Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»

А.Н. Алфимцев

Мультиагентное обучение с подкреплением

Учебное пособие

2-е издание, исправленное

BAUMANPRESS
Москва
ИЗДАТЕЛЬСТВО
МГТУ им. Н. Э. Баумана
2022

УДК 004.85
ББК 32.813
А53

Издание доступно в электронном виде по адресу
https://bmstu.press/catalog/item/7425/

Факультет «Информатика и системы управления»
Кафедра «Информационные системы и телекоммуникации»

Рекомендовано Научно-методическим советом
МГТУ им. Н.Э. Баумана в качестве учебного пособия

Алфимцев, А. Н.

А53 Мультиагентное обучение с подкреплением : учебное пособие / А. Н. Алфимцев. — 2-е изд., испр. — Москва : Издательство МГТУ им. Н. Э. Баумана, 2022. — 222, [2] с. : ил.

ISBN 978-5-7038-5851-6

Рассмотрены современные и классические алгоритмы одновременного машинного обучения множества агентов, основанные на теории игр, табличных, нейросетевых, эволюционных и роевых технологиях. Представлено последовательное развитие теоретической модели алгоритмов, базирующееся на марковских процессах принятия решений. Реализация алгоритмов выполнена на языке программирования Python с использованием библиотеки глубокого обучения PyTorch. Средой машинного обучения является компьютерная игра StarCraft II с интерфейсом кооперативного мультиагентного обучения SMAC.

Для магистрантов и аспирантов направления подготовки «Информатика и вычислительная техника».

УДК 004.85
ББК 32.813

Уважаемые читатели! Пожелания, предложения, а также сообщения о замеченных опечатках и неточностях Издательство просит направлять по электронной почте: info@bmstu.press

© МГТУ им. Н.Э. Баумана, 2022
© Оформление. Издательство МГТУ им. Н.Э. Баумана, 2022

ISBN 978-5-7038-5851-6

Предисловие

Учебное пособие предназначено для самостоятельного изучения магистрантами и аспирантами основных разделов дисциплины «Мультиагентные интеллектуальные системы» и имеет практико-ориентированный характер, обеспечивая возможность решения конкретных профессиональных задач.

Цель изучения дисциплины — формирование профессиональных компетенций, связанных с разработкой программного обеспечения для мультиагентного обучения, а также общепрофессиональных компетенций, необходимых для применения на практике новых научных принципов и методов машинного обучения.

В учебном пособии сформированы внутридисциплинарные и междисциплинарные связи, обеспечивающие преемственность знаний, полученных при изучении предшествующих дисциплин по основам программирования, а также непрерывность отдельных видов математической и технологической подготовки разработчика программного обеспечения.

Системность изложения материала обеспечена логической последовательностью и аргументированностью приводимых положений. Новые положения базируются на ранее изученных.

Учебное пособие состоит из пяти глав, каждая из которых включает девять типизированных разделов: «Классификация», «Модель», «Алгоритм», «Карта», «Технология», «Код», «Эксперимент», «Выводы», «Задачи для самоконтроля». В последовательном изложении по принципу «от простого к сложному» объединены теория и технология, поскольку для изучения и развития современного мультиагентного обучения недостаточно ограничиваться теоретическими исследованиями либо заниматься только программированием чужих изысканий.

В разделе «Классификация» рассматриваются общая идея главы и различные алгоритмы мультиагентного обучения, определяется место представляемых алгоритмов в системе научных координат.

В разделе «Модель» приводится основной используемый в главе математический аппарат.

В разделе «Алгоритм» анализируется основной алгоритм мультиагентного обучения, рассматриваемый в данной главе. Псевдокод алгоритма дается в авторской редакции, которая больше соответствует его реальной реализации.

В разделе «Карта» описывается карта игры StarCraft II, на материале которой проводится мультиагентное обучение в каждой главе. Раздел

«Технология» посвящен особенностям настройки компьютерной игры и используемых программных библиотек.

В разделе «Код» дается оригинальная реализация основного алгоритма, рассматриваемого в главе, на языке программирования Python.

В разделе «Эксперименты» приводятся результаты применения рассматриваемых в главе алгоритмов.

В разделе «Выводы» тезисно подводятся некоторые итоги.

В конце каждой главы в разделе «Задачи для самоконтроля» представлены задачи по программированию мультиагентного обучения, которые позволят закрепить рассмотренный в главе теоретический материал практическими навыками. Задачи даны по возрастанию сложности в порядке, соответствующем основным тезисам глав.

Гл. 1 учебного пособия посвящена независимому табличному Q-обучению. Алгоритмы обучения с подкреплением одними из первых были применены для мультиагентного обучения. Показаны основные преимущества и недостатки классического алгоритма Q-обучения.

В гл. 2 рассматривается мультиагентное обучение в матричных и стохастических играх с использованием теории игр. Несмотря на то что алгоритмы на основе теории игр объективно уступают современным алгоритмам, многие созданные в этот период фундаментальные концепции развиваются до сих пор.

В гл. 3 рассматриваются сильнейшие на сегодняшней день алгоритмы мультиагентного обучения, основанные на теории искусственных нейронных сетей. Показано, как алгоритмы IQN и MADDPG помогают добиться высоких результатов в микроменеджменте действий агентов в игре StarCraft II.

Считается, что агент может быть по-настоящему автономным, если он способен не просто достигать целей функционирования, но и создавать их. В гл. 4 анализируется современные принципы эволюционного и коэволюционного обучения.

Заключительная глава — гл. 5 — посвящена гомогенному кооперативному мультиагентному обучению, для которого иногда применяется термин «роевое обучение». Предполагается, что вдохновленные живой природой алгоритмы роевого обучения являются наиболее перспективной альтернативой алгоритмам нейросетевого обучения.

Каждую главу, а также введение и заключение предваряют эпиграфы, взятые из написанного в жанре фэнтези романа американского математика и писателя Юн Ха Ли «Гамбит девятихвостого лиса» (М.: Эксмо, 2018). В книге нестандартно спрогнозированы развитие мультиагентных технологий и роль человека в техногенном мире будущего. Главный герой романа учится эффективно использовать ресурсы и находить оптимальные стратегии действий, что отвечает основной идее представленного учебного пособия.

Введение

...Никто не устоял бы перед возможностью
использовать игру в качестве педагогического инструмента.
Юн Ха Ли. Гамбит девятихвостого лиса

В Национальной стратегии развития искусственного интеллекта одной из приоритетных целей является создание технологий распределенных коллективных систем. Для достижения этой цели необходимо решить задачи разработки программного обеспечения, математических моделей и алгоритмов мультиагентного обучения с подкреплением.

Мультиагентная система (*англ.* multi-agent system) — система, состоящая из нескольких взаимодействующих агентов.

Агент — компьютерная программа, которая расположена в некоторой среде и способна автономно действовать и обучаться в этой среде для достижения своих целей. Под средой обычно понимается все физическое и виртуальное пространство, находящееся вне агента.

Машинное обучение (*англ.* machine learning) — совокупность алгоритмов и моделей, которые позволяют агенту для достижения цели без предварительного программирования находить закономерности в данных, делать прогнозы, самостоятельно принимать и улучшать решения на основе полученного опыта взаимодействия.

Обучение с подкреплением (*англ.* reinforcement learning) — один из способов машинного обучения, особенностью которого является обучение агента в процессе взаимодействия со средой.

Мультиагентное обучение с подкреплением (*англ.* Multi-agent Reinforcement Learning, MARL) — совокупность алгоритмов и моделей машинного обучения с подкреплением, которые дают возможность обучать мультиагентные системы.

Мультиагентные системы как отдельная область дисциплины «Информатика и вычислительная техника» начала развиваться в конце XX в. В последние 20 лет интерес к этой области постоянно возрастает, о чем свидетельствуют более полумиллиона поисковых результатов в системе «Академия Google».

С одной стороны, этот рост связан с убеждением, что мультиагентные системы являются одной из перспективных парадигм разработки программного обеспечения. Смещение акцента с разработки приложений для персональных рабочих станций на разработку распределенных систем, взаимодействующих друг с другом дистанционно, привело к усложнению этих систем и приобретению ими свойств, характерных для мультиагентных систем.

С другой стороны, ключевым свойством агента, функционирующего в незнакомой среде, является способность к обучению на основе полученного опыта взаимодействия с этой средой или с другими агентами. Очевидные успехи глубокого одноагентного обучения в распознавательных и игровых задачах, достигнутые в последние годы, мотивируют исследователей и разработчиков использовать потенциал этих методов для мультиагентного обучения.

Мультиагентное обучение в настоящее время — фундаментальный компонент мультиагентной системы. С научной точки зрения исследование взаимодействия между множеством одновременно обучающихся агентов позволяет получить новые знания, которые могут быть полезны при анализе социальных структур общества, экономических конкурентных сред, в теории управления. С инженерной точки зрения одновременно обучающееся множество агентов может предоставить концептуально новый способ разработки коммуникационных систем, робототехники, транспортных систем, человеко-машинных интерфейсов, программного обеспечения.

В целом мультиагентный подход при построении системы в *сложной*, т. е. недетерминированной, динамической, непрерывной, *среде* дает следующие преимущества.

• *Робастность*. Мультиагентная система способна продолжить функционирование при сбоях в работе отдельных агентов.

• *Эффективность*. Параллельная обработка данных в мультиагентной системе автономными агентами ускоряет функционирование системы.

• *Адаптивность*. Децентрализованная мультиагентная система может динамически изменять свое поведение при изменениях в среде.

• *Инкапсулируемость*. Мультиагентная система модульна по своей природе, что позволяет гибко изменять ее структуру, абстрагировать данные и защищать внутренние реализации.

• *Масштабируемость*. При восприятии информации и принятии решения мультиагентная система не имеет ограничений по централизованному контролю данных, что дает возможность лучше справляться с увеличением рабочей нагрузки при добавлении ресурсов.

Однако применение методов одноагентного машинного обучения к мультиагентным системам приводит и к появлению новых вызовов.

Во-первых, мультиагентное обучение подвержено «проклятию размерности», вызванному экспоненциальным ростом пространства состояний-действий при увеличении числа агентов. Так, при табличном одноагентном обучении с подкреплением семи агентов с шестью возможными действиями для каждого состояния среды необходимо хранить 42 Q-значения. А при мультиагентном обучении совместным действиям нужно будет учесть 279 936 Q-значений.

Во-вторых, многие теоретические гарантии, которые существовали и формально доказывались при одноагентном обучении, при многоагентном обеспечении не действуют. Так, классическим алгоритмом Q-обучения

с подкреплением гарантируется сходимость к оптимальной стратегии для конечного марковского процесса принятия решений. Но при мультиагентном обучении агентам приходится довольствоваться не оптимальной стратегией, а некоторым усредненным равновесием, которого они достигли в действиях друг с другом и со средой.

В-третьих, выбор цели мультиагентного обучения, которую можно формально описать и автоматически проверить, не является столь очевидным решением, как при одноагентном обучении. На достижение цели максимизации награды агента при мультиагентном обучении с подкреплением могут влиять, например, результаты обучения других агентов. В связи с этим в практических задачах для формулирования цели мультиагентного обучения требуются отдельные, иногда уникальные, решения, касающиеся контроля стабильности обучения, адаптируемости к изменению поведения других агентов, устойчивости внутренней мотивации.

В-четвертых, нестационарность машинного обучения при мультиагентном обучении проявляется с иной стороны. Когда одновременно обучается несколько агентов, они постоянно подстраивают свои действия под действия других агентов, которые, в свою очередь, подстраиваются под них. Отдельной проблемой здесь является определение оптимальности стратегии, так как стратегия должна постоянно меняться в соответствии с изменяющимися стратегиями других агентов.

В-пятых, проблема эксплуатации-исследования также значительно усложняется при мультиагентном обучении. В случае, когда принять решение должен один агент, достаточно исследовать среду и можно начинать действовать, эксплуатировать полученный опыт. Но при обучении множества агентов этот индивидуальный переход одного агента от исследования к эксплуатации или наоборот неявно влияет на стратегии всех других агентов.

И наконец, шестой вызов связан с координацией действий агентов. Эффект от каждого действия агента зависит от действия или бездействия других агентов. В решении проблемы координации приходится постоянно балансировать между поиском эффективных совместных действий агентов и поиском оптимальных стратегий. Несмотря на то что эта проблема больше характерна для кооперативного мультиагентного обучения, отсутствие координации негативно влияет и на независимо обучающихся агентов.

В последние годы главные результаты машинного обучения обучающихся агентов достигнуты в компьютерных играх. Иногда результаты игроков-агентов сопоставимы с результатами игры профессиональных игроков-людей или даже их превосходят. Такие компьютерные игры, как стратегическая компьютерная игра StarCraft II, с точки зрения машинного обучения представляет собой сложную среду, в которой встречаются все актуальные вызовы мультиагентного обучения; в ней также можно безопасно промоделировать ситуации реального мира: частичный обзор информации, кооперативность или некооперативность взаимодействия,

децентрализованный микроменеджмент, большой объем входных данных, тысячи возможных действий, разнообразный набор сценариев, нестационарность обучения.

В связи с этим технологическая часть учебного пособия построена на базе компьютерной игры StarCraft II, разработанной кампанией Blizzard, и программной библиотеки мультиагентного обучения SMAC, разработанной исследовательской лабораторией Whiteson Оксфордского университета.

Примеры, приведенные в данном учебном пособии, опубликованы в открытом виде на GitHub, чтобы была возможность не только повторить рассматриваемые алгоритмы, но и провести модификацию предлагаемого решения, сравнить полученные результаты с результатами, приведенными в учебном пособии.

Глава 1

Независимое табличное обучение

Календарная война — война сердец.
Однако числа способны влиять на сердца при условии,
что и те, и другие верные.

Юн Ха Ли. Гамбит девятихвостого лиса

1.1. Классификация

Первые методы мультиагентного обучения базировались на алгоритмах обучения с подкреплением. Особенностью *обучения с подкреплением* является обучение агента в процессе взаимодействия со средой. Алгоритмы обучения с подкреплением не требуют априорной информации о среде или какой-либо дополнительной модели среды. Агенты $i = 1, 2, ..., n$ обучаются в процессе взаимодействия, получая от среды положительные или отрицательные подкрепляющие сигналы $r_t^1, r_t^2, \ldots, r_t^n$ и новые состояния $s_{t+1}^1, s_{t+1}^2, \ldots, s_{t+1}^n$, зависящие от действий $a_t^1, a_t^2, \ldots, a_t^n$ агентов (рис. 1.1).

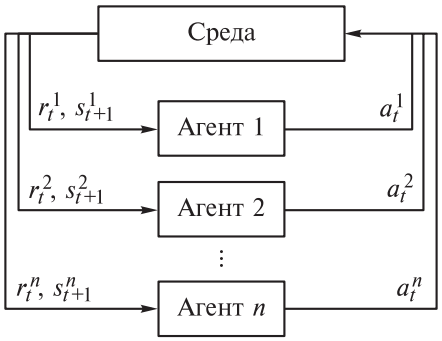


Рис. 1.1. Схема независимого мультиагентного обучения с подкреплением

Подкрепляющий сигнал менее информативен, чем в алгоритмах обучения с учителем, но все же дает некоторую информацию, в то время как в алгоритмах обучения без учителя такой сигнал отсутствует. При использовании алгоритмов обучения с подкреплением для мультиагентного обучения появляется возможность естественно совместить преимущество двух

подходов: автономное и адаптивное обучение множества агентов в сложных средах, способных децентрализованно достигать цели функционирования мультиагентной системы.

Несмотря на то что многие свойства одноагентного обучения не выполняются в случае мультиагентного обучения, в научной литературе были предложены и на практике апробированы разнообразные табличные алгоритмы мультиагентного независимого и совместного обучения (рис. 1.2). *Табличными* эти алгоритмы называют потому, что для хранения и обновления полученной в процессе обучения информации используют двухмерные таблицы данных.

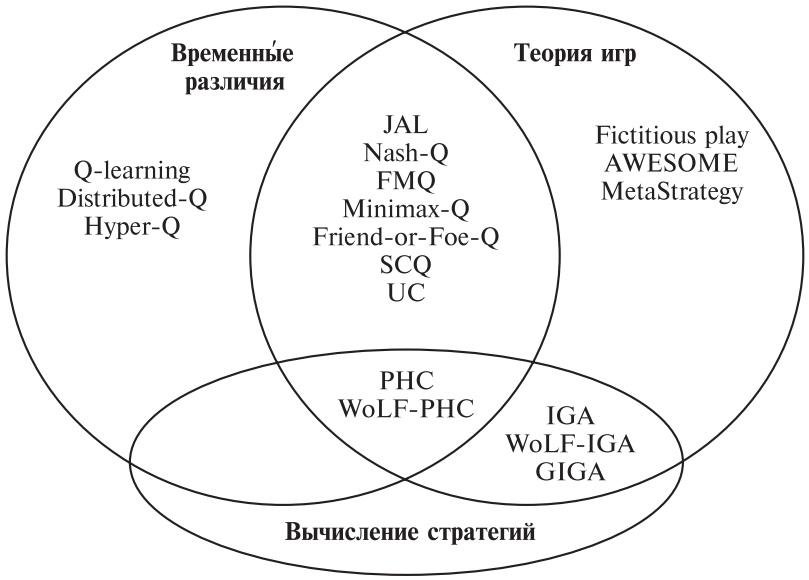


Рис. 1.2. Классификация табличных алгоритмов мультиагентного обучения с использованием принципов временны́х различий, теории игр и вычисления стратегий

В целом табличные алгоритмы мультиагентного обучения с подкреплением можно разделить на три класса: основанные на принципах временны́х различий; основанные на теории игр; позволяющие напрямую вычислять распределение вероятности для формирования стратегии действий.

Мультиагентное обучение с использованием теории игр, в том числе с вычислением стратегий, будет рассмотрено в гл. 2. В гл. 1 внимание уделено табличному алгоритму Q-обучения, в котором используется *принцип временны́х различий*. Этот принцип — один из центральных в безмодельном обучении с подкреплением, он определяет алгоритмы, с помощью которых

обучаются агенты на основе вычисления разности Q-значений или ценности состояний для смежных последовательных событий.

На принципе временны́х различий также основаны: алгоритм распределенного Q-обучения (*англ.* Distributed Q-learning), особенность которого — решение задач независимого мультиагентного обучения с исключительно положительными наградами от среды и невысокой вычислительной сложностью, сравнимой со сложностью классического алгоритма Q-обучения; алгоритм гипер-Q (*англ.* Hyper-Q), который позволяет строить модели действий других агентов и поэтому успешно адаптироваться к их нестационарному поведению, но именно вследствие этих дополнительных моделей подвержен «проклятию размерности».

В этой главе рассматривается полностью кооперативное мультиагентное обучение, при котором агенты имеют одинаковую функцию наград и награда для одного агента также является наградой для группы. Основное внимание уделено агентам, которые не взаимодействуют друг с другом и не имеют информации о действиях других агентов, — так называемым *независимо обучающимся* агентам. Такие агенты получили популярность в робототехнике в тех случаях, когда физически невозможно получить или обработать информацию о совместных действиях большого числа агентов.

1.2. Модель

Большинство алгоритмов одноагентного обучения с подкреплением изначально было разработано для математической модели МППР (марковских процессов принятия решений) (*англ.* Markov decision processes). *Марковский процесс принятия решений* — это кортеж (S, A, T, R), где S — конечное множество состояний (*англ.* States); A — конечное множество действий (*англ.* Actions) агента; $T : S \times A \times S \to [0;1]$ — функция переходов (*англ.* Transition), $R : S \to \mathbb{R}$ — функция наград (*англ.* Reward).

Марковский процесс принятия решений обладает марковским свойством, согласно которому будущие переходы и состояния полностью зависят только от текущего состояния. Вероятность перейти в состояние s' после выполнения действия a в состоянии s обозначают $T(s, a, s')$. Для всех действий a и всех состояний s и s' выполняется $0 \le T(s, a, s') \le 1$ и $\sum_{s' \in S} T(s, a, s') = 1$. Функция наград R возвращает награду $R(s, a, s')$ после выполнения действия a в состоянии s и перехода в состояние s'.

Применение МППР для мультиагентного обучения привело к появлению различных модификаций базового определения. В табл. 1.1. представлены определения МППР, разработанные с учетом стратегии агента π_i и множества стратегий агента Π_i для независимых и совместных состояний, действий и наград.

Таблица 1.1

Модифицированные МППР для мультиагентного обучения

Тип награды		
Тип действий	Независимая награда	Совместная награда
Независимые состояния		
Независимые действия	$(S_i, A_i, T, R_i, \Pi_i)$: $\pi_i \in \Pi_i$, $s_i \in S$, $a_i \in A_i$, $r_i \in R_i$	(S_i, A_i, T, R, Π_i): $\pi_i \in \Pi_i$, $s_i \in S$, $a_i \in A_i$, $r \in R$
Совместные действия	(S_i, A, T, R_i, Π_i): $\pi_i \in \Pi_i$, $s_i \in S$, $a \in A$, $r_i \in R_i$	(S_i, A, T, R, Π): π, $s_i \in S$, $a \in A$, $r \in R$
Совместные состояния		
Независимые действия	(S, A_i, T, R_i, Π_i): $\pi_i \in \Pi_i$, $s \in S$, $a_i \in A_i$, $r_i \in R_i$	(S, A_i, T, R, Π_i): $\pi_i \in \Pi_i$, $s \in S$, $a_i \in A_i$, $r \in R$
Совместные действия	(S, A, T, R_i, Π_i): $\pi_i \in \Pi_i$, $s \in S$, $a \in A$, $r_i \in R_i$	(S, A, T, R, Π): π, $s \in S$, $a \in A$, $r \in R$

Целью обучения в МППР является поиск оптимальной стратегии $\pi^*: S \to A$, которая задает действия агента, приводящие к получению максимальной награды. Качество стратегии определяется *функцией ценности* V^π (*англ.* Value). *Оптимальной стратегией* $\pi^*(s)$ называется стратегия, при которой $V^{\pi^*}(s) \geq V^\pi(s)$ для всех π и $s \in S$. *Функция ценности состояния для стратегии* π — $V^\pi(s)$ определяет общее количество награды, которую агент ожидает получить, начав из состояния s следовать стратегии π:

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma_t R(s_t) | s_0 = s\right],$$

где γ — параметр дисконтирования, $\gamma \in [0,1]$.

Однако функция ценности состояния не свидетельствует о том, какие конкретно действия следует выполнять агенту в определенном состоянии. Для этого задается *функция ценности действия для стратегии* π — $Q^\pi(s, a)$, определяющая общее количество награды, которую агент ожидает получить, если выполнит действие a в состоянии s, следуя стратегии π:

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma_t R(s_t, a_t) | s_0 = s,\ a_0 = a\right],\ \gamma \in [0,1].$$

Функция ценности состояния $V^\pi(s)$ в обучении с подкреплением и в динамическом программировании обладает фундаментальным свойством,

заключающимся в том, что эту функцию можно вычислить рекурсивно с помощью уравнения Беллмана. Функция ценности состояния для любой стратегии π и любого состояния s вычисляется на основе полученной награды $R(s)$ и предполагаемой ценности следующего состояния $V^{\pi}(s')$ взвешенной вероятностью перехода в это состояние:

$$V_{t+1}^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V_t^{\pi}(s').$$

Рассмотрим результаты решения уравнения Беллмана на примере МППР, представленного в виде графа переходов на рис. 1.3. Марковский процесс принятия решений задает независимое поведение агента в некоторой среде, где агент перемещается по карте и ищет противников. В библиотеке SMAC для компьютерной игры StarCraft II у каждого агента информация о среде ограничена окружностью с радиусом девять квадратов игрового поля для наблюдения и радиусом шесть квадратов — для стрельбы. В связи с этим

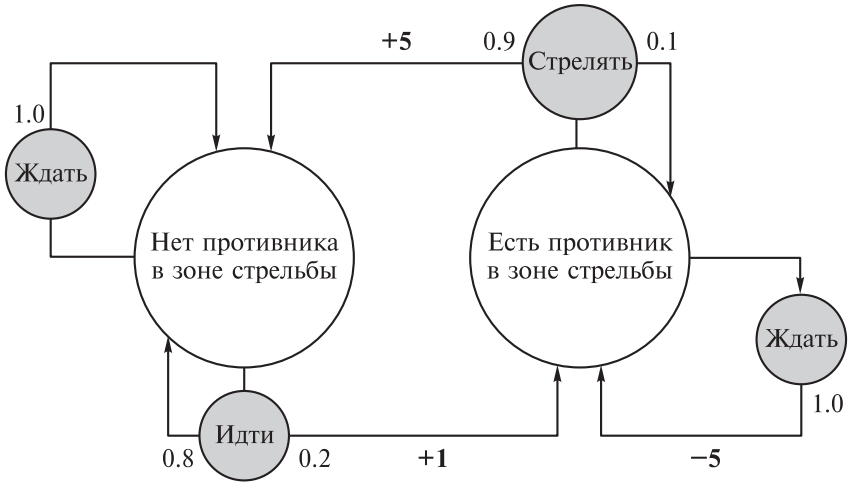


Рис. 1.3. Граф переходов марковского процесса принятия решений

зададим в МППР состояния S = {нет противника в зоне стрельбы, есть противник в зоне стрельбы}. Предположим, что если агент находится в первом состоянии s_1, то он может выполнить два действия A = {идти, ждать}. Если агент находится во втором состоянии s_2, то он может выполнить действия A = {стрелять, ждать}.

Вероятности переходов в первом состоянии s_1 при выполнении действий a_1 и a_2 определим как $T(s_1, a_1, s_2) = 0.2$; $T(s_1, a_1, s_1) = 0.8$; $T(s_1, a_2, s_1) = 1.0$. Вероятности переходов во втором состоянии s_2 при выполнении действий a_3 и a_4 определим как $T(s_2, a_3, s_1) = 0.9$; $T(s_2, a_3, s_2) = 0.1$; $T(s_2, a_4, s_2) = 1.0$.

Используя стандартный прием обучения с подкреплением, зададим цели агента с помощью дизайна наград МППР. Для того чтобы агент не просто искал противников, а вступал с ними в бой, определим награду как $R(s_1, a_1, s_2) = 1$ и $R(s_2, a_3, s_1) = 5$. В то же время, если агенту-морпеху противостоит зерглинг, то в качестве одной из целей необходимо предусмотреть сохранение дистанции с противником, потому что зерглинг — это наземный юнит ближнего боя, который может наносить урон, только находясь рядом с агентом. В данном МППР такую предосторожность можно задать лишь отрицательной наградой $R(s_2, a_4, s_2) = -5$.

В результате решения уравнения Беллмана на 10 итерациях с параметром $\gamma = 0.9$ ценность первого состояния МППР составит $V^{\pi}(s_1) = 0.243$, ценность второго состояния $V^{\pi}(s_2) = 20.804$. Ценность действий в первом состоянии рассчитана как $Q^{\pi}(s_1, a_1) = 6.868$; $Q^{\pi}(s_1, a_2) = 5.677$, во втором состоянии — как $Q^{\pi}(s_2, a_3) = 9.622$; $Q^{\pi}(s_2, a_4) = 3.317$. Таким образом, оптимальной стратегией π^* для данного МППР является в состоянии s_1 выполнение действия a_1 и в состоянии s_2 — выполнение действия a_3.

1.3. Алгоритм

Одним из прорывов в области одноагентного обучения с подкреплением стала разработка, основанная на принципах временны́х различий офлайнового алгоритма управления, больше известного в настоящее время как *алгоритм Q-обучения* (*англ.* Q-learning). Алгоритму Q-обучения не требуется модель среды в виде априорной информации о функции переходов и функции наград, алгоритм применим в средах со стохастическими переходами и наградами. Для любого конечного МППР в алгоритме Q-обучения находится оптимальная стратегия в отношении максимизации ожидаемой общей награды. Символ Q в названии алгоритма и его потомков происходит от *англ.* Quality — качество и означает качество действия, предпринятого в некотором состоянии.

Первой попыткой развития алгоритма Q-обучения применительно к мультиагентному обучению стала его реализация для независимого табличного обучения, при которой агент, использующий алгоритм, действовал рационально, но сходимость алгоритма к оптимальной стратегии не гарантировалась. При независимом Q-обучении агент игнорировал существование других агентов, предполагая, что все награды среды́ и переходы стационарны и обладают марковским свойством. Приобретаемый опыт агента i в таком случае имеет вид (a_i, r) где a_i — действие, выполненное агентом i; r — полученная награда.

Алгоритм независимого табличного Q-обучения для агента i, где α — скорость обучения; ε — параметр эксплуатации-исследования; Ne — число эпизодов, приведен ниже:

1. Инициализировать
   $\alpha \in (0, 1]$; $\gamma \in (0, 1]$; $\varepsilon \in (0, 1)$; $Ne \in [1, N]$;
   $i \leftarrow 1, \ldots, n$; $a \in A_i$;
   $Q(s, a) \leftarrow 0$, $\forall s \in S$.
2. Цикл
3. Получить состояние s.
4. Определить возможные действия $a \in A_i$ в состоянии s.
5. Выбрать возможное действие a в соответствии с максимальным значением $Q(s, a)$ и с учетом параметра ε.
6. Выполнить возможное действие a.
7. Получить награду r и следующее состояние s'.
8. Обновить значение $Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s', a') - Q(s,a)\right)$.

Главной инновацией алгоритма Q-обучения стала вдохновленная уравнением Беллмана формула итерационного обновления Q-значений на основе полученной награды и максимальной ценности действия в следующем состоянии (шаг 8). При этом Q-значения $Q(s, a)$ обычно инициализируются нулями (шаг 1) и в процессе обучения хранятся в виде Q-таблиц. В Q-таблице строки соответствуют всем возможным состояниям МППР, столбцы — всем возможным действиям (см. разд. 1.4).

Однако несмотря на инновативность алгоритма Q-обучения, в него была заложена проблема «жадного» выбора действий a^* как максимального значения $Q(s, a^*) = \max Q(s, a_i)$ (шаг 5). Иначе говоря, в алгоритме не исследуются другие варианты действий и не проверяется их эффективность для работы в динамической среде, что особенно важно при длительном периоде обучения. Альтернативой постоянного «жадного» выбора действий стала концепция ε-«жадного» выбора действий, в соответствии с которой вводится дополнительный параметр эксплуатации-исследования $\varepsilon \in (0, 1)$. В результате с небольшой вероятностью $1 - \varepsilon$ в алгоритме Q-обучения выбираются случайные действия для исследования пространства действий и с вероятностью ε осуществляется «жадный» выбор действий, т. е. эксплуатируется полученный опыт (порядок значений ε и $1 - \varepsilon$ может быть инвертирован и зависит от реализации).

Еще одной проблемой длительного обучения, которую мультиагентное обучение получило в наследство от одноагентного, стала невозможность точно определить, результатом какого действия в будущем становится та или иная награда среды. В связи с этим следующей концепцией, активно используемой в алгоритме Q-обучения, стала концепция дисконтированной награды, определяемой параметром $\gamma \in [0, 1]$. Параметр дисконтирования γ задает степень влияния будущих наград на обновление Q-значений $Q(s, a)$. Если установить значение параметра дисконтирования γ близким к нулю,

то первые полученные награды будут существенно влиять на обновление Q-значений $Q(s, a)$. Если установить значение параметра дисконтирования γ близким к единице, то отдаленная во времени награда будет оказывать большее влияние на обновление Q-значений $Q(s, a)$. Но в целом попытка максимизации ближайших наград приводит к уменьшению общей ожидаемой награды.

Помимо этого, в структуру алгоритма Q-обучения заложен параметр скорости обучения α, который определяет, в какой степени вновь полученный опыт переопределяет старую информацию. Высокая скорость обучения (значение α близко к единице) позволяет обучаться быстрее, но может привести к пропуску глобального минимума. При низкой скорости обучения (значение α близко к нулю) агент обучается медленнее, что может привести к нахождению лишь локального минимума. По этой причине параметр скорости обучения α иногда задают в виде формулы, зависящей от эпизода обучения или от промежуточных результатов обучения.

1.4. Карта

Независимое табличное мультиагентное обучение в компьютерной стратегической игре StarCraft II с использованием программной библиотеки SMAC в гл. 1 проводилась на карте 2m2zFOX.SC2Map, представленной на рис. 1.4. В начале каждого эпизода на карте слева, в точке Team 1, автоматически создаются два агента юнита морпеха, которыми управляет обучающий алгоритм, справа, в точке Team 2, автоматически создаются два агента юнита зерглинга, которыми управляет компьютер, используя стандартные правила. Длительность одного полного эпизода игры составляет 3 с.

Юнит — игровая единица (персонаж) в компьютерных стратегических играх. Морпех — базовый военный юнит терранов. Имеет 45 единиц

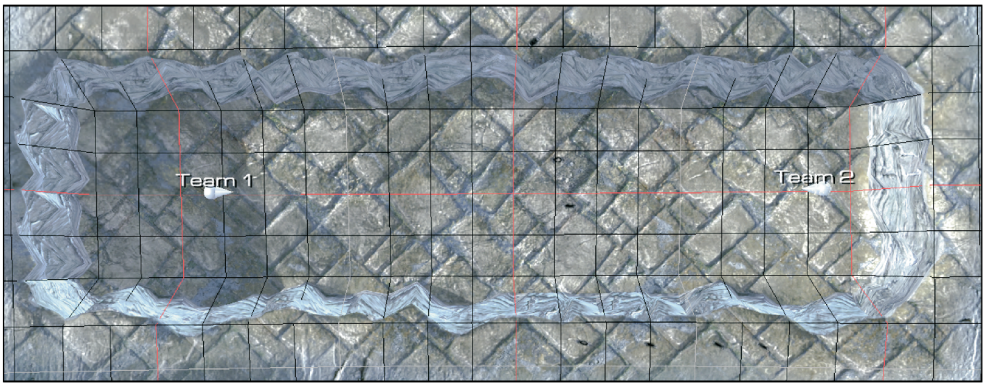


Рис. 1.4. Внешний вид карты 2m2zFOX.SC2Map в редакторе StarCraft II Editor

здоровья. Атакует наземные и воздушные юниты на расстоянии, нанося шесть единиц урона (дальность атаки — пять квадратов игрового поля). Стратегически используется при атаке пехотой и при защите от массовых атак противника. Изначально не имеет брони.

В отличие от морпеха, зерглинг расы зергов обладает меньшим количеством здоровья — 35 единиц и атакой пять единиц с дальностью 0.1. Стратегически применяется для ранних массовых атак на базы противника и разведки. Наиболее опасен для противника при использовании в комбинациях с другими юнитами.

Поскольку в этой главе рассматривается табличное обучение, для данной карты необходимо разработать МППР, представленный миром-сеткой SMAC.

Мир-сетка (*англ*. Grid World) — это двухмерное прямоугольное пространство, состоящее из ячеек одного размера (*Nx*, *Ny*). В мире-сетке агент может перемещаться только по горизонтали и по вертикали на одну клетку за одно действие. Агент всегда занимает только одну ячейку в мире-сетке и может взаимодействовать только с объектами, находящимися в данной клетке (но для действия «стрелять» в мире-сетке SMAC разрешено взаимодействие на нескольких клетках). Мир-сетка — довольно абстрактная среда, но она позволяет упростить мультиагентное обучение и пренебречь побочными факторами при проведении экспериментов. Кроме того, мир-сетка моделирует важные элементы сложных сред: зависимость действия от состояния, качественные и количественно-взвешенные переходы между состояниями, мгновенные и отложенные награды.

Мир-сетка SMAC 2×8 состоит из 16 клеток, соответствующих состояниям МППР $S = \{0, 1, 2, ..., 15\}$ (рис. 1.5). Соответствие координат карты состояниям МППР представлено в табл. 1.2. Для каждого агента задан свой МППР. Два агента $i = 1, 2$ начинают обучение в состояниях $s_1^3 = 3$ и $s_2^{11} = 11$, не имея информации о месторасположении противников и о существующих наградах. Для того чтобы применить алгоритм Q-обучения, необходимо для каждого агента также задать Q-таблицу для хранения выученных оценок действий в определенных состояниях МППР. В табл. 1.3 приведен пример обученной Q-таблицы первого агента, где реализована оптимальная стратегия (ячейки таблицы пронумерованы, № 1–4).

Каждый агент может выполнить действия следующих типов: $A_i = \{\text{no} - \text{op}$, stop, NORTH, SOUTH, EAST, WEST, attack № 1, attack № 2$\}$; здесь действие «no – op» («нет операций») означает, что агент погиб; действие stop задает полное бездействие агента; четыре действия NORTH, SOUTH, EAST, WEST определяют перемещения агента соответственно вверх, вниз, вправо, влево; два действия attack № 1, attack № 2 задают атаку агента — стрелять в первого или во второго противника (см. рис. 1.5). Каждый агент может совершить только одно действие в единицу игрового времени. Эпизод игры заканчивается, когда истекает время эпизода, погибают оба противника или оба агента.



Рис. 1.5. Мир-сетка SMAC для двух агентов

Главными целями агентов в мире-сетке SMAC являются получение максимальной награды и достижение максимального процента побед. Оптимальной стратегией π_n^*, позволяющей достичь целей за отведенный промежуток времени, является стратегия «бежать вправо и стрелять при появлении противника».

Награда изначально задается программной библиотекой SMAC и вычисляется по формуле $R = \frac{(\text{reward_battle} + \text{reward_win})}{(\text{max_reward} / \text{reward_scale_rate})}$, где reward_win — награда +1 и –1 соответственно за победу и поражение в эпизоде; reward_scale_rate — коэффициент шкалы наград, равный 20.

Награда за битву reward_battle вычисляется по формуле reward_battle = = delta_enemy+delta_deaths–delta_ally, где delta_enemy — урон, нанесенный здоровью или броне противников; delta_deaths — отношение числа потерянных союзников к числу убитых противников; delta_ally — урон, нанесенный здоровью или броне агентов.

Максимальная награда карты max_reward вычисляется по формуле max_reward = n_enemies×reward_ death_value+reward_win, где n_enemies — число противников на данной карте; reward_ death_value — награда за уничтожение юнита противника, равная 10.

В итоге максимальная награда без дисконтирования для рассматриваемого мира-сетки SMAC составляет 20.381, максимальный процент побед равен 1.0, что соответствует 100 %.

Таблица 1.2

Соответствие координат *x*, *y* карты состояниям *s* МППР

s_2^0 $6 < x < 7$ $16.2 < y < 17$	s_2^1 $7 < x < 8$ $16.2 < y < 17$	s_2^2 $8 < x < 8.9$ $16.2 < y < 17$	s_2^3 $8.9 < x < 9.1$ $16.2 < y < 17$
s_2^4 $9.1 < x < 10$ $16.2 < y < 17$	s_2^5 $10 < x < 11$ $16.2 < y < 17$	s_2^6 $11 < x < 12$ $16.2 < y < 17$	s_2^7 $12 < x < 13.1$ $16.2 < y < 17$
s_2^8 $6 < x < 7$ $15.5 < y < 16.2$	s_2^9 $7 < x < 8$ $15.5 < y < 16.2$	s_2^{10} $8 < x < 8.9$ $15.5 < y < 16.2$	$\boldsymbol{s}_2^{11}$ $8.9 < x < 9.1$ $15.5 < y < 16.2$
s_2^{12} $9.1 < x < 10$ $15.5 < y < 16.2$	s_2^{13} $10 < x < 11$ $15.5 < y < 16.2$	s_2^{14} $11 < x < 12$ $15.5 < y < 16.2$	s_2^{15} $12 < x < 13.1$ $15.5 < y < 16.2$
s_1^0 $6 < x < 7$ $15 < y < 16.5$	s_1^1 $7 < x < 8$ $15 < y < 16.5$	s_1^2 $8 < x < 8.9$ $15 < y < 16.5$	$\boldsymbol{s}_1^3$ $8.9 < x < 9.1$ $15 < y < 16.5$
s_1^4 $9.1 < x < 10$ $15 < y < 16.5$	s_1^5 $10 < x < 11$ $15 < y < 16.5$	s_1^6 $11 < x < 12$ $15 < y < 16.5$	s_1^7 $12 < x < 13.1$ $15 < y < 16.5$
s_1^8 $6 < x < 7$ $14 < y < 15$	s_1^9 $7 < x < 8$ $14 < y < 15$	s_1^{10} $8 < x < 8.9$ $14 < y < 15$	s_1^{11} $8.9 < x < 9.1$ $14 < y < 15$
s_1^{12} $9.1 < x < 10$ $14 < y < 15$	s_1^{13} $10 < x < 11$ $14 < y < 15$	s_1^{14} $11 < x < 12$ $14 < y < 15$	s_1^{15} $12 < x < 13.1$ $14 < y < 5$

Таблица 1.3

Q-таблица мира-сетки SMAC для первого агента

Состояния	Действия							
	0	1	2	3	4	5	6	7
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.433	0.343	0.332	0.532	0.245	0.201	0.249
3	0.0	0.515	0.520	0.518	0.557 № 1	0.522	0.231	0.523
4	0.0	0.528	0.531	0.530	0.589 № 2	0.528	0.532	0.531

Окончание табл. 1.3

Состояния	Действия							
	0	1	2	3	4	5	6	7
5	0.0	0.621	0.632	0.530	0.635 № 3	0.631	0.633	0.549
6	0.0	0.0	0.863	0.0	0.874 № 4	0.0	0.170	0.0
7	0.0	0.0	0.439	0.250	0.432	0.285	0.933 № 5	0.863
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.510	0.501	0.588	0.521	0.213	0.522
12	0.0	0.0	0.511	0.523	0.575	0.321	0.514	0.535
13	0.0	0.0	0.761	0.0	0.799	0.320	0.787	0.765
14	0.0	0.0	0.313	0.251	0.431	0.283	0.838	0.831
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

На рис. 1.6–1.10 проиллюстрированы различные ситуации в мире-сетке SMAC: начало эпизода игры, противники атакуют справа вне зоны стрельбы (см. рис. 1.6); противник достигает зоны стрельбы вначале первого, а затем и второго агента (см. рис. 1.7, 1.8); один из агентов погиб, и возникла проигрышная позиция «двое на одного» (см. рис. 1.9); агенты наконец-то выучили правильную стратегию (см. рис. 1.10).

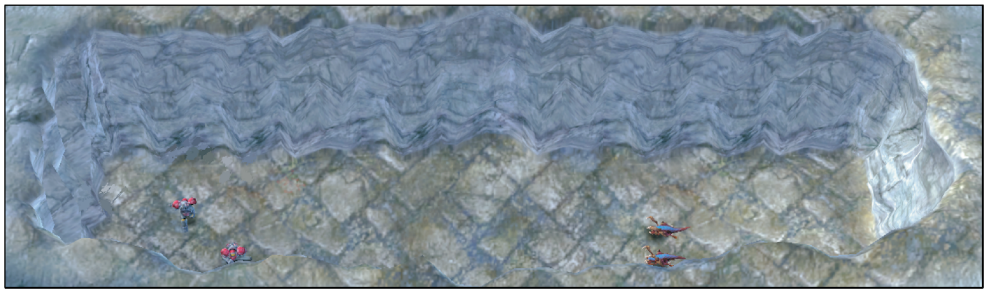


Рис. 1.6. Начало эпизода в мире-сетке SMAC

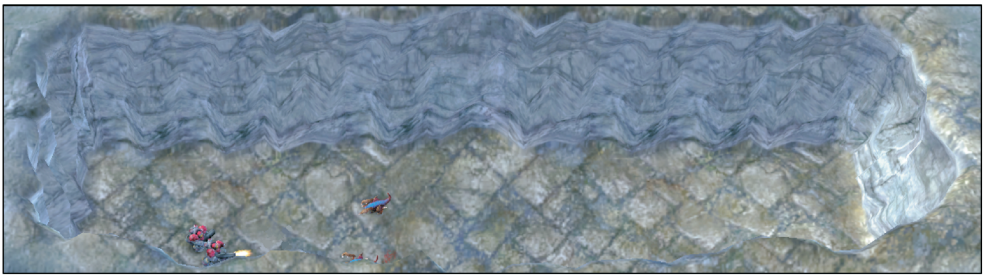


Рис. 1.7. Противник в зоне стрельбы первого агента

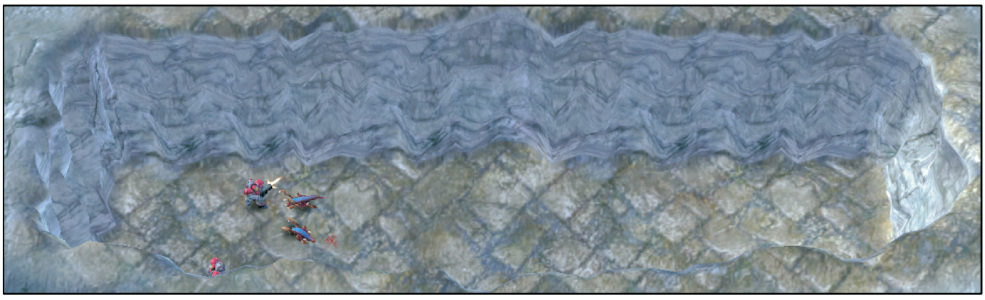


Рис. 1.8. Противник в зоне стрельбы второго агента



Рис. 1.9. Проигрышная позиция «двое на одного»



Рис. 1.10. Агенты реализуют оптимальную стратегию

Результаты экспериментального обучения в мире-сетке SMAC будут рассмотрены в разд. 1.7.

1.5. Технология

Рассматриваемый в этой главе алгоритм реализован на языке программирования Python версии 3.6. В представленном в разд. 1.6 программном коде используются стандартные библиотеки: numpy для работы с массивами, matplotlib для построения графиков, pickle для ввода/вывода данных в файл.

Основной программной библиотекой, используемой в данной главе и во всем учебном пособии, является библиотека кооперативного мультиагентного обучения SMAC (*англ.* StarCraft Multi-Agent Challenge). Кроме того, используются прикладной программный интерфейс машинного обучения StarCraft II от компании *Blizzard* и программная библиотека PySC2, разработанная компанией DeepMind. В отличие от аналогов, библиотека SMAC позволяет алгоритму машинного обучения управлять каждым агентом по отдельности и осуществлять децентрализованный микроменеджмент действий агентов.

В гл. 1 карта 2m2zFOXmatrix.SC2Map создана на базе карты SMAC в редакторе карт StarCraft II Editor. Ниже представлены изменения, которые необходимо внести в файл smac_maps.py, расположенный в папке «...\smac\env\starcraft2\maps», при создании новой карты или редактировании существующей:

```
«2m2zFOX»: { «n_agents»: 2, «n_enemies»: 2,
 «limit»: 240,
 «a_race»: «T»,
 «b_race»: «Z»,
«unit_type_bits»: 0, «map_type»: «marines»,}
```

Кроме того, файл с новой картой *.SC2Map должен быть добавлен в папку «...\StarCraft II\Maps\SMAC_Maps».

При обучении удобно ускорять визуализацию игры, а при анализе результатов нужен реальный масштаб времени. Существует возможность настройки скорости визуализации StarCraft II изменением значения переменной realtime с True на False в файле starcraft2.py, находящемся в папке «...\smac\env\starcraft2». После модификации настроек скорости или добавления новой карты необходимо перезапустить среду разработки программного обеспечения, чтобы загрузилась библиотека с изменениями.

Эксперименты по мультиагентному обучению проводились в операционной системе Windows 10. Для запуска примеров, приведенных в разд. 1.6, необходимо также установить бесплатный клиент компьютерной стратегической игры StarCraft II через приложение цифровой дистрибуции Blizzard Battle.net. Код, реализующий алгоритм независимого табличного Q-обучения, представлен в подразд. 1.6.1. Код, тестирующий выученную Q-таблицу, вынесен в отдельный файл и представлен в подразд. 1.6.2.