УДК 004.85
ББК 32.971.3
Л76

Лонца А.
Л76 Алгоритмы обучения с подкреплением на Python / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2020. – 286 с.: ил.

ISBN 978-5-97060-855-5

Эта книга поможет читателю овладеть алгоритмами обучения с подкреплением (ОП) и научиться реализовывать их при создании самообучающихся агентов.

В первой части рассматриваются различные элементы ОП, сфера его применения, инструменты, необходимые для работы в среде ОП. Вторая и третья части посвящены непосредственно алгоритмам. В числе прочего автор показывает, как сочетать Q-обучение с нейронными сетями для решения сложных задач, описывает методы градиента стратегии, TRPO и PPO, позволяющие повысить производительность и устойчивость, а также детерминированные алгоритмы DDPG и TD3. Читатель узнает о том, как работает техника подражательного обучения, познакомится с алгоритмами исследования на базе верхней доверительной границы (UCB и UCB1) и метаалгоритмом ESBAS.

Издание предназначено для тех, кто интересуется исследованиями в области искусственного интеллекта, применяет в работе глубокое обучение или хочет освоить обучение с подкреплением с нуля. Обязательное условие – владение языком Python на рабочем уровне.

УДК 004.85
ББК 32.971.3

ISBN 978-1-78913-111-6 (англ.)
ISBN 978-5-97060-855-5 (рус.)

Содержание

Об авторе

Андреа Лонца занимается глубоким обучением, одержим искусственным интеллектом и страстью создавать машины, действующие «разумно». Знания в области обучения с подкреплением, обработки естественного языка и компьютерного зрения приобрел в ходе работы над проектами по машинному обучению в университете и в промышленности. Также участвовал в нескольких конкурсах Kaggle и достигал высоких результатов. Всегда ищет интересные задачи и любит доказывать, на что способен.

О рецензенте

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.

Предисловие

Обучение с подкреплением (ОП) – популярное и многообещающее направление искусственного интеллекта, в рамках которого изучается построение моделей и агентов, способных находить идеальное поведение в условиях изменяющихся требований. Эта книга поможет вам овладеть алгоритмами ОП и научиться реализовывать их при создании самообучающихся агентов.

Книга начинается с введения в инструменты, библиотеки и процедуру установки, необходимые для работы в среде ОП, затем рассматриваются различные элементы ОП и применение методов, основанных на понятии ценности, в частности алгоритмов Q-обучения и SARSA. Вы научитесь сочетать Q-обучение с нейронными сетями для решения сложных задач. Рассматриваются также методы градиента стратегии, TRPO и PPO, позволяющие повысить производительность и устойчивость, а затем детерминированные алгоритмы DDPG и TD3. Объясняется, как работает техника подражательного обучения и как алгоритм Dagger позволяет обучить агента летать. Вы узнаете об эволюционных стратегиях и оптимизации методом черного ящика. И напоследок познакомитесь с алгоритмами исследования на базе верхней доверительной границы (UCB и UCB1) и метаалгоритмом ESBAS.

Прочитав книгу до конца, вы научитесь применять основные алгоритмы ОП для решения реальных задач и станете членом сообщества ОП.

Предполагаемая аудитория

Если вы занимаетесь исследованиями в области ИИ, применяете в работе глубокое обучение или просто хотите изучить ОП с нуля, то эта книга для вас. Она будет полезна и тем, кто хочет узнать о последних достижениях в этой области. Необходимо владение языком Python на рабочем уровне.

Структура книги

В главе 1 «Ландшафт обучения с подкреплением» описываются проблемы, которые ОП с успехом решает, и приложения, в которых алгоритмы ОП уже нашли применение. Здесь же рассматриваются инструменты, библиотеки и процедуры их установки и настройки, необходимые для выполнения обсуждаемых в книге проектов.

В главе 2 «Реализация цикла ОП и OpenAI Gym» описывается главный цикл алгоритмов ОП, инструментарий для разработки алгоритмов и различные типы сред. Мы разработаем с помощью интерфейса к OpenAI Gym агента, который будет играть в игру CartPole, совершая случайные действия. Мы также научимся использовать OpenAI Gym для работы в других средах.

Глава 3 «Решение задач методом динамического программирования» содержит введение в основные идеи, терминологию и подходы, применяемые в ОП.

Вы узнаете о главных составных частях ОП и составите общее представление о том, как алгоритмы ОП применяются к решению задач. Вы также узнаете о различиях между основанными на модели и безмодельными алгоритмами и о классификации алгоритмов обучения с подкреплением. Мы применим динамическое программирование для решения игры FrozenLake.

В главе 4 «Применения Q-обучения и алгоритма SARSA» речь пойдет о методах на основе ценности, в частности Q-обучении и SARSA, двух алгоритмах, которые отличаются от динамического программирования и хорошо масштабируются на задачи большого размера. Чтобы лучше разобраться в этих алгоритмах, мы применим их к игре FrozenLake и сравним результаты с динамическим программированием.

В главе 5 «Глубокие Q-сети» описывается использование нейронных сетей и, в частности, **сверточных нейронных сетей** (**СНС**) к Q-обучению. Вы узнаете, почему сочетание Q-обучения и нейронных сетей дает поразительные результаты и как это открывает путь к гораздо более широкому кругу задач. Кроме того, мы применим глубокую Q-сеть к игре Atari, воспользовавшись интерфейсом к OpenAI Gym.

В главе 6 «Стохастическая оптимизация и градиенты стратегии» вводится новое семейство безмодельных алгоритмов: методы градиента стратегии. Мы узнаем о различии между методами градиента стратегии и методами на основе ценности, а также об их сильных и слабых сторонах. Затем реализуем алгоритмы REINFORCE и исполнитель–критик для решения игры LunarLander.

В главе 7 «Реализация TRPO и PPO» предлагается модификация методов градиента стратегии с использованием новых механизмов контроля над улучшением стратегии. Эти механизмы позволяют улучшить устойчивость и сходимость алгоритмов градиента стратегии. В частности, мы опишем и реализуем два основных метода градиента стратегии, в которых используются эти подходы: TRPO и PPO. Они будут реализованы в семействе сред с непрерывным пространством действий RoboSchool.

В главе 8 «Применения алгоритмов DDPG и TD3» вводится новая категория алгоритмов с детерминированной стратегией, которые сочетают идею градиента стратегии с Q-обучением. Вы узнаете об идее и реализации двух глубоких детерминированных алгоритмов, DDPG и TD3, в новой окружающей среде.

В главе 9 «ОП на основе модели» иллюстрируются алгоритмы ОП, которые обучаются модели окружающей среды для планирования будущих действий, т. е. обучения стратегии. Вы узнаете, как они работают, в чем их плюсы и почему они предпочтительны во многих ситуациях. Чтобы лучше разобраться в них, мы реализуем основанный на модели алгоритм в среде RoboSchool.

В главе 10 «Подражательное обучение и алгоритм DAgger» объясняется, как работает подражательное обучение и как его можно адаптировать к конкретной задаче. Будет рассмотрен самый известный алгоритм подражательного обучения, DAgger. Ради лучшего освоения мы реализуем его и тем ускорим процесс обучения агента в игре FlappyBird.

В главе 11 «Оптимизация методом черного ящика» изучаются эволюционные алгоритмы – класс алгоритмов оптимизации методом черного ящика, которые не опираются на обратное распространение. Интерес к этим алгоритмам растет, потому что они, во-первых, быстро обучаются, а во-вторых, легко

распараллеливаются на сотни и тысячи процессорных ядер. В этой главе подводится теоретический фундамент под эти алгоритмы и приводится практическая реализация на примере алгоритма эволюционной стратегии.

В главе 12 «Разработка алгоритма ESBAS» описывается важная дилемма исследования–использования, специфичная для ОП. Она демонстрируется на примере задачи о многоруких бандитах и решается методами верхней доверительной границы: UCB и UCB1. Затем мы узнаем о проблеме выбора алгоритма и разработаем метаалгоритм ESBAS, в котором UCB1 используется для выбора наиболее подходящего алгоритма ОП в конкретной ситуации.

В главе 13 «Практические подходы к решению проблем ОП» мы рассмотрим основные проблемы, возникающие в этой области, и объясним, как их преодолевать. Вы узнаете о некоторых трудностях применения ОП к реальным задачам, о будущем глубокого ОП и о его влиянии на общество.

Что необходимо для чтения этой книги

Необходимо владение языком Python на рабочем уровне. Знакомство с ОП и различными инструментами, используемыми в этой области, также было бы полезно.

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, адреса в Twitter, например: «В этой книге используется версия Python 3.7, но должны работать все версии начиная с 3.5. Предполагается также, что пакеты `numpy` и `matplotlib` уже установлены».

Отдельно стоящие фрагменты кода набраны так:

```
import gym

# создать окружающую среду
env = gym.make("CartPole-v1")
# привести среду в исходное состояние перед началом работы
env.reset()

# повторить 10 раз
for i in range(10):
    # предпринять случайное действие
    env.step(env.action_space.sample())
    # отрисовать игру
    env.render()

# закрыть среду
env.close()
```

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

```
$ git clone https://github.com/pybox2d/pybox2d
$ cd pybox2d
$ pip install -e .
```

Новые термины, важные слова и слова на экране набраны **полужирным шрифтом**. Также выделяются элементы интерфейса, например пункты меню и поля в диалоговых окнах, например: «В **обучении с подкреплением** (**ОП**) алгоритм называется агентом, он обучается на данных, поступающих от окружающей среды».

Предупреждения и важные замечания оформлены так.

Советы и рекомендации выглядят так.

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.

Часть I

АЛГОРИТМЫ И ОКРУЖАЮЩИЕ СРЕДЫ

Эта часть представляет собой введение в обучение с подкреплением. В ней закладывается теоретический фундамент и подготавливается почва для последующих глав. Эта часть состоит из следующих глав:

- глава 1 «Ландшафт обучения с подкреплением»;
- глава 2 «Реализация цикла ОП и OpenAI Gym»;
- глава 3 «Решение задач методами динамического программирования».

Глава 1

Ландшафт обучения с подкреплением

Люди и животные обучаются методом проб и ошибок. Этот процесс основан на механизмах вознаграждения в ответ на то или иное поведение. Его цель – посредством многократного повторения побудить к выбору таких действий, которые порождают положительные отклики, и отвратить от действий, порождающих отрицательные отклики. Посредством механизма проб и ошибок мы обучаемся взаимодействовать с людьми и окружающим нас миром, преследовать сложные осмысленные цели, а не просто стремиться к получению немедленного удовольствия.

Обучение на опыте и взаимодействии принципиально важно. Представьте, что вы должны научиться играть в футбол, только глядя на то, как играют другие. Если, основываясь на таком опыте, вы выйдете на поле, то, скорее всего, будете играть отвратительно.

Это было продемонстрировано в середине XX века в известной работе Ричарда Хелда (Richard Held) и Алана Хейна (Alan Hein) 1963 года, в которой два котенка с рождения были посажены в некое подобие карусели. Один котенок мог двигаться свободно (был активен), а второй не мог производить никаких движений и перемещался пассивно – его возил первый. Впоследствии только у котенка, который мог свободно двигаться, развилось восприятие глубины и двигательные навыки. Это было доказано отсутствием у пассивного котенка мигательного рефлекса на приближающиеся объекты. Таким образом, в этом довольно жестоком эксперименте было продемонстрировано, что для обучения животного необходимо физическое взаимодействие с окружающей средой, а не только зрительная стимуляция.

В основе **обучения с подкреплением** (**ОП**) лежит идея об активном взаимодействии со средой методом проб и ошибок, подсмотренная у обучающихся людей и животных. В частности, в ОП агент постепенно обучается в процессе взаимодействия с окружающим миром. Это позволяет наделить компьютер рудиментарными навыками обучения, взяв за образец поведение человека.

Эта книга целиком посвящена обучению с подкреплением. Ее цель – помочь вам разобраться в данной области на практических примерах. В начальных главах мы познакомимся с базовыми понятиями обучения с подкреплением, а затем приступим к разработке первых алгоритмов. Постепенно мы будем создавать все более сложные и мощные алгоритмы для решения более интересных

и интригующих задач. Вы увидите, что обучение с подкреплением – очень широкий предмет и что в нем существует много алгоритмов, предназначенных для решения разных задач разными способами. Однако мы постараемся дать простое, но полное описание всех идей, а также ясные и практически полезные реализации алгоритмов.

В этой главе мы познакомимся с фундаментальными концепциями ОП, узнаем о различиях между имеющимися подходами и о таких ключевых понятиях, как функция ценности, вознаграждение и модель окружающей среды. Попутно мы расскажем об истории ОП и ее приложениях.

В этой главе рассматриваются следующие вопросы:

- введение в ОП;
- элементы ОП;
- приложения ОП.

Введение в ОП

ОП – часть машинного обучения, в которой изучается последовательное принятие решений для достижения поставленной цели. Задача ОП состоит из **агента**, принимающего решения, и физического или виртуального мира, с которым агент взаимодействует, – **окружающей среды**. Взаимодействие агента с окружающей средой сводится к **действиям**, имеющим некоторые последствия. В результате агент получает от среды обратную связь в форме нового **состояния** и **вознаграждения**. Оба этих сигнала – последствия действия, предпринятого агентом. Точнее, вознаграждение показывает, насколько хорошим или плохим было действие, а состояние – это текущее представление агента и окружающей среды. Этот цикл показан на рис. 1.1.

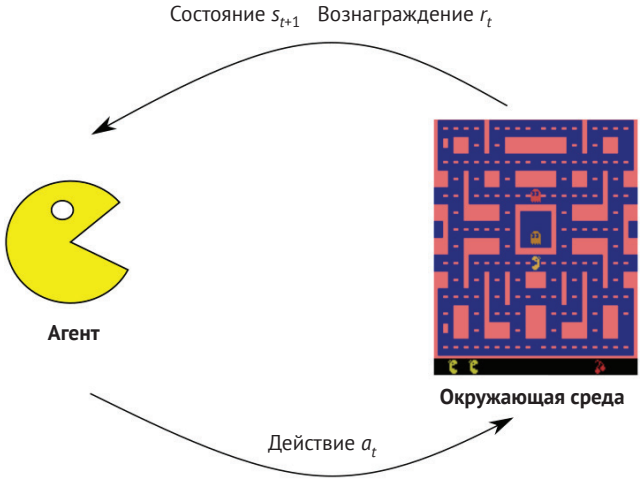

Рис. 1.1

На этом рисунке агент представлен значком пакмана[1], который в зависимости от текущего состояния окружающей среды выбирает следующее действие. Его поведение влияет на окружающую среду, а именно на его положение и положения врагов. В ответ на поведение агента среда возвращает новое состояние и вознаграждение. Этот цикл повторяется, пока игра не завершится.

Конечная цель агента – максимизировать полное вознаграждение, полученное за все время существования. Введем обозначения: если a_t – действие в момент t, а r_t – вознаграждение в момент t, то агент предпринимает действия a_0, a_1, ..., a_t, стремясь максимизировать сумму $\sum_{i=0}^{t} r_i$.

Чтобы максимизировать полное вознаграждение, агент должен обучиться наилучшему поведению в каждой ситуации. Для этого агенту необходимо произвести оптимизацию на длинном горизонте, принимая во внимание каждое действие. В окружающей среде с большим количеством дискретных или непрерывных состояний и действий обучение затруднено тем, что агент вынужден учитывать все возможные ситуации. Мало того, вознаграждение может поступать очень редко и запаздывать во времени, что дополнительно усложняет процесс обучения.

Чтобы привести пример задачи ОП и заодно объяснить, в чем сложность разреженного вознаграждения, рассмотрим хорошо известную сказку о Гензеле и Гретель. Родители завели их в лес, чтобы там оставить, но Гензель, знавший об их намерении, захватил с собой из дому ломоть хлеба и оставил дорожку из хлебных крошек, которая должна была привести его и сестру обратно домой. В системе ОП агентами являются Гензель и Гретель, а окружающей средой – лес. За каждую встреченную на обратном пути крошку агент получает вознаграждение +1, а за выход к дому – вознаграждение +10. В этом случае чем чаще хлебная дорожка, тем легче детям будет найти дорогу домой, поскольку для перехода от одной крошки к другой нужно исследовать сравнительно небольшую область. К сожалению, в реальном мире разреженные вознаграждения встречаются куда чаще плотных.

Важная характеристика ОП заключается в возможности достижения результата, даже когда окружающая среда динамическая, неопределенная и недетерминированная. Ниже приведены примеры реальных проблем, которые можно поставить как задачи ОП.

- Беспилотные автомобили – популярная идея, которая с трудом поддается ОП. Дело в том, что при движении по дороге нужно принимать во внимание много факторов (пешеходы, другие автомобили, велосипеды, светофоры и т. д.), а среда в высшей степени неопределенная. В данном случае беспилотный автомобиль – это агент, который может воздействовать на руль, акселератор и тормоза. Средой же является окружающий мир. Очевидно, что агент не может воспринять весь мир вокруг себя целиком, ему доступна лишь ограниченная информация от датчиков (например, камеры, радара и системы навигации). Цель беспилотного автомобиля – добраться до места назначения за минимальное время, соблюдая правила дорожного движения и не нанеся никакого вреда. Сле-

[1] Персонаж старой компьютерной игры PacMan. – *Прим. перев.*

довательно, агент может получить отрицательное вознаграждение, если случится какое-то неприемлемое событие, а величина положительного пропорциональна времени поездки.

- В шахматах цель состоит в том, чтобы поставить мат противнику. В терминах ОП игрок является агентом, а текущая позиция на доске – окружающей средой. Агенту разрешается переставлять фигуры согласно правилам. В итоге окружающая среда возвращает положительное вознаграждение, если агент выиграл, и отрицательное, если проиграл. Во всех остальных случаях вознаграждение нулевое, а следующим состоянием является позиция на доске после хода противника. В отличие от примера беспилотного автомобиля, здесь состояние агента совпадает с состоянием окружающей среды. Иными словами, агент знает все об окружающей среде.

Сравнение ОП и обучения с учителем

ОП и **обучение с учителем** – похожие, но все же различные парадигмы обучения на данных. Многие задачи можно решить как с помощью ОП, так и обучения с учителем, но в большинстве случаев эти методы предназначены для решения разных задач.

Цель обучения с учителем – научиться обобщать, располагая лишь фиксированным набором данных с ограниченным количеством примеров. Каждый пример состоит из входа и желательного выхода (или метки), так что реакция на выбор агента следует незамедлительно.

Напротив, в ОП акцент ставится на последовательных действиях, которые можно предпринять в конкретной ситуации. В данном случае единственное, что дает учитель, – сигнал вознаграждения. Какое действие правильно при данных условиях, неизвестно, в отличие от обучения с учителем.

ОП можно рассматривать как более общую и полную систему обучения. Перечислим основные характеристики ОП:

- вознаграждение может быть плотным, разреженным или приходить с большим запаздыванием. Во многих случаях вознаграждение становится известно только в конце задания (например, в шахматах);
- задача последовательная и зависящая от времени – выбранные действия могут влиять на следующие действия, которые, в свою очередь, влияют на возможные вознаграждения и состояния;
- агент должен совершать действия, которые с высокой вероятностью приведут к достижению цели (использование), но в то же время должен пробовать иные действия, чтобы другие части окружающей среды не остались неисследованными (исследование). Эту двойственность называют дилеммой (или компромиссом) исследования–использования, она призвана решить трудную проблему поиска баланса между исследованием и использованием окружающей среды. Она важна также и потому, что, в отличие от обучения с учителем, агент ОП может влиять на окружающую среду, т. к. вправе собирать новые данные, коль скоро считает это полезным;
- окружающая среда стохастическая и недетерминированная, и агент должен учитывать это в процессе обучения и для предсказания следующего

действия. Мы увидим, что многие компоненты ОП можно спроектировать так, что они будут выдавать либо только одно детерминированное значение, либо диапазон значений, каждому из которых сопоставлена вероятность.

Третий вид обучения – **обучение без учителя** – применяется для выявления закономерностей в данных при отсутствии какой-либо информации от учителя. Сжатие данных, кластеризация и порождающие модели – все это примеры обучения без учителя. Его можно инкорпорировать в систему ОП, чтобы исследовать окружающую среду и получать информацию о ней. Комбинация обучения без учителя и ОП называется **ОП без учителя**. В этом случае никакое вознаграждение не присуждается, а агент может генерировать внутренние мотивы, побуждающие отдать предпочтение новым ситуациям, которые можно исследовать.

Отметим, что задачи, связанные с беспилотными автомобилями, можно решать и как задачи обучения с учителем, но результаты получаются неудовлетворительные. Основная проблема в том, что распределение данных, с которым агент сталкивается на практике, может сильно отличаться от того, что он видел в процессе обучения.

История ОП

Первое математическое обоснование ОП появилось в 1960–1970-е годы в области оптимального управления. Тем самым была решена задача минимизации некоторой меры поведения динамической системы, изменяющейся во времени. Этот метод подразумевал решение системы уравнений при известной динамике системы. Тогда было сформулировано ключевое понятие **марковского процесса принятия решений** (**МППР**). Оно легло в основу общего подхода к моделированию принятия решений в стохастических условиях. В эти годы был разработан метод решения задач оптимального управления, получивший название **динамического программирования** (**ДП**). Смысл ДП в том, чтобы разбить сложную задачу на несколько более простых с целью решить систему уравнений МППР.

Отметим, что ДП упрощает поиск оптимального управления лишь для систем с известной динамикой, ни о каком обучении речи не идет. Кроме того, ему свойственна проблема **проклятия размерности**, поскольку требования к вычислительным ресурсам экспоненциально возрастают по мере роста количества состояний.

Но, как заметили Ричард Саттон и Эндрю Бартон, хотя эти методы не подразумевают обучения, мы все равно должны считать методы решения задач оптимального управления, в т. ч. ДП, методами ОП.

В 1980-х годах наконец-то появилась концепция обучения на основе предсказаний в соседние моменты времени – метод **обучения на основе временных различий** (temporal difference learning – **TD-обучение**). Открытие TD-обучения принесло с собой новое семейство эффективных алгоритмов, которые будут рассмотрены в этой книге.

Первые задачи, решенные с помощью TD-обучения, были настолько малы, что их можно было представить с помощью таблиц или массивов. Соответ-

ствующие методы получили название **табличных методов**, они часто находят оптимальное решение, но не масштабируемы. Вообще, во многих задачах ОП пространство состояний огромно, поэтому табличные методы к ним неприменимы. В таких случаях используется аппроксимация функций, позволяющая найти хорошее приближенное решение с меньшими вычислительными затратами.

Включение в ОП аппроксимации функций и, в частности, искусственных нейронных сетей (в т. ч. глубоких) – дело нетривиальное, но, как показано на целом ряде примеров, иногда это позволяет достичь поразительных результатов. Сочетание глубокого обучения с ОП называют **глубоким обучением с подкреплением** (**глубоким ОП**), оно обрело широкую популярность, после того как алгоритм **глубокой Q-сети** (**DQN**) в 2015 году продемонстрировал умение играть в игры компании Atari на уровне, превышающем возможности человека (для обучения использовались только изображения на экране). Еще одно впечатляющее достижение глубокого ОП – программа AlphaGo, которая в 2017 году стала первой программой, обыгравшей в го 18-кратного чемпиона мира Ли Седоля. Эти прорывные достижения не только показали, что программа может работать лучше человека в многомерных пространствах (воспринимая те же самые изображения, что и человек), но и что она может вести себя различными интересными способами. Примером может служить неожиданное решение, найденное системой глубокого ОП в аркадной игре Atari Breakout, в которой игрок должен уничтожить все кирпичи, как показано на рис. 1.2. Агент обнаружил, что, пробив туннель слева от кирпичей и направляя шарик в эту сторону, он сможет уничтожить гораздо больше кирпичей и тем самым увеличить свой счет всего одним ходом.

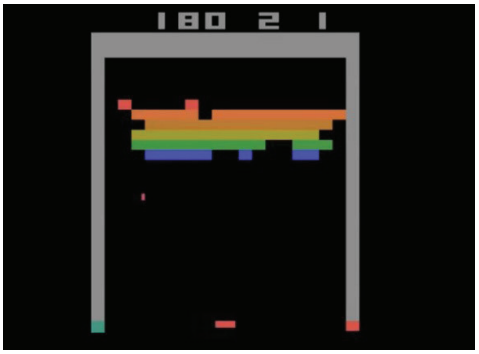

Рис. 1.2

Есть еще много интересных примеров того, как агенты демонстрировали изобретательное поведение или стратегию, неизвестную людям, в частности ход, сделанный AlphaGo в игре против Ли Седоля. С точки зрения человека, он выглядел бессмысленно, но в итоге позволил AlphaGo победить (он даже получил название – **ход 37**).

В наши дни при работе с многомерным пространством состояний или действий применение глубоких нейронных сетей для аппроксимации функций

кажется чуть ли не очевидным выбором. Глубокое ОП применялось и к более трудным задачам, например: оптимизация энергопотребления в центрах обработки данных, беспилотные автомобили, оптимизация многопериодического портфеля ценных бумаг, робототехника и многое другое.

Глубокое обучение

Теперь можно задаться вопросом: почему глубокое обучение в сочетании с ОП дает такие замечательные результаты? Главным образом потому, что глубокое обучение способно справляться с пространством состояний очень высокой размерности. До изобретения глубокого ОП пространства состояний приходилось разбивать на более простые представления, называемые **признаками**. Их было трудно проектировать, и иногда эта задача была подвластна только узким специалистам. Теперь же, пользуясь глубокими нейронными сетями, в частности **сверточными нейронными сетями** (**СНС**) или **рекуррентными нейронными сетями** (**РНС**), ОП-система может обучиться различным уровням абстракции непосредственно на исходных пикселях или последовательных данных (например, текстах на естественном языке). Такая конфигурация показана на рис. 1.3.

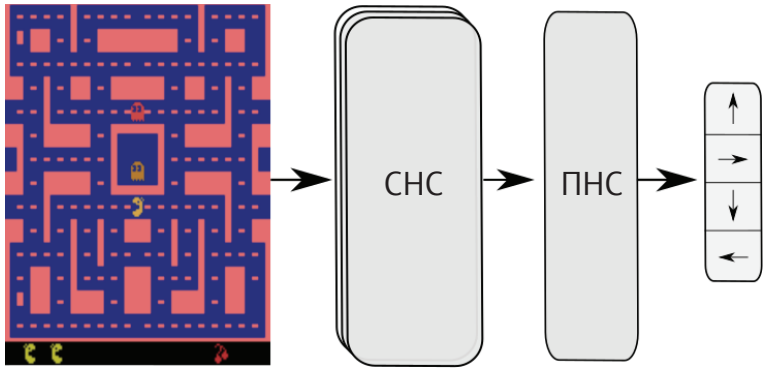

Рис. 1.3

Кроме того, задачи глубокого ОП теперь можно решать от начала до конца в едином процессе. Раньше алгоритм ОП состоял из двух разных конвейеров: один для восприятия, второй для принятия решений. Теперь же эти процессы объединены и обучаются вместе: от исходных пикселей к выбору действия. Например, пакмана на рис. 1.3 можно обучить с помощью СНС, которая обрабатывает визуальный компонент, и **полносвязной нейронной сети** (**ПНС**), которая преобразует выход СНС в действие.

В настоящее время глубокое ОП – чрезвычайно актуальная тема. И главная причина заключается в том, что глубокое ОП считается технологией, которая позволит создавать машины с высоким уровнем интеллекта. Доказательством служит тот факт, что две широко известные компании в области ИИ, DeepMind и OpenAI, ведут активные исследования в области ОП.

Но, несмотря на несомненные достижения глубокого ОП, многое еще предстоит сделать. Некоторые из открытых вопросов перечислены ниже:

- система глубокого ОП обучается слишком медленно по сравнению с человеком;
- перенос обучения в ОП остается нерешенной проблемой;
- функцию вознаграждения трудно придумать и определить;
- агенты ОП с трудом обучаются в таких сложных и динамических средах, как реальный мир.

Тем не менее фронт исследований в этой области быстро расширяется, и компании начинают включать ОП в свои продукты.

Элементы ОП

Как мы знаем, агент взаимодействует с окружающей средой посредством действий. Это заставляет среду изменяться и начислять агенту вознаграждение, пропорциональное качеству действий. Методом проб и ошибок агент постепенно обучается находить наилучшее действие в каждой ситуации, стремясь в итоге получить как можно большее полное вознаграждение. В системе ОП выбор действия в конкретном состоянии производится с помощью **стратегии**, а полное вознаграждение, достижимое при старте из некоторого состояния, называется **функцией ценности**. Короче говоря, если агент хочет вести себя оптимально, то в каждой ситуации стратегия должна выбирать такое действие, которое переводит агента в следующее состояние с наибольшей ценностью. Теперь рассмотрим эти основополагающие понятия более пристально.

Стратегия

Стратегия определяет, как агент выбирает действие в данном состоянии. Выбирается такое действие, которое максимизирует полное вознаграждение, достижимое из данного состояния, а не действие, приносящее наибольшее немедленное вознаграждение. Преследуется долгосрочная цель агента. Например, если машине нужно доехать до места назначения 30 км, но хода осталось только на 10 км, а до ближайших заправок 1 км и 60 км, то стратегия выберет заправку на первой АТС (в одном километре), чтобы не остаться без топлива посередь дороги. Это решение не оптимально в ближайшей перспективе, поскольку на заправку уйдет время, но оно необходимо для достижения конечной цели.

На рис. 1.4 показан простой пример – агент, перемещающийся по сетке 4×4, должен добраться до звезды, избегая спиралей. Действия, рекомендуемые стратегией, обозначены стрелкой в направлении хода. Слева на рис. 1.4 показана случайная начальная стратегия, а справа – окончательная оптимальная стратегия. Если имеются два одинаково хороших действия, то агент выбирает любое из них произвольно.

Существует важное различие между стохастическими и детерминированными стратегиями. Детерминированная стратегия однозначно подсказывает, какое действие предпринять, а стохастическая сообщает вероятности каждого

действия. Идея вероятности действия полезна, потому что учитывает динамичность окружающей среды и помогает исследовать ее.

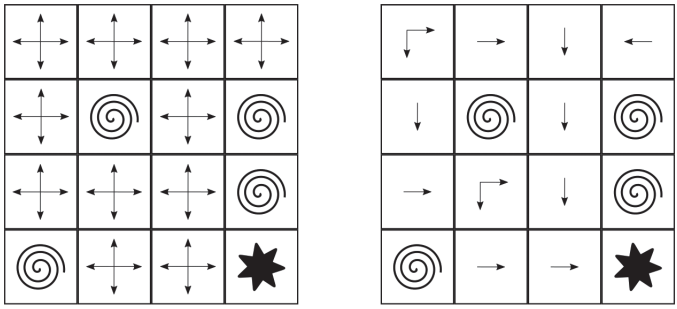

Рис. 1.4

Одна из классификаций алгоритмов ОП основана на том, как стратегии улучшаются в процессе обучения. В более простом случае стратегия, которая воздействует на окружающую среду, похожа на ту, что улучшается во время обучения. По-другому можно сказать, что стратегия обучается на тех же данных, которые генерирует. Такие алгоритмы называются алгоритмами **с единой стратегией** (on-policy algorithm). Напротив, в алгоритмах **с разделенной стратегией** (off-policy algorithm) присутствуют две стратегии: одна воздействует на среду, а другая обучается, но фактически не используется. Первая называется **поведенческой стратегией**, вторая – **целевой стратегией**. Цель поведенческой стратегии – взаимодействовать со средой и собирать о ней информацию, чтобы улучшить **пассивную** целевую стратегию. Как мы увидим в последующих главах, алгоритмы с разделенной стратегией менее устойчивы и их труднее проектировать, чем алгоритмы с единой стратегией, зато у них более высокая выборочная эффективность, т. е. для обучения нужно меньше данных.

Чтобы лучше уяснить себе обе концепции, представьте человека, который хочет обучиться новому навыку. Если он ведет себя как алгоритм с единой стратегией, то всякий раз, испробовав новую последовательность действий, будет изменять свои представления и поведение в соответствии с полученным полным вознаграждением. Наоборот, если человек ведет себя как алгоритм с разделенной стратегией, то может обучиться (целевая стратегия), глядя на старую видеозапись собственных действий (поведенческая стратегия) во время применения того же навыка, т. е. может использовать прежний опыт для самосовершенствования.

Методами градиента стратегии называется семейство алгоритмов ОП, которые обучаются параметрической стратегии (как глубокая нейронная сеть) непосредственно на данных о градиенте качества по стратегии. У таких алгоритмов много достоинств, в т. ч. способность работать с непрерывными действиями и исследовать окружающую среду с разными уровнями детализации. Мы будем подробно рассматривать их в главах 6 «Стохастическая оптимизация и градиенты стратегии», 7 «Реализация TRPO и PPO» и 8 «Применения алгоритмов DDPG и TD3».

Функция ценности

Функция ценности представляет качество состояния в долгосрочной перспективе. Это полное вознаграждение, ожидаемое в будущем, если агент стартует из данного состояния. Если вознаграждение измеряет немедленное качество действия, то функция ценности – его качество на протяженном отрезке времени. Из того, что вознаграждение велико, еще не следует, что будет велика ценность, и наоборот – если вознаграждение мало, это еще не значит, что мала ценность состояния.

Функция ценности может зависеть только от состояния или от пары состояние–действие. В первом случае она называется **функцией ценности состояний**, а во втором – **функцией ценности действий**.

На рис. 1.5 показаны окончательные ценности состояний (слева) и соответствующая оптимальная стратегия (справа).

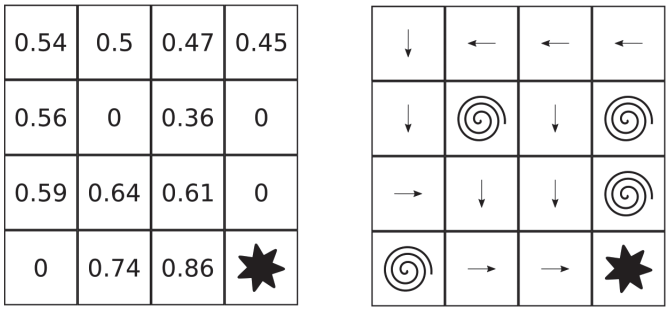

Рис. 1.5

На том же примере сетки, который использовался для иллюстрации понятия стратегии, мы можем показать, что такое функция ценности состояний. Прежде всего можно предположить, что в любой ситуации назначается вознаграждение 0, кроме случая, когда агент доходит до клетки со звездой, где получает вознаграждение +1. Далее предположим, что сильный ветер сдувает агента в другом направлении с вероятностью 0.33. Тогда ценности состояний будут такими, как в левой части рис. 1.5. Оптимальная стратегия будет выбирать действия, которые переводят агента в следующее состояние с максимальной ценностью, как показано в правой части рис. 1.5.

Методы ценности действий (или методы, основанные на функции ценности) – еще одно большое семейство алгоритмов ОП. Они сводятся к обучению функции ценности действий и использованию ее для выбора действий. Начиная с главы 3 мы многое расскажем об этих алгоритмах. Отметим, что из-за стремления взять лучшее из обоих миров в некоторых методах градиента стратегии для обучения хорошей стратегии используется функция ценности. Такие методы называются **методами исполнитель–критик**. На рис. 1.6 показаны все три основных семейства алгоритмов ОП.

Рис. 1.6

Вознаграждение

На каждом временном шаге, т. е. после каждого хода агента, окружающая среда посылает ему число, показывающее, насколько хорошим было действие. Это число называется **вознаграждением**. Как мы уже упоминали, конечная цель агента – максимизировать полное вознаграждение, полученное за время взаимодействия со средой.

В литературе считается, что вознаграждение – часть окружающей среды, но в реальности это не совсем так. Вознаграждение может порождать и сам агент, но только не та его часть, которая принимает решения. Именно по этой причине, а заодно чтобы упростить формулировки, вознаграждение всегда исходит от окружающей среды.

Вознаграждение – единственный сигнал учителя, попадающий в цикл ОП, и важно проектировать вознаграждение правильно, чтобы получить агента с хорошим поведением. Если вознаграждение в чем-то дефектно, то агент может обнаружить эти дефекты и выработать неправильное поведение. Например, *Coast Runners* – игра, цель которой – первым прийти к финишу гонок на водных катерах. В процессе гонки катера получают вознаграждение за поражение мишеней. В компании OpenAI обучили играть агента ОП. Обнаружилось, что вместо стремления как можно быстрее прийти к финишу обучаемый катер мчался по кругу, тараня возрождающиеся мишени и не обращая внимания на периодические поломки и пожары на борту. То есть катер нашел способ максимизировать полное вознаграждение, действуя не так, как ожидалось. Такое поведение стало результатом неверного баланса между краткосрочным и долгосрочным вознаграждениями.

Вознаграждение может выдаваться с разной частотой, зависящей от окружающей среды. Частое вознаграждение называют **плотным**; если же оно выдается лишь несколько раз за игру или только в самом конце, то говорят о **разреженном вознаграждении**. В таком случае агенту бывает очень трудно получить вознаграждение и найти оптимальные действия.

Подражательное обучение и **обратное ОП** – две эффективные техники обучения в условиях отсутствия вознаграждения от окружающей среды. В подражательном обучении для отображения состояний на действия исполь-

зуется демонстрация работы эксперта. А в обратном ОП функция вознаграждения выводится из оптимального поведения эксперта. То и другое – предмет главы 10.

Модель

Модель – необязательный компонент агента, т. е. она не требуется для нахождения стратегии взаимодействия со средой. Модель детально описывает поведение окружающей среды, а именно предсказывает следующее состояние и вознаграждение для данных состояния и действия. Если модель известна, то для взаимодействия с ней и рекомендации будущих действий можно использовать алгоритмы планирования. Например, в средах с дискретными действиями потенциальные траектории можно смоделировать, применяя поиск с заглядыванием вперед (скажем, поиск по дереву методом Монте-Карло).

Модель окружающей среды может быть либо задана заранее, либо обучена посредством взаимодействия с ней. Если среда сложная, то имеет смысл аппроксимировать ее глубокой нейронной сетью. Алгоритмы ОП, в которых используется уже известная или обученная модель среды, называются **методами, основанными на модели**. Они являются противоположностью безмодельным методам и подробно рассматриваются в главе 9.

Применение ОП

ОП применяется в самых разных областях, например: робототехника, финансы, здравоохранение, интеллектуальные транспортные системы. В общем случае применения можно объединить в три большие группы: автоматические машины (автономные транспортные средства, «умные» сети электроснабжения, робототехника и т. п.), оптимизация процессов (плановое техническое обслуживание, цепочки поставок, планирование процессов) и контроль (например, обнаружение отказов и контроль качества).

Поначалу ОП применялось только к простым задачам, но глубокое ОП открыло дорогу к проблемам совсем другого уровня сложности. В настоящее время глубокое ОП демонстрирует весьма многообещающие результаты. К сожалению, многие из этих прорывов ограничиваются исследовательскими приложениями или играми, и зачастую не так-то просто перекинуть мост между чисто исследовательскими приложениями и промышленными задачами. Но, несмотря на это, все больше компаний движется в направлении включения ОП в свои отрасли промышленности и продукты.

Рассмотрим основные области, в которых ОП уже внедряется или видна потенциальная выгода от него.

Игры

Игры – идеальный испытательный стенд для ОП, потому что создаются специально для того, чтобы бросить вызов человеческим возможностям. Чтобы пройти игру до конца, необходим человеческий мозг (память, способность рас-

суждать и координация движений). Поэтому компьютер, способный играть на уровне человека или лучше, обязан обладать теми же качествами. Кроме того, игру зачастую легко воспроизвести, поэтому ее нетрудно смоделировать на компьютере. Видеоигры оказались очень трудны для компьютера из-за частичной наблюдаемости (т. е. в каждый момент времени видна только часть игры) и гигантского пространства поиска (компьютер не может смоделировать все возможные конфигурации).

Прорыв в играх произошел, когда в 2015 году программа AlphaGo обыграла Ли Седоля в древнюю игру го. Это случилось вопреки всем предсказаниям. В то время считалось, что компьютер не сможет обыграть профессионального игрока в го еще по крайней мере 10 лет. В AlphaGo использовалось ОП в сочетании с обучением с учителем на играх, сыгранных профессионалами-людьми. Спустя несколько лет после матча следующая версия программы, AlphaGo Zero, разгромила AlphaGo со счетом 100:0. AlphaGo Zero научилась играть в го всего за три дня игр с самой собой.

Игра с собой – весьма эффективный способ обучения алгоритма, потому что не нужен никакой противник. К тому же в игре с собой можно выработать дополнительные навыки, которые иначе остались бы нераскрытыми.

Чтобы уловить хаотичность и непрерывную природу реального мира, группу из пяти нейронных сетей, получившую название OpenAI Five, научили играть в *DOTA 2*, стратегическую игру реального времени, в которой две команды (по пять игроков) играют друг против друга. Сложность обучения в этом случае обусловлена длительными временными горизонтами (в среднем игра продолжается 45 минут и состоит из тысяч действий), частичной наблюдаемостью (каждый игрок видит только небольшую область вокруг себя) и непрерывными пространствами действий и наблюдений очень высокой размерности. В 2018 году OpenAI Five сыграла против игроков в DOTA 2 на конкурсе The International. Она проиграла матч, но продемонстрировала прирожденные способности к сотрудничеству и выстраиванию стратегии. Затем, 13 апреля 2019 года, OpenAI Five официально победила чемпионов мира по этой игре и стала первым представителем искусственного интеллекта, превзошедшим профессиональную киберспортивную команду.

Робототехника и индустрия 4.0

ОП в промышленной робототехнике – область чрезвычайно активных исследований, поскольку является естественным внедрением этой парадигмы в практику. Потенциал интеллектуальных промышленных роботов и выгоды от их использования велики и многообразны. ОП наделяет индустрию 4.0 (или, как говорят, четвертую промышленную революцию) интеллектуальными устройствами, системами и роботами, которые могут выполнять очень сложные операции, действуя рациональным образом. Системы, умеющие прогнозировать необходимость технического обслуживания, диагностировать себя в режиме реального времени и управлять производственной деятельностью, можно объединить в единый комплекс, добиваясь улучшенного управления и более высокой производительности труда.

Машинное обучение

Гибкость ОП позволяет применять его не только для решения автономных задач, но и как своего рода метод точной настройки алгоритмов обучения с учителем. Во многих задачах **обработки естественных языков** (**ОЕЯ**) и компьютерного зрения подлежащая оптимизации функция не дифференцируема, поэтому для настройки параметров нейронной сети необходима вспомогательная дифференцируемая функция потерь. Но расхождение между двумя функциями потерь может снизить качество конечной системы. Один из способов решения этой проблемы состоит в том, чтобы сначала обучить систему, применяя методы обучения с учителем со вспомогательной функцией потерь, а затем использовать ОП для окончательной оптимизации сети относительно конечной метрики. Например, этот процесс можно с пользой применить в таких областях, как машинный перевод и вопросно-ответные системы, где метрики, по которым оценивается результат, сложны и недифференцируемы.

Кроме того, ОП может решать такие задачи ОЕЯ, как построение диалоговых систем и порождение текстов. Компьютерное зрение, локализация, анализ движения, визуальный контроль и визуальное слежение – все эти системы можно обучать методами ОП.

Глубокое обучение позволяет решить трудную задачу ручного конструирования признаков, оставляя человеку проектирование архитектуры нейронной сети. Это трудоемкая операция, смысл которой – наилучшим образом объединить несколько частей. Так почему бы не автоматизировать ее? Вообще-то можно. **Проектирование нейронной архитектуры** (neural architecture design – NAD) – подход, в котором ОП используется для проектирования архитектуры глубоких нейронных сетей (ГНС). Вычислительно он обходится очень дорого, но все же позволяет создавать архитектуры ГНС, не уступающие лучшим решениям в области классификации изображений.

Экономика и финансы

Управление бизнесом – еще одно естественное применение ОП. Оно успешно использовалось в интернет-рекламе, когда целью было максимизировать выручку от показа объявлений с платой за клик, касающихся рекомендаций по выбору продуктов, в сфере управления отношениями с клиентами и в маркетинге. А в области финансов ОП применялось для решения задач ценообразования опционов и многопериодической оптимизации.

Здравоохранение

В здравоохранении ОП используется для диагностики и лечения. С его помощью можно построить исходную оценку для снабженного искусственным интеллектом помощника врачей и медсестер. В частности, ОП помогает подготовить индивидуальные прогрессивные назначения для пациентов – этот процесс называется режимом динамического лечения. Другие примеры применения ОП в здравоохранении – персонализированный контроль уровня глюкозы в крови и персонализированное лечение сепсиса и СПИДа.

Интеллектуальные транспортные системы

В этой области ОП применяется для разработки и улучшения всех типов транспортных систем: интеллектуальные сети для управления пробками (например, управление светофорами), наблюдение за дорожным движением, безопасность (прогнозирование столкновений) и беспилотные автомобили.

Оптимизация энергопотребления и умные сети электроснабжения

Оба направления – основа интеллектуальной генерации, распределения и потребления электричества. В системы принятия решений и контроля можно внедрить методы ОП, чтобы обеспечить динамическую реакцию на изменение условий в окружающей среде. ОП можно также использовать для регулирования спроса на электроэнергию в ответ на динамическое ценообразование или для сокращения потребления.

Резюме

ОП – целеустремленный подход к принятию решения. От других парадигм он отличается прямым взаимодействием с окружающей средой и механизмом отложенного вознаграждения. Сочетание ОП с глубоким обучением очень полезно в задачах с многомерными пространствами состояний и с перцептивными входами. Понятия стратегии и функции ценности являются основными в ОП, поскольку говорят о том, какое действие предпринять, и о качестве состояний окружающей среды. В ОП модель окружающей среды необязательна, но может дать дополнительную информацию, а значит, улучшить качество стратегии.

Познакомившись с ключевыми концепциями, мы в последующих главах перейдем к самим алгоритмам ОП. Но сначала, прямо в следующей главе, заложим основы для разработки алгоритмов с помощью библиотек OpenAI и TensorFlow.

Вопросы

1. Что такое ОП?
2. Какова конечная цель агента?
3. Каковы основные различия между ОП и обучением с учителем?
4. Какие преимущества дает сочетание ОП с глубоким обучением?
5. Откуда берет начало термин «подкрепление»?
6. В чем разница между стратегией и функциями ценности?
7. Можно ли обучить модель окружающей среды посредством взаимодействия с ней?

Для дальнейшего чтения

- Пример неправильной функции вознаграждения см. по адресу https://blog.openai.com/faulty-reward-functions/.
- Для получения дополнительных сведений о глубоком ОП см. страницу по адресу http://karpathy.github.io/2016/05/31/rl/.