_book/ или напишите в издательство: dmkpress@gmail.com.

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 данной 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.

Скачивание исходного кода

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

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Packt Publishing 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 веб-сайта, чтобы мы могли применить санкции.

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

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

Глава 1

Приступаем к обучению с подкреплением и PyTorch

Мы начнем путешествие в мир обучения с подкреплением и PyTorch с простых, но важных алгоритмов: случайный поиск, восхождение на вершину и градиент стратегии. Для начала подготовим среду разработки и OpenAI Gym, чтобы для экспериментов с окружающими средами ОП можно было использовать игры Atari и CartPole. Мы также продемонстрируем пошаговую разработку алгоритмов для решения задачи о балансировании стержня. Кроме того, рассмотрим основы PyTorch и приготовимся к последующим примерам и учебным проектам.

В этой главе приводятся следующие рецепты:

- подготовка среды разработки;
- установка OpenAI Gym;
- окружающие среды Atari;
- окружающая среда CartPole;
- основы PyTorch;
- реализация и оценивание стратегии случайного поиска;
- алгоритм восхождения на вершину;
- алгоритм градиента стратегии.

Подготовка среды разработки

Прежде всего подготовим среду разработки, в т. ч. подходящие версии Python, Anaconda, а также библиотеку PyTorch, с которой будем работать на протяжении всей книги.

Python – это язык, на котором будут реализованы все алгоритмы обучения с подкреплением, описанные в этой книге. Мы будем использовать версию 3, а точнее версию 3.8 или более позднюю. Если вы по-прежнему работаете с Python 2, самое время перейти на Python 3, поскольку Python 2 после 2020 года поддерживаться не будет. Переход не сулит никаких проблем, так что не впадайте в панику.

Anaconda – это дистрибутив Python с открытым исходным кодом (www.anaconda.com/distribution/), специально предназначенный для применения в науке о данных и машинном обучении. Для установки Python-пакетов мы будем использовать входящий в Anaconda менеджер пакетов `conda`, а также программу `pip`.

PyTorch (https://pytorch.org/) – современная библиотека машинного обучения, разработанная подразделением Facebook по исследованиям в области искусственного интеллекта (FAIR) на основе каркаса Torch (http://torch.ch/). В PyTorch вместо массивов NumPy (`ndarray`) используются тензоры, обладающие большей гибкостью и совместимостью с графическими процессорами. Привлеченное широкими возможностями графов вычислений, а также простым и дружественным интерфейсом, сообщество PyTorch ежедневно растет, а библиотеку берут на вооружение все новые и новые технологические гиганты.

Теперь посмотрим, как установить и настроить все эти компоненты.

Как это делается

Начнем с установки Anaconda. Можете пропустить этот раздел, если в вашей системе уже установлен дистрибутив Anaconda для Python 3.6 или 3.7. В противном случае следуйте опубликованным на странице https://docs.anaconda.com/anaconda/install/ инструкциям для своей операционной системы:

- Installing on Windows
- Installing on macOS
- Installing on Linux

Чтобы проверить правильность установки Anaconda и Python, введите в окне терминала в Linux/Mac или в окне командной строки в Windows (начиная с этого места будем употреблять общее название – терминал) команду

```
python
```

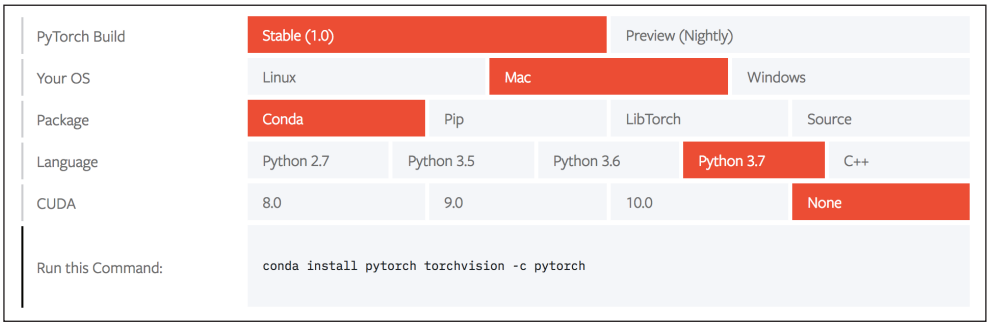
Должно появиться приглашение Python с упоминанием Anaconda:

```
Python 3.7.2 (default, Dec 29 2018, 00:00:04)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda custom (64-bit) on darwin
Type "help", "copyright", "credits" or "license" for more information.
```

Если такая картинка не появилась, проверьте список каталогов (путей), в которых ищется Python.

Следующий шаг – установка PyTorch. Перейдите по адресу https://pytorch.org/get-started/locally/ и выберите описание среды разработки из таблицы[1]:

[1] В настоящее время таблица выглядит иначе, но это типичная проблема: публикация книг отстает от развития программного обеспечения. Впрочем, изменения не принципиальны. – *Прим. перев.*



Здесь мы выбрали **Mac**, **Conda**, **Python 3.7** и локальное выполнение (без CUDA), поэтому в терминале должны ввести такую командную строку:

```
conda install pytorch torchvision -c pytorch
```

Чтобы убедиться в правильности установки PyTorch, выполните показанный ниже код на Python:

```
>>> import torch
>>> x = torch.empty(3, 4)
>>> print(x)
tensor([[ 0.0000e+00, 2.0000e+00, -1.2750e+16, -2.0005e+00],
        [ 9.8742e-37, 1.4013e-45, 9.9222e-37, 1.4013e-45],
        [ 9.9220e-37, 1.4013e-45, 9.9225e-37, 2.7551e-40]])
```

Если будет выведена матрица 3×4, значит, PyTorch установлена правильно. Итак, среда разработки успешно подготовлена.

Как это работает

Мы только что создали тензор PyTorch размера 3×4. Это пустая матрица. Слово `пустая` не означает, что значения всех элементов равны `Null`. На самом деле это неинициализированные числа с плавающей точкой, которые называются местозаместителями. Пользователь должен будет задать их впоследствии. Это очень похоже на пустой массив NumPy.

Это еще не все

Так ли необходимо устанавливать Anaconda и использовать программу `conda` для управления пакетами? Ведь можно же устанавливать пакеты с помощью менеджера `pip`. Но в некоторых отношениях `conda` лучше, чем `pip`, а именно:

- **она корректно обрабатывает зависимости между библиотеками.** Если пакет устанавливается с помощью `conda`, то автоматически будут установлены все его зависимости. А `pip` выдаст предупреждение, и установка будет отменена;
- **корректно разрешаются конфликты между пакетами.** Если для установки пакета необходим другой пакет конкретной версии (например, 2.3 или более поздней), то `conda` автоматически обновит уже установленный пакет;

- **легко создать виртуальную среду.** Виртуальная среда – это автономное дерево пакетов. Для разных приложений или проектов могут понадобиться разные виртуальные среды. Все виртуальные среды изолированы друг от друга. Рекомендуется использовать их, чтобы действия в одном приложении никак не отражались на всех остальных;
- **она совместима с pip**. Мы можем продолжать использовать `pip` вместе с `conda`, выполнив следующую команду:

```
conda install pip
```

См. также

Дополнительные сведения о `conda` можно почерпнуть из следующих ресурсов:

- **руководство пользователя по conda**: https://conda.io/projects/conda/en/latest/user-guide/index.html;
- **создание виртуальных сред и управление ими:** https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html.

Чтобы ближе познакомиться с PyTorch, перейдите в раздел «Getting Started» официального пособия по адресу https://pytorch.org/tutorials/#gettingstarted. Рекомендуем прочитать по крайней мере следующие части:

- **What is PyTorch:** https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html#sphx-glr-beginner-blitz-tensor-tutorial-py;
- **Learning PyTorch with examples:** https://pytorch.org/tutorials/beginner/pytorch_with_examples.html.

Установка OpenAI Gym

Подготовив среду разработки, мы можем перейти к установке OpenAI Gym. Этот продукт содержит разнообразные окружающие среды для разработки алгоритмов обучения, без него заниматься обучением с подкреплением невозможно.

OpenAI (https://openai.com/) – некоммерческая исследовательская компания, занимающаяся созданием безопасных систем **общего искусственного интеллекта** (artificial general intelligence – **AGI**), которые были бы полезны людям. **OpenAI Gym** – мощный комплект инструментов с открытым исходным кодом, предназначенный для разработки и сравнения алгоритмов ОП. Он предлагает интерфейс к различным имитационным моделям и задачам ОП, от обучения шагающего робота до посадки на луну, от автомобильных гонок до игр Atari. Полный список окружающих сред см. по адресу https://gym.openai.com/envs/. **Агентов** для взаимодействия со средами OpenAI Gym можно программировать с применением любой библиотеки численных расчетов, например PyTorch, TensorFlow или Keras.

Как это делается

Установить Gym можно двумя способами. Первый – с помощью `pip`:

```
pip install gym
```

Если вы пользуетесь `conda`, то не забудьте предварительно установить `pip` в `conda`, выполнив команду:

```
conda install pip
```

Дело в том, что по состоянию на начало 2019 года Gym официально не был включен в состав пакетов, поддерживаемых `conda`.

Второй вариант – собрать Gym из исходного кода.

1. Сначала клонируйте пакет из его Git-репозитория:

   ```
   git clone https://github.com/openai/gym
   ```

2. Затем перейдите в папку загрузки и оттуда установите Gym:

   ```
   cd gym
   pip install -e .
   ```

   Теперь можно экспериментировать с `gym`.

3. Проверьте правильность установки Gym, выполнив такой код:

   ```
   >>> from gym import envs
   >>> print(envs.registry.all())
   dict_values([EnvSpec(Copy-v0), EnvSpec(RepeatCopy-v0),
   EnvSpec(ReversedAddition-v0), EnvSpec(ReversedAddition3-v0),
   EnvSpec(DuplicatedInput-v0), EnvSpec(Reverse-v0), EnvSpec(CartPolev0),
   EnvSpec(CartPole-v1), EnvSpec(MountainCar-v0),
   EnvSpec(MountainCarContinuous-v0), EnvSpec(Pendulum-v0),
   EnvSpec(Acrobot-v1), EnvSpec(LunarLander-v2),
   EnvSpec(LunarLanderContinuous-v2), EnvSpec(BipedalWalker-v2),
   EnvSpec(BipedalWalkerHardcore-v2), EnvSpec(CarRacing-v0),
   EnvSpec(Blackjack-v0)
   ...
   ...
   ```

Если все правильно, то будет выведен длинный список окружающих сред. С некоторыми из них мы поэкспериментируем в следующем рецепте.

Как это работает

По сравнению с простой установкой Gym с помощью `pip`, второй способ обеспечивает большую гибкость в случае, если вы захотите добавить новые среды или модифицировать Gym самостоятельно.

Это еще не все

Возникает вопрос, зачем тестировать алгоритмы обучения с подкреплением в окружающих средах Gym, если настоящие среды могут быть совершенно дру-