

Обучение с подкреплением

Оценочная обратная связь

Динамическое
программирование

Методы Монте-Карло

ТД-методы, SARSA,
Q-обучение

Обобщение и аппроксимация
функций

Планирование и обучение



Обучение с подкреплением

Richard S. Sutton and Andrew G. Barto

Reinforcement Learning

An Introduction

A Bradford Book
The MIT Press
Cambridge, Massachusetts
London, England

АДАПТИВНЫЕ И ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

Р. С. Саттон, Э. Г. Барто

Обучение с подкреплением

Перевод с английского
Е. О. Романова

под редакцией Ю. В. Тюменцева

2-е издание (электронное)



Москва
БИНОМ. Лаборатория знаний
2014

УДК 517.11+519.92
ББК 22.18
С21

Серия основана в 2005 г.

Саттон Р. С.

С21 Обучение с подкреплением [Электронный ресурс] /
Р. С. Саттон, Э. Г. Барто ; пер. с англ. — 2-е изд. (эл.). —
Электрон. текстовые дан. (1 файл pdf : 402 с.). — М. : БИНОМ.
Лаборатория знаний, 2014. — (Адаптивные и интеллекту-
альные системы). — Систем. требования: Adobe Reader XI ;
экран 10".

ISBN 978-5-9963-2500-9

Обучение с подкреплением является одной из наиболее активно развивающихся областей, связанных с созданием искусственных интеллектуальных систем. Оно основано на том, что агент пытается максимизировать получаемый выигрыш, действуя в сложной среде с высоким уровнем неопределенности. Даётся исчерпывающее и ясное изложение идей, методов и алгоритмов обучения с подкреплением, при этом диапазон излагаемого материала — от истоков возникновения рассматриваемых концепций до современных результатов в данной области.

Для специалистов в области искусственного интеллекта, нейросетевого моделирования и управления, а также студентов и аспирантов соответствующих специальностей.

УДК 517.11+519.92
ББК 22.18

Деривативное электронное издание на основе печатного аналога: Обучение с подкреплением / Р. С. Саттон, Э. Г. Барто ;
пер. с англ. — М. : БИНОМ. Лаборатория знаний, 2011. — 399 с. :
ил. — (Адаптивные и интеллектуальные системы).

В соответствии со ст. 1299 и 1301 ГК РФ при устраниении ограничений, установленных техническими средствами защиты авторских прав, правообладатель вправе требовать от нарушителя возмещения убытков или выплаты компенсации

© 1998 Richard S. Sutton and Andrew G. Barto
Fourth printing, 2002
All rights reserved. No part of this book may be
reproduced in any form by any electronic or
mechanical means (including photocopying,
recording, or information storage and retrieval)
without permission in writing from the MIT
Press.

Published by permission of The MIT Press
(USA) via Alexander Korzhenevski Agency
(Russia)

ISBN 978-5-9963-2500-9

© БИНОМ. Лаборатория знаний, 2011

Предисловие редактора серии «Adaptive computation and machine learning»

Меня радует, что книга Ричарда Саттона и Эндрю Барто — одна из первых в новой серии «Adaptive computation and machine learning». Данное руководство представляет собой исчерпывающее введение в захватывающую область обучения с подкреплением. В число основателей этой области входят и авторы предлагаемой книги. Она позволяет студентам, специалистам-практикам и исследователям получить наглядное представление о важнейших концепциях обучения с подкреплением, а также познакомиться с математическими основами этой области знаний. В книге говорится также о новейших примерах практических применений обучения с подкреплением, о соотношении обучения с подкреплением с ключевыми проблемами искусственного интеллекта. Обучение с подкреплением обещает стать чрезвычайно важной новой технологией с огромным потенциалом практических применений, а также важным инструментом для проникновения в суть организации интеллектуальных систем.

Задача построения систем, которые могли бы приспосабливаться к окружающей среде, а также обучаться на основе получаемого опыта, давно привлекает внимание исследователей из самых различных областей, включая информатику, технические науки, математику, физику, нейробиологию и когнитологию. Проведенные ими исследования привели к созданию широкого спектра методов обучения, которые могут оказать существенное влияние на многие области науки и техники. В последние годы различные научные сообщества начинают приходить к общему пониманию проблем, связанных с обучением с учителем, обучением без учителя и с обучением с подкреплением. Серия «Adaptive computation and machine learning»¹⁾, выпускаемая издательством The MIT Press, нацелена на унификацию разнообразных направлений в исследованиях по обучению машин, а также на поощрение высококачественных исследовательских работ и инновационных приложений.

Томас Диттерих

¹⁾ Русское издание книги выходит в серии «Адаптивные и интеллектуальные системы». — Прим. ред.

Предисловие

К тому, что сейчас принято называть обучением с подкреплением, мы впервые пришли в 1979 г. Мы оба работали тогда в Массачусетском университете над одним из наиболее ранних проектов, связанных с возвращением к сетям нейроноподобных адаптивных элементов как многообещающему подходу к решению задачи адаптивного искусственного интеллекта. В этом проекте изучалась «гетеростатическая теория адаптивных систем», созданная А. Харри Клопфом. Работа Харри была богатым источником идей, и перед нами стояла задача критически изучить их, а также сопоставить с тем, что было наработано за долгую предшествующую историю исследований в области адаптивных систем. Нам надо было выявить составные элементы этих идей, понять соотношение между ними и их относительную важность. Эта работа продолжается и сейчас, но в 1979 г. мы поняли, что, несмотря на свою простоту, одна из идей, на которых основывается рассматриваемый подход, привлекала удивительно мало внимания с точки зрения вычислительной перспективы. Это была просто-напросто идея обучающейся системы, которая *хочет* чего-то, которая подстраивает свое поведение, чтобы максимизировать значение некоторого особого сигнала из окружающей среды. Это была идея «гедонистической»¹⁾ обучающейся системы, или, как сказали бы мы сейчас, идея обучения с подкреплением.

Как и другим, нам казалось, что обучение с подкреплением было уже достаточно подробно исследовано еще в ранний период развития кибернетики и искусственного интеллекта. Однако при более глубоком изучении состояния дел в данной области выяснилось, что на самом деле эти вопросы рассматривались весьма поверхностно. Хотя обучение с подкреплением явно служило отправной точкой ранних исследований в области обучения машин, большинство исследователей, которые начинали работать в этой области, переключились на такие вопросы, как

¹⁾ Гедонизм — направление в этике, в котором наслаждение, удовольствие считаются высшей целью и основным мотивом человеческого поведения. Добро при этом определяется как то, что приносит наслаждение, а зло — как то, что влечет за собой страдание. — Прим. ред.

классификация образов, обучение с учителем, адаптивное управление, либо просто перестали заниматься исследованиями в области обучения. В результате аспекты обучения, связанные с воздействием среды, пользовались относительно малым вниманием исследователей. Оглядываясь назад, можно утверждать, что эти аспекты были особенно важны, именно работа над ними обеспечила развитие данной научной области. Прогресс в ее компьютерном изучении был невелик до тех пор, пока не удалось осознать, что такая фундаментальная идея все еще слабо изучена.

С тех пор рассматриваемая область исследований прошла долгий путь, обретая зрелость и развиваясь по нескольким направлениям. Обучение с подкреплением постепенно стало одним из наиболее активно исследуемых направлений в области обучения машин, искусственного интеллекта, нейросетевых исследований. Создана прочная математическая база обучения с подкреплением, решен ряд интересных прикладных задач. Сейчас компьютерные исследования в области обучения с подкреплением — это обширная область, в которой активно работают сотни исследователей во всем мире, основные интересы которых лежат в таких разнообразных областях, как психология, теория управления, искусственный интеллект, нейробиология. В частности, большое внимание уделяется выявлению и развитию взаимосвязей обучения с подкреплением с теорией оптимального управления и динамическим программированием. В целом проблема обучения на основе взаимодействия системы со средой для достижения некоторых целей все еще весьма далека от окончательного решения, однако наше понимание данной проблемы значительно улучшилось. Уже понятна роль таких элементов, как обучение на основе временных различий, динамическое программирование, аппроксимация функций, прояснилась и перспектива решения проблемы в целом.

Наша цель при написании данной книги состояла в том, чтобы дать ясное и простое изложение важнейших идей и алгоритмов обучения с подкреплением. Нам хотелось, чтобы изложенный материал был доступен читателям из всех упоминавшихся выше научных областей, однако задачу показать во всех деталях значение для них обучения с подкреплением мы не ставили. Материалдается почти исключительно с позиций искусственного интеллекта и технических наук. Подробное изложение взаимосвязей с психологией, нейробиологией и прочими областями отложим до другого времени или оставим другим авторам. Мы не стали также давать скрупулезное формальное описание обучения с подкреплением, поскольку не стремились излагать материал на максимально возможном уровне математической абстракции в стиле

«теорема-доказательство». Мы выбрали такой уровень изложения, который позволяет передать основные математические идеи данной области, не затеняя при этом простоты и потенциальной общности ее важнейших положений.

Книга состоит из трех частей. Часть I — вводная и проблемно-ориентированная. Здесь мы акцентируем внимание читателя на простейших аспектах обучения с подкреплением, а также на основных чертах, отличающих его от родственных направлений. Целая глава в этой части посвящена описанию конкретной задачи обучения с подкреплением, решение которой изучается затем на протяжении всей оставшейся части книги. В части II представлены три важнейших базисных метода поиска решений: динамическое программирование, простые методы Монте-Карло, обучение на основе временных различий. Первый из перечисленных методов представляет собой одну из разновидностей методов планирования и предполагает, что имеется явным образом сформулированное знание обо всех аспектах решаемой проблемы, тогда как оставшиеся два метода — это методы обучения. Часть III посвящена обобщению этих методов, а также построению ряда комбинированных подходов на их основе. Используя следы приемлемости, можно унифицировать методы Монте-Карло и обучения на основе временных различий, а методы аппроксимации функций, в частности на основе искусственных нейронных сетей, развиваются все эти методы таким образом, что они становятся применимыми к задачам существенно более высокой размерности. Мы опять возвращаемся к идее объединения методов, основанных на планировании и на обучении, а также соотносим их с эвристическим поиском. Наконец, мы обсуждаем состояние исследований в области обучения с подкреплением, а также рассматриваем ряд конкретных случаев, включая несколько наиболее интересных и впечатляющих примеров применения обучения с подкреплением, существующих на сегодняшний день.

Наша книга задумывалась как основа для односеместрового курса, ее можно дополнить другой литературой, например более математизированным руководством Бертsekаса и Цициклиса (Bertsekas, Tsitsiklis 1996). Данная книга может быть использована также в качестве части курса более широкой направленности, например, машинного обучения, искусственного интеллекта или нейронных сетей. В таком случае может оказаться желательным ограничиться лишь частью материала, содержащегося в книге. Мы рекомендовали бы тогда взять гл. 1 в качестве краткого обзора тематики обучения с подкреплением, гл. 2 по разд. 2.2, гл. 3 без разд. 3.4, 3.5 и 3.9, а затем взять какое-то количество материа-

ла из оставшихся глав соответственно интересам и имеющемуся в распоряжении времени. Главы 4, 5 и 6 взаимосвязаны, и их лучше давать именно в такой последовательности, из них гл. 6 наиболее важна как для всей области обучения с подкреплением, так и для остального материала книги. Курс, основным содержанием которого является обучение машин или нейронные сети, должен включать гл. 8, а курс, связанный с тематикой искусственного интеллекта или планированием, — гл. 9. Материал гл. 10 надо включать практически во всех случаях, поскольку глава краткая и содержит обобщающий материал, дающий единую точку зрения на излагаемые методы обучения с подкреплением. В течение всей книги разделы, более трудные для восприятия и не очень существенные для понимания оставшейся части книги, помечаются символом «★». Они могут быть опущены при первом чтении, что не создаст особых затруднений при чтении последующих разделов. Некоторые упражнения также помечены символом «★», чтобы показать, что они относятся к повышенному типу сложности и не существенны для понимания основного материала соответствующей главы.

Книга большей частью самодостаточна. Не предполагается, что читатель обладает какой-либо специальной математической подготовкой, он должен быть знаком лишь с элементарными сведениями из теории вероятностей, в частности с понятием математического ожидания случайной величины. Материал гл. 8 будет значительно проще понять, если читатель обладает некоторыми знаниями в области искусственных нейронных сетей или знаком с какими-либо другими подходами к организации обучения с учителем. Этот материал, однако, можно читать и без такой предварительной подготовки. Мы настоятельно рекомендуем прорабатывать упражнения, содержащиеся в книге. Преподаватели могут воспользоваться соответствующими руководствами по решению этих задач. Эти руководства и другие регулярно обновляемые материалы по теме доступны через Интернет.

В конце большинства глав содержится раздел, озаглавленный «Библиографические и исторические замечания». В этих разделах указываются источники идей, излагаемых в соответствующей главе, даются ориентиры для дальнейшего изучения предмета и исследований в нем, а также описывается история вопроса. Несмотря на наше стремление сделать эти разделы возможно более корректными и полными, вне всякого сомнения, какие-то из важных работ не попали в поле нашего зрения. Мыносим свои извинения по этому поводу и будем всячески приветствовать поправки и дополнения, которые могли бы быть включены в последующие издания.

В каком-то смысле мы работаем над данной книгой вот уже двадцать лет, и у нас есть причины высказать свою благодарность очень многим людям. Первым делом, мы благодарим тех, кто очень помог нам выработать общую точку зрения, представленную в этой книге, — это Харри Клопф (Harry Klopff), который помог нам осознать необходимость возобновления исследований в области обучения с подкреплением; Крис Уоткинс (Chris Watkins), Дмитрий Бертsekas (Dimitri Bertsekas), Джон Цициклис (John Tsitsiklis) и Пол Вербос (Paul Werbos), которые помогли нам понять значение связей обучения с подкреплением и динамического программирования; мы благодарны Джону Муру (John Moore) и Джиму Кехоэ (Jim Kehoe) за озарения и вдохновение, связанные с теорией обучения животных; Оливеру Селфриджу (Oliver Selfridge) за подчеркивание широты и важности такой области, как адаптация. В более широком плане мы признательны нашим коллегам и студентам, которые воздействовали на нас очень многими способами — это Ron Williams, Charles Anderson, Satinder Singh, Sridhar Mahadevan, Steve Bradtke, Bob Crites, Peter Dayan и Leemon Baird. Наши идеи в области обучения с подкреплением существенно обогатились в результате дискуссий с такими коллегами, как Paul Cohen, Paul Utgoff, Martha Steenstrup, Gerry Tesauro, Mike Jordan, Leslie Kaelbling, Andrew Moore, Chris Atkeson, Tom Mitchell, Nils Nilsson, Stuart Russell, Tom Dietterich, Tom Dean и Bob Narendra. Мы признательны Michael Littman, Gerry Tesauro, Bob Crites, Satinder Singh и Wei Zhang за помощь в работе со специфическим материалом разд. 4.7, 11.1, 11.4, 11.5 и 11.6 соответственно. Мы признательны также Управлению по научным исследованиям BBC США, Национальному научному фонду и фирме GTE Laboratories за их долгую и дальновидную поддержку.

Мы хотели бы также поблагодарить многочисленных коллег, которые прочитали черновой вариант данной книги и высказали ценные замечания. Среди этих коллег Tom Kalt, John Tsitsiklis, Paweł Cichosz, Olle Gällmo, Chuck Anderson, Stuart Russell, Ben Van Roy, Paul Steenstrup, Paul Cohen, Sridhar Mahadevan, Jette Randlov, Brian Sheppard, Thomas O'Connell, Richard Coggins, Cristina Versino, John H. Hiett, Andreas Badelt, Jay Ponte, Joe Beck, Justus Piater, Martha Steenstrup, Satinder Singh, Tommi Jaakkola, Dimitri Bertsekas, Ben Van Roy, Sascha Engelbrecht, Torbjörn Ekman, Christina Björkman, Jakob Carlström и Olle Palmgren. В заключение мы признательны Gwyn Mitchell за многообразную помощь, а Harry Stanton и Bob Prior — за отстаивание наших интересов в издательстве MIT Press.

I

Постановка задачи и подходы к ее решению

1 Введение

Когда мы размышляем о самой природе обучения, первой, вероятно, возникает идея о том, что мы обучаемся через взаимодействие с окружающей средой. Когда играет младенец, размахивая руками или глядя по сторонам, явно выраженного учителя у него нет, но зато у него есть прямой сенсомоторный контакт со средой. Постоянно повторяясь, контакты такого рода в изобилии дают информацию относительно причин и следствий, относительно последовательностей действий, а также о том, что надо делать, чтобы добиться определенных целей. В течение всей нашей жизни такие взаимодействия представляют собой, несомненно, важный источник знаний об окружающем мире и о нас самих. Учимся мы водить автомобиль или поддерживать разговор, мы четко сознаем, каким образом среда реагирует на наши действия, и стараемся добиться нужных нам результатов, организуя соответствующим образом свое поведение. Обучение через взаимодействие — основополагающая идея, на которой базируются почти все теории обучения и интеллекта.

В этой книге мы принимаем *вычислительный* подход к обучению на основе взаимодействия. Вместо того чтобы напрямую строить теории того, как обучаются люди и животные, мы исследуем идеализированную ситуацию обучения и оцениваем эффективность различных подходов к обучению. Иначе говоря, мы рассматриваем данную проблему с позиций исследователя или специалиста в области искусственного интеллекта. Мы изучаем проекты машин, которые были бы эффективны для решения задач обучения, интересных с научной и практической точек зрения, оценивая эти проекты с привлечением математических методов и компьютерных экспериментов. Изучаемый подход именуется *обучением с подкреплением*. Он, в отличие от других подходов к обучению машин, значительно более ориентирован на обучение, направляемое целями и основанное на взаимодействии со средой.

1.1. Обучение с подкреплением

Обучение с подкреплением — это обучение тому, что надо делать, как следует отображать ситуации в действия, чтобы максимизировать некоторый сигнал поощрения (вознаграждения), принимающий числовые

значения. Обучаемому не говорят, какое действие следует предпринять, как это имеет место в большинстве подходов к обучению машин. Вместо этого он, пробуя выполнять различные действия, должен найти, какие из них принесут ему наибольшее вознаграждение. В наиболее интересных и важных случаях действия могут влиять не только на вознаграждение, получаемое немедленно, но также и на возникающую ситуацию, а через нее — на все последующие поощрения. Эти две характеристики — поиск методом проб и ошибок, а также отсроченные поощрения — представляют собой две наиболее важные отличительные черты обучения с подкреплением.

Определение обучения с подкреплением дается не через описание методов обучения, а путем выявления характерных свойств *задачи обучения*. Любой метод, который хорошо подходит для решения сформулированной задачи, будем рассматривать как метод обучения с подкреплением. Подробное описание задачи обучения с подкреплением в терминах оптимального управления марковскими процессами принятия решений отложим до гл. 3. Но уже сейчас можно сказать, что основная идея состоит просто в том, чтобы уловить и зафиксировать наиболее важные аспекты реальной задачи, имея в виду организацию взаимодействия агента со средой для достижения некоторой цели. Совершенно ясно, что такого рода агент должен иметь возможность в какой-то мере воспринимать состояние среды, а также быть в состоянии предпринимать действия, которые могут повлиять на состояние среды. Агент должен иметь также цель или цели, связанные с состояниями среды. Формулировка задачи обучения с подкреплением должна учитывать все три аспекта (восприятие, действие, цели) в их наиболее простых формах, но без их вульгаризации.

Обучение с подкреплением отличается от *обучения с учителем*, которое рассматривается в большинстве современных исследований по обучению машин, статистическому распознаванию образов, искусственным нейронным сетям. Обучение с учителем — это обучение по примерам, предъявляемым некоторой информированной внешней инстанцией. Это важный вид обучения, однако без привлечения дополнительных средств он не пригоден для обучения через взаимодействие. В задачах, решаемых на основе взаимодействия, зачастую непрактично пытаться получать примеры требуемого поведения, которые были бы одновременно корректными и представительными для всех ситуаций, в которых должен действовать агент. На «ничейных» территориях, в ситуациях, когда обучение более всего и нужно, агент должен быть в состоянии учиться только на основе своего собственного опыта.

Одна из наиболее серьезных проблем, возникающих в обучении с подкреплением и отсутствующих в других видах обучения, — это проблема поиска компромисса между *изучением и применением*¹⁾. Чтобы получить большее вознаграждение, агент, обучающийся с подкреплением, должен предпочитать действия, которые он уже проверил в прошлой своей деятельности и обнаружил, что они эффективны с точки зрения получения поощрения. Однако, чтобы обнаруживать их, надо пробовать выполнять такие действия, которые еще не выполнялись ранее. Агент должен *применять* те действия, про которые уже известно, что они позволяют получить вознаграждение, но он должен также и *изучать* новые действия, чтобы иметь возможность делать лучший выбор в будущем. Проблема состоит в том, что нельзя только использовать уже проверенные действия или только искать новые эффективные действия, поскольку это ведет к провалу попыток решения задачи. Агент должен пытаться предпринимать разнообразные действия и благоприятствовать тем из них, которые окажутся лучшими. В задачах стохастического характера каждое из действий должно быть повторено многократно, чтобы добиться получения надежной оценки ожидаемого вознаграждения. Диллемма изучения–применения интенсивно исследуется математиками вот уже в течение нескольких десятилетий (см. гл. 2). Пока же просто отметим, что в области обучения с учителем, в том виде, как эту область принято обычно определять, проблема равновесия между изучением и применением в полном объеме даже и не возникает.

Еще одна характерная черта обучения с подкреплением состоит в том, что в явном виде рассматривается *целостная* проблема целенаправленного агента, взаимодействующего с неопределенной средой²⁾. Это существенно отличается от многих других подходов, где рассмотрение проводится на уровне подзадач без всяких указаний на то, как эти подзадачи можно увязать в общую картину. Например, уже упоминалось, что в большинстве исследований в области обучения машин внимание фокусируется на обучении с учителем, причем явно никак не указывается, как такого рода возможность можно использовать на практике для получения каких-либо полезных результатов. В других исследованиях разрабатываются теории планирования, оперирующего с общими целями, но без рассмотрения роли планирования в процессах принятия решения, идущих в реальном масштабе времени, или же во-

¹⁾ То есть между поведением, направленным на получение знания, и поведением, основанным на использовании уже имеющегося знания. — Прим. ред.

²⁾ То есть со средой, которая содержит неопределенность, неопределенные факты. — Прим. ред.

проса о том, откуда взять предсказывающие модели, которые необходимы для планирования. Такой подход позволил получить много важных результатов, однако концентрация внимания в нем на изолированных подзадачах существенно ограничивает возможности его применения.

Обучение с подкреплением ставит прямо противоположную задачу и исходит из целостного, взаимодействующего со средой, целенаправленного агента. Все агенты, обучающиеся с подкреплением, имеют явным образом выраженные цели, могут воспринимать особенности среды, а также выбирать действия, влияющие на эту среду. Более того, обычно с самого начала предполагается, что агент должен действовать, несмотря на существенную неопределенность в среде. Когда в обучение с подкреплением вовлекается планирование, приходится предпринимать усилия для обеспечения взаимодействия между планированием и выбором действий в реальном масштабе времени, а также для поиска способов получения и улучшения моделей среды. Рассмотрение специфичных для данной задачи соображений относительно того, какие характеристики критичны, а какие нет, может привести к включению в обучение с подкреплением средств обучения с учителем. Чтобы добиться успеха в области обучения, надо выделить и изучить важные подзадачи общей задачи обучения, но это должны быть такие подзадачи, роль которых ясна с точки зрения целостного, взаимодействующего со средой, целенаправленного агента, даже если нет возможности во всех подробностях описать этого агента.

Исследования в области обучения с подкреплением можно считать частью общего процесса, развивающегося в последние годы. Он состоит во все расширяющихся контактах и взаимодействиях между искусственным интеллектом и другими инженерными дисциплинами. Еще совсем недавно искусственный интеллект рассматривался как область исследований, никак не связанная с такими областями, как теория управления и статистика. Искусственный интеллект имел дело с логикой и символами, не с числами. Его моделями были большие программы на языке LISP, а не дифференциальные уравнения, соотношения линейной алгебры и статистики. В последние десятилетия эта точка зрения постепенно размывалась. Современные исследователи в области искусственного интеллекта допускают использование моделей статистики и алгоритмов управления, например, в качестве вполне подходящих конкурирующих методов или же просто как часть арсенала методов анализа. Области, которым ранее не придавали значения, находящиеся на стыке искусственного интеллекта и традиционных инженерных методов, сейчас развиваются наиболее активно. В их число входят такие новые на-

правления, как нейронные сети, интеллектуальное управление, а также предмет нашего рассмотрения — обучение с подкреплением. В обучении с подкреплением развиваются идеи, почерпнутые в теории оптимального управления и в стохастической аппроксимации, следя более общим и более амбициозным целям искусственного интеллекта.

1.2. Примеры

Хороший способ понять суть обучения с подкреплением — рассмотреть для него несколько примеров и возможные варианты применений, которые подсказывают, в каком направлении должна идти разработка данного метода.

- Опытный шахматист делает ход. Выбор этого хода обусловлен как планированием — ожиданием ответной реакции противника и реакции на эту реакцию, так и текущими интуитивными соображениями относительно желательности конкретных позиций и ходов.
- Адаптивный регулятор подстраивает параметры процесса перегонки нефти в реальном масштабе времени. Данный регулятор оптимизирует соотношение между выходом продукции, ее стоимостью и качеством на основе заданных граничных значений стоимости без строгой привязки к соображениям, которые закладывались изначально инженерами.
- Детеныш газели старается встать на ноги сразу после рождения. Попутно позже он уже бежит со скоростью более 30 км/ч.
- Мобильный робот решает, должен ли он войти в очередную комнату при сборе мусора или же ему уже пора начинать искать дорогу назад, к месту, где он сможет зарядить свои аккумуляторы. Он принимает соответствующее решение на основе данных о том, насколько быстро и просто удавалось найти зарядную станцию в прошлом.
- Фил готовит себе завтрак. При более подробном рассмотрении оказывается, что даже такая сугубо утилитарная деятельность представляет собой сложный комплекс действий, подчиненных определенным условиям, взаимосвязанных целей и подцелей: подойти к шкафу, открыть его, выбрать коробку с крупой, протянуть к ней руку, взять коробку, вытащить ее. Другие сложные последовательности поведенческих актов, корректируемые по ходу действия, требуются для того, чтобы взять чашку, ложку, кувшин с молоком. Каждый из перечисленных шагов требует выполнения последовательности движений глаз для получения информации, а также для

управления процессами перемещения руки к требуемому предмету и перемещения этого предмета. Постоянно принимаются решения относительно того, что следует делать с тем или иным объектом, стоит ли его перенести на обеденный стол, прежде чем браться за остальные. Выполнение каждого шага направляется целями, такими как «взять ложку» или «подойти к холодильнику». Эти цели могут быть подчинены другим целям, например «обеспечить наличие ложки для еды», когда завтрак готов к употреблению.

Эти примеры имеют ряд общих фундаментальных черт, которые нельзя упускать из виду. Все они включают *взаимодействие* между активным агентом, принимающим решения, и окружающей его средой, в которой агент осуществляет поиск для достижения своей цели, несмотря на наличие *неопределенности* в среде. Действия агента направлены на то, чтобы повлиять на будущее состояние среды (им может быть, например, следующая шахматная позиция, уровень жидкости в резервуаре при перегонке нефти, следующее местоположение робота) и через него повлиять на те возможности, которые окажутся в распоряжении агента в следующие моменты времени. Для правильного выбора необходимо принимать во внимание непрямые, отложенные последовательности действий, что может потребовать использования прогнозирования или планирования.

Вместе с тем во всех приведенных примерах нет возможности полностью предсказать влияние предпринимаемых действий. По этой причине агент должен достаточно часто контролировать состояние среды и ответственным образом реагировать на изменения. Например, Фил должен контролировать количество молока, которое он наливает в чашку, чтобы молоко не перелилось через край. Все эти примеры включают явные цели в том смысле, что агент может оценивать характер продвижения к требуемой цели, основываясь на том, что он может воспринимать непосредственно. Шахматист знает, выигрывает он или нет, регулятор нефтеперегонной установки имеет данные о том, сколько нефти должно быть обработано, мобильный робот обнаруживает, что его батарея заряжена недостаточно, а Фил знает, нравится ему завтрак или нет.

Во всех этих примерах агент может использовать накопленный опыт, чтобы со временем улучшать свои характеристики. Шахматист совершенствует свою интуицию, помогающую в оценке позиций, что позволяет ему более успешно играть; детеныш газели развивает свою способность бегать; Фил учится более рационально готовить завтрак. Знание, которое привносится агентом в задачу в начале процесса ее ре-

шения (из прошлого опыта работы с аналогичными задачами, встроенное при проектировании агента или приобретенное в процессе эволюции агента), оказывает влияние на то, насколько полезным или простым будет процесс обучения, но решающим фактором, обеспечивающим корректировку поведения агента с учетом специфических особенностей решаемой задачи, является взаимодействие со средой.

1.3. Элементы обучения с подкреплением

Помимо агента и среды, можно указать следующие четыре основных элемента, входящие в состав систем, реализующих обучение с подкреплением: *стратегия*, *функция поощрения*, *функция ценности* и, возможно, *модель среды*.

Стратегия определяет характер поведения обучающегося агента в каждый данный момент времени. Упрощенно говоря, стратегия представляет собой отображение воспринимаемых состояний среды в действия, отвечающие этим состояниям. Данный подход соответствует тому, что в психологии принято называть набором правил «стимул–реакция» или набором ассоциаций. В некоторых случаях стратегия может представлять собой просто функцию или же просмотровую таблицу, тогда как в других случаях приходится выполнять большие объемы вычислений, например в виде некоторого процесса поиска. Стратегия — это ключевой элемент агента, реализующего обучение с подкреплением, в том смысле, что ее одной достаточно для того, чтобы определить поведение агента. В общем случае стратегии могут быть случайными.

Функция поощрения определяет цель в задаче обучения с подкреплением. Упрощенно говоря, она ставит в соответствие каждому воспринятыму состоянию среды (или паре «состояние–действие») единственное число, вознаграждение, показывающее степень желательности данного состояния. Единственной целью агента, реализующего обучение с подкреплением, является максимизация суммарного вознаграждения, получаемого агентом в течение достаточно продолжительного промежутка времени. Функция поощрения определяет, насколько хороши или плохи те или иные события для данного агента. Для биологических систем вознаграждение вполне можно отождествить с удовольствием и болью. Оно немедленно и определяющим образом характеризует проблему, с которой сталкивается агент. Раз так, необходимо, чтобы агент не имел возможности воздействовать на функцию поощрения. Однако она может послужить основой для изменения стратегии. Например, если некоторое действие, задаваемое стратегией, сопровождается невысоким воз-

награждением, то стратегия может быть изменена, чтобы в будущем, когда опять встретится такая же ситуация, использовать уже какое-либо другое действие. В общем случае функции поощрения могут быть случайными.

Если функция поощрения показывает, что хорошо прямо сейчас, то *функция ценности* — что хорошо в долгосрочной перспективе. Упрощенно говоря, *ценность* некоторого состояния — это общая сумма вознаграждения, которую агент рассчитывает получить в будущем, если данное состояние будет для него начальным. Если, как отмечалось, вознаграждение определяет текущую желательность некоторого состояния среды, ценность показывает *долгосрочную* желательность данного состояния с учетом наиболее вероятной последовательности состояний, а также вознаграждений, получаемых в этих состояниях. Например, некоторое состояние может всегда приносить невысокое вознаграждение, но тем не менее иметь высокую ценность в силу того, что за ним регулярно следуют другие состояния, обеспечивающие большое вознаграждение. Возможна, конечно, и прямо противоположная ситуация. Используя аналогию с человеком, вознаграждение подобно удовольствию (при высоком его значении) или боли (при малом значении), а ценность отвечает более тонким и дальновидным суждениям о нашей степени удовлетворенности или неудовлетворенности применительно к некоторому конкретному состоянию среды. Можно надеяться, что из этого объяснения становится ясна фундаментальная роль функции ценности, а также то, что она формализует хорошо знакомую уже идею.

Вознаграждение в известном смысле первично, а ценность — вторична, как предсказание вознаграждения. Без вознаграждения не было бы и ценности, а вычисление ценности проводится только для того, чтобы получить более высокое вознаграждение. Тем не менее именно ценности уделяется основное внимание при принятии и оценке решений. Выбор действий осуществляется на основе суждений о ценности. Отыскиваются не действия, приносящие наибольшее вознаграждение, а действия, которые обладают наибольшей ценностью для данного состояния, поскольку они позволяют получить в итоге наиболее высокое суммарное вознаграждение. Поэтому в принятии решений и планировании вторичная величина, именуемая ценностью, привлекает к себе наибольшее внимание. К сожалению, значение ценности найти намного сложнее, чем значение вознаграждения. Вознаграждения определяются в основном непосредственно средой, тогда как значения ценности приходится снова и снова вычислять на основе последовательностей наблюдений, осуществляемых агентом в течение всего его жизненного цикла. Фак-

тически наиболее важным элементом почти всех алгоритмов обучения с подкреплением является метод эффективного вычисления значений функции ценности. Можно сказать, что выявление центральной роли данного метода является наиболее важным результатом последних десятилетий, относящимся к обучению с подкреплением.

Все методы обучения с подкреплением, изучаемые в данной книге, включают в качестве отдельного элемента вычисление значений функции ценности. Это, однако, не является строго необходимым для решения задач обучения с подкреплением. В частности, для решения этих задач могут использоваться поисковые методы, такие как генетические алгоритмы, генетическое программирование, имитационный отжиг и другие методы минимизации функций. Эти методы осуществляют поиск непосредственно в пространстве стратегий, не обращаясь к аппарату функций ценности. Методы, упомянутые выше, принято называть *эволюционными*, поскольку они действуют в определенной степени аналогично биологическим механизмам совершенствования поведения организмов, даже если эти организмы не обучаются в течение их индивидуального периода жизни. Эволюционные стратегии могут быть эффективными, если пространство стратегий невелико или может быть структурировано так, чтобы хорошие стратегии найти было достаточно просто. Кроме того, у эволюционных методов имеется преимущество в тех задачах, где обучающийся агент не может воспринимать состояния среды с достаточной точностью.

Однако обучение с подкреплением в том виде, как его принято понимать, осуществляет обучение через взаимодействие со средой, чего лишены методы эволюционного типа. Представляется, что методы, которые способны обращаться к деталям индивидуальных поведенческих взаимодействий, во многих случаях могут оказаться намного более эффективными, чем эволюционные методы. Эволюционные методы пренебрегают многим полезным из того, что характерно для задачи обучения с подкреплением: они не используют тот факт, что искомая стратегия представляет собой функциональную зависимость между состояниями и действиями; они не отмечают, какие состояния объекта-индивидуума встречаются в течение всей его жизни или же какие действия им выбираются. В некоторых случаях эта информация может оказаться дезориентирующей (например, если состояния воспринимаются ошибочно), однако значительно чаще она позволяет организовать более эффективный поиск. Хотя у эволюции и обучения много общих свойств и они могут, естественно, работать вместе, как это и имеет место в природе, мы не будем рассматривать эволюционные методы сами по себе, а со-

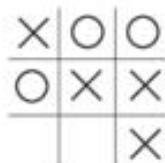
средоточимся только на проблемах обучения с подкреплением. Для простоты в этой книге термин «обучение с подкреплением» не будет предполагать использования эволюционных методов.

Четвертый и последний элемент, присутствующий в ряде систем, реализующих обучение с подкреплением, это *модель* среды. Назначение модели — имитировать поведение рассматриваемой среды. Например, если известны состояние и действие, такая модель может предсказать следующие значения для состояния и вознаграждения. Модели используются для *планирования*, под которым будем понимать некоторый способ решить, какими должны быть последующие действия применительно к возможным в будущем ситуациям, которые пока еще реально не возникали. Включение моделей и средств планирования в системы, реализующие обучение с подкреплением, представляет собой относительно новую область исследований. Ранние варианты систем, реализующих обучение с подкреплением, в явном виде воспроизводили метод проб и ошибок. То, что делали эти системы, выглядит практически как *противоположность* планированию. Однако постепенно стало ясно, что методы обучения с подкреплением тесно связаны с опирающимися на модели методами динамического программирования, а они, в свою очередь, тесно связаны с методами планирования в пространстве состояний. В главе 9 будут изучены системы, реализующие обучение с подкреплением, которые одновременно обучаются методом проб и ошибок, строят с помощью обучения модель среды, а также используют эту модель для планирования. Диапазон подходов современного обучения с подкреплением простирается от низкоуровневого обучения на основе метода проб и ошибок до высокоуровневого совещательного планирования.

1.4. Подробный пример: крестики-нолики

Чтобы проиллюстрировать основную идею обучения с подкреплением и сопоставить данный подход с другими, рассмотрим более подробно один из характерных примеров.

Вспомним известную детскую игру в крестики-нолики. Два игрока совершают ходы на доске размером три на три клетки. Один из игроков играет «крестиками» **X**, а второй — «ноликами» **O**, записывая их в те или иные клетки. Выигрывает тот игрок, который сумеет поместить три своих значка горизонтально, в строку, вертикально, в столбец, или диагонально, как это сделал игрок, пользовавшийся крестиками **X**, на приводимом рисунке:



Если клетки доски заполнены, но ни один из игроков не сумел расположить три символа в линию, победитель определяется жребием. Поскольку опытный игрок может играть так, что никогда проигрывать не будет, предположим, что мы играем против «несовершенного» соперника, который иногда делает неверные ходы и за счет этого у нас появляются шансы на выигрыш. Будем считать пока, что жребий и проигрыш для нас одинаково неблагоприятны. Как должен быть устроен игрок, чтобы он находил промахи в игре соперника и обучался максимизировать свои шансы на выигрыш?

Хотя рассматриваемая задача и не относится к числу сложных, ее нельзя удовлетворительно решить с помощью классических методов. Например, классическое минимаксное решение из теории игр здесь не подойдет, поскольку оно основано на предположении о том, что поведение соперника подчиняется определенным правилам специального вида. Например, «минимаксный» игрок никогда не достигнет такого состояния в игре, из которого он мог бы проиграть, даже если он всегда выигрывал из данного состояния только из-за неверных действий соперника. Классические оптимизационные методы для задач последовательного принятия решений, такие как динамическое программирование, могут вычислить оптимальное решение для любого из соперников, однако им потребуется его полное описание в качестве входной информации, включая вероятности, с которыми этот игрок совершает ходы для каждого из текущих состояний игры. Будем считать, что такого рода информация для данной проблемы априори недоступна, — именно так и обстоит дело для многих задач, представляющих практический интерес. С другой стороны, такая информация может быть получена из опыта, в рассматриваемом случае — путем розыгрыша большого числа партий против данного соперника. Одним из лучших вариантов в этом случае было бы вначале получить с помощью обучения модель поведения соперника, обеспечивающую заданный уровень доверия к ней, а затем воспользоваться динамическим программированием для нахождения оптимального решения, основанного на этой модели. Такой подход, в общем-то, не очень отличается от тех методов обучения с подкреплением, которые изучаются далее в данной книге.

В эволюционном подходе к решению этой проблемы производился бы прямой поиск в пространстве возможных стратегий для того, чтобы отыскать стратегию с наиболее высокой вероятностью выигрыша. В данном случае стратегия представляет собой некоторое правило выбора следующего хода для каждого состояния игры, т. е. для каждой возможной комбинации символов **X** и **O** на доске (игровом поле) размером три на три клетки. Для каждой из рассматриваемых стратегий оценку вероятности выигрыша можно было бы получить, разыгрывая некоторое количество партий с данным соперником. Эта оценка могла бы затем служить ориентиром при решении вопроса о том, какую из стратегий надо рассматривать следующей. Типичный эволюционный метод мог бы состоять из поиска максимума в пространстве стратегий, последовательном формировании и оценке стратегий с тем, чтобы обеспечить их постепенное улучшение. Можно было бы также использовать алгоритм типа генетического, который работает с популяцией стратегий. Для решения рассматриваемой задачи можно было бы применить буквально сотни различных методов оптимизации. При этом под *прямым* поиском в пространстве стратегий подразумевается то, что на основе скалярных оценок предлагаются и сравниваются *стратегии в целом*.

Рассмотрим теперь, каким образом можно было бы подойти к решению задачи игры в крестики-нолики с помощью обучения с подкреплением и аппроксимации функций ценности. Для начала создадим таблицу чисел, по одному для каждого из возможных состояний игры. Каждое из этих чисел будет завершающей оценкой вероятности выигрыша, если данное состояние взять в качестве начального. Будем трактовать эту оценку как *ценность* состояния, а таблицу в целом — как функцию ценности, получаемую путем обучения. Состояние А имеет более высокую ценность, чем состояние В, т. е. оно «лучше», чем состояние В, если текущая оценка вероятности выигрыша из состояния А больше, чем из состояния В. Пусть мы всегда играем «крестиками», тогда для всех состояний с тремя символами **X**, расположенными подряд, вероятность выигрыша равна 1, поскольку мы уже выиграли. Аналогично, для всех состояний с тремя символами **O** на одной линии соответствующее значение вероятности будет равно 0, поскольку возможность выиграть из такого состояния для нас отсутствует. Для всех прочих состояний зададим начальные значения, равные 0,5, отвечающие предположению о том, что наши шансы на выигрыш составляют 50%.

Сыграем достаточно много партий против данного соперника. Для выбора ходов будем анализировать состояния игры, которые будут след-

ствием выполнения каждого из возможных ходов (по одному на каждую из незанятых клеток доски), после чего по просмотровой таблице найдем текущие значения ценности для этих состояний. Большую часть времени при этом осуществляется *перебор ходов*, направленный на то, чтобы перейти к следующему состоянию, обладающему наибольшей ценностью, т. е. наибольшим значением оценки вероятности выигрыша. Время от времени, однако, вместо этого выполняется просто случайный выбор одного из возможных ходов. Такие ходы именуются *зондирующими*, поскольку они приводят к просмотру состояний, которые в другом случае никогда бы не встретились. Последовательность ходов, выполненных и просмотренных в ходе игры, можно изобразить в виде диаграммы наподобие показанной на рис. 1.1.

В ходе игры происходит изменение ценности ее состояний. Попытаемся дать для них более точные оценки вероятностей выигрыша. Для этого будем «восстанавливать» после каждого «жадного» хода предыдущее состояние, как это показано стрелками на рис. 1.1. Выражаясь более точно, текущая ценность заданного предыдущего состояния подстраивается таким образом, чтобы она была как можно ближе к ценности более позднего состояния. Это можно сделать изменением ценности требуемого предыдущего состояния на величину, которая составляет некоторую часть от разности между ценностью двух рассматриваемых состояний. Если обозначить через s состояние игры перед жадным ходом, а через s' — состояние после него, то значение $V(s)$ ценности состояния s корректируется согласно правилу вида

$$V(s) \leftarrow V(s) + \alpha [V(s') - V(s)],$$

где α — небольшое положительное число, называемое *размером шага*, влияющее на скорость обучения. Это правило корректировки представляет собой один из примеров метода обучения, основанного на *временных различиях* (temporal-difference learning — TDL). Своё название этот метод получил из-за того, что выдаваемые им изменения значений ценности основаны на разностях $V(s') - V(s)$ между их оценками для двух различных моментов времени.

Метод, описанный выше, прекрасно подходит для решения данной задачи. Например, если размер шага с течением времени постепенно уменьшать, данный метод для любого заданного соперника приводит к истинным значениям вероятностей выигрыша для каждого из состояний в оптимальной игре. Более того, исполняемые в данном случае ходы (за исключением зондирующих ходов) являются, по сути дела, оптимальными против данного соперника. Другими словами, рассмат-

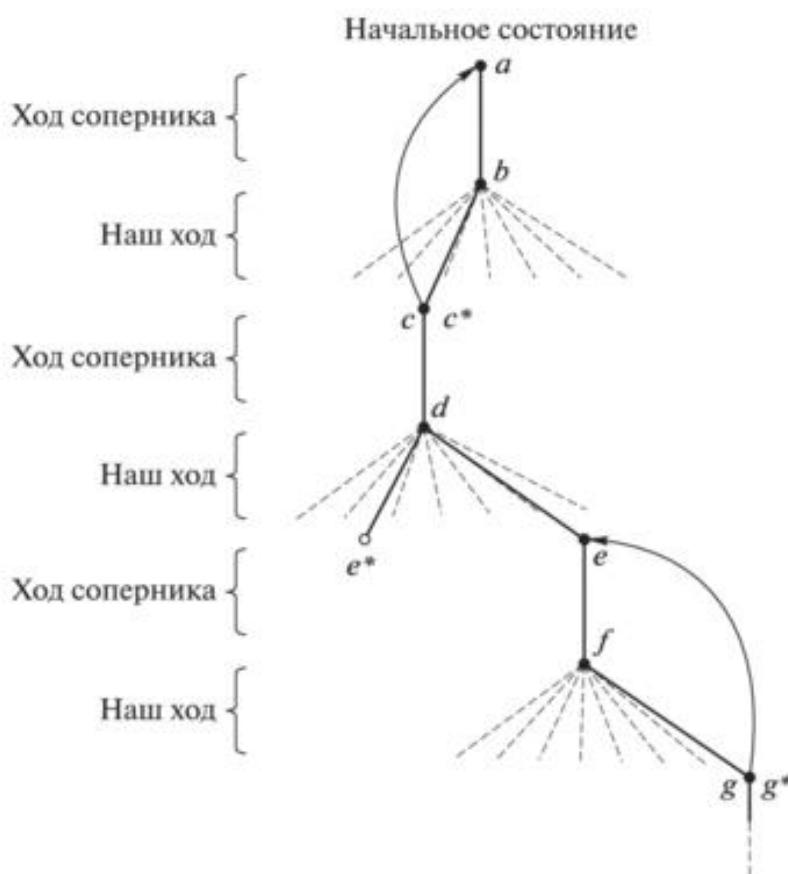


Рис. 1.1. Пример записи последовательности ходов в игре «крестики-нолики». Сплошные линии показывают ходы, выполненные в процессе игры; пунктирные линии соответствуют ходам, которые игрок (обучающийся с подкреплением) просматривал, но не сделал. Второй ход был зондирующим. Это означает, что он был сделан игроком несмотря на то, что есть другой возможный ход, который переводит игру в состояние e^* с более высоким значением ценности, чем у полученного состояния e . Зондирующие ходы не отражаются непосредственно на результате обучения, но каждый из них расширяет состав *возможных вариантов действий*, как это показано изогнутыми стрелками и разъясняется в тексте

риаемый метод сходится к оптимальной стратегии поведения в данной игре. Если размер шага не уменьшать до нуля по ходу игры, то данный игрок может также хорошо играть и с соперником, стиль игры которого медленно изменяется с течением времени.

Этот пример демонстрирует разницу между эволюционными методами и методами обучения на основе функций ценности. Чтобы оценить стратегию, эволюционный метод должен зафиксировать ее и применить много раз против заданного соперника либо многократно имитировать игру, используя какую-либо модель этого соперника. Частота выигры-

шой дает несмешенную оценку вероятности выигрыша при использовании данной стратегии и может быть использована при выборе следующей стратегии. Однако любое изменение стратегии осуществляется только после большого числа разыгранных партий, причем по каждой игре берется лишь конечный результат, а то, что происходит *в течение* игры, во внимание не принимается. Например, если игрок добился победы, то все его поведение *в целом* оценивается положительно, независимо от того, насколько критичными для достижения победы были те или иные конкретные ходы. Такая положительная оценкадается даже ходам, никогда не совершившимся! В противоположность этому, методы функции ценности позволяют оценивать индивидуальные состояния игры. В конечном счете, как эволюционные методы, так и методы функций ценности осуществляют поиск в пространстве стратегий, однако обучение на основе функции ценности позволяет с выгодой использовать информацию, доступную по ходу игры.

Рассматриваемый простой пример позволяет показать некоторые принципиально важные аспекты методов обучения с подкреплением. Во-первых, акцент в них делается на обучение через взаимодействие со средой, в случае рассматриваемой игры — с игроком-соперником. Во-вторых, имеется четко выраженная цель, а реализация надлежащего поведения требует такого планирования или прогнозирования, при котором принималось бы во внимание отдаленное влияние возможных решений. Например, игрок, использующий простое обучение с подкреплением, мог бы научиться выстраивать многоходовые комбинации-ловушки для не слишком дальновидного соперника. Поразительная черта решений, получаемых с помощью обучения с подкреплением, состоит в том, что можно организовать планирование, а также прогнозирование последствий принимаемых решений, не прибегая к использованию модели соперника, а также не предпринимая явного поиска среди возможных последовательностей будущих состояний и действий.

Хотя этот пример и иллюстрирует некоторые из основных свойств обучения с подкреплением, он все же настолько прост, что данный подход может показаться более ограниченным, чем это имеет место на самом деле. Хотя «крестики-нолики» представляет собой игру двух участников, обучение с подкреплением применимо и в случае, когда соперник отсутствует, т. е. в случае «игр с Природой». Кроме того, обучение с подкреплением не ограничивается задачами, в которых поведение разделяется на отдельные эпизоды, подобно отдельным партиям в игре «крестики-нолики», с вознаграждением только в конце каждого эпизода. Обучение с подкреплением в равной степени применимо и в случае

непрерывного поведения, когда выигрыши различной величины могут быть получены в любой момент.

Игра «крестики-нолики» имеет относительно небольшое конечное множество состояний, но обучение с подкреплением можно использовать и в случаях, когда это множество очень велико или даже бесконечно. Например, в работах (Tesauro 1992, 1995) алгоритм, описанный выше, был объединен с искусственной нейронной сетью для обучения игре в нарды, в которой примерно 10^{20} состояний. В такой ситуации какие-то эксперименты можно осуществить лишь для очень небольшой доли возможных состояний. Программа, реализующая предложенный алгоритм, научилась играть в нарды на уровне лучших людей-игроков мира (см. гл. 11). Нейронная сеть, использованная в данной программе, обеспечивает ей возможность обобщать накапливаемый опыт, так что в новых состояниях игры выбор ходов основывается на информации, полученной, когда в прошлом встречались аналогичные состояния. Насколько хорошо система, основанная на обучении с подкреплением, будет работать с задачами, имеющими такие большие множества возможных состояний, зависит от того, насколько успешно она сможет обобщать прошлый опыт. Именно в такой ситуации очень нужны методы обучения с учителем, основанные на подкреплении. Нейронные сети для решения данной задачи не являются ни единственным, ни даже наилучшим средством.

В рассматриваемом примере игры в крестики-нолики обучение началось без какой-либо априорной информации о правилах игры, однако обучение с подкреплением отнюдь не следует концепции *tabula rasa*¹⁾ в отношении обучения и интеллекта. Напротив, априорная информация может быть встроена в обучение с подкреплением различными способами, от которых зависит, насколько эффективным будет обучение. В этом примере использовалась также полная и неискаженная информация о состояниях игры, однако обучение с подкреплением можно применять и в тех случаях, когда часть состояний недоступна для наблюдения или когда различные состояния представляются обучаемому одними и теми же. Этот случай, однако, существенно труднее и подробно в данной книге он рассматриваться не будет.

Наконец, в представленном примере игрок в «крестики-нолики» мог прогнозировать состояния, которые будут следствием каждого из возможных ходов. Для этого он должен располагать некоторой моделью

¹⁾ *Tabula rasa* (лат.) — чистая доска; нечто нетронутое, не испорченное посторонним влиянием. — Прим. ред.

рассматриваемой игры, позволяющей «размышлять» о том, как среда могла бы измениться в ответ на ходы, которые никогда ранее не совершались. Многие из существующих задач похожи на эту, но есть и такие, в которых недоступна даже модель краткосрочного влияния предпринимаемых действий на среду. Обучение с подкреплением можно применять в любом из этих случаев. Никаких моделей в обязательном порядке не требуется, но в то же время, если они существуют или же могут быть получены обучением, их можно весьма просто использовать.

Упражнение 1.1 [Игра «сам-с-собой»]. Пусть алгоритм обучения с подкреплением, описанный выше, играет не с каким-то определенным соперником, а против самого себя. Как вы думаете, что произойдет в этом случае? Научится ли алгоритм играть как-то по-другому?

Упражнение 1.2 [Симметрии]. Многие позиции в игре «крестики-нолики» выглядят различными, однако на самом деле они совпадают вследствие симметрии. Каким образом следует изменить алгоритм обучения с подкреплением, описанный выше, чтобы извлечь пользу из данного обстоятельства? За счет чего можно было бы улучшить этот алгоритм? А теперь подумайте вот над чем. Предположим, что наш соперник не использует информацию о симметриях. Следует ли в такой ситуации использовать эту информацию нам? Будет ли тогда верным утверждение о том, что симметрично эквивалентные позиции обязательно имеют одну и ту же ценность?

Упражнение 1.3 [Жадный игрок]. Предположим, что игрок, использующий обучение с подкреплением, является *жадным*, т. е. он всегда делает такие ходы, которые переводят его в позицию с наивысшей ценностью. Научится ли он играть лучше или хуже, чем игрок, не отличающийся жадностью? Какие проблемы могут возникнуть у жадного игрока?

Упражнение 1.4 [Обучение путем зондирования]. Примем, что обучающие корректировки осуществляются после всех ходов, включая и зондирующие (пробные) ходы. Если размер шага с течением времени соответствующим образом уменьшается, то значения ценности состояний будут сходиться к некоторому набору вероятностей. Какими будут эти наборы для случаев, когда зондирующие ходы используются в обучении и когда они не используются? Если выполнение зондирующих ходов продолжится, какой из этих наборов вероятностей лучше использовать для обучения? Какой из них привел бы к большему числу выигранных партий?

Упражнение 1.5 [Другие улучшения]. Придумайте другие улучшения для игрока, использующего обучение с подкреплением. Предложите более эффективные способы игры в крестики-нолики в том ее виде, как она представлена здесь.

1.5. Итоги

Обучение с подкреплением представляет собой вычислительный подход к пониманию и реализации целенаправленного обучения и принятия решений. Оно отличается от других аналогичных подходов к решению этих задач тем, что акцент делается на обучение агента через прямое взаимодействие его со средой, без использования обучения по примерам или полных моделей среды. По нашему мнению, обучение с подкреплением — это первая область, серьезно изучающая вычислительные аспекты, которые возникают при попытке удовлетворить долгосрочным целям за счет обучения посредством взаимодействия со средой.

Обучение с подкреплением использует формальную основу, которая определяет взаимодействие между обучающимся агентом и его средой в терминах состояний, действий и вознаграждений. Эта основа должна давать некоторый простой способ выражать существенные черты задач искусственного интеллекта, включая смысл таких понятий, как причина и следствие, неопределенность, недетерминированность, а также существование в этих задачах явно выраженных целей.

Использование концепций ценности и функции ценности — одна из главнейших отличительных особенностей методов обучения с подкреплением, рассматриваемых в этой книге. По нашему мнению, функции ценности существенно важны для эффективного поиска в пространстве стратегий. Способ использования функций ценности отличает методы обучения с подкреплением от эволюционных методов, которые осуществляют прямой поиск в пространстве стратегий, направляемый скалярными оценками стратегий в целом.

1.6. История обучения с подкреплением

В истории обучения с подкреплением можно выделить два основных направления, богатых результатами, долго развивавшихся независимо друг от друга до той поры, пока они не переплелись в современном варианте обучения с подкреплением. Одно из этих направлений, имеющее дело с обучением на основе проб и ошибок, возникло в психологии обучения животных. Это направление повлияло на некоторые из ранних

работ в области искусственного интеллекта, а также привело к возрождению работ по обучению с подкреплением в начале 1980-х гг. Второе направление связано с задачей оптимального управления и ее решением, используя функции ценности и динамическое программирование. Как правило, в задачах из этой области обучение не используется. Хотя эти два направления и развивались независимо друг от друга, были и исключения, которые группировались в третьей, менее явно выраженной, области, в которой использовались методы обучения, основанные на использовании временных различий наподобие тех, что были использованы в примере игры в крестики-нолики, описанном в данной главе. Все эти три области сошлись вместе в конце 1980-х гг., и на их основе возникла область обучения с подкреплением в ее современном понимании, излагаемом в данной книге.

Направление, связанное с обучением путем проб и ошибок, наиболее нам знакомо, и именно о нем и будет идти речь на протяжении большей части предлагаемого краткого рассказа об истории обучения с подкреплением. Но вначале обсудим кратко направление, связанное с оптимальным управлением.

Термин «оптимальное управление» вошел в употребление в конце 1950-х гг. для описания задачи разработки регулятора, минимизирующего некоторую меру поведения динамической системы, определенную на заданном интервале времени. Один из подходов к решению данной задачи был разработан в середине 1950-х гг. Ричардом Беллманом и другими исследователями путем развития теории Гамильтона—Якоби, созданной еще в XIX в. Этот подход использует понятия состояния динамической системы и функции ценности или «функции оптимальной платы», чтобы определить функциональное уравнение, часто называемое теперь уравнением Беллмана. Класс методов решения задач оптимального управления, основанных на решении данного уравнения, известен как динамическое программирование (Bellman 1957a). В работе (Bellman 1957b) вводится также дискретный стохастический вариант задачи оптимального управления, именуемый марковскими процессами принятия решений (МППР), а в (Howard 1960) для МППР разработан метод итераций по стратегиям. Все эти элементы лежат в основе теории и алгоритмов современного подхода к обучению с подкреплением.

Динамическое программирование обычно рассматривается как единственный возможный путь решения задач оптимального стохастического управления общего вида. Оно, однако, страдает от того, что было названо Беллманом «проклятием размерности» — потребности в вычис-

литеральных ресурсах растут в этих задачах экспоненциально с ростом числа переменных состояния. Тем не менее динамическое программирование остается намного более эффективным и намного более широко распространенным, чем любой другой общий метод. Динамическое программирование усиленно разрабатывалось, начиная со второй половины 1950-х гг., включая расширение этого подхода на марковские процессы принятия решений с частичной наблюдаемостью (обзор работ этого направления см. в (Lovejoy 1991)), многочисленные приложения (см. обзоры (White 1985, 1988, 1993)), методы аппроксимации (см. обзор (Rust 1996)) и асинхронные методы (Bertsekas 1982, 1983). Имеется также и много современных трактовок динамического программирования, например (Bertsekas 1995; Puterman 1994; Ross 1983), а также (Whittle 1982, 1983). В книге Брайсона (Bryson 1996) дается авторитетное изложение истории оптимального управления.

В этой книге работы в области оптимального управления рассматриваются с точки зрения потребностей обучения с подкреплением. Когда идет речь об обучении с подкреплением, подразумевается наличие некоторого эффективного способа решения задач такого обучения. Ясно, что эти задачи тесно связаны с задачами оптимального управления, особенно с теми, которые формулируются как марковские процессы принятия решений. Соответственно, следует рассматривать методы оптимального управления, в частности динамическое программирование, как возможные методы обучения с подкреплением. Разумеется, почти все эти методы требуют полного знания об объекте управления, и по этой причине, наверное, может показаться несколько странным говорить о них как о методах, имеющих отношение к *обучению с подкреплением*. Но, с другой стороны, многие методы динамического программирования являются многошаговыми и итеративными. Подобно методам обучения, они приводят к ответу постепенно, через последовательность приближений. Как будет показано в оставшейся части данной книги, это сходство вовсе не является поверхностным. Теории и методы решения рассматриваемой задачи для случаев с полной и неполной информацией настолько тесно связаны друг с другом, что, на наш взгляд, их следует рассматривать совместно, как части одной и той же темы.

Вернемся теперь к одному из главных направлений современного обучения с подкреплением. В основе этого направления лежит идея обучения методом проб и ошибок, а возникло оно в психологии, где общеприняты теории «подкрепления» при обучении. По-видимому, первым суть обучения методом проб и ошибок выразил в сжатой форме Эдвард Торндайк (Edward Thorndike). Дело в том, что действия, приводящие

к хорошим или плохим последствиям, в следующий раз будут соответственно повторяться или отвергаться. Или, словами Торндайка:

Из нескольких откликов на одну и ту же ситуацию те из них, которым сопутствует удовлетворение потребности животного (или хотя бы такое удовлетворение следует вскоре после соответствующего отклика), при прочих равных условиях будут более тесно связаны с данной ситуацией, так что если она повторится, то и соответствующие действия, скорее всего, также будут повторены. Для тех же откликов, которым сопутствует или вскоре за которыми следует затруднение в удовлетворении потребности животного, при прочих равных условиях связи с данной ситуацией станут более слабыми, и если она повторится, то использование соответствующих действий станет менее вероятным. Чем сильнее полученное удовлетворение или затруднение, тем больше будет степень усиления или ослабления соответствующей связи (Thorndike 1911, с. 244).

Данному положению Торндайк дал наименование «Закон влияния», поскольку с его помощью описывается воздействие подкрепляющих событий на характер выбираемых действий. Хотя в ряде случаев данный закон не выглядит бесспорным (см., например, (Kimble 1961, 1967; Mazur 1994)), он, тем не менее, широко используется как очевидный базисный принцип, который лежит в основе многих поведенческих актов (см., например, (Hilgard, Bower 1975; Dennett 1978; Campbell 1960; Cziko 1995)).

Закон влияния включает два наиболее важных аспекта того, что принято называть обучением методом проб и ошибок. Во-первых, он *избирателен*, т. е. включает проверку и отбор альтернатив путем сравнения их последствий. Во-вторых, он *ассоциативен*, т. е. альтернативы, найденные путем отбора, связываются с конкретными ситуациями. Естественный отбор в эволюции представляет собой важнейший пример процесса отбора, однако он не ассоциативен. Обучение с учителем является ассоциативным, но оно не избирательно. Именно комбинация этих двух факторов представляет собой существенную отличительную черту закона влияния и обучения методом проб и ошибок. Еще один способ показать, что собой представляет закон влияния, состоит в объединении *поиска* и *памяти*. При этом поиск трактуется как проверка различных вариантов действий применительно к каждой из ситуаций с выбором одного из этих вариантов, а память — как способ сохранения действий, позволивших получить наилучший результат, связывая их с теми ситуациями, в которых они дали этот результат. Именно такой способ объединения поиска и памяти характерен для обучения с подкреплением.

Идея запрограммировать компьютер так, чтобы он мог обучаться методом проб и ошибок, восходит к наиболее ранним исследованиям взаимосвязей компьютеров и интеллекта (см., например, (Turing 1950)). Самыми первыми из опубликованных результатов в области компьютерных исследований обучения методом проб и ошибок были, по-видимому, работы Минского (Minsky), а также Фарли (Farley) и Кларка (Clark), изданные в 1954 г. Минский в своей докторской диссертации обсуждал компьютерные модели обучения с подкреплением и описал конструкцию предложенной им аналоговой машины, собранной из элементов, названных им стохастическими нейро-аналоговыми калькуляторами с подкреплением (SNARC – Stochastic Neural-Analog Reinforcement Calculators). Фарли и Кларк описали другую нейросетевую обучающуюся машину, разработанную для обучения ее с помощью метода проб и ошибок. В 1960-х гг. термины «подкрепление» и «обучение с подкреплением» впервые стали широко использоваться в технической литературе (см., например, (Waltz, Fu 1965; Mendel 1966; Fu 1970; Mendel, McLaren 1970)). Очень большое влияние оказала статья Минского «На пути к искусственному интеллекту» (Minsky 1961), в которой обсуждался ряд проблем, принципиально важных для обучения с подкреплением. Одну из них Минский назвал *задачей распределения коэффициентов уверенности*: «Как распределить значения коэффициентов, описывающих уверенность в успехе, между многими возможными вариантами решения? Все методы, рассматриваемые в данной книге, направлены, в известном смысле, на решение этой проблемы.

Интересы Фарли и Кларка (Farley, Clark 1954; Clark, Farley 1955) сместились с методов обучения с подкреплением на решение проблем обобщения и распознавания образов, т. е. из области обучения с подкреплением в область обучения с учителем. С этого времени началась путаница в трактовке взаимосвязей между данными двумя видами обучения. Создается впечатление, что многие исследователи считали, будто они занимаются обучением с подкреплением, хотя фактически они изучали обучение с учителем. Например, такие пионеры нейросетевых исследований, как Розенблatt (Rosenblatt 1962), а также Уидроу и Хофф (Widrow, Hoff 1960), с полной очевидностью руководствовались идеями обучения с подкреплением (очевидно, используя язык поощрений и наказаний), однако работали с системами, обучаемыми с учителем и предназначенными для решения задач распознавания образов и перцептивного обучения. Даже сейчас исследователи и учебники часто сводят к минимуму различия между этими двумя видами обучения или излагают их очень неясно. В некоторых современных учебниках используется

термин «пробы и ошибки» при описании нейронных сетей, которые обучаются по примерам, поскольку для корректировки весов связей в рассматриваемой сети используется информация об ошибке¹⁾. Такого рода путаница вполне объяснима, однако из-за нее упускается существенно избирательный характер обучения методом проб и ошибок.

Отчасти вследствие такого смешения понятий исследования в области «подлинных» методов обучения путем проб и ошибок в 60–70-е годы стали очень редкими. В следующих абзацах мы обсудим некоторые исключения из этого общего правила.

Одно из них — работа (Andreae 1963) новозеландского исследователя Джона Андрэ (John Andreae), где описывается созданная им система STeLLA, которая обучалась методом проб и ошибок через взаимодействие со средой. Эта система включала внутреннюю модель мира, а впоследствии еще и «внутренний монолог», обеспечивающий возможность работы со скрытыми состояниями (Andreae 1969a). В более поздней работе Андрэ (Andreae 1977) существенное внимание уделяется обучению с учителем, однако она по-прежнему включает в себя метод проб и ошибок. К сожалению, его пионерская работа не была достаточно хорошо известна и поэтому не оказала большого влияния на последующие работы в области обучения с подкреплением.

Большее влияние оказали работы Дональда Мичи (Donald Michie). В 1961 и 1963 гг. он описал простую систему (Michie 1961, 1963), обучающуюся методом проб и ошибок, которая могла научиться играть в крестики-нолики. Эта система была названа MENACE (Matchbox Educable Naughts and Crosses Engine). Она содержала по одному спичечному коробку для каждой возможной позиции в игре, в каждом спичечном коробке было несколько цветных бусинок, по одному цвету для каждого из возможных ходов из данной позиции. Ход в системе MENACE определялся извлечением некоторой бусинки из спичечного коробка, отвечающего данной позиции в игре. Это извлечение осуществлялось случайным образом. По окончании игры бусинки добавлялись в коробки, использованные в игре, или же удалялись из них, чтобы поощрить или наказать MENACE за принятые решения. Мичи и Чамберс (Michie, Chambers 1968) описали также и другую систему, названную GLEE (Game Learning Expectimaxing Engine), использующую обучение с подкреплением при игре в крестики-нолики. Кроме того, был построен еще и регулятор BOXES, также основанный на обучении с подкреп-

¹⁾ То есть о рассогласовании между требуемым и фактическим выходом сети. — Прим. ред.

лением. Он применялся для решения задачи обучения поддержанию равновесия стержня, закрепленного шарниром на движущейся тележке. Это обучение было основано только на сигнале ошибки, который возникал, если стержень падал или же тележка достигала конца дорожки, по которой она двигалась. Данной работе предшествовала статья Уидроу и Смита (Widrow, Smith 1964), в которой использовались методы обучения с учителем, причем предполагалось, что инструкции, выдаваемые учителем, обеспечивают решение задачи поддержания равновесия стержня. Вариант решения данной задачи, предложенный Мичи и Чамберсом, представляет собой один из лучших ранних примеров использования обучения с подкреплением в условиях неполного знания. Эта работа оказала влияние на другие работы, намного более поздние, в том числе и на наши собственные исследования (Barto, Sutton, Anderson 1983; Sutton 1984). Мичи последовательно подчеркивал роль проб и ошибок, а также обучения в качестве существенных аспектов проблемы искусственного интеллекта (Michie 1974).

Уидроу, Гупта и Мэйтре (Widrow, Gupta, Maitra 1973) модифицировали алгоритм Уидроу—Хоффа (Widrow, Hoff 1960), основанный на методе наименьших квадратов, чтобы получить правило обучения с подкреплением, использующее сигналы об успехе или неудаче предпринятых действий вместо применения обучающих примеров. Авторы назвали эту форму обучения «избирательная самонастраивающаяся адаптация» и описали ее как «обучение с критиком» взамен «обучения с учителем». Они проанализировали этот подход и показали, как его можно применить для обучения карточной игре в очко. Это был единственный случай, когда Уидроу работал в области обучения с подкреплением. Его вклад в область обучения с учителем намного более весом.

Исследования по *обучающимся автоматам* прямо повлияли на направление в обучении, основанное на методе проб и ошибок, что способствовало появлению обучения с подкреплением в его современном понимании. Речь идет о методах, предназначенных для решения одной неассоциативной, чисто переборной проблемы обучения, получившей название «задачи об *n*-руком бандите» по аналогии с жаргонным называнием игрового автомата («однорукий бандит»), только с *n* рычагами (см. гл. 2). Обучающиеся автоматы, позволяющие решать данную задачу, представляют собой простые машины с небольшим объемом памяти. Первоначально обучающиеся автоматы были предложены в России в работах Цетлина (Tsetlin 1973), а затем эта концепция широко разрабатывалась применительно к решению технических задач (см. (Narendra,

Thathachar 1974, 1989)). Барто и Анандан (Barto, Anandan 1985) развили эти методы применительно к ассоциативному случаю.

Джон Холланд (John Holland) в 1975 г. построил основы общей теории адаптивных систем (Holland 1975), основанной на принципах отбора. В его ранней работе метод проб и ошибок применялся непосредственно в неассоциативной форме, как в эволюционных методах и в задаче об *n*-руком бандите. В 1986 г. им были введены *классифицирующие системы* (Holland 1986), которые представляли собой уже настоящие системы, использующие обучение с подкреплением, включая использование таких механизмов, как ассоциации и функции ценности. Ключевым элементом классифицирующих систем Холланда всегда был *генетический алгоритм*, т. е. некоторый эволюционный метод, роль которого состояла в постепенном выявлении успешных решений. Работы в области классифицирующих систем интенсивно проводились многими исследователями и сформировали одно из основных направлений в обучении с подкреплением (см., например, (Goldberg 1989; Wilson 1994)), однако генетические алгоритмы, которые сами по себе не относятся к области систем обучения с подкреплением, привлекали и привлекают к себе намного больше внимания.

Наиболее существенные заслуги в части возрождения работ, связанных с применением метода проб и ошибок для обучения с подкреплением в рамках исследований по искусственноому интеллекту, принадлежат Харри Клопфу (Klopf 1972, 1975, 1982). Клопф обнаружил, что после того, как исследователи, работающие в области обучения машин, сконцентрировались почти исключительно на задачах обучения с учителем, были утеряны существенные аспекты, присущие адаптивному поведению. А именно, потеряны были, согласно Клопфу, гедонистические аспекты поведения, т. е. то, что обеспечивает настойчивость в достижении результата во взаимодействия со средой, в управлении средой таким образом, чтобы обеспечить получение желаемого результата и избежать нежелательных последствий. Эта идея представляет собой существенный элемент подхода к обучению методом проб и ошибок. Идеи Клопфа значительно повлияли на исследования авторов данной книги, поскольку наши попытки оценить значимость его работ (Barto, Sutton 1981a) привели к пониманию различия между обучением с учителем и обучением с подкреплением, а также к переносу акцентов в проводимых исследованиях на обучение с подкреплением. Многие из ранних работ, как наших, так и наших коллег, были направлены на то, чтобы показать, что между обучением с подкреплением и обучением с учителем и в самом деле существует разница (Barto, Sutton, Brouwer 1981; Barto, Sutton

1981b; Barto, Anandan 1985). В других исследованиях было показано, каким образом обучение с подкреплением можно было бы использовать для решения важных проблем, связанных с обучением нейронных сетей, в частности как можно было бы получить обучающие алгоритмы для многослойных сетей (Barto, Anderson, Sutton 1982; Barto, Anderson 1985; Barto, Anandan 1985; Barto 1985, 1986; Barto, Jordan 1987).

Обратимся теперь к третьему направлению в истории обучения с подкреплением — обучению на основе временных различий. Отличительная черта методов обучения такого вида состоит в том, что в них используются упорядоченные по времени оценки одной и той же величины, например вероятности выигрыша в примере игры в крестики-нолики. Данное направление работ меньше по объему и не так ярко выражено, как два других, однако оно играет важную роль в рассматриваемой области. Это связано с тем, в частности, что методы временных различий можно трактовать как новый и уникальный элемент в составе комплекса средств, обеспечивающего решение задачи обучения с подкреплением.

Истоки обучения на основе временных различий лежат частично в психологии обучения животных, в частности в идеи *вторичных подкрепляющих стимулов*. Вторичный подкрепляющий стимул — это некоторый стимул, являющийся парным по отношению к определенному первичному подкрепляющему стимулу, такому как пища или боль, приводящий к тем же самым результатам подкрепления. По-видимому, Минский (Minsky 1954) был первым, кто осознал важность данного психологического принципа для искусственных обучающихся систем. Артур Сэмюэль (Samuel 1959) первым предложил и разработал метод обучения, основанный на идеи временных различий, в качестве составного элемента его знаменитой программы для игры в шашки. У Сэмюэля не было ссылок на работу Минского или на возможные связи с проблемой обучения животных. Источником вдохновения для него явно послужило высказывание Клода Шеннона (Shannon 1950) о том, что компьютер можно было бы запрограммировать для игры в шахматы с использованием некоторой оценочной функции, а также о том, что при этом можно было бы улучшать игру путем оперативной корректировки данной функции. (Вполне возможно, что эти идеи Шеннона оказали также влияние и на Беллмана, но у нас нет никаких свидетельств на этот счет.) Минский в 1961 г. подробно разобрал работу Сэмюэля в своей статье (Minsky 1961), рассмотрев связь с теориями вторичного подкрепления применительно как к естественным системам, так и к искусственным.

Как уже говорилось, в десятилетие, прошедшее с момента выхода работ Минского и Сэмюэля, усилий по компьютерной реализации обучения методом проб и ошибок предпринималось совсем немного, а в области обучения на основе временных различий таких попыток не предпринималось вовсе. В 1972 г. Клопф объединил обучение методом проб и ошибок с одним из важных элементов, относящихся к обучению на основе временных различий. Клопф интересовался принципами, на которых можно было бы организовать обучение в больших системах. По этой причине его внимание привлекли идеи локального подкрепления, в соответствии с которыми компоненты обучающейся системы в целом могли бы стимулировать друг друга. Он разработал идею «обобщенного подкрепления», в соответствии с которой каждый элемент (номинально, каждый нейрон) системы воспринимает все свои входы в терминах подкреплений: возбуждающие входы как вознаграждения и тормозящие входы как наказания. Эта идея не соответствует современному пониманию обучения на основе временных различий. Оглядываясь назад, можно заключить, что она дальше от современной трактовки, чем работа Сэмюэля. Однако Клопф связал эту идею с обучением на основе метода проб и ошибок, а также установил для нее связи с обширной эмпирической базой данных о психологии обучения животных.

Саттон в дальнейшем (Sutton 1978a, 1978b, 1978c) развил идеи Клопфа, в особенности в том, что касается их связей с теориями обучения у животных. Он описал обучающие правила, которые работают на основе учета изменений в последовательностях предсказаний, упорядоченных по времени. Саттон с Барто усовершенствовали эти идеи и разработали психологическую модель выработки классического (павловского) условного рефлекса на основе обучения по временным различиям (Sutton, Barto 1981a; Barto, Sutton 1982). За этой работой последовал ряд других важных публикаций, где излагались психологические модели формирования павловского условного рефлекса, также основанные на обучении по временным различиям (см., например, (Klopf 1988; Moore et al. 1986; Sutton, Barto 1987, 1990)). В терминах временных различий можно прекрасно интерпретировать и ряд нейрофизиологических моделей, разработанных в это же время (Hawkins, Kandel 1984; Butne, Gingrich, Baxter 1990; Gelperin, Hopfield, Tank 1985; Tesauro 1986; Friston et al. 1994), хотя в большинстве случаев связей и взаимодействия между этими областями не существовало. Недавний обзор взаимосвязей между обучением на основе временных различий и идеями нейробиологии содержится в работе (Schultz, Dayan, Montague 1997).

Наши ранние работы по методам обучения на основе временных различий находились под сильным влиянием теорий обучения животных, а также исследований Клопфа. Взаимосвязи со статьей Минского «На пути к искусственному интеллекту» (Minsky 1961) и программами Сэмюэля для игры в шашки (Samuel 1959) были нами выявлены и осознаны лишь впоследствии. К 1981 г., однако, мы уже знали обо всех предшествующих работах, упоминавшихся выше в качестве составных элементов направлений, связанных с обучением на основе временных различий и с обучением методом проб и ошибок. К этому времени мы разработали метод использования временных различий в обучении на основе проб и ошибок. Этот метод получил наименование «архитектура исполнитель-критик». Он был применен к решению задачи Мичи-Чамберса о поддержании равновесия стержня, закрепленного шарнирно на подвижной тележке¹⁾ (Barto, Sutton, Anderson 1983). Обстоятельное изучение этого метода было предпринято в докторской диссертации Саттона (Sutton 1984). В диссертации Андерсона он был расширен для использования нейронных сетей, обучаемых по методу обратного распространения ошибки (Anderson 1986). Примерно в это же время Холланд (Holland 1986) ввел в явном виде идею временного различия в свои классифицирующие системы. Важнейший шаг в 1988 г. сделал Саттон (Sutton 1988), когда отдал обучение на основе временных различий от управления, с трактовкой такого обучения как метода предсказания общего вида. В этой работе был введен также TD(λ)-алгоритм²⁾ и изучены основные его свойства с точки зрения сходимости.

После завершения в 1981 г. работы над архитектурой «исполнитель-критик» мы обнаружили статью (Witten 1977), написанную Айаном Уиттеном (Ian Witten), в которой содержалось наиболее раннее из известных нам изложений идей обучающего правила на основе временных различий. Уиттен предложил метод, который мы теперь называем табличным TD(0)-методом, для использования в качестве составной части адаптивного регулятора при решении задач управления марковскими процессами принятия решений. Работа Уиттена основывалась на ранних экспериментах Андрэ с системой STeLLA, а также на результатах, полученных для других систем, реализующих обучение методом проб и ошибок. Таким образом, статья Уиттена 1977 года охватывает оба главных направления исследований в области обучения с подкреплени-

¹⁾ Задачу о поддержании равновесия стержня, закрепленного шарнирно на подвижной тележке (pole-balancing problem), часто называют также задачей о перевернутом (обратном) маятнике (inverted pendulum problem). — Прим. ред.

²⁾ От Temporal-Difference — временное различие. — Прим. ред.

ем, связанных с методом проб и ошибок, а также с оптимальным управлением; кроме того, она представляет собой одну из наиболее ранних попыток использования обучения на основе временных различий.

Наконец, в 1989 г. направления, связанные с временными различиями и оптимальным управлением, были полностью объединены в *Q*-обучении, созданном Крисом Уоткинсом (Chris Watkins) в его диссертации (Watkins 1989). Эта работа развивала и объединяла предшествующие результаты, связанные со всеми тремя основными направлениями исследований в области обучения с подкреплением. В это объединение внес вклад также и Пол Вербос (Paul Werbos), который начиная с 1977 года уделял внимание слиянию обучения на основе метода проб и ошибок и динамического программирования (Werbos 1977, 1987). Ко времени выхода работы Уоткинса исследования в области обучения с подкреплением переживали период бурного роста в первую очередь в той части исследований по искусственному интеллекту, что была связана с обучением машин, но еще и применительно к нейронным сетям, а также к искусственному интеллекту в более широкой его трактовке. В 1992 г. поразительный успех программы TD-Gammon, написанной Джерри Тезауро (Gerry Tesauro) и предназначенный для игры в нарды¹⁾ (Tesauro 1992), привлек к данной области дополнительное внимание. Другие важные достижения в области обучения с подкреплением, полученные в последние годы, настолько многочисленны, что нет возможности даже упомянуть их в данном кратком обзоре. Ссылки на эти работы будут приводиться в конце глав, с которыми данные работы связаны тематически.

1.7. Библиографические и исторические справки

За дополнительной информацией об обучении с подкреплением в целом мы отсылаем читателя к книгам (Bertsekas, Tsitsiklis 1996) и (Kaelbling 1993а). Обучению с подкреплением были посвящены два специальных выпуска журнала *Machine Learning* («Обучение

¹⁾ Нарды (англ. backgammon) — игра на специальной доске с 24 игровыми полями для двух партнеров. У каждого из партнеров по 15 фишек, ходы выполняются с использованием двух костей в виде кубиков с числами от 1 до 6 на гранях. Цель игры — первым вывести свои фишки за пределы игрового поля. В нардах гаммон — двукратная победа (выигрыш двух очков вместо одного, получаемого при обычном выигрыше), которая достигается в случае, если игрок вывел все свои фишки, тогда как его соперник — ни одной. — Прим. ред.

машин»): (Sutton 1992) и (Kaelbling 1996). Полезны также обзоры (Barto 1995b; Kaelbling, Littman, Moore 1996; Keerthi, Ravindran 1997).

Пример с завтраком Фила в данной главе возник под влиянием работы (Agre 1988). Библиографические ссылки для того варианта метода временных различий, который был использован в рассмотренном примере игры в крестики-нолики, можно найти в шестой главе.

Современные попытки установить, какого вида алгоритмы используются при обучении с подкреплением нервной системой, освещены в работах (Hampson 1989; Friston et al. 1994; Barto 1995a; Houk, Adams, Barto 1995; Montague, Dayan, Sejnowski 1996; Schultz, Dayan, Montague 1997).

2 Оценочная обратная связь

Наиболее важной особенностью, отличающей обучение с подкреплением от других видов обучения, является использование обучающей информации, которая *оценивает* осуществляемые действия вместо того, чтобы *наставлять*, показывая правильные действия. По этой причине для получения приемлемого поведения требуется активное выполнение пробных действий, явным образом организованный поиск методом проб и ошибок. Чисто оценочная обратная связь показывает лишь, насколько хорошим является предпринятое действие, но не говорит о том, является ли оно лучшим или худшим из всех возможных. Оценочная обратная связь служит основой методов оптимизации функций, в том числе и эволюционных методов. В противоположность этому, инструктивная обратная связь указывает, каким должно быть правильное действие, независимо от того, что за действие было реализовано фактически. Данный вид обратной связи представляет собой основу обучения с учителем, которая широко используется в классификации образов, искусственных нейронных сетях, идентификации систем. В чистом виде два этих вида обратной связи полностью различны: оценочная обратная связь зависит только от предпринятого действия, тогда как инструктивная обратная связь от него не зависит вообще. Существуют также и представляющие интерес промежуточные варианты обратной связи, в которых сочетаются оценочная и инструктивная составляющие.

В этой главе рассматривается оценочный аспект обучения с подкреплением в упрощенной постановке, когда обучение задействуется не более чем для одной ситуации. Эта постановка является *неассоциативной*, в ее рамках предварительно рассматриваются все основные вопросы, включая оценочную обратную связь. Такой подход позволяет не касаться пока многих сложных вопросов, связанных с рассмотрением всей проблемы обучения с подкреплением в целом. Изучение упрощенного случая позволяет совершенно ясно увидеть основные различия оценочных и инструктивных обратных связей, а также возможности их сочетания.

В качестве конкретной задачи с неассоциативной оценочной обратной связью мы рассмотрим несложный вариант задачи об *n*-руком бандите. Мы введем ряд основных методов обучения, которые

в последующих главах изучим более подробно применительно к задаче обучения с подкреплением в полной постановке. В конце этой главы мы рассмотрим ассоциативную задачу об n -руком бандите, т. е. такую, в которой действие предпринимается в более чем одной ситуации. Такой анализ позволит сделать еще один шаг в направлении полной задачи обучения с подкреплением.

2.1. Задача об n -руком бандите

Рассмотрим следующую задачу обучения. Пусть приходится многократно осуществлять выбор одной из n различных альтернатив (вариантов действий). Каждый выбор влечет за собой получение определенного вознаграждения, значение которого получается с использованием некоторого стационарного распределения вероятностей, зависящего от выбранного действия. Целью последовательности действий является максимизация ожидаемого полного вознаграждения за заданный период времени, например за 1000 попыток выбора варианта действия. Каждая такая попытка именуется игрой.

Это исходная форма представления *задачи об n -руком бандите*. В отличие от игрового автомата или «однорукого бандита» мы рассмотрим автомат, в котором n рычагов вместо одного. Каждый выбор действия уподобляется воздействию на один из рычагов автомата в ходе игры, а вознаграждениями являются выигрыши в данной игре. В серии повторяющихся игр требуется максимизировать выигрыш путем выбора лучших рычагов. Другая аналогия связана с ситуацией, когда врач выбирает курс лечения для каждого из группы серьезно больных пациентов. Каждая игра в данном случае — это выбор лечения, а каждое вознаграждение — выживание или выздоровление пациента. В настоящее время термин «задача об n -руком бандите» часто обозначает обобщенный вариант описанной выше задачи, однако в этой книге он будет использоваться только применительно к рассмотренному простому случаю.

В рассматриваемой задаче об n -руком бандите каждому действию поставлено в соответствие ожидаемое или среднее вознаграждение, называемое при выборе этого действия; будем называть его *ценностью* данного действия. Если бы ценность каждого действия была известна, решение задачи об n -руком бандите было бы тривиальным: всегда следовало бы выбирать действие с наивысшей ценностью. Будем считать, что точные значения ценности действий неизвестны, но можно получить оценки этих значений.

Если ценности действий оцениваются постоянно, то в каждый момент времени найдется хотя бы одно действие, для которого предполагаемая ценность будет наибольшей. Назовем такое действие *жадным*. Если выбирается жадное действие, то говорят, что *применяется* текущее знание о ценности действий. Если вместо этого выбирается одно из нежадных действий, то говорят об *изучении*, поскольку таким образом можно точнее оценить ценность нежадного действия. Применение текущих знаний позволяет максимизировать ожидаемое вознаграждение в одной игре, тогда как изучение может привести к увеличению общего вознаграждения в долгосрочной перспективе. Пусть, например, точно известна ценность жадного действия, в то время как ценность ряда других действий близка к нему по величине, но она найдена со значительной погрешностью. Погрешность эта такова, что хотя бы одно из этих действий, скорее всего, лучше жадного действия, но какое именно — неизвестно. Если еще предстоит сыграть много игр, то, может быть, стоит изучить эти нежадные действия и выяснить, какие из них лучше данного жадного действия. В течение непродолжительного времени, пока выполняется изучение, вознаграждение снизится, но оно будет более высоким в долгосрочной перспективе, поскольку после обнаружения действий лучше жадного появляется возможность применять их. Из-за того что невозможно одновременно получать и применять знания в рамках выбора единственного действия, часто говорят о «конфликте» между изучением и применением.

В каждом конкретном случае выбор между изучением и применением сложным образом зависит от точных значений оценок, имеющихся неопределенностей и количества оставшихся игр. Существует много достаточно сложных методов достижения баланса между изучением и применением для различных частных вариантов математических формулировок задачи об n -рукум бандите, а также для задач аналогичного характера. Однако большинство из этих методов основываются на достаточно сильных предположениях относительно стационарности и априорных знаний. В прикладных задачах и в общей задаче обучения с подкреплением, рассматриваемой в последующих главах, эти предположения либо нарушаются, либо их не удается подтвердить. Тот факт, что эти методы гарантируют наилучший результат или ограничивают размеры потерь, не слишком обнадеживает в тех случаях, когда заложенные в этих методах допущения оказываются неприемлемыми.

В этой книге не будут рассматриваться какие-то изощренные способы достижения баланса между изучением и применением, вполне достаточным будет любой доступный метод такого рода. В данной главе

представлены несколько простых методов обеспечения такого баланса в задаче об n -руком бандите и показано, что они гораздо лучше методов, основанных только на применении. Отметим также, что методы обучения с учителем (или, скорее, методы, близкие к методам обучения с учителем, адаптированные к данной задаче) хуже справляются с решением поставленной задачи, поскольку в них вообще не предполагается наличия баланса между изучением и применением. Потребность в таком балансе является проблемой, характерной именно для обучения с подкреплением; простота задачи об n -руком бандите позволяет показать это в очень наглядной форме.

2.2. Методы вычисления значений ценности действий

Вначале рассмотрим более подробно некоторые простые методы, позволяющие приближенно вычислить ценности действий, а также использовать получаемые оценки при выборе действий. В данной главе будем обозначать истинную (фактическую) ценность действия a через $Q^*(a)$, а предполагаемое значение для нее в t -й игре через $Q_t(a)$. Напомним, что истинной ценностью действия является среднее значение вознаграждения, получаемого за выбор данного действия. Она вычисляется как среднее значение по совокупности вознаграждений, фактически полученных при выборе данного действия. Другими словами, если к моменту времени t начала t -й игры действие a было выбрано k_a раз, что привело к получению соответствующих вознаграждений r_1, r_2, \dots, r_{k_a} , то ценность этого действия можно оценить как

$$Q_t(a) = \frac{r_1 + r_2 + \dots + r_{k_a}}{k_a}. \quad (2.1)$$

Если $k_a = 0$, то вместо этой оценки используется некоторое значение $Q_t(a)$, принимаемое по умолчанию, например $Q_0(a) = 0$. Если $k_a \rightarrow \infty$, то по закону больших чисел $Q_t(a)$ стремится к $Q^*(a)$. Назовем этот подход методом *среднего выборочного* для оценивания ценности действия, поскольку каждая оценка представляет собой среднее арифметическое выборки рассматриваемых вознаграждений. Это, разумеется, лишь один из способов оценивания значения ценности действия, и он не обязательно наилучший. Тем не менее мы пока на нем остановимся и обратимся к вопросу о том, как полученные оценки можно использовать при выборе действий.

Проще всего выбирать такое действие (или одно из действий), которое обладает наивысшей оценкой значения ценности. Это значит, что в игре t следует выбирать одно из жадных действий a^* , для которого

$$Q_t(a^*) = \max_a Q_t(a).$$

Такой метод для максимизации немедленно получаемого вознаграждения всегда использует текущие знания. При этом вообще не предпринимаются попытки исследовать действия, представляющиеся заведомо худшими, чтобы проверить, не будут ли они на самом деле лучшими. Простой альтернативный вариант состоит в том, чтобы использовать жадные действия большую часть времени, но иногда, например с некоторой малой вероятностью ε , выбирать действие произвольно, равномерным образом, независимо от ожидаемой ценности. Такие методы, использующие близкое к жадному правило выбора действий, называются ε -жадными методами. Их преимущество состоит в том, что в пределе, когда количество игр возрастает, каждое действие будет выбрано бесконечное число раз, т. е. обязательно выполняется условие $k_a \rightarrow \infty$ для всех действий a , и вследствие этого все оценки $Q_t(a)$ будут стремиться к $Q^*(a)$. Отсюда, конечно, вытекает, что вероятность выбора оптимального действия стремится к величине большей, чем $1 - \varepsilon$, т. е. это событие является практически достоверным. Но это всего лишь асимптотическая оценка, не отражающая реальную эффективность методов данного вида.

Чтобы примерно оценить сравнительную эффективность жадного и ε -жадного методов, мы сравнивали их численно на наборе тестовых задач. Этот набор состоял из сформированных случайным образом 2000 задач об n -руком бандите, для которых было принято значение $n = 10$. Для каждого действия a вознаграждение выбиралось на основе нормального распределения вероятностей с математическим ожиданием $Q^*(a)$ и дисперсией 1. Было сформировано 2000 задач об n -руком бандите путем выборки 2000 раз значений $Q^*(a)$, каждый раз согласно нормальному распределению с математическим ожиданием 0 и дисперсией 1. Усредняя по совокупности задач, можно показать графически характеристики и поведение различных методов (см. рис. 2.1). Из приведенных данных видно, как улучшаются характеристики этих методов при количестве игр, приближающемся к 1000. Такого рода набор тестовых задач назовем *10-рукой испытательной средой*.

На рис. 2.1 жадный метод сравнивается с двумя ε -жадными методами (для $\varepsilon = 0,01$ и $\varepsilon = 0,1$) с применением сформированной 10-рукой испытательной среды так, как это было описано выше. В обоих случа-

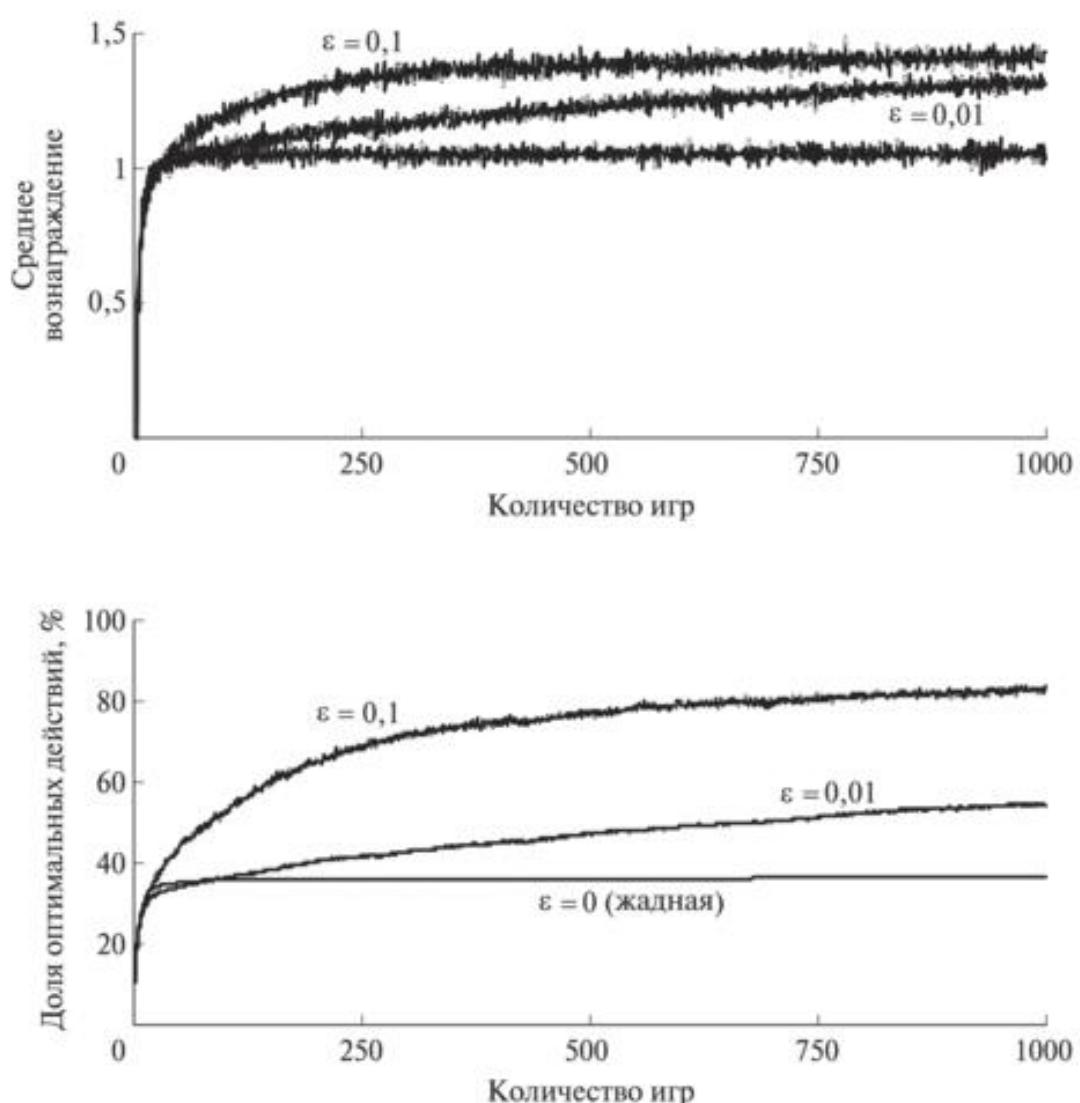


Рис. 2.1. Осредненные характеристики для жадного и ε -жадного методов выбора действия на примере 10-рукой испытательной среды. Результаты осреднены по 2000 задачам. Все методы используют средние выборочные в качестве оценок значений ценности действий

ях при оценке значения ценности действия используется метод среднего выборочного. На верхнем графике показано возрастание ожидаемого вознаграждения по мере накопления числа проведенных игр. Жадный метод обеспечивает немного более быстрый, чем остальные, рост вознаграждения в самом начале процесса, но затем этот рост прекращается на уровне, самом низком из трех сравниваемых методов. Жадный метод приводит к вознаграждению, примерно равному 1, тогда как для лучшего из методов оно составляет 1,55 на данном наборе тестовых задач. Жадный метод на больших интервалах времени значительно проигры-

вает в эффективности, поскольку в нем часто используются субоптимальные действия. На нижнем графике видно, что при использовании жадного метода оптимальные действия были найдены лишь примерно в одной трети случаев. В остальных случаях первоначальный выбор оптимального действия оказался неудовлетворительным, и оно никогда больше не использовалось. Результат в случае ϵ -жадных методов в итоге оказывается лучше, потому что процесс изучения, а следовательно, и повышения шансов выбора оптимального действия, в них не прекращается. При $\epsilon = 0,1$ изучающие действия предпринимаются чаще, и обычно оптимальное действие обнаруживается раньше, но оно никогда не будет выбрано более чем в 91% случаев. Если принять $\epsilon = 0,01$, рост получаемого вознаграждения происходит медленнее, но в итоге этот вариант метода оказывается лучше, чем при $\epsilon = 0,1$ по обоим критериям. Можно также уменьшать значение ϵ с течением времени, чтобы попытаться улучшить как значение получаемого вознаграждения, так и вероятность выбора наилучшего действия.

Преимущество ϵ -жадного метода над жадным зависит от того, какая конкретно решается задача. Пусть, к примеру, дисперсия вознаграждения больше, чем в рассмотренном выше случае, скажем 10 вместо 1. Это означает более высокий уровень неопределенности в значениях вознаграждения, откуда следует, что потребуется большее число попыток изучающего характера, чтобы отыскать требуемое оптимальное действие. В такой ситуации ϵ -жадные методы окажутся еще более эффективными в сравнении с жадным методом. С другой стороны, если дисперсия значений вознаграждения равна 0, то в случае жадного метода истинная ценность каждого действия будет известна после первой же попытки его применить. В этом случае жадный метод может оказаться лучшим, потому что после быстрого нахождения оптимального действия необходимость в изучении отпадает. Но даже в этом детерминированном случае изучение будет очень выгодным, если ослабить другие предположения. Пусть, например, рассматриваемая задача о бандите является нестационарной, т. е. истинная ценность действий меняется со временем. В этом случае изучение необходимо даже и в детерминированном случае для обеспечения уверенности в том, что нежадное действие не изменилось так, что оно стало лучше жадного. Как будет показано в следующих нескольких главах, такого рода нестационарность очень часто встречается в задачах обучения с подкреплением. Даже если исходная проблема является стационарной и детерминированной, в процессе обучения приходится иметь дело с задачами поиска решений наподобие рассмотренных, каждая из которых меняется с течением времени в зависимости

от хода процесса обучения. Обучение с подкреплением требует поддержания баланса между изучением и применением.

Упражнение 2.1. Сравните данные, приведенные на рис. 2.1. Какой метод можно считать наилучшим на большом интервале времени по совокупному вознаграждению и интегральной вероятности выбора наилучшего действия? Насколько он будет лучше?

2.3. Выбор действия с помощью операции softmax

Хотя ϵ -жадный выбор действия является эффективным и распространенным способом достижения баланса между изучением и применением в задаче обучения с подкреплением, его недостаток состоит в том, что при изучении выбор среди всех действий осуществляется на равновероятной основе. Это означает, что с одинаковой вероятностью может быть выбрано как наихудшее действие, так и близкое к наилучшему. Если наихудшие действия в рассматриваемой задаче совершенно неудовлетворительны, такой способ выбора может оказаться неприемлемым. Очевидное решение этой проблемы состоит в варьировании вероятностей действий, представляемых посредством некоторой функции от предполагаемых значений ценности. Для жадного действия вероятность выбора по-прежнему будет наибольшей, но все остальные ранжируются согласно их предполагаемой ценности. Такой принцип выбора действия называется правилом на основе операции softmax или softmax-методом. В самом распространенном варианте softmax-метода используется распределение Гиббса или распределение Больцмана. Выбор действия a в t -й игре, согласно распределению Гиббса, осуществляется с вероятностью

$$\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^n e^{Q_t(b)/\tau}}, \quad (2.2)$$

где τ — положительный параметр, называемый *температурой*. Высокие температуры делают все действия примерно равновероятными. Низкие температуры вызывают более значительное различие в вероятности выбора действий, имеющих разную предполагаемую ценность. В пределе, при $\tau \rightarrow 0$, выбор действия softmax-методом становится аналогичным жадному выбору действия. Конечно, softmax-метод не обязательно должен быть основан на распределении Гиббса. Например, можно просто добавить случайное число из распределения с длинным хвостом к каждому $Q_t(a)$, а затем выбрать действие с наибольшей суммой.

Какой метод лучше: softmax или ϵ -жадный, зависит от конкретной задачи и предпочтений человека, решающего ее. Оба метода имеют только один параметр, значение которого требуется задать. Многие считают, что легче правильно задать значение параметра ϵ , тогда как для нахождения величины τ необходимо знать ценность возможного действия и значение показателя степени экспоненты. Насколько нам известно, точного сопоставления и сравнительного анализа этих двух простых принципов выбора действия не проводилось.

Упражнение 2.2 [Программирование]. Какой результат можно получить в случае 10-рукой испытательной среды, если для выбора действий применить правило softmax, использующее распределение Гиббса? Реализуйте данный метод и запустите его с различными температурами, чтобы получить графики, аналогичные показанным на рис. 2.1. Чтобы проверить правильность работы написанной программы, реализуйте вначале ϵ -жадные методы и воспроизведите результаты рис. 2.1.

Упражнение 2.3*. Покажите, что в случае с двумя действиями операция softmax, использующая распределение Гиббса, принимает вид логистической или сигмоидальной функции, применяемой в искусственных нейронных сетях. Какое влияние на эту функцию оказывает температурный параметр?

*2.4. Оценивание в сравнении с инструктированием

Рассмотренная выше задача об n -руком бандите представляет собой пример, в котором обратная связь является полностью оценочной. Вознаграждение, получаемое после каждого действия, дает некоторую информацию о том, насколько хорошим было это действие, но ничего не говорит о том, был ли произведенный выбор корректным или нет, т. е. было ли данное действие наилучшим. Корректность представляет собой сравнительную характеристику действия, значение которой можно найти только перебором всех действий и сравнением вознаграждений, получаемых при их выполнении. В этом смысле задача требует, по сути, явного перебора всех возможных альтернативных вариантов действий. Необходимо определить некоторый способ порождения и проверки таких вариантов, посредством которого задаются действия и получаются результаты их исполнения, сопоставляемые между собой для выбора наиболее эффективного действия. Такой подход является обучением на основе выбора, которое противопоставляется обучению на ос-

нове инструктирования. Все методы обучения с подкреплением должны использовать перебор в той или иной форме.

Это резко отличается от обучения с учителем, когда обратная связь с окружающей средой прямо указывает, какое действие должно быть осуществлено. В таком случае нет необходимости в поиске. Какими бы ни были предпринимаемые действия, всегда будет указано, какое из них можно было бы считать правильным. Нет необходимости проверять разнообразные действия, поскольку инструктивная «обратная связь» не зависит от выбранного действия (таким образом, она в общем-то и не является фактически обратной связью). Поиск в пространстве параметров для системы, обучаемой с учителем, по-прежнему может понадобиться (в форме поиска в пространстве весов нейронной сети), но поиск в пространстве действий не требуется.

Конечно, обучение с учителем обычно применяется в том или ином виде для решения задач, гораздо более сложных, чем задача об *n*-руком бандите. В обучении с учителем не существует единственной ситуации, в которой выполняется некоторое действие, такое обучение оперирует с большим числом разнообразных ситуаций, на каждую из которых требуется правильно отреагировать. Для системы обучения с учителем основной является проблема отображения ситуаций в действия. Такое отображение должно определять правильные действия, отвечающие рассматриваемым воздействиям среды, а также обладать обобщающими свойствами, позволяющими выдавать корректные ответы и для новых ситуаций. От системы, обучающейся с учителем, нельзя требовать, чтобы она *управляла* окружающей средой, поскольку система, скорее, следует получаемой инструктивной информации, чем влияет на нее. Система не пытается подстроить под себя поведение окружающей среды, но вместо этого *сама* старается вести себя так, как предписывает среда.

Достаточно просто различие между оцениванием и инструктированием можно показать, если рассмотреть конкретный случай с одной многократно повторяющейся ситуацией. Пусть имеется 100 возможных действий и из них выбирается действие с номером 32. Оценочная обратная связь назначит за это действие, к примеру, 7,2 балла, тогда как, согласно инструктивной обучающей информации корректным будет не это действие, а некоторое другое, например с номером 67. Ясно, что инструктивная информация намного более содержательна, чем оценочная. Даже если инструктивная информация не вполне точна, она все равно будет более содержательной, чем та, которую дает оценочная обратная связь. Несомненно, *отдельную* инструкцию можно использовать для прямого внесения изменений в правило выбора действия, тогда как

в случае оценочной обратной связи требуется сравнивать ее с таковой для других действий, прежде чем можно будет сделать какие-то выводы о выборе действия¹⁾.

Разница между оценочной обратной связью и инструктивной информацией остается значительной даже в случае, когда существуют только два действия и два возможных вида вознаграждения. Для такого рода *бинарной задачи о бандите* назовем эти два вида вознаграждения *успехом* и *неудачей*. Если достигнут успех, разумно заключить, что было выбрано правильное действие, а в случае неудачи можно сделать вывод, что правильным являлось *небыранное* действие. Можно тогда подсчитать, насколько часто каждое из действий оказывалось правильным (согласно непосредственной оценке или логическому умозаключению), после чего выбрать то действие, которое являлось правильным в большинстве случаев. Назовем такое правило алгоритмом выбора *с учителем*, поскольку этот подход близок по своей идеи к методу обучения с учителем в случае с единственным входным примером. Если вознаграждения детерминированы, то все заключения, получаемые с помощью алгоритма выбора с учителем, правильны, и алгоритм работает прекрасно. Если же вознаграждения стохастические, то картина становится намного более сложной.

В последнем случае конкретная бинарная задача о бандите определяется двумя числами — вероятностями успешного исхода для каждого из возможных действий. Тогда пространство всех возможных задач представляет собой единичный квадрат, как это показано на рис. 2.2. Верхний левый и нижний правый квадранты отвечают сравнительно простым задачам, в которых алгоритм выбора с учителем будет работать хорошо. В этих задачах вероятность успеха для лучшего действия больше 0,5, а для худшего — меньше 0,5. Для таких задач действие,

¹⁾ Различие между инструктированием и оцениванием можно наглядно продемонстрировать, если сопоставить два вида алгоритмов оптимизации функций. Один из этих видов используется, когда непосредственно доступна информация о градиенте минимизируемой (или максимизируемой) функции. Градиент сообщает алгоритму, куда следует двигаться в пространстве поиска. Многие алгоритмы обучения с учителем основаны на использовании градиента (точного или приближенного) ошибки. Другой вид алгоритмов оптимизации использует только значения функции, соответствующие оценочной информации. Такие алгоритмы должны активно исследовать функцию в дополнительных точках пространства поиска, чтобы решить, куда двигаться дальше. Классическими примерами этих двух видов алгоритмов являются соответственно алгоритмы стохастической аппроксимации Робинса—Монро и Кифера—Волфвица — см. (Kashyap, Blaydon and Fu 1970).

считающееся правильным, окажется на самом деле таковым более чем в половине случаев.

Однако бинарные задачи о бандите, относящиеся к оставшимся двум квадрантам на рис. 2.2, оказываются более сложными и не решаются эффективно с помощью алгоритма выбора с учителем. Рассмотрим, к примеру, задачу, соответствующую точке A в левой нижней четверти на рис. 2.2. Для этой точки вероятности успеха действий 1 и 2 равны соответственно 0,2 и 0,1. Поскольку оба этих действия неуспешны по крайней мере в 80% случаев, любой метод, считающий подобную неудачу показателем того, что другое действие было бы правильным, будет совершать непрекращающиеся переходы между двумя рассматриваемыми действиями и никогда не сможет выбрать лучшее из них. Рассмотрим теперь задачу с вероятностями успешности действий, равными соответственно 0,9 и 0,8, отвечающую точке B в правом верхнем квадранте на рис. 2.2. В этом случае оба действия почти всегда оказываются успешными. Любой метод, считающий успех показателем правильности действия, легко может завести процесс выбора в логический тупик, выбрав неправильное действие.

На рис. 2.3 показано поведение в среднем алгоритма выбора с учителем и нескольких других алгоритмов в случае бинарных задач о бандите, соответствующих точкам A и B . Для сравнения здесь показано также поведение ε -жадного метода оценки действий ($\varepsilon = 0,1$), работающего так, как это было описано в разд. 2.2. Для обеих задач алгоритм выбора с учителем обучается находить лучшее действие только чуть больше, чем в половине случаев.

На рис. 2.3 представлены также графики поведения в среднем для двух других алгоритмов, обозначаемых обычно как L_{R-I} и L_{R-P} . Это классические методы из области *обучающихся автоматов*, которые следуют той же логике, что и алгоритм выбора с учителем. Оба метода являются стохастическими, они осуществляют корректировку вероятностей $\pi_t(1)$ и $\pi_t(2)$ выбора каждого из действий. При этом L_{R-P} метод вначале находит правильное действие по тому же принципу, что и алгоритм выбора с учителем, а затем корректирует вероятности следующим



Рис. 2.2. Простые и сложные области в пространстве всех бинарных задач о бандите

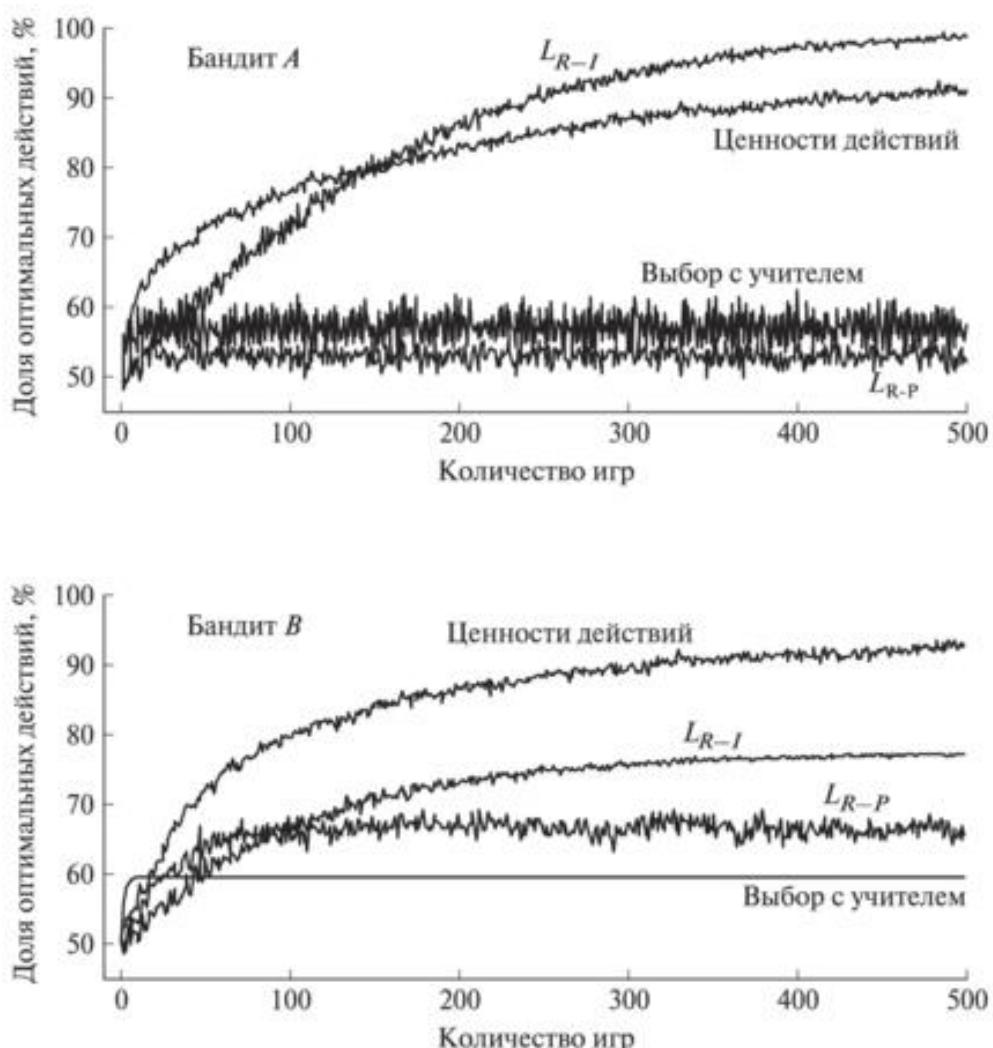


Рис. 2.3. Характеристики некоторых алгоритмов решения бинарных задач о бандите, соответствующих точкам *A* и *B* на рис. 2.2. Данные осреднены по 2000 попыток

образом. Если действие d_t оказалось правильным в игре t , то вероятность $\pi_t(d_t)$ получает приращение α , т. е. новое значение данной вероятности увеличится и станет ближе к единице:

$$\pi_{t+1}(d_t) = \pi_t(d_t) + \alpha[1 - \pi_t(d_t)]. \quad (2.3)$$

Вероятность другого действия изменяется так, чтобы эти две вероятности в сумме давали 1. Для результатов, показанных на рис. 2.3, значение параметра α было принято равным 0,1. Идея, на которой основан L_{R-P} метод, такая же как и для алгоритма выбора с учителем, только этот метод имеет стохастический характер. Вместо того чтобы всегда исполь-

зователь действие, считающееся лучшим, L_{R-P} метод лишь постепенно увеличивает вероятность этого действия¹⁾.

Обозначение L_{R-P} означает «linear, reward-penalty» (линейный, вознаграждение-штраф). Это значит, что выражение (2.3) является линейным и что корректировка значений вероятностей происходит как для успешных игр (вознаграждение), так и для неуспешных (штраф). Обозначение L_{R-I} означает «linear, reward-inaction» (линейный, вознаграждение-бездействие). Этот алгоритм идентичен алгоритму L_{R-P} за исключением того, что значения вероятностей корректируются только в случае успешных игр; неуспешные игры полностью игнорируются. Результаты, приведенные на рис. 2.3, показывают, что алгоритм L_{R-P} в бинарных задачах о бандите, соответствующих точкам A и B на рис. 2.2, если и ведет себя лучше, чем алгоритм поиска с учителем, то совсем ненамного. Что касается алгоритма L_{R-I} , в итоге он показывает себя очень хорошо в A -задачах, но не в B -задачах, обучается же он медленно в обоих случаях.

Бинарные задачи о бандите представляют собой поучительный частный случай, в котором сочетаются аспекты обучения с учителем и обучения с подкреплением. Поскольку вознаграждения в этих задачах бинарны, появляется возможность сделать некоторые выводы о правильном действии, которое приводит к получению однократного вознаграждения. Для ряда задач данного вида такие выводы будут вполне обоснованными и приводят к появлению эффективных алгоритмов. Однако в других задачах этого вида подобные выводы могут оказаться менее пригодными и приведут к ухудшению результата. В задачах о бандите с вознаграждениями, не являющимися бинарными, например для 10-рукой экспериментальной среды, совсем не очевидно, каким образом идеи, лежащие в основе этих выводов, можно применить для выработки эффективного алгоритма. Все это очень простые задачи, но уже на их примере видна потребность в методах, возможности которых выше, чем у методов обучения с учителем.

Упражнение 2.4. Рассмотрим класс простейших задач обучения с учителем, в которых имеется только одна ситуация (входной пример) и два действия. Одно действие, пусть это будет a , правильное, а другое действие, обозначим его b , неправильное. Инструктивный сигнал со-

¹⁾ Это описание на самом деле представляет собой значительно упрощенный вариант алгоритмов обучения автоматов. Например, такие алгоритмы можно определить также и для $n > 2$, они также часто используют разные значения величины параметра α для случаев успеха и неудачи. Тем не менее ограничения, выявленные в этом разделе, остаются в силе.

держит неопределенность: он предписывает выполнение неправильного действия с вероятностью p , т. е. с вероятностью p действие b считается правильным. Проблемы такого рода можно трактовать как бинарные задачи о бандите, если интерпретировать рассматриваемый входной сигнал (возможно, неправильный) как *успешный* в случае согласия с ним, или как *неуспешный* — в случае несогласия. Проанализируйте получившийся класс бинарных задач о бандите. Есть ли что-то особенное в этих задачах? Как алгоритм выбора с учителем работает с этими задачами?

2.5. Пошаговая реализация обучения

Все методы, основанные на ценности действий, рассмотренные до сих пор, определяли эту ценность как среднее выборочное для уже полученных вознаграждений. Очевидный способ реализации такого подхода состоит в том, чтобы для каждого действия a записывать значения всех вознаграждений, которые последуют за выбором данного действия. Тогда при необходимости рассчитать значение ценности действия a в момент времени t его можно найти согласно выражению (2.1), приводимому еще раз ниже:

$$Q_t(a) = \frac{r_1 + r_2 + \dots + r_{ka}}{k_a},$$

где r_1, \dots, r_{ka} — это все вознаграждения, полученные за все случаи выбора действия a , вплоть до момента времени t . Основная проблема, которая возникает при такой прямой реализации расчета ценности действия, состоит в том, что потребности в памяти и других вычислительных ресурсах растут неограниченно с течением времени. Таким образом, при каждом добавлении вознаграждения, полученного за действие a , требуется все больше памяти для его хранения, а также все больше вычислительных ресурсов, чтобы найти значение $Q_t(a)$.

Легко видеть, что на самом деле в этом нет необходимости. Можно достаточно просто получить выражения пошагового типа для вычисления требуемых средних значений. Потребности в вычислительных ресурсах для таких выражений при обработке каждого нового вознаграждения будут совсем невелики и постоянны. Для некоторого действия a обозначим как Q_k среднее значение его k первых вознаграждений (не следует путать это значение с $Q_k(a)$, средним значением для действия a в k -й игре). Получив это среднее значение, а также $(k+1)$ -е вознаграждение, среднее значение для всех $k+1$ вознаграждений можно

вычислить с помощью соотношения:

$$\begin{aligned}
 Q_{k+1} &= \frac{1}{k+1} \sum_{i=1}^{k+1} r_i = \frac{1}{k+1} \left(r_{k+1} + \sum_{i=1}^k r_i \right) = \\
 &= \frac{1}{k+1} (r_{k+1} + kQ_k + Q_k - Q_k) = \\
 &= \frac{1}{k+1} (r_{k+1} + (k+1)Q_k - Q_k) = \\
 &= Q_k + \frac{1}{k+1} [r_{k+1} - Q_k].
 \end{aligned} \tag{2.4}$$

Это выражение остается верным даже и при $k = 0$. В этом случае оно в качестве результата дает $Q_1 = r_1$ для произвольного Q_0 . При такой реализации память необходима только для хранения значений Q_k и k . Требуемые вычисления также невелики по объему и состоят в расчете по формуле (2.4) для каждого нового вознаграждения.

Правило корректировки (2.4) представляет собой выражение, часто встречающееся на страницах этой книги. В общем виде его можно представить следующим образом:

$$\begin{aligned}
 \text{Новая Оценка} &\leftarrow \text{Старая Оценка} + \text{Длина Шага} \times \\
 &\quad \times [\text{Цель} - \text{Старая Оценка}].
 \end{aligned} \tag{2.5}$$

Выражение $[\text{Цель} - \text{Старая Оценка}]$ представляет собой *ошибку* в оценке. Ошибка уменьшается с каждым шагом по направлению к «Цели». Предполагается, что эта цель указывает желаемое направление, по которому следует двигаться, хотя она и может содержать в себе неопределенность. Например, в случае, рассмотренном выше, целью является $(k+1)$ -е вознаграждение.

Отметим, что параметр *Длина Шага* в рассмотренном выше пошаговом методе меняется от одного временного шага к другому. При обработке k -го вознаграждения для действия a этот метод в качестве длины шага использует величину $1/k$. В этой книге длина шага обозначается как α или, в более общем случае, как $\alpha_k(a)$. Например, рассмотренная выше пошаговая реализация метода среднего выборочного описывается соотношением $\alpha_k(a) = 1/k_a$. Иногда используется менее формальная запись $\alpha = 1/k$, в которой зависимость от действия подразумевается неявно.

Упражнение 2.5. Напишите символический код полного алгоритма решения задачи об n -руком бандите. Используйте жадный выбор действия и пошаговое вычисление ценности действий при длине шага

$\alpha = 1/k$. Создайте функцию *бандит(a)*, которая получает на входе действие и возвращает в качестве выхода вознаграждение. Используйте массивы и переменные; не применяйте в качестве индекса временной показатель t . Покажите, каким образом задаются начальные значения ценности действий и как они корректируются после каждого вознаграждения.

2.6. Нестационарные задачи

Рассмотренные до сих пор методы осреднения пригодны для задач со стационарной средой, но их нельзя применять в случае изменяющегося во времени бандита. Как уже отмечалось ранее, часто встречаются существенно нестационарные задачи обучения с подкреплением. В таких случаях более поздним вознаграждениям имеет смысл придавать больший вес, чем более ранним. Наиболее распространенный способ добиться этого состоит в использовании постоянного по величине шага. Например, пошаговое правило корректировки (2.4), с помощью которого вычисляется значение среднего выборочного Q_k для k последних вознаграждений, в этом случае принимает следующий вид:

$$Q_{k+1} = Q_k + \alpha[r_{k+1} - Q_k], \quad (2.6)$$

где размер шага α , $0 < \alpha \leq 1$, принимается постоянным. В результате получаем значение для Q_k , которое будет средневзвешенным для предыдущих вознаграждений и начальной оценки Q_0 :

$$\begin{aligned} Q_k &= Q_{k-1} + \alpha[r_k - Q_{k-1}] = \alpha r_k + (1 - \alpha)Q_{k-1} = \\ &= \alpha r_k + (1 - \alpha)\alpha r_{k-1} + (1 - \alpha)^2 Q_{k-2} = \\ &= \alpha r_k + (1 - \alpha)\alpha r_{k-1} + (1 - \alpha)^2 r_{k-2} + \dots + \\ &\quad + (1 - \alpha)^{k-1} r_1 + (1 - \alpha)^k Q_0 = \\ &= (1 - \alpha)^k Q_0 + \sum_{i=1}^k \alpha(1 - \alpha)^{k-i} r_i. \end{aligned} \quad (2.7)$$

Значение, получаемое с помощью выражения (2.7), будем называть взвешенным средним (средневзвешенным), поскольку для суммы весов, как это легко проверить, имеет место соотношение

$$(1 - \alpha)^k + \sum_{i=1}^k \alpha(1 - \alpha)^{k-i} = 1.$$

Обратите внимание на то, что значение веса $\alpha(1 - \alpha)^{k-i}$ для вознаграждения r_i зависит от величины $k - i$, т. е. от того, сколько шагов назад это

вознаграждение было получено. Величина $1 - \alpha$ меньше 1, следовательно, вес вознаграждения r_i уменьшается с ростом числа полученных вознаграждений. Фактически значение этого веса уменьшается экспоненциально с показателем $1 - \alpha$. Соответственно, такое значение среднего выборочного называют иногда *экспоненциальным средним* или *средне-звешенным с учетом новизны*.

Иногда удобно от шага к шагу изменять размер шага. Обозначим через $\alpha_k(a)$ величину шага, используемую при расчете вознаграждения, полученного после k -го выбора действия a . Как отмечалось, выбор значения шага $\alpha_k(a) = 1/k$ приводит к методу среднего выборочного, который гарантирует сходимость к истинному значению ценности действия согласно закону больших чисел. Однако сходимость не гарантируется, конечно, для всех возможных вариантов последовательности $\{\alpha_k(a)\}$. Условия, которые должны быть выполнены, чтобы обеспечить сходимость с вероятностью 1, можно получить на основе хорошо известного результата из теории стохастической аппроксимации:

$$\sum_{k=1}^{\infty} \alpha_k(a) = \infty \quad \text{и} \quad \sum_{k=1}^{\infty} \alpha_k^2(a) < \infty. \quad (2.8)$$

Выполнение первого из этих условий требуется, чтобы обеспечить достаточную величину шага, позволяющую, в конечном счете, справиться с любыми начальными условиями или случайными флюктуациями. Второе условие гарантирует уменьшение со временем величины шага в степени, достаточной для обеспечения сходимости.

Отметим, что оба условия сходимости удовлетворяются в случае среднего выборочного значения, когда

$$\alpha_k(a) = \frac{1}{k},$$

но не для постоянного размера шага

$$\alpha_k(a) = \alpha.$$

В последнем случае второе условие не удовлетворяется, показывая тем самым, что процесс вычисления оценок не сходится. Значения их будут постоянно меняться в зависимости от очередного полученного вознаграждения. Как указывалось выше, именно это и происходит в нестационарной среде, а существенно нестационарные задачи типичны в обучении с подкреплением. Кроме того, последовательности значений размера шага, удовлетворяющие условиям (2.8), часто сходятся очень медленно или требуют значительной корректировки для получения приемлемой скорости сходимости. И хотя такие последовательности, удо-

влетворяющие упомянутым условиям сходимости, часто используются в работах теоретического характера, на практике и в экспериментальных исследованиях они применяются редко.

Упражнение 2.6. Если значения размера шага $\alpha_k(a)$ непостоянны, то оценка $Q_k(a)$ будет средневзвешенным значением предыдущих вознаграждений с весом, отличным от рассчитанного согласно выражению (2.7). Каким будет вес каждого предыдущего вознаграждения в общем случае?

Упражнение 2.7 [Программирование]. Подготовьте и проведите эксперимент, который демонстрирует трудности, возникающие при применении метода среднего выборочного к нестационарным задачам. Воспользуйтесь модифицированным вариантом 10-рукой испытательной среды, в которой все оценки $Q^*(a)$ изначально имеют одинаковые значения, а в дальнейшем претерпевают независимые случайные изменения. Постройте графики наподобие рис. 2.1 для метода, основанного на ценности действия и использующего средние выборочные значения, вычисляемые пошаговым образом для $\alpha = 1/k$, а также для другого метода этого же типа, в котором используется постоянный размер шага, равный $\alpha = 0,1$. Используйте значение $\varepsilon = 0,1$ и, если в этом возникнет необходимость, можете превысить лимит в 1000 игр.

2.7. Оптимистичные начальные оценки

Все рассмотренные до сих пор методы зависели в некоторой степени от начальных значений оценок ценности действия, т. е. от величины $Q_0(a)$. В терминах языка статистики эти методы оперируют со *смещёнными* начальными оценками. В случае методов среднего выборочного смещение оценок исчезает после того, как каждое из действий выбрано хотя бы один раз, но в случае методов с постоянным шагом α смещение присутствует постоянно, хотя и уменьшается со временем согласно соотношению (2.7). На практике такой вид смещения обычно не вызывает трудностей, а иногда оказывается и очень полезным. Недостаток подобного подхода состоит в том, что требуемые начальные оценки представляют собой фактически некоторый набор параметров, значения которых пользователь либо должен задать, либо приравнять их все к нулю. Преимуществом же здесь является то, что таким образом достаточно просто учитывается имеющееся априорное знание об ожидаемых значениях вознаграждений.

Начальные значения ценности действий могут поощрять к проведению деятельности исследовательского характера. Предположим, что в качестве начальных оценок мы выбрали не нуль, как это было сделано для 10-рукой испытательной среды, а число +5. Напомним, что величины $Q^*(a)$ в этой задаче выбираются с использованием нормального распределения с математическим ожиданием 0 и дисперсией 1. Начальная оценка +5, таким образом, является чрезмерно оптимистичной. Но такой оптимизм стимулирует методы, основанные на ценности действия, предпринимать исследовательские попытки. Какие бы действия ни выбирались на начальном этапе, вознаграждение будет меньше начальных оценок; обучающийся переключается на другие действия, будучи «разочарованным» полученным вознаграждением. В результате все действия будут испробованы по несколько раз, прежде чем сойдетесь процесс вычисления оценок. Система в этом случае будет проводить значительный объем исследований, даже если постоянно выбираются жадные действия.

На рис. 2.4 показаны результаты применения жадного метода к 10-рукой испытательной среде. Здесь использовалась величина $Q_0(a) = +5$ для всех значений a . Для сравнения даются также результаты использования ε -жадного метода при $Q_0(a) = 0$. Оба метода используют постоянный размер шага, равный $\alpha = 0,1$. Вначале оптимистичный метод показывает себя хуже, поскольку он предпринимает больше исследовательских действий, но в дальнейшем его поведение улучшается, потому что объем проводимых им исследований со временем уменьшается. Такой способ стимулирования исследований будем называть мето-

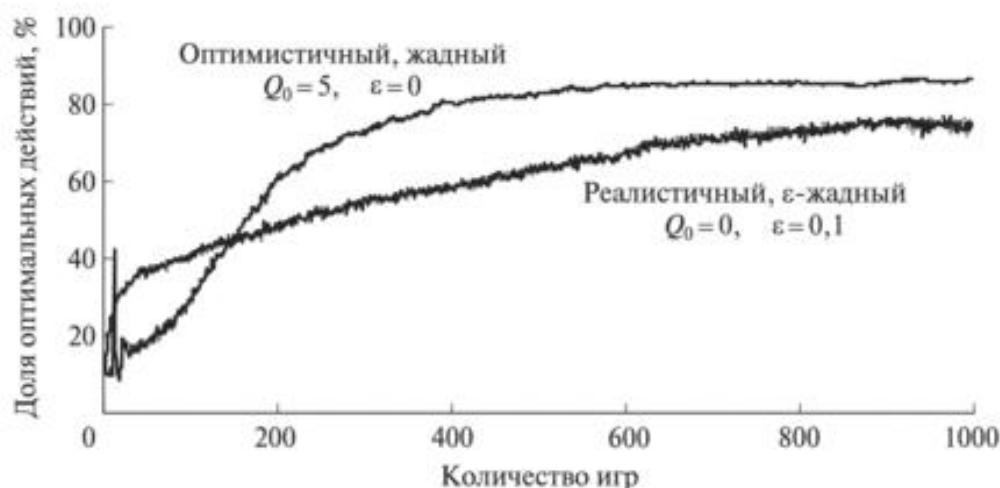


Рис. 2.4. Влияние оптимистичных начальных оценок на результаты решения применительно к 10-рукой испытательной среде

дом оптимистичных начальных оценок. Этот метод представляет собой довольно простой прием, который может оказаться достаточно эффективным в стационарных задачах, но он малопригоден в качестве общего метода стимулирования исследовательских действий. Например, он плохо подходит для решения нестационарных задач, потому что его стремление исследовать носит временный характер. Если задача изменяется, появляется необходимость в исследованиях, но в данной ситуации указанный метод помочь не сможет. На самом деле любой метод, в работе которого повышенное внимание уделяется начальным условиям, едва ли будет пригодным в случае общей нестационарной задачи. Начальный момент времени имеет место только один раз, и по этой причине не следует слишком сильно на нем сосредоточиваться. Все эти критические замечания относятся также и к методам среднего выборочного, в которых начало отсчета времени трактуется как событие специального вида, с использованием одинаковых значений весов при осреднении всех последующих вознаграждений. Однако все эти методы, в силу их простоты, по отдельности или в несложных комбинациях часто применяются на практике. В этой книге далее нередко будут использоваться некоторые из этих простых приемов исследования.

Упражнение 2.8. Результаты, приведенные на рис. 2.4, должны быть достаточно достоверными, так как они являются результатом осреднения для более чем 2000 отдельных, случайным образом выбранных задач о 10-руком бандите. Почему же тогда имеют место такие колебания и пики в начальной части графика функции, полученного для оптимистичного метода? Каким образом можно значительно улучшить или ухудшить получаемые результаты в среднем или только для игр в начальный период времени?

*2.8. Сравнение с подкреплением

Интуитивно ясно, что основой обучения с подкреплением является стремление повысить вероятность выбора действий, за которыми следуют большие вознаграждения, и в то же время понизить вероятность выбора действий, за которыми следуют малые вознаграждения. Но каким образом обучающийся узнает, что считать большим вознаграждением, а что — малым? Если предпринято действие и от среды получено вознаграждение, равное 5, это много или мало? Чтобы получить требуемый ответ, необходимо сравнить вознаграждение с некоторым стандартным или эталонным уровнем, который будем именовать *эталонным вознаграждением*. Естественно принять за эталонное вознаграждение сред-

нее значение вознаграждений, полученных ранее. Другими словами, вознаграждение считается большим, если оно превышает это среднее значение, и малым, если не превышает. Методы обучения, основанные на такой идеи, называются методами *сравнения с подкреплением*. Иногда эти методы более эффективны, чем методы, основанные на ценности действия. Они также являются предшественниками методов, основанных на взаимодействии исполнителя и критика действий. Такого рода методы относятся к классу средств, предназначенных для решения полной задачи обучения с подкреплением, и будут рассмотрены позже.

Методы сравнения с подкреплением обычно имеют дело не с оценками значений ценности действий, а только с оценкой общего уровня вознаграждения. При выборе действия эти методы работают со значениями показателя предпочтения, поставленными в соответствие каждому из действий. Будем обозначать предпочтение для действия a в игре t через $p_t(a)$. Эти предпочтения можно следующим образом использовать для определения вероятности выбора действия в соответствии с softmax-правилом:

$$\pi_t(a) = \frac{e^{p_t(a)}}{\sum_{b=1}^n e^{p_t(b)}}, \quad (2.9)$$

где $\pi_t(a)$ обозначает вероятность выбора действия a в t -й игре. При обновлении значений предпочтений для рассматриваемых действий используется идея сравнения с подкреплением. После каждой игры значение предпочтения для действия, выбранного в данной игре a_t , получает приращение в виде разницы между вознаграждением r_t и эталонным вознаграждением \bar{r}_t :

$$p_{t+1}(a_t) = p_t(a_t) + \beta[r_t - \bar{r}_t], \quad (2.10)$$

где β — положительный размер шага. Это соотношение реализует идею, согласно которой высокие вознаграждения должны увеличивать вероятность повторного выбора предпринятого действия, а низкие — уменьшать эту вероятность.

Эталонное вознаграждение является постепенно возрастающим средним значением *всех* ранее полученных вознаграждений, независимо от того, какие действия выбирались. После изменения значений предпочтений, согласно формуле (2.10), эталонное вознаграждение обновляется:

$$\bar{r}_{t+1} = \bar{r}_t + \alpha[r_t - \bar{r}_t], \quad (2.11)$$

где α , $0 < \alpha \leq 1$, как обычно, размер шага. Начальное значение эталонного вознаграждения r_0 можно выбрать либо оптимистично, что-

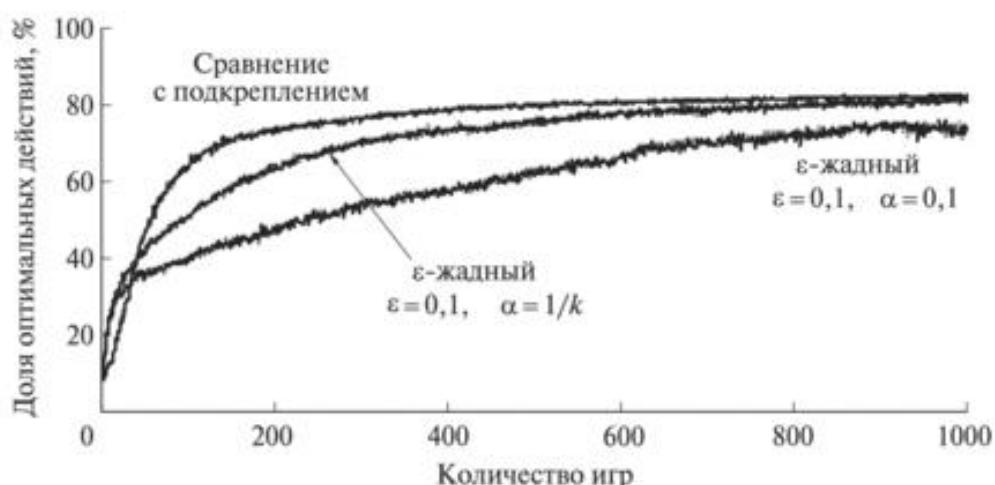


Рис. 2.5. Сопоставление методов сравнения с подкреплением и методов, основанных на ценности действия, применительно к 10-рукой испытательной среде

бы стимулировать выполнение изучающих действий, либо основываясь на уже имеющихся знаниях. Все начальные значения предпочтений для действий можно принять равными нулю. Постоянный шаг α в рассматриваемом случае является хорошим выбором, так как распределение вознаграждений меняется во времени по мере того, как улучшается выбор действий. Это первый встретившийся нам случай, когда изучаемая задача является существенно нестационарной, несмотря даже на то, что исходная проблема имеет стационарный характер.

Методы сравнения с подкреплением могут быть очень эффективными. Иногда они могут работать даже лучше, чем методы, основанные на ценности действия. На рис. 2.5 показано, как ведет себя рассмотренный выше алгоритм (при $\alpha = 0,1$) в случае с 10-рукой испытательной средой. Для сравнения здесь показано также поведение ϵ -жадного ($\epsilon = 0,1$) метода, основанного на ценности действия при $\alpha = 0,1$ и $\alpha = 1/k$.

Упражнение 2.9. В softmax-правиле выбора действий (2.9), предложенном для методов сравнения с подкреплением, исключен температурный параметр τ , входивший в выражение (2.2). Как вы считаете, почему это было сделано? Не была ли утеряна в определенной степени гибкость метода из-за исключения параметра τ ?

Упражнение 2.10*. В описанных здесь методах сравнения с подкреплением используются два различных размера шага, α и β . Можно ли в общем случае сократить их число до одного, если принять $\alpha = \beta$? Что будет утрачено при таком сокращении?

Упражнение 2.11* [Программирование]. Примем, что начальное эталонное вознаграждение \bar{r} очень мало по величине. Какое бы действие

ни было выбрано первоначально, вероятность его последующего выбора увеличится. Соответственно, это действие, скорее всего, будет выбрано опять, и снова вероятность его последующего выбора возрастет. Таким образом, начальное действие, которое не лучше по сравнению с какими-либо другими, может надолго вытеснить все остальные действия. Чтобы не допустить такого эффекта, обычно к приращению (2.10) добавляют коэффициент $(1 - \pi_t(a_t))$. Подготовьте и проведите эксперимент, позволяющий установить, действительно ли эта мера позволит улучшить поведение рассматриваемого алгоритма.

*2.9. Методы преследования

Еще одним классом эффективных методов обучения для задачи об n -руком бандите являются методы *преследования*. Методы преследования работают как с оценками ценности действий, *так и с предпочтениями действий*, непрерывно «преследуя» то действие, которое является жаждым согласно текущим оценкам ценности действий. В простейшем варианте метода преследования предпочтение $\pi_t(a)$ действия a представляет собой вероятность, с которой действие a выбирается в игре t .

После каждой игры указанные вероятности корректируются таким образом, чтобы повысить возможность выбора жаждого действия. Предположим, что t -я игра завершена и обозначим

$$a_{t+1}^* = \arg \max_a Q_{t+1}(a)$$

жадное действие (или одно из жадных действий, выбранное случайно, если их больше одного) для $(t + 1)$ -й игры. Тогда вероятность выбора действия

$$a_{t+1} = a_{t+1}^*$$

для $(t + 1)$ -й игры возрастает на величину, равную некоторой доле, задаваемой коэффициентом β , от разности между 1 и вероятностью выбора этого же действия для t -й игры:

$$\pi_{t+1}(a_{t+1}^*) = \pi_t(a_{t+1}^*) + \beta[1 - \pi_t(a_{t+1}^*)], \quad (2.12)$$

в то время как вероятности выбора других действий аналогичным образом уменьшаются:

$$\pi_{t+1}(a) = \pi_t(a) + \beta[0 - \pi_t(a)] \quad \text{при всех } a \neq a_{t+1}^*. \quad (2.13)$$

Ценности действий $Q_{t+1}(a)$ вычисляются одним из способов, рассмотренных в предыдущих разделах, например, как среднее выборочное значений полученных вознаграждений, согласно соотношению (2.1).

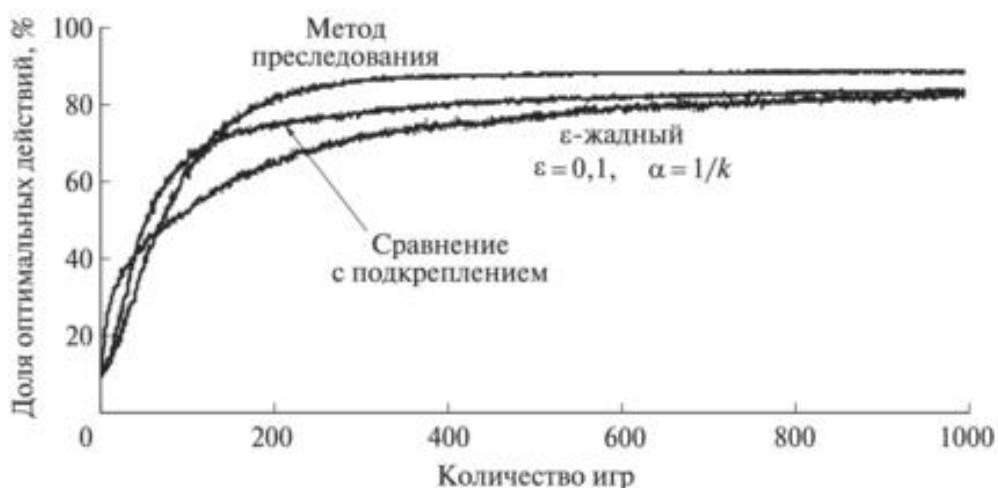


Рис. 2.6. Характеристики метода преследования в сопоставлении с методами сравнения с подкреплением и методами, основанными на ценности действия, применительно к 10-рукой испытательной среде

На рис. 2.6 показано, как работает рассмотренный выше алгоритм преследования, если ценность действия определяется как среднее выборочное, вычисляемое с шагом $\alpha = 1/k$. Эти результаты получены при начальных значениях вероятностей выбора действий $\pi_0(a)$, равных $1/n$ для всех a , а значение параметра β было принято равным 0,01. Для сравнения здесь же показано поведение ε -жадного метода (при $\varepsilon = 0,1$), в котором значение ценности действий также определялось как среднее выборочное. Кроме того, представлено поведение метода сравнения с подкреплением из предыдущего раздела. Хотя в данной задаче при данных значениях параметров алгоритм преследования оказался лучше остальных двух алгоритмов, в других случаях результат вполне может оказаться и не таким. Все три этих метода имеют свои преимущества и свои области применения.

Упражнение 2.12. ε -жадный метод всегда выбирает некоторое случайное действие для части выполняемых шагов. А как с этим обстоит дело в методе преследования? Будет ли он в конечном счете выбирать оптимальное действие с вероятностью, приближающейся к 1?

Упражнение 2.13. Для многих задач, с которыми предстоит еще встретиться в этой книге, нет возможности непосредственно корректировать значения вероятностей действий. Чтобы в этих случаях можно было использовать методы преследования, необходимо изменить их таким образом, чтобы они использовали предпочтения действий, не являющиеся вероятностями. Они должны определять вероятности действий согласно softmax-правилу, такому как распределение Гиббса (2.9). Как

можно изменить алгоритм преследования, рассмотренный выше, чтобы его можно было использовать подобным образом? Опишите такой алгоритм детально, включая соотношения для ценностей действий, предпочтений и вероятностей для каждой игры.

Упражнение 2.14 [Программирование]. Как показал себя алгоритм, построенный при выполнении упражнения 2.13? Подготовьте и проведите эксперимент, позволяющий оценить работоспособность этого алгоритма. Проанализируйте, каким образом влияют значения параметров данного алгоритма на результаты эксперимента.

Упражнение 2.15. Алгоритм преследования, рассмотренный выше, подходит только для случая стационарной среды, поскольку вероятности действий в нем, хотя и медленно, но стремятся к вероятности достоверного события. Как можно было бы объединить идею преследования с идеей ε -жадности, чтобы получить метод, который был бы по своему поведению близким к алгоритму преследования, но при этом всегда, хотя и в небольшой степени, предпринимал бы изучающие действия?

*2.10. Ассоциативный поиск

До сих пор в этой главе рассматривались только неассоциативные задачи, в которых не возникало необходимости ставить в соответствие различные действия отличающимся ситуациям. В таких задачах обучающийся либо делает попытку найти единственное наилучшее действие, если задача стационарная, либо пытается отслеживать меняющееся во времени наилучшее действие, когда задача нестационарная. Однако в задаче обучения с подкреплением общего вида ситуация не единственна, и целью в такой задаче является формирование некоторой стратегии поведения, т. е. некоторого отображения ситуаций в действия, наилучшего для имеющейся совокупности ситуаций. В качестве подготовительного этапа перед рассмотрением полной задачи рассмотрим кратко простейший способ, с помощью которого неассоциативные задачи можно расширить так, чтобы ввести в них настройки ассоциативного характера.

Пусть, например, есть несколько различных задач об n -руком бандите. При этом в каждой игре случайным образом выбирается одна из этих задач. Другими словами, задача о бандите меняется от игры к игре по некоторой случайной схеме. Можно было бы попытаться применить один из методов, рассмотренных в этой главе, обеспечивающих решение нестационарных задач, но за исключением случаев, когда ис-

тические значения ценностей действий меняются медленно, эти методы будут работать не слишком хорошо. Предположим, однако, что, когда некоторая задача о бандите уже выбрана, доступна некоторая информация относительно ее особенностей (но не о значениях ценности действий). Пусть, например, выполняются действия с реальным игровым автоматом, у которого с изменением ценности действия меняется цвет экрана. Тогда можно найти стратегию, отвечающую каждой из задач, основываясь при этом на видимом цвете экрана автомата. Вследствие этого возникает возможность при появлении определенной задачи выбрать отвечающее ей наилучшее действие. Например, если цвет экрана красный, использовать рычаг 1, если зеленый — рычаг 2. Если следовать правильной стратегии, можно обычно получить результат, который будет намного лучше, чем если выбирать действия в отсутствие какой-либо информации, отличающей одну задачу о бандите от другой.

Это пример задачи *ассоциативного поиска*, названной так потому, что она включает в себя как обучение методом проб и ошибок в форме *поиска наилучших действий*, так и установление связей (*ассоциирование*) этих действий с ситуациями, в которых данные действия являются наилучшими. Задачи ассоциативного поиска занимают промежуточное положение между задачей об n -руком бандите и полной задачей обучения с подкреплением. В них, как и в полной задаче обучения с подкреплением, осуществляется поиск стратегии, но в то же время, как в рассмотренном варианте задачи об n -руком бандите, предпринимаемое действие влияет только на немедленно получаемое вознаграждение. Если действия могут влиять на *следующую ситуацию*, как и на вознаграждение, то имеет место полная задача обучения с подкреплением. Данная задача будет представлена в следующей главе, а ответвления от нее будут рассмотрены в оставшейся части этой книги.

Упражнение 2.16. Пусть требуется решить бинарную задачу о бандите, в которой истинные ценности действий меняются случайным образом от игры к игре. Примем, что для любой игры истинные ценности действий 1 и 2 равны соответственно 0,1 и 0,2 с вероятностью 0,5 (случай А) или 0,9 и 0,8 с вероятностью 0,5 (случай В). Если нет возможности определить, какой случай имеет место в каждой из игр, какого наилучшего результата можно ожидать и как необходимо действовать, чтобы его добиться? Предположим теперь, что для каждой игры становится известно, что это за случай, А или В (хотя истинные значения ценностей действий по-прежнему не известны). Данная задача относится к классу

су задач ассоциативного поиска. Какого наилучшего результата можно ожидать в этой задаче и как необходимо действовать для его получения?

2.11. Итоги

В этой главе были представлены некоторые простые способы достижения баланса между изучением и применением. При этом ϵ -жадные методы расходуют на исследовательские действия, реализуемые случайным образом, некоторую малую часть общего времени решения задачи, softmax-методы варьируют вероятности выполняемых ими действий в зависимости от текущих оценок действий, а методы преследования постоянно совершают шаги в направлении текущего жадного действия. Действительно ли эти методы являются наилучшими с точки зрения построения алгоритмов, полезных с практической точки зрения? Судя по всему, на этот вопрос следует дать положительный ответ. Несмотря на свою простоту, представленные в этой главе методы вполне отвечают, на наш взгляд, современному уровню развития рассматриваемой области. Существуют и более изощренные методы, но из-за их сложности, а также в силу постулатов, принятых при их построении, они непрактичны в случае полной задачи обучения с подкреплением, которая наиболее интересна для нас. Начиная с гл. 5 будут рассматриваться методы обучения, предназначенные для решения полной задачи обучения с подкреплением, в которых частично будут использованы и простые методы, изучавшиеся в этой главе.

Хотя те простые методы, что исследованы в настоящей главе, и являются, наверное, наилучшими на сегодня, они далеко не обеспечивают по-настоящему удовлетворительного решения проблемы баланса между изучением и применением. Завершая эту главу, дадим краткий обзор некоторых из разрабатываемых сейчас в данной области идей. Их пока еще нельзя назвать практически полезными, но они могут указать путь получения хороших решений.

Одна из многообещающих идей состоит в использовании оценок неточности предположительных значений ценности действий, чтобы направлять и стимулировать процесс изучения. Пусть, например, имеются два действия, предполагаемые ценности которых лишь немного меньше ценности жадного действия, но они значительно различаются по степени неопределенности. Одна из оценок близка к достоверности, соответствующее действие, возможно, уже неоднократно проверялось, в результате было получено много вознаграждений. Неточность этого предположительного значения ценности действия настолько мала, что его истин-

ное значение с большой вероятностью не будет выше ценности жадного действия. Другое действие изучено не так хорошо, и оценка значения его ценности содержит значительную неопределенность. Истинная ценность этого действия вполне может оказаться выше ценности жадного действия. Очевидно, будет больше смысла в исследовании второго действия, чем первого.

Рассуждения такого рода приводят к методам *интервального оценивания*. Эти методы для каждого действия дают оценку доверительного интервала значений его ценности. В таком случае результатом обучения будет, например, утверждение, что значение ценности действия находится между 9 и 11 с уверенностью, к примеру, 95%, вместо утверждения о том, что ценность действия равна приблизительно 10. Выбор действия здесь осуществляется на основе полученных доверительных интервалов: выбрано будет то действие, для которого доверительный интервал имеет наибольшую верхнюю границу. Это стимулирует изучение таких действий, которые определены не полностью и, *кроме того*, имеют шанс в итоге оказаться наилучшими. В некоторых случаях можно гарантировать, что оптимальное действие найдено с уверенностью, численно равной доверительной вероятности (т. е. 95%). К сожалению, методы интервального оценивания для практических применений малопригодны из-за сложности статистических средств, используемых при нахождении доверительных интервалов. Кроме того, зачастую не удовлетворяются и основные статистические допущения, на базе которых построены данные методы. Тем не менее сама идея использования доверительных интервалов или какой-либо другой меры неопределенности для стимулирования изучения отдельных действий представляется здравой и привлекательной.

Существует также хорошо известный байесовский алгоритм вычисления оптимального баланса между изучением и применением. В точной трактовке вычислительная реализация этого метода очень сложна, однако можно найти эффективные варианты его приближенного представления. В данном методе предполагается, что известно вероятностное распределение для этапов решения задачи, т. е. вероятности для всех возможных наборов истинных значений ценности действий. После того как выбрано некоторое действие, можно вычислить вероятность каждого возможного немедленного вознаграждения, а также получающееся в результате апостериорное распределение вероятностей для значений ценности действий. Это распределение, эволюционирующее во времени, становится *информационным состоянием* задачи. Пусть рассматривается, например, совокупность из тысячи игр, для которых можно рас-

смотреть все возможные действия, все возможные порождаемые ими вознаграждения, все возможные последующие действия, все последующие вознаграждения и т. д. для всей этой тысячи игр. Зная принятые допущения, можно найти вознаграждения и вероятности для любой из возможных цепочек событий, после чего останется только выбрать лучшую из них. Но дерево возможностей растет чрезвычайно быстро; даже при наличии всего лишь двух действий и двух вознаграждений дерево будет иметь 2^{2000} ветвей. Такой подход фактически превращает задачу о бандите в пример полной задачи обучения с подкреплением. В конечном счете, по-видимому, появится возможность использования методов обучения с подкреплением для аппроксимации этого оптимального решения. Это, однако, составляет предмет исследований, пока еще не завершенных, а тема их выходит за рамки данной книги, имеющей вводный характер.

Классическое решение проблемы достижения баланса между изучением и применением в задаче об n -руком бандите основывается на вычислении функций специального вида, называемых *показателями Гиттингса*. Они позволяют получить оптимальное решение для задач о бандите определенного вида, более общего, чем рассмотренные выше, однако в этом случае требуется знать априорное распределение возможных задач. К сожалению, нет ни теоретических, ни вычислительных результатов, позволяющих распространить данный метод на полную задачу обучения с подкреплением, рассматриваемую в оставшейся части книги.

2.12. Библиографические и исторические справки

- 2.1. Задача о бандите уже достаточно давно изучается в статистике, технических науках и психологии. В статистике задача о бандите возникла под наименованием «последовательный план экспериментов». Она рассматривалась первоначально в работах (Thompson 1933, 1934) и (Robbins 1952), а затем дополнительно изучалась в работе (Bellman 1956). В книге (Berry, Fristedt 1985) задачи о бандите подвергаются всестороннему рассмотрению с точки зрения статистики. Книга (Narendna, Thathachar 1989) посвящена рассмотрению задачи о бандите с инженерной точки зрения, с детальным изучением различных теоретических подходов, связанных с этими задачами. В психологии задачи о бандите играют заметную роль в статистических теориях обучения, см. работы (Bush, Mosteller 1955) и (Estes 1950). Термин *жадный* часто используется в литературе по эвристическому поиску, см., например, (Pearl 1984). Конфликт между изучением и применением известен в теории автоматического управления как противоречие между

идентификацией (или оцениванием) и управлением, см., например, (Witten 1976). В книге (Feldbaum 1965) эта проблема названа проблемой *дальнего управления*, что указывает на необходимость решать одновременно как задачу идентификации, так и задачу управления в случаях, когда предпринимаются попытки управлять системой в условиях неопределенности. В книге (Holland 1975), посвященной рассмотрению различных аспектов генетических алгоритмов, подчеркивается значимость упомянутого конфликта, который интерпретируется как противоречие между необходимостью применения уже имеющегося знания и необходимостью получать новую информацию.

- 2.2. Методы, основанные на ценности действий для рассматриваемой задачи об *n*-румбом бандите, были впервые предложены в статье (Thathachar, Sastry 1985). В литературе по обучающимся автоматам их часто называют *алгоритмами оценивания*. Термин *ценность действия* обязан своим появлением диссертации (Watkins 1989), но идея, выражаемая им, настолько проста, что вполне возможно это понятие уже применялась и ранее.
- 2.3. Операция softmax, которая используется для реализации правила выбора действия (2.2), вводится в работе (Bridle 1990). Это правило впервые было предложено в книге (Luce 1959). Параметр τ называется температурой в алгоритмах имитационного отжига (Kirkpatrick, Gelatt, Vecchi 1983).
- 2.4. Основная идея и результаты этого раздела впервые были представлены в диссертации (Sutton 1984). Дальнейший анализ взаимосвязей между оцениванием и инструктированием дается в работах (Barto 1985, 1991, 1992) и (Barto, Anandan 1985). Представление бинарной задачи о бандите в виде единичного квадрата, показанное на рис. 2.2, в экспериментальной психологии было названо пространством сопряженности, см., например, книгу (Staddon 1983). Книга (Narendra, Thathachar 1989) содержит результаты всестороннего рассмотрения современной теории обучающихся автоматов, а также их применений. Кроме того, в этой книге рассматриваются сходные алгоритмы из статистической теории обучения, развивающейся в психологии. Другие методы, основанные на использовании опыта, получаемого в ходе обучения с подкреплением, для выработки действий, направленных на достижение требуемых целей, изучались в работах (Widrow, Gupta, Maitra 1973) и (Gallmo, Asplund 1995).
- 2.5–6. Материал этих разделов основан на подходах, известных под общим наименованием «стохастические итерационные алгоритмы». Они довольно подробно рассмотрены в книге (Bertsekas, Tsitsiklis 1996).
- 2.7. Методы сравнения с подкреплением были в значительной степени разработаны в диссертации (Sutton 1984) и впоследствии усовершенствованы в статьях (Williams 1986, 1992; Kaelbling 1993a; Dayan 1991). В этих работах проанализировано большое число различных вариантов данной идеи, в том числе и для других условий приемлемости, что дает возможность значительно улучшить получаемые результаты. По-видимому, впервые сравне-

ние с подкреплением было использовано в работе (Barto, Sutton, Brouwer 1981).

- 2.8. Алгоритм преследования обязан своим появлением работе (Thathachar, Sastry 1985).
- 2.9. Термин *ассоциативный поиск* и соответствующая задача были введены в статье (Barto, Sutton, Brouwer 1981). Термин *ассоциативное обучение с подкреплением* применялся также в отношении ассоциативного поиска (Barto, Anandan 1985), однако авторы данной книги предпочитают закрепить его в качестве синонима для полной задачи обучения с подкреплением, как это было принято в работе (Sutton 1984). Отметим, что закон влияния Торндайка (упоминавшийся в гл. 1) описывает ассоциативный поиск, опираясь на формирование ассоциативных связей между ситуациями (состояниями) и действиями. Согласно терминологии инструментального (оперантного) условного рефлекса, см., например, (Skinner 1938), различительным является стимул, который сигнализирует о присутствии конкретной подкрепляющей непредвиденной ситуации. Другими словами, несходные различительные стимулы соответствуют несходным состояниям.
- 2.10. Методы интервального оценивания обязаны своим появлением работам (Lai 1987) и (Kaelbling 1993a). В статье Беллмана (Bellman 1956) впервые было показано, как динамическое программирование можно использовать для вычисления оптимального баланса между изучением и применением в рамках байесовской формулировки задачи. Обзор (Китмаг 1985) породил плодотворную дискуссию относительно байесовских и небайесовских подходов к решению таких задач. Термин *информационное состояние* заимствован из литературы о частично наблюдаемых марковских процессах принятия решений; см., например, (Lovejoy 1991). Подход на основе показателя Гиттинса был предложен в статье (Gittins, Jones 1974). В работе (Duff 1995) было продемонстрировано, как можно находить значения показателей Гиттинса в задачах о бандите посредством обучения с подкреплением.
-

3 Задача обучения с подкреплением

В этой главе происходит знакомство с задачей, решению которой посвящена оставшаяся часть этой книги. Для нас данная задача определяет область обучения с подкреплением: любой метод, подходящий для решения этой задачи, считается методом обучения с подкреплением. Цель данной главы — дать общее описание задачи обучения с подкреплением. Будет рассмотрена обширная область возможных прикладных проблем, которые могут считаться задачами обучения с подкреплением. Кроме того, мы опишем математически идеализированную модель задачи обучения с подкреплением, для которой можно дать точные теоретические формулировки. Мы познакомимся с ключевыми элементами математической структуры задачи, такими как функции ценности и уравнения Беллмана. Как и во всем, что связано с искусственным интеллектом, существуют некоторые расхождения между широтой области применений и возможностями имеющегося математического аппарата. В этой главе рассматриваются такие расхождения и обсуждаются некоторые связанные с ними проблемы и компромиссные решения.

3.1. Взаимосвязь агент — окружающая среда

Задача обучения с подкреплением является дальнейшим развитием задачи обучения на основе взаимодействия для достижения поставленной цели. Обучающийся и принимающий решения здесь называется *агентом*. Совокупность всех объектов, находящихся вне агента, с которыми этот агент взаимодействует, обозначается термином *окружающая среда*. Такое взаимодействие происходит постоянно: агент выбирает действия, окружающая среда реагирует на них, создавая новые ситуации для агента¹⁾. Окружающая среда также является источником вознаграждений — особых численных значений, которые агент постоянно пытается увели-

¹⁾ Будем использовать термины *агент*, *окружающая среда* и *действие* вместо технических терминов *регулятор*, *регулируемая система* (или *объект*) и *управляющий сигнал*, поскольку они понятны более широкому кругу читателей.



Рис. 3.1. Взаимосвязь агент — окружающая среда в обучении с подкреплением

чить. Полное описание характеристик окружающей среды определяет *задание*, один конкретный вариант задачи обучения с подкреплением.

Более конкретно, агент и окружающая среда взаимодействуют на каждом из дискретных временных шагов, составляющих последовательность $t = 0, 1, 2, 3, \dots$ ¹⁾. На каждом временном шаге t агент получает некоторое описание *состояния* окружающей среды $s_t \in S$, где S — множество возможных состояний, и на основании этого описания выбирает *действие* $a_t \in A(s_t)$, где $A(s_t)$ — множество действий, возможных в состоянии s_t . На следующем временном шаге, отчасти как результат предпринятого действия, агент получает числовое *вознаграждение* $r_{t+1} \in \mathbb{R}$ и оказывается в новом состоянии s_{t+1} ²⁾. Рисунок 3.1 иллюстрирует взаимодействие агент — окружающая среда.

На каждом временном шаге агент осуществляет отображение из множества состояний на множество вероятностей выбора каждого из возможных действий. Это отображение называется *стратегией* агента и обозначается π_t , где $\pi_t(s, a)$ — вероятность того, что $a_t = a$, если $s_t = s$. Методы обучения с подкреплением определяют, как агент меняет свою стратегию в зависимости от имеющегося опыта. Упрощенно говоря, целью агента является максимизация суммы всех вознаграждений, которую он получит в долгосрочной перспективе.

¹⁾ Для простоты мы ограничиваемся рассмотрением дискретного времени, тем не менее, многие из рассматриваемых идей могут быть распространены и на случаи с непрерывным временем (см. Bertsekas and Tsitsiklis, 1996; Werbos, 1992; Doya, 1996).

²⁾ Мы используем r_{t+1} вместо r_t для обозначения вознаграждения соответствующего действию, выполненному в момент времени t , чтобы подчеркнуть, что следующее вознаграждение и следующее состояние s_{t+1} определены совместно.

Эта достаточно общая и гибкая структура может быть использована для решения самых разнообразных задач различными способами. Например, временные шаги не обязательно должны соответствовать фиксированным интервалам реального времени, они могут отвечать произвольным последовательным этапам принятия решений и выполнения действий. Действия могут быть низкоуровневыми управлениями, такими как подача нужного напряжения на электродвигатели руки-манипулятора робота, или могут представлять собой высокоуровневые решения, такие как пойти на обед или нет, поступать в магистратуру или нет. Точно так же состояния могут быть представлены в разнообразных формах. Они могут полностью определяться низкоуровневыми сенсорными данными, получаемыми непосредственно от датчиков, или же они могут быть высокоуровневыми и абстрактными, например символным описанием предметов в комнате. Частично то, что определяет состояние, может основываться на ощущениях, полученных в прошлом, или же полностью быть результатом мыслительного процесса и иметь субъективный характер. Например, агент может быть в «состоянии» неуверенности относительно месторасположения объекта, или в состоянии «удивления», в некотором точно определенном смысле. Аналогично, некоторые действия могут полностью являться результатом мыслительного процесса или математического расчета. Например, некоторые действия могут определять выбор агента — о чем ему подумать или на чем сфокусировать свое внимание. В общем, действиями могут быть любые решения, которые необходимо научиться принимать, а состояниями могут быть любые знания, которые помогают принимать решения.

Строго говоря, граница между агентом и окружающей средой не всегда совпадает с физической границей тела робота или живого организма. Обычно эта граница находится ближе к агенту, чем физическая граница его тела. Например, двигатели и механические сочленения робота, а также его сенсорные устройства, скорее, должны рассматриваться как объекты окружающей среды, нежели как части агента. Если рассматривать с этой точки зрения человека или животное, мышцы, скелет и органы восприятия тоже следует считать частью окружающей среды. Вознаграждения рассчитываются обычно внутри живого тела или искусственной обучающейся системы, но также считаются внешними по отношению к агенту.

Будем придерживаться следующего общего правила: все то, что не может быть произвольно изменено агентом, считается находящимся вне агента и, таким образом, является частью окружающей среды. Мы не будем исходить из того, что агенту ничего не известно об окружаю-

щей среде. В частности, агент обычно довольно много знает о том, каким образом рассчитываются его вознаграждения, являющиеся функциями от его действий и состояний, в которых эти действия осуществляются. Однако процесс расчета вознаграждений всегда будет считаться внешним по отношению к агенту, поскольку этот процесс определяет задания агента, и по этой причине агент не должен иметь возможность произвольно менять его. Фактически в некоторых случаях агент может знать все о функционировании окружающей среды и, тем не менее, выполнение задания на обучение с подкреплением окажется для него сложным, подобно тому, как мы можем точно знать, как устроен кубик Рубика, но не можем решить эту головоломку. Граница агент — окружающая среда показывает предел *абсолютного контроля* агента, но не его знаний.

Граница агент — окружающая среда может располагаться в разных местах для разных целей. В сложном роботе одновременно может функционировать много разных агентов, каждый со своей собственной границей. Например, один агент может принимать высокоуровневые решения, влияющие на формирование состояний. С этими состояниями имеет дело низкоуровневый агент, обеспечивающий выполнение высокоуровневых решений. На практике граница агент — окружающая среда определяется выбором конкретных состояний, действий и вознаграждений, поскольку это задает требуемую задачу принятия решений.

В основе обучения с подкреплением лежит существенно обобщенная задача целенаправленного обучения на основе взаимодействия. Это предполагает, что какими бы ни были сенсорные, запоминающие и управляющие устройства, какие бы цели ни ставились, любая задача целенаправленного обучения поведению может быть сведена к трем сигналам, которыми обмениваются агент и окружающая среда: один сигнал отражает выбор, который делает агент (действие), еще один сигнал содержит основание для сделанного выбора (состояние), третий сигнал определяет цель агента (вознаграждение). Возможно, эта модель и не достаточна для решения всех задач обучения, но она доказала свою пригодность к широкому практическому применению.

Конечно, конкретные состояния и действия сильно отличаются в различных прикладных проблемах, и то, как они представлены, может существенно влиять на получаемые результаты. В обучении с подкреплением, как и в других видах обучения, пока что выбор подходящего способа представления является, скорее, искусством, чем наукой. В этой книге даются некоторые советы и приводятся примеры хорошего выбора представления состояний и действий, однако основное внимание уделяется общим принципам обучения и тому, что и как делать, когда способы представления уже выбраны.

Пример 3.1 (Биореактор). Предположим, что обучение с подкреплением применяется для нахождения в заданные моменты времени температур и скоростей перемешивания для биореактора (большого бака с питательными веществами и бактериями, используемого для получения химикатов определенного вида). В этом случае действиями могут быть требуемые температуры и скорости перемешивания, значения которых передаются низкоуровневой системе управления, которая, в свою очередь, напрямую действует нагревательные элементы и электродвигатели для получения нужных значений указанных параметров. Состояниями могут быть данные от термопар и других сенсорных элементов, возможно, подвергнутые фильтрации и сохраненные в памяти, в сочетании с входными сигналами, в которых содержится информация об ингредиентах в баке и получаемых химикатах. Вознаграждениями в заданные моменты могут быть показатели скорости, с которой необходимые химикаты производятся биореактором. Отметим, что каждое состояние здесь — это список или вектор данных, состоящих из показаний сенсоров и значений входных сигналов, а каждое действие — это вектор, содержащий требуемую температуру и скорость перемешивания. Такое структуризованное представление состояний и действий типично для задач обучения с подкреплением. Однако вознаграждения — это всегда просто числа.

Пример 3.2 (Робот, перекладывающий предметы). Рассмотрим использование обучения с подкреплением для управления движением руки-манипулятора робота при выполнении повторяющихся заданий, состоящих в перекладывании предметов с одного места на другое. Если требуется научиться быстрым и плавным движениям, обучающемуся агенту необходимо напрямую управлять электродвигателями и без промедления получать информацию о текущих положениях и скоростях механических звеньев манипулятора. Действиями в этом случае могут быть напряжения, подаваемые на каждый двигатель в каждом из сочленений, а состояниями могут быть данные об углах и скоростях в сочленениях. Вознаграждением может быть $+1$ за каждый объект, успешно переложенный с места на место. Для поощрения плавных движений на каждом временном шаге может налагаться небольшой штраф (отрицательное вознаграждение) как функция «неравномерности» движения в заданные моменты времени.

Пример 3.3 (Робот, собирающий банки). Мобильный робот собирает пустые жестяные банки от напитков в офисе. У робота есть сенсоры для обнаружения таких банок, рука и захватывающее устройство, кото-

рое может их подбирать и помещать во встроенный накопитель; робот питается от подзаряжаемой батареи. Система управления робота имеет компоненты для обработки информации от сенсоров, а также компоненты, обеспечивающие навигацию и управление рукой и захватывающим устройством. Высокоуровневые решения о том, как искать жестяные банки, принимаются агентом обучения с подкреплением на основании текущего уровня заряда батареи. Агент решает, должен ли робот

- (1) активно искать банки в течение определенного периода времени,
- (2) оставаться неподвижным и ждать, пока кто-нибудь принесет банку, или

(3) направиться обратно на базу для зарядки батареи.

Эти решения должны приниматься или периодически, или по мере наступления определенных событий, таких как обнаружение пустой банки. Таким образом, у агента есть три действия и его состояние определяется состоянием батареи. Большую часть времени вознаграждения могут быть нулевыми и принимать положительные значения, когда робот помещает в себя пустую банку, или принимать большие отрицательные значения, если батарея полностью разрядилась. В этом примере не весь робот является агентом обучения с подкреплением. Состояния, за которыми он ведет наблюдение, описывают условия внутри самого робота, а не условия окружающей робота среды. Окружающая среда агента, таким образом, включает в себя часть робота, которая, как и окружающая среда робота, может содержать другие сложные системы принятия решений.

Упражнение 3.1. Придумайте самостоятельно примеры заданий, отвечающих сути обучения с подкреплением. Для каждого задания определите состояния, действия и вознаграждения. Сделайте эти примеры настолько *разными*, насколько это возможно. Подход на основе обучения с подкреплением является достаточно общим и гибким, его можно использовать различными способами. Попытайтесь расширить границы данного подхода каким-либо образом хотя бы в одном из примеров.

Упражнение 3.2. Будет ли пригоден подход на основе обучения с подкреплением для эффективного решения *всех* задач целенаправленного обучения? Можете ли вы придумать какие-либо очевидные случаи, не укладывающиеся в схему данного подхода?

Упражнение 3.3. Рассмотрим задачу управления автомобилем. Можно определить требуемые действия как изменения положений акселератора, руля и тормозов, т. е. тех органов управления, с помощью которых водитель взаимодействует с машиной. Их можно задать и другим спосо-

бом, например рассматривать точку, в которой покрышка соприкасается с дорогой, считая действиями крутящие моменты, передаваемые на колеса. Еще один способ — в качестве действий рассматривать сокращения мышц, управляемых командами от мозга; эти сокращения приводят к изменению пространственного положения конечностей водителя. Или же, когда управление является действительно высокоуровневым, воспользоваться речевым каналом, тогда действиями будут произнесенные команды, показывающие *куда* надо ехать. Какой из этих вариантов следует признать правильным, где надо провести границу между агентом и средой? Какими должны быть критерии выбора этой границы? Существуют ли какие-нибудь фундаментальные принципы, согласно которым одно положение границы будет лучше другого, или же это произвольный выбор?

3.2. Цели и вознаграждения

В обучении с подкреплением цель агента формализуется посредством специального сигнала вознаграждения, поступающего к агенту из окружающей среды. На каждом временном шаге t вознаграждение представляет собой обычное число $r_t \in \mathbb{R}$. Цель агента — максимизация суммарного вознаграждения. Это означает, что требуется максимизировать не вознаграждение на текущем шаге, а совокупное итоговое вознаграждение по результатам достаточно продолжительной последовательности шагов.

Использование сигнала вознаграждения для формализации цели является наиболее характерной особенностью обучения с подкреплением. Несмотря на то что поначалу такой способ формализации целей может показаться ограниченным, он на практике доказал свою гибкость и широкую применимость. Чтобы в этом убедиться, лучше всего рассмотреть примеры того, как этот способ используется или мог бы использоваться. Например, чтобы научить робота ходить, на каждом временном шаге устанавливается вознаграждение, пропорциональное поступательному движению робота. Если робота обучают искать выход из лабиринта, вознаграждение обычно остается нулевым, пока выход не найден, а когда он найден — становится равным +1. Согласно другому распространенному методу, на каждом временном шаге в процессе поиска выхода из лабиринта агент получает вознаграждение, равное -1; это поощряет его усилия выйти из лабиринта как можно быстрее. Чтобы научить робота находить и собирать для переработки пустые жестяные банки из-под напитков, можно установить вознаграждение, равное нулю боль-

шую часть времени и равное +1 за каждую найденную банку, которая окажется пустой. Кроме того, можно давать роботу отрицательные вознаграждения в случае, если он сталкивается с предметами или если кто-нибудь на него кричит. Чтобы научить агента играть в шашки или шахматы, естественно было бы давать вознаграждение, равное +1 за победу, -1 за проигрыш и 0 за ничейный результат и за все промежуточные позиции в игре.

Рассмотрим, что происходит во всех этих примерах. Агент постоянно учится максимизировать получаемое вознаграждение. Если мы хотим, чтобы он делал что-либо для нас, мы должны устанавливать вознаграждения таким способом, чтобы, максимизируя их, агент добивался наших целей. Таким образом, необходимо, чтобы установленные нами вознаграждения правильно указывали, чего мы хотим добиться. В частности, сигнал вознаграждения не должен наделять агента априорными знаниями о том, *как* достичь того, чего мы от него хотим¹⁾. Например, играющий в шахматы агент должен получать вознаграждения только за фактические победы, но не за достижение каких-либо промежуточных целей, таких как взятие фигур соперника или контроль центра доски. Если бы достижение таких промежуточных целей вознаграждалось, агент мог бы найти способ их удовлетворения, не достигая при этом главной цели. Например, он может найти способ захватывать фигуры соперника даже ценой проигрыша игры. Сигнал вознаграждения является способом сообщить агенту, *чего* он должен добиваться, но не *как* он должен это сделать.

Новичков в области обучения с подкреплением иногда удивляет то, что вознаграждения, которые определяют цель обучения, рассчитываются в окружающей среде, а не в агенте. Разумеется, у животных большинство конечных целей определяется функционированием органов, находящихся внутри их тел, например рецепторов, распознающих еду, порождающих чувство голода, боль, удовольствие. Тем не менее, как уже говорилось в предыдущем разделе, можно переопределить границу взаимодействия агента со средой таким образом, что упомянутые части тела будут находиться вне агента (и таким образом, будут являться частью окружающей агента среды). Например, если цель имеет отношение к внутренним источникам энергии робота, то они считаются частью среды; если цель имеет отношение к положению конечностей робота, то они

¹⁾ Априорные знания такого типа лучше задавать через исходную стратегию, или функцию ценности, или через влияющие на них факторы. См. работы [Lin (1992)], [Maclin, Shavlik (1994)] и [Clouse (1996)].

тоже считаются частью окружающей среды — это означает, что граница агента проходит между конечностями робота и контролирующими их системами. Эти объекты считаются внутренними по отношению к роботу и внешними по отношению к агенту обучения. Для наших целей удобно располагать границу обучающегося агента не на рубеже его физического тела, а на рубеже его управляющих систем.

Это делается потому, что агент не должен иметь конечную цель, которую он может даже частично контролировать: к примеру, у него не должно быть возможности просто *устанавливать* получаемые вознаграждения таким же образом, как он может произвольно менять свои действия. По этой причине источник вознаграждений должен находиться вне агента, что не мешает агенту определить для себя какое-то внутреннее вознаграждение или последовательность таких вознаграждений. В действительности многие методы обучения с подкреплением именно так и делают.

3.3. Выгода

До сих пор мы не определяли точно цель обучения. Было сказано, что целью агента является максимизация совокупного вознаграждения. Каким образом это можно определить формально? Если обозначить через $r_{t+1}, r_{t+2}, r_{t+3}, \dots$ последовательность вознаграждений, полученных после временного шага t , то какие именно аспекты этой последовательности необходимо максимизировать? Вообще говоря, мы пытаемся максимизировать *ожидаемую выгоду*, где выгода R_t определяется как некоторая функция, определенная на последовательности вознаграждений. В простейшем случае выгода представляет собой сумму вознаграждений:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \dots + r_T, \quad (3.1)$$

где T — завершающий временной шаг. Такой подход имеет смысл в тех прикладных задачах, где завершающий шаг можно определить естественным образом, исходя из природы решаемой задачи, т. е., когда взаимодействие агент — окружающая среда можно разбить на подпоследовательности, называемые *эпизодами*¹⁾. Примерами эпизодов могут служить партия в игре, однократное прохождение лабиринта, любые другие типы повторяющихся взаимодействий. Каждый эпизод заканчивается особым состоянием, называемым *терминальным состоянием*, за которым следует возврат к обычному начальному состоянию или к выборке

¹⁾ В литературе эпизоды часто называют «испытаниями».

из стандартного распределения начальных состояний. Задания с такими эпизодами называются *заданиями, составленными из эпизодов*. В таких заданиях иногда необходимо отличать набор всех нетерминальных состояний, обозначаемый S , от совокупности всех состояний плюс терминальное состояние, что обозначается как S^+ .

С другой стороны, часто взаимодействие агент — окружающая среда нельзя разбить естественным образом на различимые эпизоды, поскольку это взаимодействие осуществляется непрерывно, причем без фиксации завершающего момента времени. Например, подобным образом можно трактовать задания, связанные с управлением процессами с непрерывным временем и непрерывной областью значений состояний, а также задания для робота с продолжительным сроком службы. Такие задания будем называть *непрерывными*. Формулировка выгоды (3.1) не подходит для непрерывных заданий, поскольку в этом случае заключительным временным шагом оказалось бы значение $T = \infty$, а выгода, которую необходимо максимизировать, может оказаться бесконечной. (Для примера, предположим, что на каждом временном шаге агент получает вознаграждение, равное +1.) Поэтому в данной книге будем обычно использовать определение выгоды, которое несколько сложнее для понимания, но намного проще с математической точки зрения.

Введем дополнительное понятие *приведенной величины*. Используя это понятие, можно говорить о том, что агент старается выбирать действия, стараясь максимизировать сумму приведенных вознаграждений, которые он получит в дальнейшем. В частности, он выбирает действия a_t так, чтобы максимизировать ожидаемую *приведенную выгоду*:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (3.2)$$

где γ — коэффициент приведения, $0 \leq \gamma \leq 1$.

Коэффициент приведения определяет текущую ценность будущих вознаграждений: вознаграждение, которое будет получено после k временных шагов в будущем, отличается от вознаграждения, получаемого немедленно, в γ^{k-1} раз. Если $\gamma < 1$, сумма бесконечного ряда имеет конечное значение, до тех пор пока последовательность $\{r_k\}$ будет ограниченной. Если $\gamma = 0$, то агент «близорукий», и его интересует максимизация только ближайших вознаграждений: в этом случае цель агента — научиться так выбирать действия a_t , чтобы максимизировать только ближайшее вознаграждение r_{t+1} . Если бы каждое из действий агента влияло только на ближайшее вознаграждение и не влияло на бу-

дущие вознаграждения, то близорукий агент мог бы максимизировать приведенную выгода (3.2), максимизируя по отдельности каждое ближайшее вознаграждение. Однако в общем случае стремление максимизировать только ближайшее вознаграждение может ограничить воздействие на будущие вознаграждения и выгода может оказаться заниженной. По мере того как коэффициент γ приближается к 1, растет значимость будущих вознаграждений: агент становится более дальновидным.



Рис. 3.2. Задача балансировки стержня

считается исход, когда стержень падает, отклонившись от вертикали на угол, больший заданного, или если тележка достигает границ площадки. Стержню возвращается вертикальное положение после каждой неудачи. Это задание может рассматриваться как состоящее из эпизодов; эпизодами здесь являются повторяющиеся попытки балансировки стержня. Вознаграждением может быть +1 за каждый временной шаг, на котором не было неудачи. Таким образом, выгода все время будет равна числу шагов до неудачи. Если использовать приведенные значения, то можно рассматривать балансировку стержня как непрерывное задание. В этом случае вознаграждением может быть -1 за каждую неудачу и 0 в течение остального времени. Выгода в этом случае будет зависеть от $-\gamma^k$, где k — количество временных шагов до неудачи. В обоих случаях, чтобы максимизировать выгоду, необходимо сохранять баланс стержня как можно дольше.

Упражнение 3.4. Пусть балансировка стержня рассматривается как задание, состоящее из эпизодов, но при этом используются также приведенные значения, причем все вознаграждения равны нулю, за исключением момента неудачи, когда вознаграждение равно -1. Какой в этом случае будет выгода в каждый момент времени? Как эта выгода будет отличаться от выгоды в случае, если рассматривать это задание как приведенное и непрерывное?

Упражнение 3.5. Представьте, что разрабатывается робот для прохождения лабиринта. Принимается решение установить вознагражде-

Пример 3.4 (Балансировка стержня). Рисунок 3.2 демонстрирует задание, которое является несложной иллюстрацией применения обучения с подкреплением. В этом задании требуется таким образом приложить силы к тележке, передвигающейся по площадке ограниченных размеров, чтобы шарнирно закрепленный на ней стержень не падал. Неудачей

размеров, чтобы шарнирно закрепленный на ней стержень не падал. Неудачей

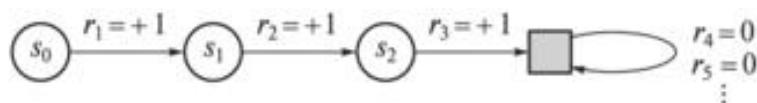
ние +1 за выход из лабиринта и 0 в течение всего остального времени. Задание естественным образом разбивается на эпизоды — последовательность попыток прохождения лабиринта. Таким образом, задание считается состоящим из эпизодов, целью в нем является максимизация ожидаемого совокупного вознаграждения (3.1). После некоторого количества попыток обнаруживается, что обучающийся агент не демонстрирует никаких улучшений в прохождении лабиринта. Где допущена ошибка? Правильно ли агенту было указано, чего он должен добиваться?

3.4. Единые обозначения для непрерывных заданий и заданий, состоящих из эпизодов

В предыдущем разделе были рассмотрены два вида заданий, которые выполняются с помощью обучения с подкреплением. В заданиях первого вида взаимодействие агент — окружающая среда естественным образом разбивается на последовательность отдельных эпизодов (задания, состоящие из эпизодов). В заданиях второго вида разбиения на эпизоды не происходит (непрерывные задания). Первый вид математически более простой, так как каждое действие влияет только на конечное число вознаграждений, получаемых в течение эпизода. В этой книге иногда рассматриваются задания первого вида, иногда — второго, но чаще всего речь будет идти о смешанных заданиях. Поэтому полезно будет ввести единые обозначения, позволяющие точно говорить об обоих видах заданий одновременно.

Чтобы быть точными при рассмотрении заданий, состоящих из эпизодов, требуется ввести дополнительные обозначения. Вместо одной длинной последовательности временных шагов надо иметь возможность рассматривать серии эпизодов, каждый из которых состоит из конечной последовательности временных шагов. Нумерация временных шагов в каждой серии начинается с нуля. Поэтому вместо обозначения s_t для состояния в момент времени t будем использовать обозначение $s_{t,i}$ для состояния в момент времени t в i -м эпизоде (и аналогично $a_{t,i}, r_{t,i}, \pi_{t,i}, T_i$ и т. д.). Следует отметить, что при рассмотрении заданий, состоящих из эпизодов, почти никогда не придется делать различия между разными эпизодами. Почти всегда будет рассматриваться один конкретный эпизод или же иногда будут формулироваться положения, справедливые для всех эпизодов. Соответственно, на практике индекс, указывающий номер эпизода, почти всегда будет опускаться для упрощения записи, т. е. вместо $s_{t,i}$ будем писать s_t и т. д.

Введем еще одно соглашение, чтобы получить единую систему обозначений как для заданий, состоящих из эпизодов, так и для непрерывных заданий. Выгода была ранее определена как сумма конечного числа элементов в одном случае (3.1) и как сумма бесконечного числа элементов во втором случае (3.2). Эти два подхода можно объединить, если ввести для конца эпизода специальное *поглощающее состояние*, которое переходит только само в себя и которое порождает только нулевые вознаграждения. Рассмотрим, например, следующую диаграмму переходов:



Здесь затененный квадрат обозначает специальное поглощающее состояние, соответствующее концу некоторого эпизода. Начиная с состояния s_0 будет порождаться последовательность вознаграждений $+1, +1, +1, 0, 0, 0, \dots$. Суммируя первые T вознаграждений (здесь $T = 3$), получаем ту же самую выгоду, что и при суммировании всей бесконечной последовательности. Этот результат будет верен даже при использовании приведенных величин. Таким образом, можно определить выгоду для общего случая согласно соотношению (3.2), опуская номера эпизодов, когда они не нужны, и включая в качестве возможного варианта значение $\gamma = 1$, если сумма остается конечной (например, если все эпизоды завершились). Выражение для выгоды можно записать и в другом варианте:

$$R_t = \sum_{k=0}^T \gamma^k r_{t+k+1}. \quad (3.3)$$

Здесь допускается возможность того, что $T = \infty$ или $\gamma = 1$ (но не одновременно¹⁾). Эти соглашения будут использоваться на протяжении всей книги для упрощения обозначений и для того, чтобы подчеркнуть схожесть непрерывных заданий и задач, состоящих из эпизодов.

*3.5. Марковское свойство

В обучении с подкреплением решения, принимаемые агентом, представляют собой функцию от сигнала из окружающей среды, называемого *событием*.

¹⁾ Отыскание способов формулирования задач, которые одновременно являются непрерывными и не используют приведенных величин, является объектом текущих исследований; см. [Mahadevan (1996)]; [Schwartz (1993)]; [Tadepalli, Ok (1994)]. Некоторые из этих идей рассматриваются в разд. 6.7.

стоянием среды. Обсудим в этом разделе, что именно требуется от сигнала, представляющего состояния, какую информацию из него можно извлечь, а какую нельзя. В частности, дадим формальное определение свойству окружающих сред и сигналов, характеризующих их состояния, называемому марковским свойством и представляющему особенный интерес.

В этой книге под термином «состояние» понимается любая информация, доступная агенту. Предполагается, что состояние определяется некоторой системой предварительной обработки, которая номинально является частью окружающей среды. В этой книге не рассматриваются вопросы формирования, изменения или изучения сигнала состояния. Это делается не потому, что вопросы представления состояния можно считать несущественными, но только для того, чтобы полностью сосредоточиться на вопросах принятия решений. Другими словами, основной интерес для нас представляет не то, каким образом был получен сигнал, характеризующий состояние, но принятие решения о том, какое действие следует осуществить как функцию любого доступного сигнала состояния.

Несомненно, сигнал состояния должен включать непосредственные ощущения, такие как сенсорные данные, но он может содержать и гораздо больше этого. Представления состояний могут быть существенно переработанными вариантами непосредственных ощущений, или же они могут быть сложными структурами, построенными на основе последовательности ощущений, которые воспринимались в течение определенного периода времени. Например, мы можем переводить взгляд с места на место, наблюдая в каждый отдельный момент во всех деталях лишь небольшой участок некоторого изображения, однако получить в итоге точное и подробное представление о картине в целом. Или еще проще, можно посмотреть на объект, затем отвести взгляд и, тем не менее, знать, что объект все еще там, где он был. Услышав слово «да», мы оказываемся в совершенно различных состояниях, в зависимости от вопроса, который прозвучал ранее. В более прагматичном примере система управления по результатам измерений положения объекта для двух различных моментов времени может сформировать представление состояния, включающее информацию о скорости. Во всех этих случаях состояние формируется на базе непосредственных ощущений, а также прошлых ощущений в виде информации о предыдущих состояниях или в какой-либо еще форме. В этой книге не рассматривается вопрос о том, как это сделать, несомненно только, что это может и должно быть сделано. Нет никаких причин ограничивать представления состояний только

лишь непосредственными ощущениями. В типичных прикладных задачах следует ожидать, что представление состояния способно обеспечивать агента более обширной информацией.

С другой стороны, не следует ожидать, что сигнал состояния будет информировать агента обо всем в окружающей среде или даже обо всем том, что может повлиять на принятие решения. Если агент играет в «двадцать одно», не следует ожидать, что он будет знать следующую карту на столе. Если агент отвечает на звонок по телефону, не следует ожидать, что он будет знать, кто именно звонит. Если агент является спасателем, которого вызывают на автомобильную аварию, не следует ожидать, что он тотчас же будет знать, какие внутренние органы повреждены у пострадавшего, находящегося в бессознательном состоянии. Во всех этих случаях в окружающей среде присутствует скрытая информация, и эта информация была бы полезна, если бы агент владел ею, но агент ею не владеет, так как она недоступна его ощущениям. Одним словом, нет вины агента, если он не знает чего-либо важного, его вина есть, только если он забывает имевшуюся ранее информацию.

В идеале сигнал состояния должен кратко суммировать прошлые ощущения таким образом, чтобы сохранялась вся необходимая информация. Обычно для этого требуется больше, чем наличие лишь непосредственных ощущений, но ни в коем случае не больше, чем наличие предыстории всех прошлых ощущений. Сигнал состояния, содержащий всю необходимую информацию, называется *марковским*, или обладающим *марковским свойством* (формальное определение будет дано ниже). Например, шашечная позиция — текущее положение всех фигур на доске — представляет собой марковское состояние, поскольку содержит в себе наиболее важную информацию обо всех предыдущих позициях, приведших к текущему расположению дел. Значительная часть информации о последовательности ходов утрачена, но информация, действительно влияющая на дальнейшее развитие игры, сохранена. Аналогично, скорость и текущее местоположение пушечного ядра полностью определяют его дальнейший полет. И не важно, каким образом были получены данная скорость и данное местоположение. Это иногда называют свойством «независимости от пути», так как все необходимое заключено в текущем сигнале состояния; значение сигнала не зависит от «пути» или предыстории сигналов, приведших к данному состоянию.

Дадим теперь формальное определение марковскому свойству для задачи обучения с подкреплением. Чтобы упростить математические выкладки, ограничимся здесь рассмотрением конечного числа состояний и значений вознаграждений. Это позволяет использовать суммы

вместо интегралов и вероятности вместо плотностей вероятности. Однако все рассуждения легко обобщаются на случай с непрерывными состояниями и вознаграждениями. Рассмотрим, каким образом окружающая среда может в момент времени $t + 1$ реагировать на действие, осуществленное в момент времени t . В наиболее общем случае эта реакция может зависеть от какого угодно события в прошлом. В этом случае динамика среды может быть определена только заданием полного распределения вероятности:

$$\Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0\} \quad (3.4)$$

для всех s', r и всех возможных значений предыдущих событий: $s_t, a_t, r_t, \dots, r_1, s_0, a_0$. С другой стороны, если сигнал состояния обладает *марковским свойством*, то реакция окружающей среды в момент времени $t + 1$ зависит лишь от представления состояния и действия в момент времени t . В таком случае динамику окружающей среды можно определить, задавая только

$$\Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\} \quad (3.5)$$

для всех s', r, s_t и a_t . Другими словами, состояние обладает марковским свойством и, следовательно, является марковским состоянием тогда и только тогда, когда (3.5) равно (3.4) для всех s', r и предшествовавших $s_t, a_t, r_t, \dots, r_1, s_0, a_0$. В этом случае говорят, что окружающая среда и задание в целом обладают марковским свойством.

Если окружающая среда обладает марковским свойством, то ее одиночковая динамика (3.5) позволяет предсказать следующее состояние и следующее ожидаемое вознаграждение, которое является результатом текущего состояния и действия. Можно показать, проводя итеративные вычисления для данного соотношения, что имеется возможность предсказать все будущие состояния и ожидаемые вознаграждения и всю историю состояний, зная только текущее состояние. Отсюда следует, что марковские состояния обеспечивают наилучший из всех возможных базис для выбора действий. Таким образом, наилучшая стратегия выбора действия, определенная как функция марковского состояния, эффективна в такой же степени, как и наилучшая стратегия, сформированная как функция всей истории состояний.

Даже если сигнал состояния не марковский, все же уместно будет рассматривать его в задаче обучения с подкреплением как аппроксимацию марковского состояния. В частности, всегда желательно, чтобы состояние являлось хорошим базисом для предсказания последующих вознаграждений и для выбора действий. В случаях, когда модель окру-

жающей среды получается через обучение (см. гл. 9), хотелось бы также, чтобы данное состояние являлось хорошим базисом для предсказания последующих состояний. Марковские состояния являются наилучшим базисом для осуществления всех этих целей. Чем ближе характеристики данного состояния будут к характеристикам марковских состояний, тем лучше будет проявлять себя система обучения с подкреплением. Таким образом, на каждом временном шаге состояние выгодно рассматривать как аппроксимацию марковского состояния, хотя необходимо помнить, что оно может не полностью удовлетворять марковскому свойству.

Марковское свойство играет важную роль в обучении с подкреплением, так как принимаемые решения и получаемые значения величин являются функциями только текущего состояния. Для того чтобы они были эффективны и содержательны, представление состояния должно быть информативным. Все теоретические построения, излагаемые в этой книге, исходят из того, что сигналы состояния являются марковскими. Это означает, что разработанная теория не полностью применима в тех случаях, когда допущение о наличии марковского свойства некорректно. Тем не менее разработанная для марковского случая теория помогает нам понять работу алгоритмов, которые можно было бы успешно применять к заданиям с состояниями, не являющимися строго марковскими. Полное понимание теории в марковском случае является важнейшей основой для дальнейшего ее развития в направлении более сложных и реалистичных немарковских случаев. В заключение отметим, что допущения, сделанные в представлении марковского состояния, имеют место не только в обучении с подкреплением, они также присутствуют в большинстве, если не во всех, методов изучения искусственного интеллекта.

Пример 3.5 (Сигнал состояния в задаче балансировки стержня). В рассмотренной ранее задаче балансировки стержня сигнал состояния являлся бы марковским, если бы он точно задавал или позволял бы точно воссоздать положение и скорость тележки на площадке, угол между тележкой и стержнем, а также скорость, с которой этот угол меняется (угловую скорость). В идеализированной системе тележка—стержень этой информации было бы достаточно для точного предсказания дальнейшего поведения тележки и стержня при заданных действиях, предпринимаемых регулятором. На практике, однако, такую точную информацию получить невозможно, поскольку в результатах измерений, поступающих от любого реального датчика, присутствуют запаздывания и искажения. Более того, на поведение реальной системы

тележка—стержень всегда будет оказывать некоторое влияние еще ряд факторов, таких как искривление стержня, температура в подшипниках тележки и стержня, различного рода люфты. Эти факторы могли бы вызвать нарушения марковского свойства, если бы сигнал состояния сообщал только положения и скорости тележки и стержня.

Тем не менее достаточно часто положения и скорости вполне можно использовать в качестве состояний. В некоторых ранних работах, изучавших задачу балансировки стержня, использовалось весьма грубое представление для сигнала состояния, который подразделял положения тележки на три зоны: правая, левая и средняя (такое же грубое разбиение применялось и для остальных трех переменных, определяющих состояние). Такое немарковское состояние позволяло легко справиться с заданием методами обучения с подкреплением. Фактически такое укрупненное представление позволяло обеспечить быстрое обучение, заставляя агента игнорировать тонкие различия, бесполезные при решении рассматриваемой задачи.

Пример 3.6 (Покер). В одной из разновидностей покера (пятикарточный покер, draw poker) каждому игроку раздается пять карт. Далее следует первый круг торговли, после которого игроки могут заменить одну или несколько карт, полученных при первой раздаче, после чего следует завершающий круг торговли. На каждом круге торговли каждый из игроков должен подтвердить или перебить наивысшую ставку среди всех других игроков или выбыть из розыгрыша (спасовать). После окончания торговли выигрывает тот игрок, который не выбыл из розыгрыша и у которого на руках оказывается наилучшая комбинация карт. Этот игрок и забирает банк.

Сигнал состояния в покере различен для каждого из игроков. Каждый игрок знает свои карты и может только догадываться о картах соперников. Распространенной ошибкой является мнение, что марковский сигнал состояния должен содержать информацию о картах всех игроков и картах, оставшихся в колоде. Однако в честной игре предполагается, что игроки в принципе не могут знать таких вещей, основываясь на предыдущих наблюдениях. Если бы игрок знал что-либо подобное, он бы смог предсказывать некоторые события (например, какие карты мог заменить соперник) лучше, чем если бы основывался только на всех своих предыдущих наблюдениях.

Кроме информации о собственных картах состояние в покере должно содержать информацию о ставках и количестве карт, сброшенных игроками. Например, если один из игроков сбросил три карты, можно

предполагать, что он оставил на руках пару, и в соответствии с этим строить свои предположения о силе его позиции. Ставки игроков также влияют на предположения о силе их карт. Фактически частью марковского состояния является опыт предыдущих игр с данными конкретными игроками. Любит ли Элен блефовать или она играет осторожно? Можно ли по ее лицу или поведению как-то судить о силе ее карт? Как меняется манера игры Джо поздней ночью или когда он уже выиграл много денег?

Хотя все наблюдения за игроками могут повысить вероятность правильного предсказания имеющихся у них на руках комбинаций карт, на практике оказывается, что информации, которую необходимо запомнить и проанализировать, слишком много, и большей частью она не влияет напрямую на правильность прогнозирования и принятий решений. Очень хорошие игроки в покер умеют запоминать всякого рода важные сведения и способны быстро оценивать новых игроков, но и они никогда не стараются запомнить все происходящее в игре. В результате представления состояний, которые люди используют для принятия решений при игре в покер, являются, несомненно, немарковскими, и решения сами по себе, очевидно, не будут идеальными. Тем не менее люди принимают очень хорошие решения, рассматривая такие задачи. Можно сделать вывод, что невозможность использовать *идеальное* марковское представление состояния не является, по-видимому, серьезной проблемой для агента обучения с подкреплением.

Упражнение 3.6 [Сломанная видеосистема]. Вообразите, что вы являетесь видеосистемой. Когда вас впервые включают, в вашу камеру поступает поток изображений. Вы можете видеть множество вещей, но не все. Вы не можете видеть скрытые объекты, и конечно, вы не можете видеть объекты, расположенные за вами. После того как вы впервые увидели обстановку вокруг, будет ли у вас доступ к марковскому состоянию окружающей среды? Предположим, что ваша камера сломалась, и к вам не поступает никаких изображений весь день. Будет ли у вас в этом случае доступ к марковскому состоянию?

3.6. Марковские процессы принятия решений

Задание обучения с подкреплением, удовлетворяющее марковскому свойству, называется *марковским процессом принятия решений*, или *МППР*. Если пространства состояний и действий являются конечными, то задание называется *финитным марковским процессом принятия решений* (*финитным МППР*). Финитные МППР особенно важны для тео-

рии обучения с подкреплением. Они будут часто рассматриваться в этой книге, поскольку современное обучение с подкреплением на 90% связано именно с такими процессами.

Конкретный финитный МППР определяется множествами своих состояний и действий, а также одношаговой динамикой окружающей среды. При заданном состоянии s и действии a вероятность каждого возможного следующего состояния s' определяется выражением

$$\mathcal{P}_{ss'}^a = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}. \quad (3.6)$$

Эти величины называются *вероятностями перехода*. Аналогично, при заданном текущем состоянии s и действии a в сочетании со следующим состоянием s' ожидаемое значение последующего вознаграждения равно

$$\mathcal{R}_{ss'}^a = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}. \quad (3.7)$$

Величины $\mathcal{P}_{ss'}^a$ и $\mathcal{R}_{ss'}^a$ полностью описывают наиболее важные аспекты динамики финитного МППР (утерянной оказывается только информация о распределении вознаграждений в окрестности ожидаемого значения). В теоретических построениях, представленных в оставшейся части книги, большей частью предполагается, что окружающая среда является финитным МППР.

Пример 3.7 (МППР Робот для сбора банок). Робот для сбора банок (пример 3.3) может быть превращен в несложный пример МППР, если несколько упростить задачу и ввести в нее некоторые дополнительные элементы. (Наша цель — привести простой пример, не обязательно реалистичный.) Необходимо помнить, что агент принимает решения в моменты времени, определяемые внешними событиями (или компонентами системы управления робота). В каждый такой момент времени робот решает, должен ли он

- (1) активно искать банки,
- (2) оставаться неподвижным и ждать, пока кто-нибудь принесет банку, или
- (3) направиться обратно на базу для зарядки батареи.

Предположим, что окружающая среда функционирует следующим образом. Лучшим способом находить банки является их активный поиск, но это разряжает батарею робота, тогда как в периоды ожидания батарея не разряжается. Если робот осуществляет поиск, существует вероятность того, что его батарея окажется полностью разряженной. В этом случае робот должен отключиться и ждать технической помощи (получая низкие вознаграждения).

Принимаемые агентом решения являются функцией только уровня заряда батареи. Агент может различать два уровня, **high** (высокий) и **low** (низкий), таким образом, множество состояний имеет вид $S = \{\text{high}, \text{low}\}$. Назовем возможные решения — действия агента — **wait** (ожидать), **search** (искать) и **recharge** (подзаряжаться). Когда уровень заряда батареи имеет значение **high**, подзарядка не имеет смысла, таким образом, она не включается во множество действий для данного состояния. Множества действий агента имеют вид

$$\begin{aligned} \mathcal{A}(\text{high}) &= \{\text{search}, \text{wait}\} \\ \mathcal{A}(\text{low}) &= \{\text{search}, \text{wait}, \text{recharge}\}. \end{aligned}$$

Если уровень зарядки батареи равняется **high**, то период активного поиска всегда может быть завершен без риска полностью разрядить батарею. Период поиска, начавшийся с уровнем заряда батареи, равным **high**, сохранит заряд батареи на уровне **high** с вероятностью α и понизит уровень до **low** с вероятностью $1 - \alpha$. С другой стороны, период поиска, начавшийся, когда уровень зарядки батареи равняется **low**, сохранит уровень **low** с вероятностью β и разрядит батарею с вероятностью $1 - \beta$. В последнем случае работу должна быть оказана помощь, после чего батарею следует зарядить до уровня **high**. За каждую найденную банку робот получает единичное вознаграждение, тогда как при оказании ему помощи робот получает отрицательное вознаграждение, равное -3 . Обозначим через $\mathcal{R}^{\text{search}}$ и $\mathcal{R}^{\text{wait}}$, $\mathcal{R}^{\text{search}} > \mathcal{R}^{\text{wait}}$, ожидаемое количество собранных роботом банок (и, следовательно, полученных вознаграждений) в процессе поиска и в процессе ожидания соответственно. Наконец, чтобы не усложнять задачу, будем считать, что робот не может собирать банки по пути на базу для подзарядки и что он не может их собирать, когда батарея разряжена. Такая система представляет собой финитный МППР. Для нее можно записать вероятности перехода и ожидаемые вознаграждения так, как это сделано в табл. 3.1.

Граф переходов представляет собой полезное средство, позволяющее показать динамику финитного МППР в целом. На рис. 3.3 показан граф переходов для робота, собирающего банки. В этом графе используются два вида вершин: *вершины-состояния* и *вершины-действия*. Каждому возможному состоянию соответствует вершина-состояние (большой белый кружок с названием состояния внутри), каждой паре состояние—действие соответствует вершина-действие (маленький черный кружок с расположенным рядом названием действия, соединенный линией с вершиной-состоянием). Начиная с состояния s и выбирая действие a , двигаемся по линии от вершины-состояния s к вершине-

Таблица 3.1
Вероятности перехода и ожидаемые вознаграждения
для финитного МППР для примера с роботом для сбора банок

s	s'	a	$\mathcal{P}_{ss'}^a$	$\mathcal{R}_{ss'}^a$
high	high	search	α	$\mathcal{R}^{\text{search}}$
high	low	search	$1 - \alpha$	$\mathcal{R}^{\text{search}}$
low	high	search	$1 - \beta$	-3
low	low	search	β	$\mathcal{R}^{\text{search}}$
high	high	wait	1	$\mathcal{R}^{\text{wait}}$
high	low	wait	0	$\mathcal{R}^{\text{wait}}$
low	high	wait	0	$\mathcal{R}^{\text{wait}}$
low	low	wait	1	$\mathcal{R}^{\text{wait}}$
low	high	recharge	1	0
low	low	recharge	0	0

Примечание. Каждой возможной комбинации текущего состояния s , следующего состояния s' и возможного в данном текущем состоянии действия $a \in A(s)$ соответствует своя строка.

действию (s, a) . Окружающая среда отвечает переходом в следующую вершину-состояние с помощью одной из стрелок, идущих из вершины-действия (s, a) . Каждая стрелка соответствует набору из трех величин (s, s', a) , где s' — следующее состояние. Каждой из стрелок ставится в соответствие метка, показывающая вероятность перехода $\mathcal{P}_{ss'}^a$ и ожидаемое вознаграждение за этот переход $\mathcal{R}_{ss'}^a$. Отметим, что вероятности

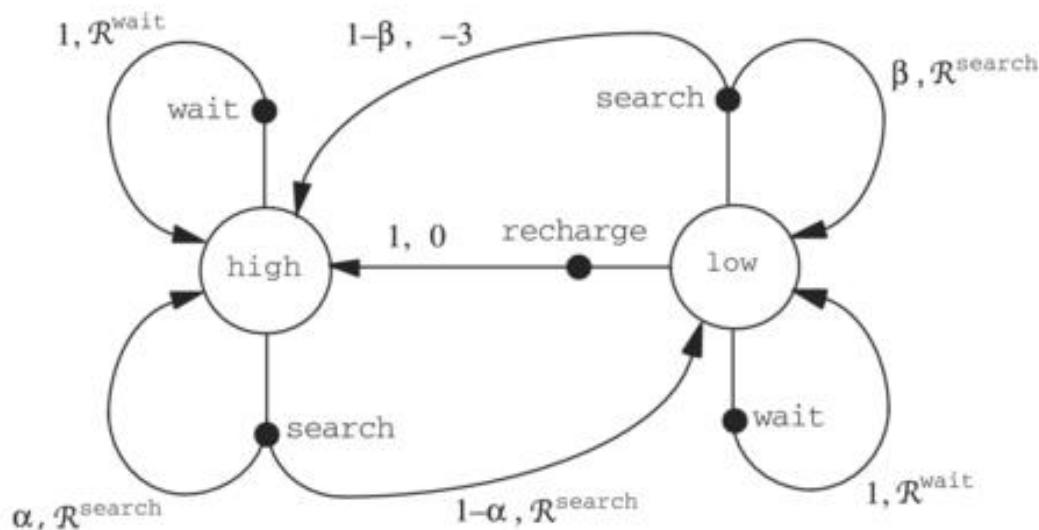


Рис. 3.3. Граф переходов для робота, собирающего банки (high — высокий, low — низкий, wait — ожидать, search — искать, recharge — подзаряжаться)

перехода, закрепленные за стрелкой, выходящей из вершины-действия, всегда в сумме равны 1.

Упражнение 3.7. Рассматривая финитный МППР с конечным количеством значений вознаграждений, напишите уравнение для вероятностей перехода и ожидаемых вознаграждений, используя совместное условное распределение (3.5).

3.7. Функции ценности

В основе почти всех алгоритмов обучения с подкреплением лежат оценочные *функции ценности* — функции состояний (или пар состояния—действие), которые оценивают, *насколько хорошо* для агента находиться в данном состоянии (или насколько хорошо для агента осуществить данное действие в данном состоянии). Понятие «насколько хорошо» определяется здесь с учетом ожидаемых в будущем вознаграждений или, более точно, ожидаемой выгоды. Конечно, вознаграждения, которые агент ожидает получить в будущем, зависят от действий, которые он осуществит. Соответственно, функции ценности определяются с учетом конкретной стратегии.

Напомним, что стратегия π определяет соответствие каждого состояния $s \in \mathcal{S}$ и действия $a \in A(s)$ вероятности $\pi(s, a)$, с которой действие a будет осуществлено в состоянии s . Фактически *ценность* состояния s при стратегии π , обозначаемая $V^\pi(s)$, является ожидаемой выгодой при начальном состоянии s и со стратегией π . Для МППР *ценность* $V^\pi(s)$ формально определяется как

$$V^\pi(s) = E_\pi\{R_t \mid s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\}, \quad (3.8)$$

где $E_\pi\{\}$ обозначает математическое ожидание при условии, что агент следует стратегии π . Заметим, что ценность заключительного состояния, если таковое присутствует, всегда равна нулю. Функция V^π называется *функцией ценности состояния для стратегии π* .

Аналогично ценность действия a в состоянии s при стратегии π , обозначаемая $Q^\pi(s, a)$, определяется как ожидаемая выгода при начальном состоянии s , осуществленном действии a и со стратегией π :

$$Q^\pi(s, a) = E_\pi\{R_t \mid s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}. \quad (3.9)$$

Будем называть Q^π *функцией ценности действия для стратегии π* .

Функции ценности V^π и Q^π могут быть получены из опыта. Например, если агент реализует стратегию π и получает при каждом состоянии, в котором оказывается, значение выгода, среднее из тех, которые следуют за данным состоянием, то эта средняя выгода будет стремиться к ценности состояния $V^\pi(s)$ в случае, когда число повторений рассматриваемого состояния стремится к бесконечности. Если каждому действию, осуществленному в данном состоянии, соответствуют разные значения среднего, то эти средние значения будут аналогичным образом стремиться к ценностям действий $Q^\pi(s, a)$. Методы оценивания такого вида называются *методами Монте-Карло*, так как включают в себя усреднение по случайной выборке значений фактической выгода (см. гл. 5). Конечно, если состояний очень много, то может оказаться непрактичным каждому отдельному состоянию ставить в соответствие свое значение среднего. Вместо этого агент должен обращаться с V^π и Q^π как с параметризованными функциями и подбирать их параметры так, чтобы эти функции соответствовали полученной выгоде. Так тоже можно получить точные оценки, хотя многое зависит от характера аппроксимации, задаваемого параметризованной функцией (см. гл. 8).

Фундаментальным свойством функций ценности, которые используются в обучении с подкреплением и динамическом программировании, является то, что они удовлетворяют определенным рекуррентным соотношениям. Для любой стратегии π и любого состояния s выполняется следующее условие совместности ценности s и ценности возможных последующих состояний:

$$\begin{aligned} V^\pi(s) &= E_\pi\{R_t \mid s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s\right\} = \\ &= E_\pi\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s\right\} = \\ &= \sum_a \pi(s, a) \sum_{s'} \mathcal{P}_{ss'}^a \left[\mathcal{R}_{ss'}^a + \gamma E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_{t+1} = s'\right\} \right] = \\ &= \sum_a \pi(s, a) \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^\pi(s')], \end{aligned} \tag{3.10}$$

где подразумевается, что действия a берутся из множества $A(s)$, а следующие состояния s' принадлежат множеству S или S^+ в случае с задачей, состоящей из эпизодов. Уравнение (3.10) называется *уравнением Беллмана для V^π* . Оно выражает зависимость между ценностью некоторого состояния и ценностями других состояний, следующих за ним.

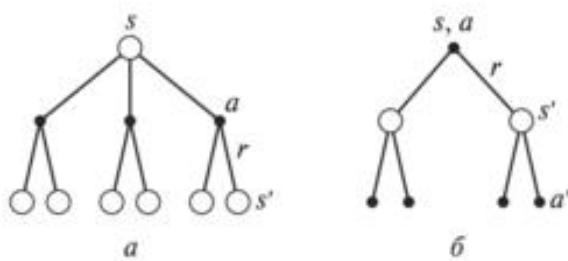


Рис. 3.4. Диаграммы предшествующих состояний для функций (a) V^π и (б) Q^π

Рассмотрим некоторое состояние и последующие состояния для него (рис. 3.4, a). Каждый белый кружок здесь соответствует состоянию, а каждый черный кружок — паре состояния—действие. Если агент в качестве начального использует состояние s , т. е. верхнюю вершину-корень, то он сможет осуществить любое из данного множества действий, три возможных варианта которых изображены на рис. 3.4, a. Реакцией окружающей среды на каждое из этих действий может быть одно из нескольких состояний s' с соответствующим вознаграждением r . Уравнение Беллмана (3.10) осуществляет усреднение по всем возможным вариантам, взвешенным в соответствии с вероятностью их появления. Согласно этому уравнению, ценность начального состояния должна быть равна (приведенной) ценности ожидаемого следующего состояния плюс вознаграждение, которое ожидается получить.

Функция ценности V^π является единственным решением уравнения Беллмана. В последующих главах будет показано, каким образом это уравнение Беллмана можно использовать для получения функций V^π и выполнения различных действий с ними. Диаграммы, изображенные на рис. 3.4, называются *диаграммами предшествующих состояний*, так как они графически изображают связи, формирующие совокупность операций корректировки (или операций восстановления предшествующих состояний), которые лежат в основе методов обучения с подкреплением. С помощью этих операций состояние (или пара состояния—действие) получает информацию о ценности от своих же последующих состояний (или пар состояния—действие). Диаграммы предшествующих состояний используются нами на протяжении всей книги для графической демонстрации обсуждаемых алгоритмов. (Отметим, что в отличие от графов переходов вершины-состояния в диаграммах предшествующих состояний не обязательно соответствуют определенным состояниям; например, действие может являться преемником самого себя. Кроме того, в диаграммах предшествующих состояний опущены стрелки, так как ось времени всегда направлена вниз по диаграмме.)

Пример 3.8 (Сетка). На рис. 3.5, a, показана прямоугольная сетка, с помощью которой удобно показать значения функции ценности для простого финитного МПР. Клетки сетки соответствуют состояниям

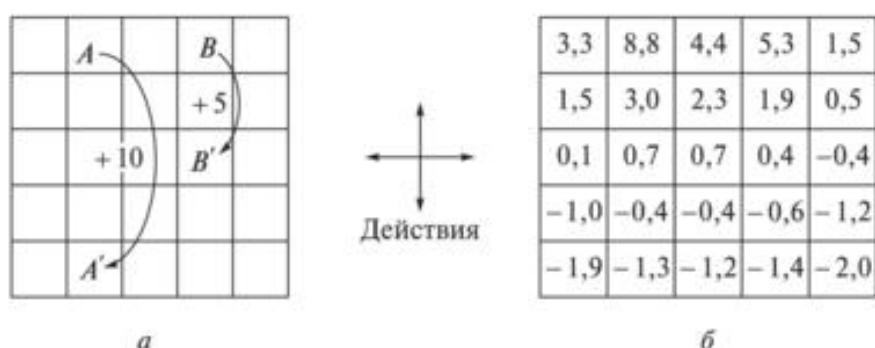


Рис. 3.5. Пример с сеткой: (а) динамика вознаграждений для особых состояний; (б) функция оценки состояния для равновероятной случайной стратегии

окружающей среды. В каждой клетке возможны четыре действия: «север», «юг», «восток» и «запад», каждое из которых предопределяет перемещение агента на один шаг в соответствующем направлении по сетке. Действие, которое вывело бы агента за пределы сетки, оставляет его местоположение без изменений, но результатом такого действия является вознаграждение -1 . Результатом других действий является вознаграждение 0 , за исключением тех, которые выводят агента из особых состояний A и B . В случае с состоянием A результатом любого из четырех действий является вознаграждение $+10$ и переход в состояние A' . В случае с состоянием B результатом любого из четырех действий является вознаграждение $+5$ и переход в состояние B' .

Пусть во всех состояниях агент с одинаковой вероятностью выбирает любое из четырех действий. На рис. 3.5, б, показана функция ценности V^π для такой стратегии в случае приведенного вознаграждения с коэффициентом $\gamma = 0,9$. Эта функция ценности была получена решением системы уравнений (3.10). Обратите внимание на отрицательные вознаграждения вблизи нижней границы. Они обусловлены высокой вероятностью выхода на нижнюю границу при случайной стратегии. Состояние A является лучшим для данной стратегии, но ожидаемая выгода для него меньше 10 , т. е. она меньше значения его текущего вознаграждения, так как из состояния A агент попадает в состояние A' , откуда возможен выход на границу сетки. С другой стороны, ценность состояния B больше его текущего вознаграждения 5 , так как из состояния B агент попадает в состояние B' , которое имеет положительную ценность. В случае с B' ожидаемый штраф (отрицательное вознаграждение) за возможный выход на границу с запасом компенсируется ожидаемым выигрышем, результатом возможной задержки в состоянии A или B .

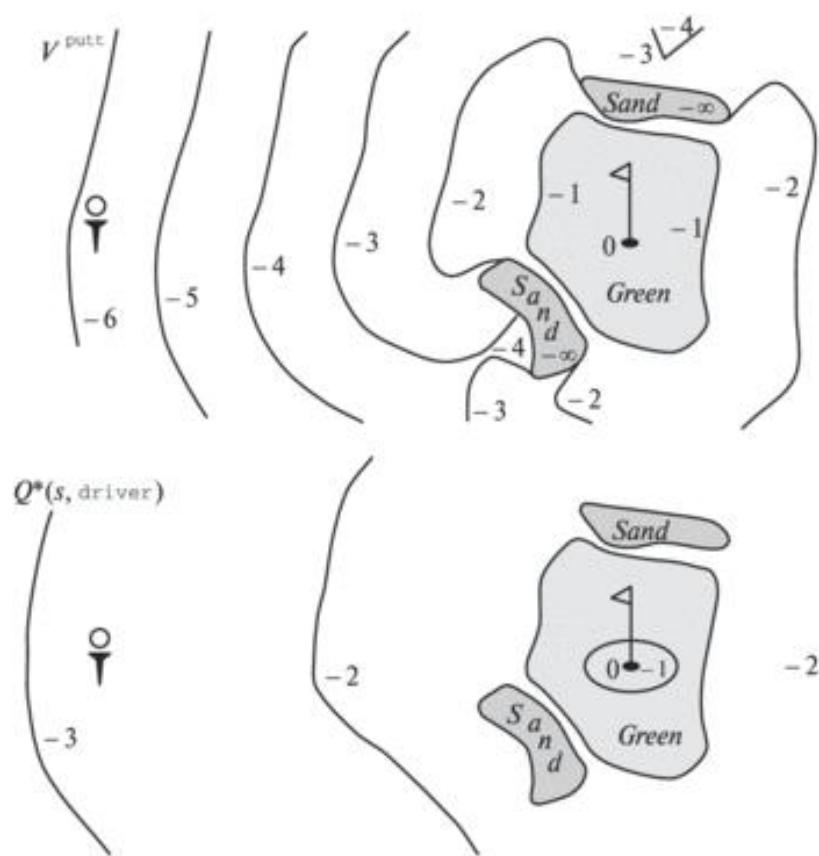


Рис. 3.6. Пример игры в гольф: функция ценности состояния при использовании патта (вверху) и оптимальная функция ценности действия при использовании драйвера (внизу); здесь green — лужайка с короткой травой вокруг лунки, sand — песчаная ловушка

Пример 3.9 (Гольф). Сформулируем задание обучения с подкреплением, в котором требуется попасть мячом в лунку при игре в гольф. Для этого примем значение штрафа (отрицательного вознаграждения), равным -1 за каждый удар, пока мяч еще не попал в лунку. За состояние примем положение мяча. За ценность состояния примем умноженное на -1 число ударов, необходимое для перемещения мяча из данного положения в лунку. В число предпринимаемых действий входит то, как осуществляется выбор направления и силы удара, а также, конечно, выбор клюшки. Примем, что два первых фактора предопределены, и ограничимся только выбором клюшки. Будем считать, что клюшечек всего две — паттер и драйвер¹⁾. В верхней части рис. 3.6 показана воз-

¹⁾ Драйвер (driver) — клюшка для самого далекого удара, имеющая наименьший угол наклона головки и самую длинную ручку. Паттер (putter) — короткая деревянная клюшка для патта (putt), т. е. катящегося удара на участке-лужайке (green) с самой короткой травой непосредственно вокруг лунки. — Прим. ред.

можная функция ценности состояния $V^{putter}(s)$ для стратегии, всегда использующей паттер. Ценность терминального состояния *in-the-hole* (мяч в лунке) равна 0. Предполагается, что из любого положения на лужайке вокруг лунки можно сделать патт; эти состояния имеют ценность -1 . Вне этой лужайки патт сделать нельзя, следовательно, отрицательное значение ценности такого состояния больше. Если из некоторого состояния можно достичь лужайки около лунки с помощью одного удара, то это состояние должно иметь ценность, на единицу меньшую ценности состояния на лужайке, т. е. -2 . Для простоты будем считать, что все удары очень точные, но с ограниченной силой. Это дает нам четко очерченную линию, обозначенную -2 на рисунке; все положения между этой линией и лужайкой требуют в точности два удара, чтобы закатить мяч в лунку. Аналогичным образом очерчивается граница зоны с ценностью состояния -3 и остальные линии, показанные на рисунке. Из песочных ловушек с помощью удара выбраться невозможно, таким образом, их ценность равна $-\infty$. В общем случае от площадки для первого удара до лунки необходимо сделать шесть ударов.

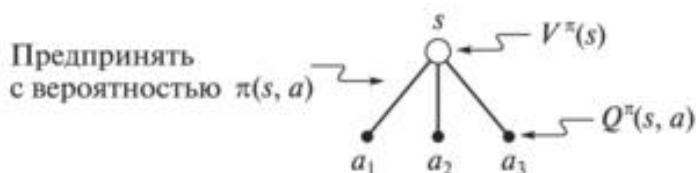
Упражнение 3.8. Каким будет уравнение Беллмана для ценности действия, т. е. для Q^π ? Ценность действия $Q^\pi(s, a)$ в этом уравнении должна быть выражена через ценности действий $Q^\pi(s', a')$, возможных последующих пар состояния—действие (s, a) . В качестве подсказки на рис. 3.4, б, изображена соответствующая этому уравнению диаграмма предшествующих состояний. Напишите ряд уравнений, аналогичных (3.10), но для значений ценности действий.

Упражнение 3.9. Уравнение Беллмана (3.10) должно выполняться для любого состояния в случае функции V^π , показанной на рис. 3.5, б. В качестве примера покажите численно, что это уравнение выполняется для центрального состояния с ценностью $+0,7$, учитывая четыре соседних с ним состояния с ценностями $+2,3$, $+0,4$, $-0,4$ и $+0,7$. (Эти числа приведены с точностью только до одного знака после запятой.)

Упражнение 3.10. В примере с сеткой вознаграждения положительны при выполнении поставленной цели, отрицательны при выходе на границу и равны нулю во всех остальных случаях. Важен ли знак, стоящий перед численными значениями вознаграждений, или имеет значение только разница между ними? Докажите, используя соотношения (3.2), что добавление ко всем вознаграждениям постоянной C приводит к добавлению постоянной K ко всем ценностям состояний и, таким образом, не влияет на относительные значения ценностей состояний при любой стратегии. Выразите K через C и γ .

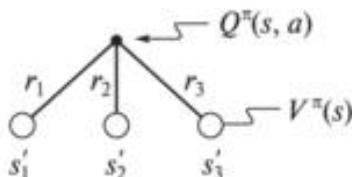
Упражнение 3.11. Рассмотрим теперь добавление постоянной C ко всем вознаграждениям в задании, *составленном из эпизодов*, таком как прохождение лабиринта. Изменится ли что-нибудь в задании или это не окажет никакого влияния, как в рассмотренном выше случае с непрерывным заданием? Объясните и приведите пример.

Упражнение 3.12. Ценность состояния зависит от ценностей действий, возможных в данном состоянии, и от того, насколько вероятно осуществление данного действия при текущей стратегии. Чтобы проиллюстрировать данную задачу, рассмотрим небольшую диаграмму предшествующих состояний с анализируемым состоянием в корневой вершине, рассматривая каждое возможное действие:



Напишите соответствующее такому подходу и данной диаграмме уравнение для ценности состояния корневой вершины $V^\pi(s)$, выраженной через ценность действия ожидаемой краевой вершины $Q^\pi(s_t, a_t)$ при заданном состоянии $s_t = s$. Это ожидание зависит от стратегии π . Затем напишите второе уравнение, в котором ожидаемая ценность в явной форме выражена через $\pi(s, a)$ таким образом, чтобы в уравнение не входила ожидаемая ценность.

Упражнение 3.13. Ценность действия $Q^\pi(s, a)$ может быть разделена на две части: ожидаемое следующее вознаграждение, которое не зависит от стратегии π , и ожидаемая сумма оставшихся вознаграждений, которая зависит от следующего состояния и стратегии. Вновь изобразим небольшую диаграмму предшествующих состояний. На этот раз корневой вершине соответствует действие (пара состояние—действие), а от нее ответвляются возможные следующие действия:



Напишите соответствующее такому подходу и данной диаграмме уравнение для ценности действия $Q^\pi(s, a)$, выраженной через ожидаемое следующее вознаграждение r_{t+1} и ожидаемую ценность следующего

состояния $V^\pi(s_{t+1})$, при заданных состоянии $s_t = s$ и действии $a_t = a$. Затем напишите второе уравнение, в котором ожидаемая ценность в явной форме выражена через вероятность $\mathcal{P}_{ss'}^a$ и вознаграждение $\mathcal{R}_{ss'}^a$, заданные соответственно формулами (3.6) и (3.7) таким образом, чтобы в уравнение не входила ожидаемая ценность.

3.8. Оптимальные функции ценности

Решить задачу обучения с подкреплением — это значит, упрощенно говоря, найти стратегию, которая достигает большого вознаграждения в течение длительного периода времени. Для финитного МППР можно точно определить оптимальную стратегию следующим способом. Функции ценности определяют частичный порядок на множестве стратегий. Говорят, что стратегия π лучше либо равна стратегии π' , если ее ожидаемая выгода лучше или равна выгоде стратегии π' для всех состояний. Другими словами, $\pi \geq \pi'$ тогда и только тогда, когда $V^\pi(s) \geq V^{\pi'}(s)$ для всех состояний $s \in \mathcal{S}$. Всегда существует по крайней мере одна стратегия, которая лучше всех остальных стратегий или равна им. Это и есть *оптимальная стратегия*. Хотя таких стратегий может быть больше одной, будем обозначать все оптимальные стратегии через π^* . Они основываются на использовании одной и той же функции ценности состояния, называемой *оптимальной функцией ценности состояния*, которая обозначается V^* и определяется как

$$V^*(s) = \max_\pi V^\pi(s) \quad (3.11)$$

для всех состояний $s \in \mathcal{S}$.

Оптимальным стратегиям также соответствует одна и та же оптимальная функция ценности действия, которая обозначается Q^* и определяется как

$$Q^*(s, a) = \max_\pi Q^\pi(s, a) \quad (3.12)$$

для всех состояний $s \in \mathcal{S}$ и действий $a \in \mathcal{A}(s)$. Для пары состояни—действие (s, a) эта функция дает ожидаемую выгоду при осуществлении действия a в состоянии s при оптимальной стратегии. Таким образом, можно выразить Q^* через V^* следующим образом:

$$Q^*(s, a) = E\{r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a\}. \quad (3.13)$$

Пример 3.10 (Оптимальная функция ценности применительно к игре в гольф). На нижней части рис. 3.6 показаны контуры возможной оптимальной функции ценности действия $Q^*(s, \text{driver})$. Они

соответствуют ценности каждого состояния, если первый удар делается драйвером, и уже затем делается выбор между паттерном и драйвером, в зависимости от ситуации. Драйвер позволяет сделать сильный удар, но не очень точный. Одним ударом загнать мяч в лунку драйвером можно, только находясь очень близко от лунки; таким образом, контур -1 для функции $Q^*(s, \text{driver})$ охватывает только небольшой участок лужайки возле лунки. Однако с помощью двух ударов можно достичь лунки со значительно большего расстояния, как показывает контур -2 . В этом случае нет необходимости сразу доставлять мяч в область, очерченную контуром -1 , достаточно доставить мяч в ту область лужайки, где можно будет воспользоваться паттерном. Оптимальная функция ценности действия дает значения ценности после выбора *первого* действия, в нашем случае использования драйвера, но в дальнейшем просто выбираются наилучшие действия. Контур -3 находится еще дальше и включает в себя площадку для первого удара. Начиная с этой площадки наилучшей комбинацией действий будет два драйвера и один патт, при которой мяч забивается с трех ударов.

Так как функция ценности V^* зависит от стратегии, она должна удовлетворять условию непротиворечивости, описанному уравнением Беллмана для ценности действий (3.10). Но так как она является оптимальной функцией ценности, это условие может быть записано в особой форме, не связанной с какой-либо конкретной стратегией. Такая форма будет являться уравнением Беллмана для функции V^* или *уравнением оптимальности Беллмана*. Уравнение оптимальности Беллмана выражает тот факт, что ценность состояния при оптимальной стратегии должна быть равна ожидаемой выгоде для лучшего действия в данном состоянии:

$$\begin{aligned} V^*(s) &= \max_{a \in \mathcal{A}(s)} Q^{\pi^*}(s, a) = \max_a E_{\pi^*} \left\{ R_t \mid s_t = s, a_t = a \right\} = \\ &= \max_a E_{\pi^*} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\} = \\ &= \max_a E_{\pi^*} \left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s, a_t = a \right\} = \\ &= \max_a E \left\{ r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a \right\} = \quad (3.14) \\ &= \max_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^*(s')]. \quad (3.15) \end{aligned}$$

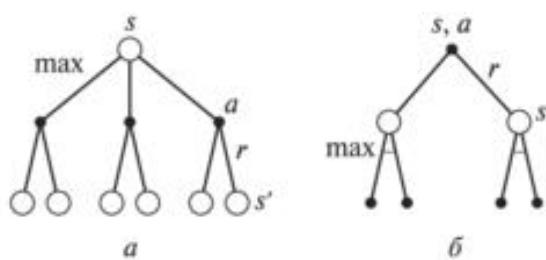


Рис. 3.7. Диаграммы предшествующих состояний для функций (а) V^* и (б) Q^*

Последние два уравнения представляют собой две формы записи уравнения оптимальности Беллмана для функции V^* . Уравнение оптимальности Беллмана для функции Q^* имеет следующий вид:

$$\begin{aligned} Q^*(s, a) &= E\left\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a\right\} = \\ &= \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma \max_{a'} Q^*(s', a')]. \end{aligned}$$

Диаграммы предшествующих состояний на рис. 3.7 графически показывают разброс будущих состояний и действий, входящих в уравнения оптимальности Беллмана для функций V^* и Q^* . Эти диаграммы отличаются от диаграмм для функций V^π и Q^π только тем, что в них добавлены дуги в точках принятия решения агентом, показывающие, что выбором является максимум, а не ожидаемая ценность при некоторой заданной стратегии. На рис. 3.7, а, графически показано уравнение оптимальности Беллмана (3.15).

Для финитного МППР уравнение оптимальности Беллмана (3.15) имеет единственное решение, независимо от принятой стратегии. Уравнение оптимальности Беллмана фактически является системой уравнений, по одному уравнению для каждого состояния. Таким образом, при наличии N состояний имеем N уравнений и N неизвестных. Если известна динамика окружающей среды (т. е. известны значения $\mathcal{P}_{ss'}^a$ и $\mathcal{R}_{ss'}^a$), то в принципе можно решить эту систему уравнений для функции V^* , используя любой из известных методов решения систем нелинейных уравнений. Таким же образом можно решить аналогичную систему уравнений для функции Q^* .

Зная функцию V^* , сравнительно легко определить оптимальную стратегию. Каждому состоянию будет соответствовать одно или несколько действий, при которых достигается максимум в уравнении оптимальности Беллмана. Любая стратегия, которая устанавливает ненулевую вероятность только для этих действий, является оптимальной.

Эту процедуру можно считать одношаговым поиском. Если известна оптимальная функция ценности V^* , то действия, к которым приводит одношаговый поиск, будут оптимальными. Другими словами, любая *жадная* по отношению к оптимальной функции ценности V^* стратегия является оптимальной. Термин «жадный» используется в теории вычислительных систем для описания любого поиска или процедуры принятия решения, которая осуществляет выбор, основанный только на анализе локальных или ближайших событий, и не учитывает вероятность того, что такой выбор может ограничить в будущем доступ к лучшим возможностям. Таким образом, этот термин описывает стратегии, которые выбирают действия, основываясь только на краткосрочных последствиях. Привлекательность функции V^* заключается в том, что при ее применении для оценки краткосрочных последствий действий, особенно одношаговых последствий, жадная политика становится, вообще говоря, оптимальной с точки зрения долгосрочной перспективы, которая и представляет для нас интерес. Это свойство обусловлено тем, что функция V^* уже учитывает последующие вознаграждения для всех возможных вариантов поведения в будущем. Посредством функции V^* ожидаемая долгосрочная выгода превращается в величину, которая локально и непосредственно доступна для любого состояния. Следовательно, одношаговый поиск приводит к действиям, оптимальным в долгосрочном плане.

Если известна функция Q^* , то выбрать оптимальное действие еще проще. В этом случае агенту даже не нужно осуществлять одношаговый поиск: для любого состояния s он может легко найти действие, которое максимизирует величину $Q^*(s, a)$. Функция ценности действия эффективно запоминает результаты всех одношаговых поисков. Она выдает оптимальную ожидаемую долгосрочную выгоду как ценность, которая локально и непосредственно доступна для любой пары состояния—действие. Таким образом, результатом представления функции как зависимости от пары состояния—действие, а не просто от состояния, является то, что оптимальная функция ценности действия позволяет выбирать оптимальные действия, не зная ничего о возможных последующих состояниях и их ценностях, т. е. не зная ничего о динамике окружающей среды.

Пример 3.11 (Уравнение оптимальности Беллмана для робота, собирающего банки). Используя соотношение (3.15), можно получить уравнение оптимальности Беллмана для примера с роботом, собирающим банки. Для краткости обозначим состояния `high` и `low` и действия

`search`, `wait` и `recharge` через `h`, `l`, `s`, `w` и `t` соответственно. Так как имеется только два состояния, уравнение оптимальности Беллмана состоит из двух уравнений. Уравнение для функции $V^*(h)$; может быть записано в следующем виде:

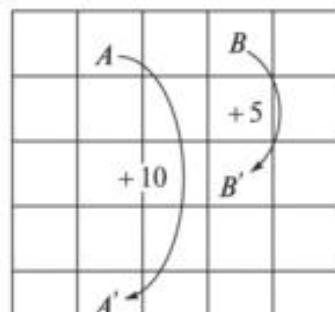
$$\begin{aligned} V^*(h) &= \max \left\{ \begin{array}{l} \mathcal{P}_{hh}^s [\mathcal{R}_{hh}^s + \gamma V^*(h)] + \mathcal{P}_{hl}^s [\mathcal{R}_{hl}^s + \gamma V^*(l)], \\ \mathcal{P}_{hh}^w [\mathcal{R}_{hh}^w + \gamma V^*(h)] + \mathcal{P}_{hl}^w [\mathcal{R}_{hl}^w + \gamma V^*(l)] \end{array} \right\}, = \\ &= \max \left\{ \begin{array}{l} \alpha [\mathcal{R}^s + \gamma V^*(h)] + (1 - \alpha) [\mathcal{R}^w + \gamma V^*(h)], \\ 1 [\mathcal{R}^w + \gamma V^*(h)] + 0 [\mathcal{R}^s + \gamma V^*(l)] \end{array} \right\}, = \\ &= \max \left\{ \begin{array}{l} \mathcal{R}^s + [\alpha V^*(h) + (1 - \alpha)V^*(l)], \\ \mathcal{R}^w + \gamma V^*(h). \end{array} \right\} \end{aligned}$$

Следуя той же самой процедуре для $V^*(l)$, получим уравнение

$$V^*(l) = \max \left\{ \begin{array}{l} \beta \mathcal{R}^s - 3(1 - \beta) + \gamma [(1 - \beta)V^*(h) + \beta V^*(1)], \\ \mathcal{R}^w + \gamma V^*(1), \\ \gamma V^*(h). \end{array} \right\}$$

Любому выбору значений \mathcal{R}^s , \mathcal{R}^w , α , β и γ , при $0 \leq \gamma < 1$, $0 \leq \alpha$, $\beta < 1$, соответствует в точности одна пара чисел $V^*(h)$ и $V^*(l)$, которая одновременно удовлетворяет двум выписанным нелинейным уравнениям.

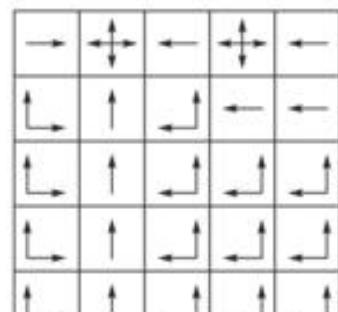
Пример 3.12 (Пример с сеткой). Предположим, что необходимо решить уравнение оптимальности Беллмана для функции V^* для простого задания с сеткой, описанного в примере 3.8 и изображенного еще раз на рис. 3.8, а. Напомним, что состоянию A сопутствует вознаграждение $+10$ и переход в состояние A' , тогда как состоянию B отвечает вознаграждение $+5$ и переход в состояние B' . На рис. 3.8, б, показана оптимальная функция ценности, а на рис. 3.8, в, показаны соответствующие



Сетка
а

22,0	24,4	22,0	19,4	17,5
19,8	22,0	19,8	17,8	16,0
17,8	19,8	17,8	16,0	14,4
16,0	17,8	16,0	14,4	13,0
14,4	16,0	14,4	13,0	11,7

V^*
б



π^*
в

Рис. 3.8. Оптимальные решения в примере с сеткой

оптимальные стратегии. В случаях с несколькими стрелками в одной клетке любая из стратегий является оптимальной.

Решение уравнения оптимальности Беллмана в явном виде представляет собой один из подходов к поиску оптимальной стратегии и, соответственно, к решению задачи обучения с подкреплением. Однако такой подход полезным оказывается достаточно редко. Он аналогичен поиску методом полного перебора, когда рассматриваются все возможности, рассчитываются вероятности их возникновения и их желательность с точки зрения ожидаемых вознаграждений. Это решение основывается по крайней мере на трех предположениях, которые на практике выполняются редко:

- (1) точно известна динамика окружающей среды;
- (2) имеется достаточно вычислительных ресурсов для нахождения решения;
- (3) сигнал состояния обладает марковским свойством.

Для интересующих нас задач в большинстве случаев невозможно получить точное решение, так как эти предположения не выполняются в различных комбинациях. Например, хотя первое и третье предположения не вызывают проблем при реализации игры в нарды, второе из них представляет собой серьезное препятствие. Так как в игре возможны 10^{20} состояний, наиболее быстрым современным компьютерам потребуются миллионы лет, чтобы решить уравнение Беллмана для функции V^* , и то же самое относится к функции Q^* . Поэтому в обучении с подкреплением приходится работать с приближенными решениями.

Многие из методов принятия решений можно трактовать как способы приближенного решения уравнения оптимальности Беллмана. Например, методы эвристического поиска могут рассматриваться как многократное разложение правой части уравнения (3.15) до получения «дерева» возможностей требуемой глубины, после чего следует применение эвристической оценочной функции для аппроксимации функции V^* в «листьях» построенного дерева. (Эвристические методы поиска, такие как A^* , почти всегда базируются на варианте задания, состоящего из эпизодов.) Еще более близкое отношение к уравнению оптимальности имеют методы динамического программирования Беллмана. Многие методы обучения с подкреплением можно, несомненно, считать методами, приближенно решающими уравнение оптимальности Беллмана, используя опытные данные о фактических переходах вместо информации об ожидаемых переходах. Такие методы будут рассмотрены в последующих главах.

Упражнение 3.14. Нарисуйте или опишите оптимальную функцию ценности состояний для примера с игрой в гольф.

Упражнение 3.15. Нарисуйте или опишите контуры оптимальной функции ценности действий при использовании функции $Q^*(s, \text{driver})$ в примере с игрой в гольф.

Упражнение 3.16. Напишите уравнение Беллмана для функции Q^* для робота, собирающего банки.

Упражнение 3.17. Рисунок 3.8 показывает, что оптимальная ценность наилучшего состояния в случае с сеткой равна 24,4 с точностью до одного знака после запятой. Используйте сведения об оптимальной стратегии, а также соотношение (3.2) для записи этой ценности в символьном виде, а затем рассчитайте ее с точностью до третьего знака после запятой.

3.9. Оптимальность и аппроксимация

Мы дали определение оптимальной функции ценности и оптимальной стратегии. Очевидно, агент, придерживающийся оптимальной стратегии, будет действовать очень хорошо, но на практике такое происходит редко. Для интересующих нас видов заданий оптимальная стратегия может быть выработана только ценой недопустимо большого расхода вычислительных ресурсов. Использование точных представлений об оптимальности позволяет сформировать метод обучения, описываемый в этой книге, а также помогает понять теоретические свойства различных алгоритмов обучения. К сожалению, к этому идеалу агент может только приблизиться с той или иной степенью точности. Как уже отмечалось выше, даже если есть полная и точная модель динамики окружающей среды, обычно не представляется возможным просто взять и рассчитать оптимальную стратегию, решая уравнение оптимальности Беллмана. Например, настольные игры, такие как шахматы, являются лишь очень небольшой частью жизненного опыта человека, но, тем не менее, специально разработанные компьютеры не могут рассчитать для них оптимальных ходов. Для агента же очень важно, какие вычислительные мощности ему доступны, в частности какой объем вычислений может быть выполнен за один временной шаг.

Доступная память также является важным сдерживающим фактором. Для построения аппроксимаций функций ценности, стратегий и моделей часто требуются большие объемы памяти. В задачах с небольшим

конечным набором состояний можно строить эти аппроксимации с помощью массивов и таблиц с одной позицией для каждого состояния (или пары состояние—действие). Будем называть такой случай *табличным*, а соответствующие методы — табличными методами. Во многих интересных с практической точки зрения случаях, однако, намного больше состояний, чем можно было бы поместить в таблицу. В таких случаях функции приходится аппроксимировать, используя какие-то более компактные параметризованные представления.

Наше понимание задачи обучения с подкреплением требует использования аппроксимации. В то же время оно предоставляет хорошие возможности для получения требуемых аппроксимаций. Например, при аппроксимации оптимального поведения может быть много состояний, в которых агент окажется с настолько низкой вероятностью, что выбор неоптимальных действий для них окажет лишь незначительное влияние на вознаграждение, получаемое агентом. В частности, программа по игре в нарды, написанная Джерри Тезауро, играет необычайно сильно, несмотря даже на то, что она может принимать очень плохие решения относительно позиций на доске, которые никогда не возникают в игре против профессионалов. Фактически может быть так, что программа TD-Gammon будет принимать плохие решения для большей части игровых состояний. Интерактивная природа обучения с подкреплением позволяет искать приближенные оптимальные стратегии способами, в которых больше внимания уделяется обучению тому, как принимать хорошие решения для часто встречающихся состояний, за счет того, что на состояния, встречающиеся нечасто, тратится меньше усилий. Это ключевое свойство отличает обучение с подкреплением от других методов приближенного решения для МПР.

3.10. Итоги

Перечислим все компоненты задачи обучения с подкреплением, которые были представлены в этой главе. Можно сказать, что обучение с подкреплением является обучением на основе взаимодействия тому, какого поведения придерживаться для достижения цели. *Агент* обучения с подкреплением и *окружающая его среда* взаимодействуют на протяжении последовательности дискретных временных шагов. Подробное описание границы между агентом и средой определяет конкретное задание: *действия* выбираются агентом; *состояния* являются базисом для осуществления этого выбора; а *вознаграждения* служат базисом для оценки сделанного выбора. Все, что находится внутри агента, ему полностью

известно и подконтрольно; все, что находится вне агента, не полностью подконтрольно, но может быть (или не быть) полностью известно. *Стратегия* представляет собой вероятностное правило, согласно которому агент выбирает действие как функцию состояния. Цель агента — максимизировать суммарное вознаграждение, которое он получит за некоторый период времени.

Выгода является функцией будущих вознаграждений, которую агент старается максимизировать. Она имеет несколько определений в зависимости от задания и от того, являются ли *приведенными* отсроченные вознаграждения. Неприведенная форма присуща *задачам, состоящим из эпизодов*, в которых взаимодействие агента — окружающая среда естественным образом разбивается на *эпизоды*; приведенная форма присуща *непрерывным задачам*, в которых взаимодействие не разбивается на эпизоды, а осуществляется непрерывно.

Окружающая среда обладает *марковским свойством*, если ее сигнал состояния кратко резюмирует прошлое без ухудшения способности предсказывать будущее. Марковское свойство редко реализуется в точности, чаще оно выполняется лишь приблизительно. Сигнал состояния следует выбирать или формировать так, чтобы марковское свойство осуществлялось настолько полно, насколько это возможно. В данной книге предполагается, что это уже сделано, и основное внимание уделяется задаче принятия решения: каким образом выбрать действие как функцию любого доступного сигнала состояния. Если марковское свойство выполняется, то окружающая среда определяет *марковский процесс принятия решения* (МППР). *Финитный МППР* — это МППР с конечными наборами состояний и действий. Большая часть теории обучения с подкреплением в настоящее время связана с рассмотрением финитных МППР, но рассматриваемые методы и идеи могут применяться и в более общих случаях.

Функции ценности стратегии присваивают каждому состоянию или паре состояния—действие ожидаемую выгоду от этого состояния или от пары состояния—действие при условии, что агент придерживается данной стратегии. *Оптимальные функции ценности* присваивают каждому состоянию или паре состояния—действие ожидаемую наибольшую выгоду, достижимую при данной стратегии. Стратегия, функция ценности которой оптимальна, называется *оптимальной стратегией*. Тогда как данному МППР соответствует единственная оптимальная функция ценности состояния или пары состояния—действие, оптимальных стратегий может быть много. Любая *жадная* по отношению к оптимальной функции ценности стратегия должна быть оптимальной стратеги-

ей. Уравнения оптимальности Беллмана являются особыми условиями непротиворечивости, которым должны удовлетворять оптимальные функции ценности и которые в принципе могут быть решены для оптимальных функций ценности. С помощью этих функций сравнительно просто определить оптимальную стратегию.

Задача обучения с подкреплением может быть сформулирована многими способами в зависимости от предположений об уровне знания, изначально доступного агенту. В задачах с *полным знанием* в распоряжении агента полная и точная модель динамики окружающей среды. Если окружающая среда представляет собой МППР, то такая модель состоит из одношаговых *вероятностей переходов и ожидаемых вознаграждений* для всех состояний и разрешенных в них действий. В задачах с *неполным знанием* завершенная и идеальная модель окружающей среды недоступна.

Даже если агент располагает полной и точной моделью окружающей среды, он обычно не имеет возможности в полной мере ею воспользоваться, поскольку не располагает ресурсами, требуемыми для выполнения необходимых вычислений на каждом временном шаге. Доступный объем памяти также может ограничивать возможности агента. Память может потребоваться для построения хороших аппроксимаций функций ценности, стратегий и моделей. В большинстве случаев, представляющих практический интерес, число имеющихся состояний слишком велико, чтобы можно было воспользоваться точными табличными представлениями, поэтому приходится прибегать к аппроксимациям.

Использование точных представлений об оптимальности позволяет построить метод обучения, описываемый в этой книге, а также помогает понять теоретические свойства различных алгоритмов обучения. К сожалению, к этому идеалу агент обучения с подкреплением может только приблизиться с той или иной степенью точности. В обучении с подкреплением особое место занимают задачи, в которых оптимальные решения получить нельзя, поэтому следует искать их приближенные варианты.

3.11. Библиографические и исторические справки

Большое значение для задачи обучения с подкреплением имеет идея марковского процесса принятия решений (МППР), заимствованная из области оптимального управления. Эта тема, а также значительное влияние на обучение с подкреплением идей из психологии описаны в кратком историческом обзоре в гл. 1. Обучение с подкреплением применительно к МППР

фокусируется на вопросах аппроксимации и неполной информации применительно к серьезным и крупным прикладным задачам. МППР и задача обучения с подкреплением довольно слабо связаны с традиционными задачами обучения и принятия решений в области искусственного интеллекта. Тем не менее формулировки МППР для планирования и принятия решений изучаются сейчас активно и всесторонне. Формулировки МППР являются более общими, чем те, которые использовались ранее в области искусственного интеллекта, что позволяет работать с целями и неопределенностями более общего вида.

Определенное влияние на наше представление задачи обучения с подкреплением оказала работа [Watkins (1989)].

3.1. Пример с биореактором основан на работах [Ungar (1990)] и [Miller and Williams (1992)]. Пример робота, собирающего банки, появился благодаря работе [Jonathan Cornell (1989)], в которой описана похожая конструкция.

3.3–4. Термины *состоящее из эпизодов* и *непрерывное* задание нельзя назвать общепринятыми в литературе по МППР, где обычно различают три типа заданий:

- (1) задания с конечным числом шагов, в которых взаимодействие завершается после некоторого *фиксированного* числа временных шагов;
- (2) задания с неопределенным числом шагов, в которых взаимодействие может длиться как угодно долго, но в итоге все же должно прекратиться;
- (3) задания с неограниченным числом шагов, в которых взаимодействие не прекращается.

Рассматриваемые в книге состоящие из эпизодов задания и непрерывные задания аналогичны заданиям с неопределенным и с неограниченным числом шагов соответственно, хотя и следует учитывать различие в природе взаимодействия для этих видов заданий. Это различие представляется более фундаментальным, чем различие между целевыми функциями, выражаемое в обычных терминах. Часто в заданиях, состоящих из эпизодов, используется целевая функция для случая неопределенного числа шагов, а в непрерывных задачах — целевая функция в варианте для неограниченного числа шагов, однако будем считать это скорее совпадением, нежели фундаментальным различием.

Пример с балансировкой стержня взят из работ [Michie and Chambers (1968)] и [Barto, Sutton, and Anderson (1983)].

3.5. Дальнейшее рассмотрение понятия состояния можно найти в работе [Minsky (1967)].

3.6. Теория МППР рассматривается, например, в работах [Bertsekas (1995)], [Ross (1983)], [White (1969)] и [Whittle (1982, 1983)]. Она изучается также в рамках вероятностного оптимального управления, где методы *адаптивного* оптимального управления очень тесно связаны с методами обучения с подкреплением (см. [Kumar (1985)]; [Kumar and Varaiya (1986)]).

Теория МППР явилась результатом попыток решить задачу принятия последовательностей решений в условиях неопределенности, когда каждое решение может зависеть от предыдущих решений и их последствий. Иногда эту задачу называют теорией многошагового или последовательного процесса принятия решений, которая уходит корнями в статистическую литературу по последовательной выборке, начиная с работ [Thompson (1933, 1934)] и [Robbins (1952)]. На них были ссылки в гл. 2 в связи с задачами о бандите (которые являются прообразом МППР, если формулируются как задачи с множеством состояний).

Наиболее ранним известным нам примером, в котором обучение с подкреплением рассматривалось с использованием формализма МППР, является работа [Andreae (1969b)], где предпринимается попытка сформировать единый взгляд на обучающиеся машины. В работе [Witten and Corbin (1973)] описаны эксперименты с системой обучения с подкреплением, которая позднее была проанализирована в статье [Witten (1977)] с использованием формализма МППР. Без упоминания в явной форме МППР в работе [Werbos (1977)] предложены методы приближенного решения для задач вероятностного оптимального управления, имеющие отношение к современным методам обучения с подкреплением (см. также [Werbos (1982, 1987, 1988, 1989, 1992)]). Хотя в то время идеи Вербоса не получили широкого признания, впоследствии они оказались востребованными, поскольку в них подчеркивалась важность задач оптимального управления, решаемых приближенно в самых различных областях, включая искусственный интеллект. Наиболее существенное влияние на интеграцию обучения с подкреплением и МППР оказала диссертация [Watkins (1989)]. Принятый в ней подход к обучению с подкреплением с использованием формализма МППР получил широкое признание.

Наше определение динамики вознаграждений для МППР с помощью величины $\mathcal{R}_{ss'}^a$ является несколько необычным. В литературе по МППР более принято описывать динамику вознаграждений, используя ожидаемое следующее вознаграждение при заданном текущем состоянии и действии, т. е. величину

$$\mathcal{R}_s^a = E\{r_{t+1} | s_t = s, a_t = a\}.$$

Эти две величины связаны следующим образом:

$$\mathcal{R}_s^a = \sum_{s'} \mathcal{P}_{ss'}^a \mathcal{R}_{ss'}^a.$$

В традиционной теории МППР величина $\mathcal{R}_{ss'}^a$ всегда появляется под знаком суммы, как в этой формуле, и, следовательно, проще пользоваться величиной \mathcal{R}_s^a . Тем не менее в обучении с подкреплением намного чаще приходится иметь дело с отдельными фактическими или выборочными результатами. Мы обнаружили, что обозначение $\mathcal{R}_{ss'}^a$ концептуально проще и, соответственно, легче осознается в процессе преподавания методов обучения с подкреплением.

3.7–8. Мы определяли значения ценности, учитывая долгосрочную перспективу. Такой подход имеет длинную историю. Отображение состояний в числовые значения, определяющие долгосрочные последствия управленических решений, представляет собой одну из важнейших составных частей теории оптимального управления, которая разрабатывалась в 1950-х гг. как развитие методов классической механики, созданных в девятнадцатом веке и основанных на понятии функции состояния (см. [Schultz and Melsa (1967)]). При описании того, как можно запрограммировать компьютер для игры в шахматы, в работе [Shannon (1950)] предлагается использовать оценочную функцию, которая учитывает долгосрочные преимущества и недостатки шахматных позиций.

Обучающий Q -алгоритм из диссертации [Watkins (1989)] для оценивания значений функции Q^* (гл. 6) сделал функции ценности действий важным элементом обучения с подкреплением. По этой причине такие функции часто называются Q -функциями. Но идея функции ценности действия появилась намного раньше. В работе [Shannon (1950)] было высказано предположение, что функция $h(P, M)$ могла бы быть использована программой игры в шахматы для решения, заслуживает ли исследования ход M в позиции P . Можно считать, что система MENACE, предложенная в работе [Michie (1961, 1963)], а также система BOXES, описанная в работе [Michie and Chambers (1968)], находят значения функции ценности действия. В классической физике функция, определяемая принципом Гамильтона, является функцией ценности действия; ньютоновская динамика является жаждой по отношению к этой функции (см. [Goldstein, 1957]). Функции ценности действий играют также главную роль в теоретической трактовке процессов принятия решений, основанной на понятии сжимающего отображения [Denardo (1967)].

То, что мы называем уравнением оптимальности Беллмана для функции V^* , было впервые представлено Ричардом Беллманом в работе [Bellman (1957a)] и называлось «основным функциональным уравнением». Аналогом уравнения оптимальности Беллмана для задач с непрерывным временем и состояниями является уравнение Гамильтона-Якоби-Беллмана (или чаще просто уравнение Гамильтона-Якоби), корнями уходящее в классическую физику (см. [Schultz and Melsa (1967)]).

Пример с игрой в гольф был предложен Крисом Уоткинсом.

II

Фундаментальные методы решения

В этой части книги мы опишем три фундаментальных класса методов обучения с подкреплением: динамическое программирование, методы Монте-Карло и методы временных различий. Все эти методы подходят для решения задачи в ее полном варианте, включая отсроченные вознаграждения.

Каждый класс методов имеет свои сильные и слабые стороны. Методы динамического программирования хорошо разработаны математически, но требуют точной и полной модели окружающей среды. Методы Монте-Карло (методы статистических испытаний) не требуют модели окружающей среды, они концептуально просты, но не подходят для пошаговых инкрементных вычислений. Наконец, методы временных различий не требуют модели и являются полностью инкрементными, однако они сложнее для анализа. Кроме того, все методы отличаются друг от друга эффективностью и скоростью сходимости. В третьей части книги будет рассмотрено, каким образом эти методы можно сочетать, чтобы использовать преимущества каждого из них.

4

Динамическое программирование

Термин «динамическое программирование» (ДП) относится к набору алгоритмов для построения оптимальной стратегии при заданной идеальной модели окружающей среды как марковского процесса принятия решений (МППР). Использование классических алгоритмов ДП в обучении с подкреплением весьма ограничено как в связи с предположением об идеальности модели, так и в связи с большими затратами на вычисления; тем не менее эти алгоритмы важны с теоретической точки зрения. Динамическое программирование является основой для понимания методов, далее представленных в этой книге. Фактически все эти методы служат тем же целям, что и ДП, но с меньшим потребным объемом вычислений и без предположения об идеальности модели окружающей среды.

Начиная с этой главы мы обычно предполагаем, что окружающая среда является финитным МППР, т. е. предполагается, что множества ее состояний и действий \mathcal{S} и $\mathcal{A}(s)$, $s \in \mathcal{S}$, конечны и что ее динамика задана множеством вероятностей переходов

$$\mathcal{P}_{ss'}^a = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$

и ожидаемых вознаграждений, получаемых немедленно

$$\mathcal{R}_{ss'}^a = E\{r_{t+1} | a_t = a, s_t = s, s_{t+1} = s'\},$$

для всех $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$ и $s' \in \mathcal{S}^+$ (множество \mathcal{S}^+ — это множество \mathcal{S} , дополненное заключительным состоянием, если рассматривается задача, состоящая из эпизодов). Хотя идеи ДП могут применяться и в задачах с непрерывными пространствами состояний и действий, точные решения при этом возможны только в отдельных частных случаях. Распространенным способом получения приближенных решений для задач с непрерывными состояниями и действиями является дискретизация пространств состояний и действий с последующим применением методов ДП для задач с конечным числом состояний. В гл. 8 будут рассмотрены методы, которые применяются для решения непрерывных задач и позволяют значительно расширить границы применимости данного подхода.

Основной идеей ДП и обучения с подкреплением в целом является использование функции ценности для создания и структуризации поиска хороших стратегий. В этой главе будет показано, как можно использовать ДП для вычисления функции ценности,

определенной в гл. 3. Как уже говорилось, оптимальную стратегию можно легко получить, если задать функции ценности V^* или Q^* , удовлетворяющие уравнению оптимальности Беллмана:

$$\begin{aligned} V^*(s) &= \max_a E\{r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a\} = \\ &= \max_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^*(s')] \end{aligned} \quad (4.1)$$

или

$$\begin{aligned} Q^*(s, a) &= E\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a\} = \\ &= \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma \max_{a'} Q^*(s', a')] \end{aligned} \quad (4.2)$$

для всех $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$ и $s' \in \mathcal{S}^+$. Как будет показано, алгоритмы ДП можно получить преобразованием уравнений Беллмана, наподобие представленных выше, в последовательность операторов присваивания, т. е. в правила корректировки, позволяющие получать все более точные приближения искомой функции ценности.

4.1. Оценка стратегии

Для начала рассмотрим, как вычислить функцию ценности состояния V^π для произвольной стратегии π . В литературе по процессам принятия решения эти действия именуют *оценкой стратегии*. Кроме того, эту задачу еще называют *задачей прогнозирования*. Как уже отмечалось в гл. 3, для всех $s \in \mathcal{S}$

$$\begin{aligned} V^\pi(s) &= E_\pi\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \mid s_t = s\} = \\ &= E_\pi\{r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s\} = \end{aligned} \quad (4.3)$$

$$= \sum_a \pi(s, a) \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^\pi(s')], \quad (4.4)$$

где $\pi(a, s)$ — вероятность осуществления действия a в состоянии s при стратегии π . Нижний индекс π указывает, что ожидания обусловлены текущей стратегией π . Условие существования и единственности функции V^π выполняется до тех пор, пока либо $\gamma < 1$, либо гарантировано завершение процесса из всех состояний при стратегии π .

Если динамика окружающей среды полностью известна, то система (4.4) является системой $|S|$ совместных линейных уравнений с $|S|$ неизвестными (величинами $V^\pi(s)$, $s \in \mathcal{S}$). В принципе данную систему можно решить непосредственно, хотя это и будет трудоемкая в вычислительном плане задача. В нашем случае для этой цели лучше всего

подходят итерационные методы. Рассмотрим последовательность приближенных функций ценности V_0, V_1, V_2, \dots , отображающих множество \mathcal{S}^+ на \mathbb{R} . Начальное приближение V_0 будем выбирать произвольно (лишь в случае, когда существует заключительное состояние, ему должна быть присвоена нулевая ценность), а каждое последующее приближение получаем, используя уравнение Беллмана для функции V^π (4.3) в форме правила уточнения:

$$\begin{aligned} V_{k+1}(s) &= E_\pi \{ r_{t+1} + \gamma V_k(s_{t+1}) \mid s_t = s \} = \\ &= \sum_a \pi(s, a) \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V_k(s')] \end{aligned} \quad (4.5)$$

для всех $s \in \mathcal{S}$. Очевидно, $V_k = V^\pi$ является неподвижной точкой для этого уточняющего правила, так как из уравнения Беллмана для функции V^π следует существование данного равенства. Действительно, можно показать, что, в общем, последовательность $\{V_k\}$ сходится к V^π при $k \rightarrow \infty$ при тех же условиях, при которых гарантируется существование функции V^π . Этот алгоритм получил название *итеративного оценивания стратегии*.

Для получения любого последующего приближения V_{k+1} на основе V_k для каждого состояния s при итеративном оценивании стратегии выполняется одна и та же операция: старая ценность состояния s заменяется на новую, полученную из старых ценностей состояний, следующих за s , и ожидаемых вознаграждений, получаемых немедленно, в соответствии со всеми одношаговыми переходами, возможными при данной оцениваемой стратегии. Такой тип операций будем называть *полным дублированием*. На каждой итерации однократно выполняется дублирование ценности каждого действия для последующего получения новой приближенной функции ценности V_{k+1} . Существует несколько различных видов полного дублирования, в зависимости от того, дублируется ли состояние (как в рассматриваемом сейчас случае) или пара состояния—действие, а также каким конкретно способом сочетаются расчетные ценности последующих состояний. Любое дублирование, выполненное при помощи алгоритмов динамического программирования, называется полным дублированием, так как базируется на всех возможных следующих состояниях, а не на каком-то одном из них. Суть дублирования может быть выражена посредством уравнения, как это было сделано выше, или же через диаграммы предшествующих состояний, представленные в гл. 3. Например, на рис. 3.4, *a*, изображена диаграмма предшествующих состояний, соответствующая полному дублированию, применяемому при итеративном оценивании стратегии.

Чтобы написать компьютерную программу последовательного типа, реализующую итеративное оценивание стратегии согласно соотношению (4.5), необходимо создать два массива — один для старых ценности $V_k(s)$ и второй для новых ценностей $V_{k+1}(s)$. Таким способом значения новых ценностей можно вычислить одно за другим на основе старых ценностей, сохраняя при этом и старые значения ценностей. Было бы, конечно, легче использовать лишь один массив и обновлять ценности «на месте», чтобы каждое новое значение ценности заменяло старое значение. Тогда, в зависимости от порядка, в котором дублировались состояния, иногда новые значения ценности будут использоваться вместо старых значений в правой части соотношения (4.5). Этот немного измененный алгоритм также сходится к функции V^π ; фактически, данный вариант алгоритма обычно сходится быстрее, чем вариант с двумя массивами, так как новые значения в нем используются сразу, как только они становятся доступными. Подобного рода дублирование рассматривается нами как *прогонка* в пространстве состояний. Для такого алгоритма замещения порядок, в котором состояния заменяются при прогонке, в значительной степени влияет на скорость сходимости. При рассмотрении алгоритма ДП обычно подразумевают его вариант с замещением.

Еще один важный элемент реализации рассматриваемого алгоритма связан с завершением его работы. Формально процесс итеративного оценивания стратегии сходится только в пределе, однако на практике он должен завершаться раньше. Чтобы определить момент остановки данного процесса, обычно проверяют значение величины $\max_{s \in \mathcal{S}} |V_{k+1}(s) - V_k(s)|$ после каждой прогонки. Остановка производится, когда эта величина достаточно мала. На рис. 4.1 приведен полный

```

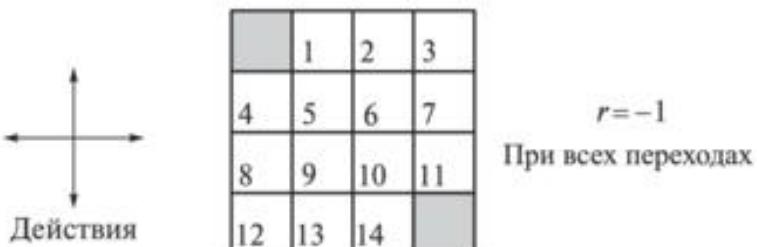
Ввести  $\pi$ , оцениваемую стратегию
Инициализировать  $V(s) = 0$  для всех  $s \in \mathcal{S}^+$ 
Повторять
     $\Delta \leftarrow 0$ 
    Для каждого  $s \in \mathcal{S}$ :
         $v \leftarrow V(s)$ 
         $V(s) \leftarrow \sum_a \pi(s, a) \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V(s')]$ 
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ 
    до тех пор, пока  $\Delta < \theta$  (малое положительное число)
    Выход  $V \approx V^\pi$ 

```

Рис. 4.1. Итеративная оценка стратегии

алгоритм для итеративного оценивания стратегии с таким критерием остановки.

Пример 4.1. Рассмотрим представленную ниже сетку размером 4×4 .



Нетерминальными состояниями здесь являются $\mathcal{S} = \{1, 2, \dots, 14\}$. В каждом состоянии возможны четыре действия

$$\mathcal{A} = \{\text{up}, \text{down}, \text{right}, \text{left}\},$$

которые вызывают соответствующие переходы из одного состояния в другое, не считая тех действий, которые должны выводить агента за пределы сетки и которые, фактически, оставляют состояния неизмененными. Таким образом, например,

$$\mathcal{P}_{5,6}^{\text{right}} = 1, \quad \mathcal{P}_{5,10}^{\text{right}} = 0 \quad \text{и} \quad \mathcal{P}_{7,7}^{\text{right}} = 1.$$

Это неприведенная задача, состоящая из эпизодов. Вознаграждение равно -1 при всех переходах до достижения заключительного состояния. Заключительные состояния на рисунке обозначены затененными квадратами (несмотря на то, что эти квадраты находятся в двух разных местах, это, по сути, одно и то же состояние). Функция ожидаемых вознаграждений будет равна соответственно $\mathcal{R}_{ss'}^a = -1$ для всех состояний s, s' и действий a . Предположим, что агент придерживается равновероятной произвольной стратегии (все действия равновероятны). На рис. 4.2 слева показана последовательность функций ценности $\{V_k\}$, рассчитанных методом итеративного оценивания стратегии. Завершающей оценкой является, фактически, V^π , которая в этом случае ставит в соответствие каждому состоянию величину, численно равную ожидаемому числу шагов из данного состояния до завершающего, взятую с обратным знаком.

Упражнение 4.1. Если π является равновероятной произвольной стратегией, чему равно $Q^\pi(11, \text{down})$? Чему равно $Q^\pi(7, \text{down})$?

Упражнение 4.2. Пусть в сетку сразу после состояния 13 добавлено новое состояние 15 и действия для него `left`, `up`, `right` и `down` перемещают агента в состояния 12, 13, 14 и 15 соответственно. Предположим,

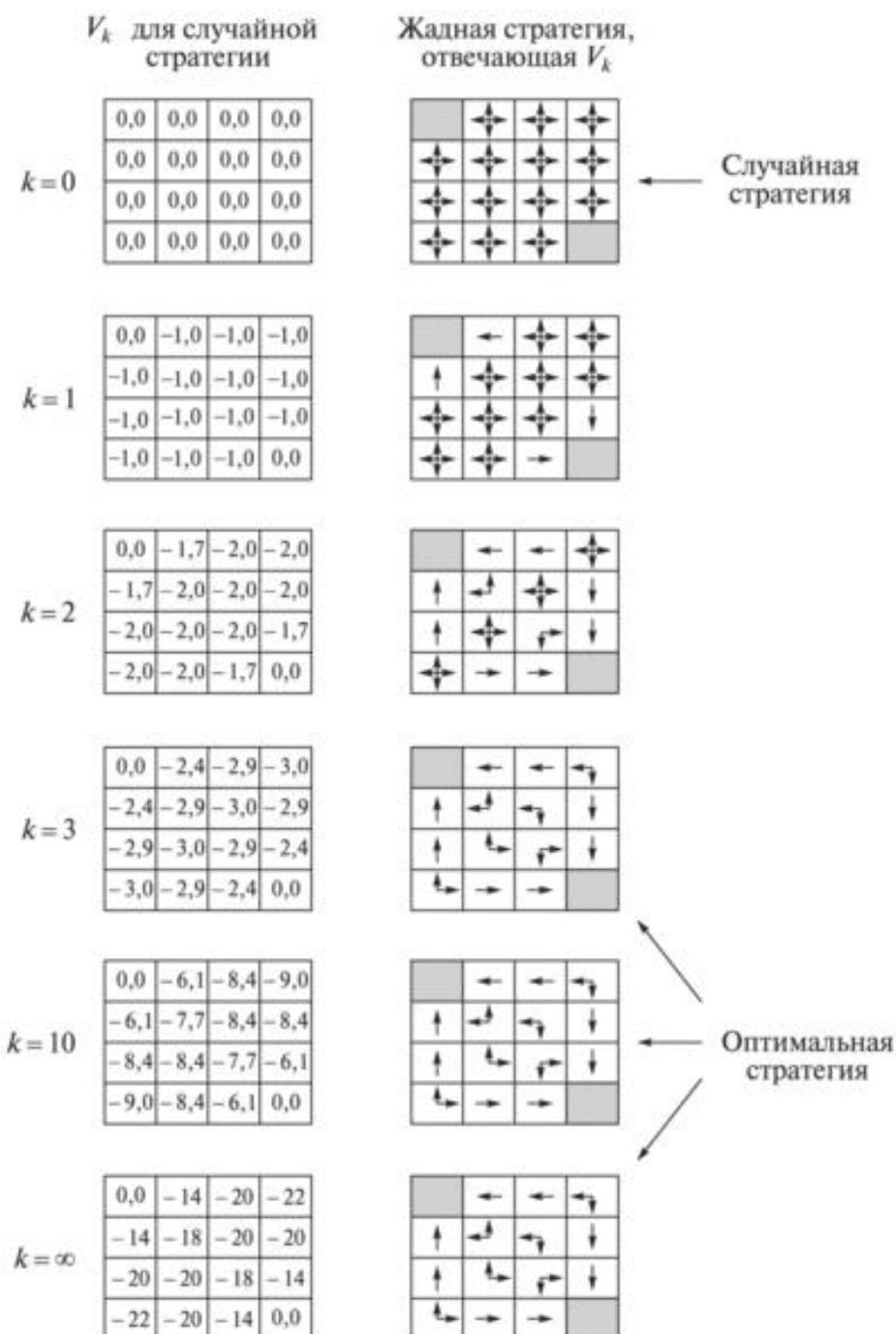


Рис. 4.2. Сходимость процесса итеративного оценивания стратегии в небольшой сетке. В левой колонке изображена последовательность приближений функции ценности состояния для случайной стратегии (все действия равновероятны). В правой колонке приведена последовательность жадных стратегий, отвечающих расчетным величинам функции ценности (стрелками показаны все действия, при которых достигается максимум). Гарантировать здесь можно лишь то, что последняя стратегия будет лучше случайной, но в данном случае и последняя, и все стратегии после третьей итерации оптимальны

что переходы из исходных состояний остались прежними. Каким в этом случае будет значение $V^\pi(15)$ для равновероятной случайной стратегии? Теперь пусть динамика состояния 13 изменена так, что действие `down` в состоянии 13 перемещает агента в новое состояние 15. Каким в этом случае будет значение $V^\pi(15)$ для равновероятной случайной стратегии?

Упражнение 4.3. Какой вид имеют аналоги уравнений (4.3), (4.4) и (4.5) для функции ценности действия Q^π и ее приближений в виде последовательности функций Q_0, Q_1, Q_2, \dots ?

4.2. Улучшение стратегии

Функция ценности для некоторой стратегии вычисляется для того, чтобы найти с использованием полученных результатов новую, улучшенную стратегию. Пусть нами определена функция ценности V^π для произвольной детерминированной стратегии π . Для некоторого состояния s нам хотелось бы знать, нужно ли менять стратегию для отдельно взятого действия $a \neq \pi(s)$. Нам известно, насколько выгодна текущая стратегия в состоянии s , т. е. $V^\pi(s)$, но не окажется ли более выгодной новая стратегия? Для того чтобы ответить на этот вопрос, можно рассмотреть выбор действия a в состоянии s и затем следовать существующей стратегии π . Ценность такого поведения равна

$$\begin{aligned} Q^\pi(s, a) &= E_\pi \{ r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s, a_t = a \} = \\ &= \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^\pi(s')]. \end{aligned} \quad (4.6)$$

Ключевой критерий здесь — больше или меньше эта величина, чем $V^\pi(s)$. Если она больше, т. е. если выгоднее один раз выбрать a в состоянии s и в дальнейшем следовать стратегии π , чем всегда следовать стратегии π , то можно ожидать, что выгоднее будет каждый раз выбирать a , находясь в состоянии s , и что новая стратегия в целом будет выгоднее.

Это утверждение является частным случаем общего результата, называемого *теоремой об улучшении стратегии*. Пусть π и π' — пара детерминированных стратегий таких, что для всех $s \in \mathcal{S}$ выполняется неравенство

$$Q_\pi(s, \pi'(s)) \geq V^\pi(s). \quad (4.7)$$

Тогда стратегия π' должна быть не хуже, чем π , т. е. она должна принести такую же или большую ожидаемую выгоду при всех состояниях $s \in \mathcal{S}$:

$$V_{\pi'}(s) \geq V^\pi(s). \quad (4.8)$$

Более того, если неравенство (4.7) строго выполняется при любом состоянии, то и неравенство (4.8) должно быть строгим при одном или больше состояниях. Полученный результат можно применить, в частности, к двум стратегиям, которые были рассмотрены в предыдущем абзаце, т. е. к исходной детерминированной стратегии π и измененной стратегии π' , которая идентична стратегии π , за исключением того, что $\pi'(s) = a \neq \pi(s)$. Очевидно, что неравенство (4.7) выполняется для всех состояний кроме s . Таким образом, если $Q^\pi(s, a) > V^\pi(s)$, то измененная стратегия действительно лучше, чем π .

Идею, лежащую в основе доказательства теоремы об улучшении стратегии, легко понять. Начиная с неравенства (4.7) будем осуществлять разложение функции Q^π , используя соотношение (4.6), и проверять неравенство (4.7) до тех пор, пока не будет получена функция $V^{\pi'}(s)$:

$$\begin{aligned} V_\pi(s) &\leq Q_\pi(s, \pi'(s)) = E_{\pi'}\{r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s\} \leq \\ &\leq E_{\pi'}\{r_{t+1} + \gamma Q^\pi(s_{t+1}, \pi'(s_{t+1})) \mid s_t = s\} = \\ &= E_{\pi'}\{r_{t+1} + \gamma E_{\pi'}\{r_{t+2} + \gamma V^\pi(s_{t+2})\} \mid s_t = s\} = \\ &= E_{\pi'}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V^\pi(s_{t+3}) \mid s_t = s\} \leq \\ &\vdots \\ &\leq E_{\pi'}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \dots \mid s_t = s\} = V^{\pi'}(s). \end{aligned}$$

Мы видели уже, как, имея стратегию и ее функцию ценности, легко можно оценить изменения стратегии для отдельного состояния при выборе конкретного действия. В развитие этого естественным будет рассмотреть изменения во *всех* состояниях при осуществлении *всех* возможных действий, когда в каждом состоянии выбирается действие, наилучшее согласно $Q^\pi(s, a)$. Другими словами, рассмотреть новую жадную стратегию π' , заданную следующим образом:

$$\begin{aligned} \pi'(s) &= \arg \max_a Q^\pi(s, a) = \\ &= \arg \max_a E\{r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s, a_t = a\} = \quad (4.9) \\ &= \arg \max_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^\pi(s')], \end{aligned}$$

где $\arg \max_a$ обозначает ценность действия a , при котором рассматриваемое выражение достигает своего максимума (со связями, разорванными произвольным образом). Жадная стратегия выбирает действия, которые кажутся наилучшими в краткосрочной перспективе — после просчета одного шага вперед — согласно функции V^π . По построению

жадная стратегия удовлетворяет условиям теоремы об улучшении стратегии (4.7). Таким образом, известно, что жадная стратегия выгодна в такой же или большей степени, что и исходная. Процесс формирования новой стратегии, улучшающей исходную стратегию, делая ее жадной или приблизительно жадной по отношению к функции ценности исходной стратегии, называется *улучшением стратегии*.

Предположим, что новая жадная стратегия π' настолько же (но не более) выгодна, что и старая стратегия π . Тогда $V^\pi = V^{\pi'}$, и из равенства (4.9) следует, что для всех состояний $s \in \mathcal{S}$ выполняется соотношение

$$\begin{aligned} V^{\pi'}(s) &= \max_a E\{r_{t+1} + \gamma V^{\pi'}(s_{t+1}) \mid s_t = s, a_t = a\} = \\ &= \max_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^{\pi'}(s')]. \end{aligned}$$

Но это уравнение оптимальности Беллмана (4.1), и, следовательно, функция $V^{\pi'}$ должна совпадать с V^* , а стратегии π и π' должны быть оптимальными. Таким образом, улучшение стратегии должно давать другую стратегию, которая будет строго лучше исходной, за исключением тех случаев, когда исходная стратегия уже является оптимальной.

До сих пор в этом разделе рассматривался частный случай детерминированных стратегий. В общем случае вероятностная стратегия π задает вероятности $\pi(s, a)$ осуществления действия a в каждом состоянии s . Не вдаваясь в подробности, скажем лишь, что все идеи этого раздела легко распространяются на случай вероятностных стратегий. В частности, обобщая теорему об улучшении стратегии на вероятностный случай, получим:

$$Q^\pi(s, \pi'(s)) = \sum_a \pi'(s, a) Q^\pi(s, a).$$

Кроме того, если некоторые шаги при улучшении стратегии, такие как (4.9), приводят к неопределенности, т. е. если существует несколько действий, при которых достигается максимум, то в вероятностном случае нет необходимости выбирать среди них одно действие. Вместо этого каждому максимизирующему действию может быть присвоена доля вероятности того, что оно будет осуществлено в новой жадной стратегии. Разрешено любое пропорциональное распределение, до тех пор пока всем субмаксимальным действиям присвоена нулевая вероятность.

На рис. 4.2, в последнем ряду, показан пример улучшения стратегии для вероятностного случая. Здесь первоначальная стратегия π является равновероятной случайной стратегией, а новая стратегия π' является жадной по отношению к функции V^π . Функция ценности V^π пред-

ставлена нижней левой диаграммой, а набор возможных стратегий π' — нижней правой диаграммой. Состояния с несколькими стрелками на диаграмме π' являются состояниями, в которых максимум в соотношении (4.9) достигается при нескольких действиях; распределение вероятностей среди этих действий допускается в любой пропорции. Функцию ценности любой такой стратегии $V^{\pi'}(s)$ можно найти методом подбора. Она равна либо -1 , -2 , либо -3 при всех состояниях $s \in \mathcal{S}$, тогда как $V^\pi(s)$ не превышает -14 . Таким образом, $V^{\pi'}(s) \geq V^\pi(s)$ для всех состояний $s \in \mathcal{S}$, что показывает улучшение стратегии. Несмотря на то что в данном случае новая стратегия π' оказалась оптимальной, в общем случае гарантировано только улучшение стратегии.

4.3. Итерация по стратегиям

Как только стратегия π улучшена с использованием V^π (улучшенный вариант стратегии обозначим через π'), можно вычислить $V^{\pi'}$, после чего получить стратегию π'' , которая будет еще лучше. Таким образом, можно получить последовательность монотонно улучшаемых стратегий и функций ценности:

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^*,$$

где \xrightarrow{E} обозначает *оценивание* стратегии, а \xrightarrow{I} — *улучшение* стратегии. Каждая стратегия гарантированно является строго улучшенной по отношению к предыдущему ее варианту (если только предыдущая стратегия не является уже оптимальной). Так как финитный МППР имеет конечное число стратегий, этот процесс должен сходиться к оптимальной стратегии и оптимальной функции ценности за конечное число итераций.

Такой способ поиска оптимальной стратегии называется *итерацией по стратегиям*. Полный алгоритм этого поиска приведен на рис. 4.3. Отметим, что каждое оценивание стратегии представляет собой итерационный вычислительный процесс, который начинается с использованием функции ценности для предыдущей стратегии. Это обычно приводит к резкому увеличению скорости сходимости процесса оценивания стратегии (предположительно потому, что функция ценности меняется незначительно от одной стратегии к другой).

Итерация по стратегиям зачастую требует удивительно малого числа шагов. Пример такой ситуации показан на рис. 4.2. Нижняя левая диаграмма изображает функцию ценности для равновероятной случайной стратегии, а нижняя правая диаграмма изображает жадную стратегию

1. Инициализация
 $V(s) \in \mathcal{R}$ и $\pi(s) \in \mathcal{A}(s)$ произвольно для всех $s \in \mathcal{S}$
2. Оценка стратегии
 Повторять

$$\Delta \leftarrow 0$$
 Для каждого $s \in \mathcal{S}$:

$$v \leftarrow V(s)$$

$$V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} [\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$
 пока $\Delta < \theta$ (малое положительное число)
3. Улучшение стратегии

Стратегия устойчива \leftarrow истинно

Для каждого $s \in \mathcal{S}$:

$$b \leftarrow \pi(s)$$

$$\pi(s) \leftarrow \arg \max_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V(s')]$$

Если $b \neq \pi(s)$, то стратегия устойчива \leftarrow ложно

Если стратегия устойчива, то stop; иначе перейти к шагу 2.

Рис. 4.3. Итерация по стратегиям (с использованием итеративного оценивания стратегии) для V^* . В операторе $\arg \max$ на шаге 3 предполагается, что связи разрываются в последовательном порядке

для данной функции ценности. Теорема об улучшении стратегии гарантирует, что эти стратегии лучше исходной случайной стратегии. В рассматриваемом случае эти стратегии не просто лучше — они являются оптимальными, достигая заключительного состояния за минимальное количество шагов. В данном примере итерация по стратегиям находит оптимальную стратегию после первой же итерации.

Пример 4.2 (Компания «Автомобили напрокат»). Джек управляет двумя местными отделениями общенациональной компании, выдающей автомобили напрокат. Каждый день некоторое число клиентов приезжает в одно из отделений, чтобы арендовать автомобиль. Если в наличии имеется свободный автомобиль, Джек сдает его в аренду и получает от национальной компании 10\$. Если в данном отделении нет свободных машин, то выгода упущена. Машины становятся доступными для следующей аренды через день после их возврата. Чтобы быть уверенным, что машины доступны там, где они необходимы, Джек может за ночь перегнать машину от одного отделения к другому, это будет стоить 2\$ за машину. Будем полагать, что число требуемых и возвращенных ма-

шин в каждом отделении является пуссоновской случайной величиной. Это означает, что вероятность того, что рассматриваемое число равно n , равна $(\lambda^n/n!)e^{-\lambda}$, где λ — математическое ожидание. Предположим, что λ равно 3 и 4 для требующихся машин в первом и втором отделении и 3 и 2 для возвращенных. Чтобы несколько упростить задачу, будем считать, что в каждом отделении количество машин не может превышать 20 (каждая машина сверх этого числа возвращается общенациональной компании и, таким образом, в задаче не учитывается) и что за одну ночь от одного отделения к другому можно перегнать не более 5 машин. Примем $\gamma = 0,9$ за значение коэффициента приведения и сформулируем задачу как непрерывный ограниченный МППР, где временными шагами являются дни, состоянием является число машин в каждом отделении на конец дня и действием является общее число машин, перегоняемых между отделениями за ночь. На рис. 4.4 изобра-

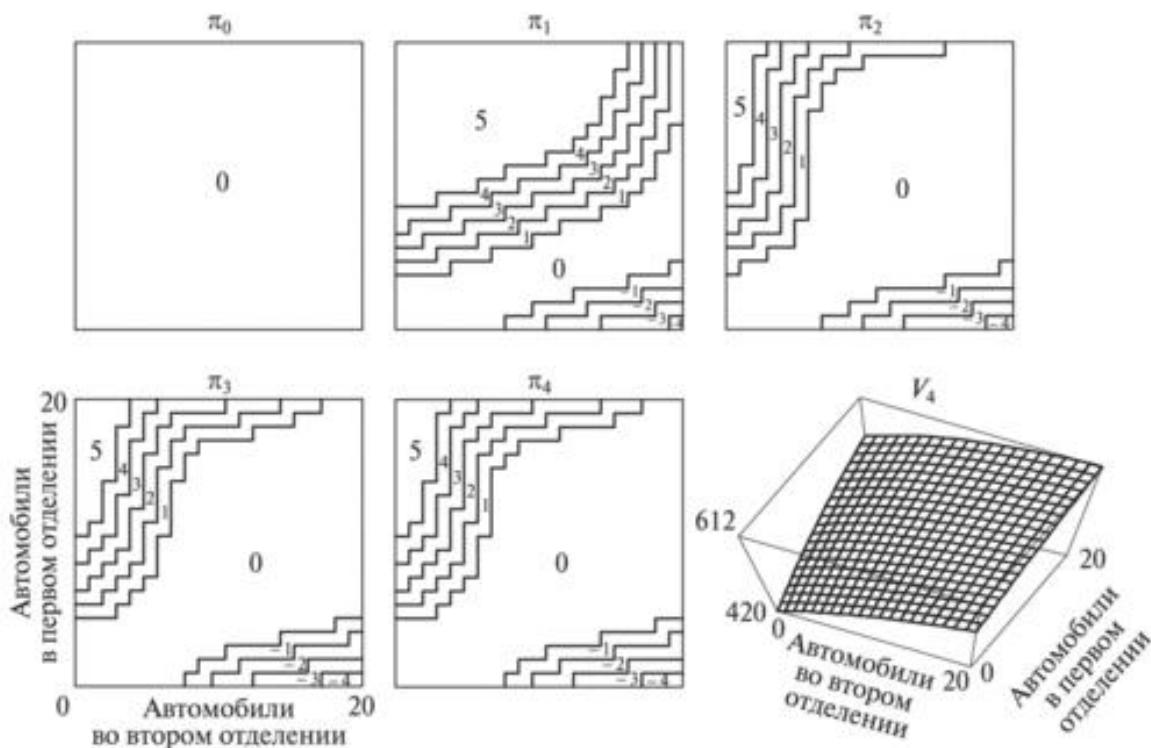


Рис. 4.4. Последовательность стратегий, найденных с помощью итерации по стратегиям в задаче об аренде автомобилей, а также функция ценности состояния в окончательном виде. На первых пяти диаграммах для каждого числа автомобилей в каждом отделении на конец дня показано число машин, которые необходимо перегнать из первого отделения во второе (отрицательное число означает, что машины перегоняются из второго отделения в первое). Каждая последующая стратегия строго лучше предыдущей стратегии, а последняя стратегия является оптимальной

жена последовательность стратегий, найденных итерационно, начиная со стратегии, согласно которой машины не перегоняются никогда.

Упражнение 4.4 [Программирование]. Напишите программу для итерации по стратегиям и решите задачу об аренде автомобилей со следующими изменениями. Одна из сотрудниц первого отделения каждый вечер едет домой на автобусе, а ее дом расположен около второго отделения. Она была бы рада перегнать одну из машин во второе отделение бесплатно. Каждый дополнительный перегон машины по-прежнему стоит 2\$, как и перегон машины в противоположном направлении. Кроме того, в каждом отделении число парковочных мест ограничено. Если за ночь в отделение пригнали более 10 машин (после всех перемещений машин), то возникает необходимость в использовании второй автостоянки стоимостью 4\$ (независимо от того, сколько на ней находится машин). Такого типа нелинейности и произвольная динамика часто возникают в реальных задачах. Подобного рода задачи непросто решать с помощью методов оптимизации, отличных от динамического программирования. Для проверки полученной программы сначала воспроизведите результаты исходной задачи. Если используемому компьютеру не хватает ресурсов для решения полной задачи, сократите в два раза количество автомобилей.

Упражнение 4.5. Как определить итерацию по стратегиям для ценностей действий? Напишите полный алгоритм для вычисления функции Q^* , аналогичный алгоритму, приведенному на рис. 4.3 для вычисления функции V^* . Обратите особое внимание на это упражнение, поскольку используемые здесь идеи будут применяться далее в этой книге.

Упражнение 4.6. Предположим, что можно рассматривать только ε -гибкие стратегии, когда вероятность выбора каждого действия в каждом состоянии s не меньше $\varepsilon/|\mathcal{A}(s)|$. Опишите на качественном уровне изменения, которые потребуются на каждом из шагов 3, 2 и 1 (именно в таком порядке) в алгоритме итерации по стратегиям для функции V^* (рис. 4.3).

4.4. Итерация по ценностям

Одним из недостатков итерации по стратегиям является то, что каждая итерация включает в себя процесс оценивания стратегии, который сам по себе может потребовать длительных итерационных вычислений, требующих большого числа прогонок в множестве состояний. Если оценивание стратегии происходит итерационно, то сходимость к точному

значению функции V^π имеет место только в пределе. Необходимо добиваться точной сходимости или можно остановиться на более ранней стадии? Представленный на рис. 4.2 пример показывает, что процесс оценивания стратегии можно сократить. В этом примере оценивание стратегии уже после трех итераций не влияет на соответствующую жадную стратегию.

Фактически, число шагов при оценивании стратегии с помощью итерации по стратегиям может быть сокращено несколькими способами с гарантией сходимости процесса итерации по стратегиям. В одном важном частном случае процесс оценивания стратегии прекращается после всего лишь одной прогонки (одного дублирования каждого состояния). Такой алгоритм называется *итерацией по ценностям*. Он может быть записан в виде очень простой операции дублирования, которая объединяет улучшение стратегии и усеченный вариант процесса оценивания стратегии:

$$\begin{aligned} V_{k+1}(s) &= \max_a E\{r_{t+1} + \gamma V_k(s_{t+1}) \mid s_t = s, a_t = a\} = \\ &= \max_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V_k(s')] \end{aligned} \quad (4.10)$$

для всех состояний $s \in \mathcal{S}$. Для произвольной функции V_0 можно показать, что последовательность $\{V_k\}$ сходится к V^* при тех же условиях, которые гарантируют существование V^* .

Чтобы лучше понять процесс итерации по ценностям, обратимся к уравнению оптимальности Беллмана (4.1). Отметим, что итерацию по стратегиям можно легко получить путем превращения уравнения оптимальности Беллмана в правило уточнения. Отметим также, что дублирование при итерации по ценностям во всем повторяет дублирование при оценивании стратегии (4.5) за исключением того, что в первом случае требуется достижение максимума по всем действиям. Чтобы увидеть схожесть алгоритмов, сравните их диаграммы предшествующих состояний: на рис. 3.4, а, показана такая диаграмма для оценивания стратегии, а на рис. 3.7, а, — для итерации по ценностям. Эти алгоритмы реализуют естественные операции дублирования при вычислении функций V^π и V^* .

Рассмотрим в заключение, как завершается итерация по ценностям. Как и итерация по стратегиям, итерация по ценностям формально требует бесконечного числа повторений для обеспечения сходимости к точному значению функции V^* . На практике остановка процесса производится, когда функция ценности после очередного прогона изменяет-

Инициализировать V произвольно, например $V(s) = 0$ для всех $s \in \mathcal{S}^+$

Повторять

$$\Delta \leftarrow 0$$

Для каждого $s \in \mathcal{S}$:

$$v \leftarrow V(s)$$

$$V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

пока $\Delta < \theta$ (малое положительное число)

Вывести детерминированную стратегию π , такую что

$$\pi(s) = \arg \max_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V(s')]$$

Рис. 4.5. Итерация по ценности

ся на незначительную величину. На рис. 4.5 приведен полный алгоритм итерации по стратегиям с условием завершения данного вида.

Итерация по ценностям на каждом из ее шагов эффективно сочетает прогон оценивания стратегии и прогон улучшения стратегии. Более быстрая сходимость обычно достигается за счет того, что между прогонами улучшения стратегии вклиниваются многократные прогоны оценивания стратегии. В общем случае весь класс усеченных алгоритмов итерации по стратегиям можно интерпретировать как последовательность прогонов, в части которых используется дублирование оценки стратегии, а в других применяется дублирование итерации по стратегиям. Поскольку единственным различием между этими вариантами дублирования является использование операции \max в соотношении (4.10), это означает, что в некоторые прогоны оценивания стратегии добавляется такая операция. Все эти алгоритмы сходятся к оптимальной стратегии для приведенного финитного МППР.

Пример 4.3 (Задача игрока). Игрок делает ставки на результаты бросаний монетки. Если выпадает орел, игрок выигрывает столько долларов, сколько он поставил на это подбрасывание монетки; если выпадает решка, игрок проигрывает свою ставку. Игра заканчивается, когда игрок выигрывает, достигая заданного уровня в 100\$, или когда игрок проигрывает, теряя все деньги. При каждом подбрасывании игрок должен решить, на какую сумму в целых числах долларов сделать ставку. Данная задача может быть сформулирована как неприведенный финитный МППР, состоящий из эпизодов. Состоянием процесса является капитал игрока, $s \in \{1, 2, \dots, 99\}$, действиями являются став-

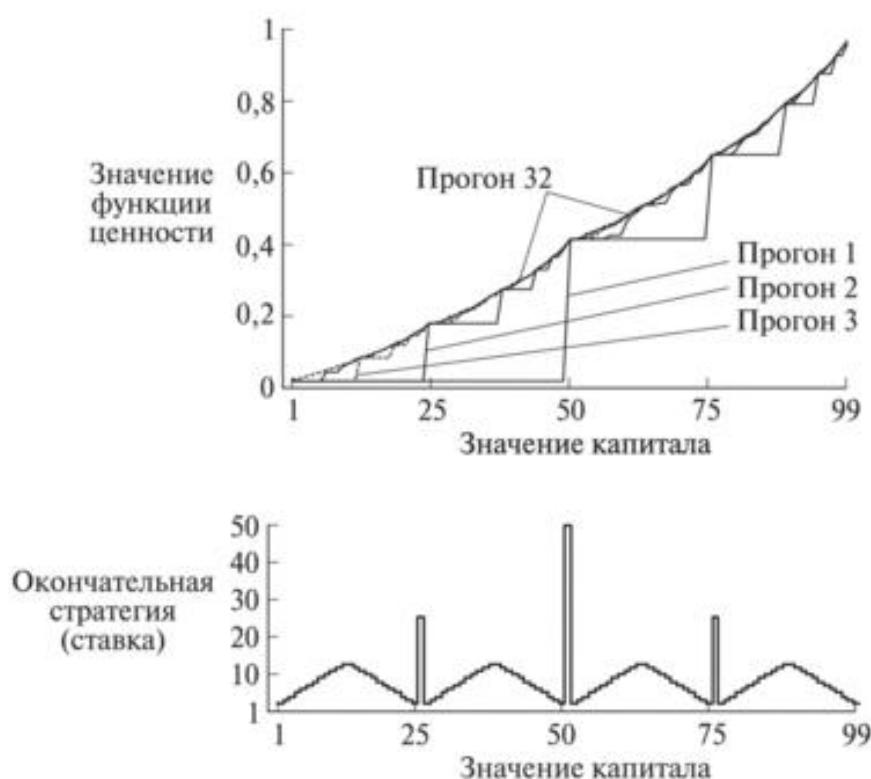


Рис. 4.6. Решение задачи игрока для $p = 0,4$. На верхнем графике показаны функции ценности, найденные при помощи последовательных прогонов итерации по ценностям. На нижнем графике показана окончательная стратегия

ки, $a \in \{1, 2, \dots, \min(s, 100 - s)\}$. Вознаграждение равно нулю при всех переходах кроме тех, при которых игрок достигает своей цели; тогда вознаграждение равно $+1$. Функция ценности состояния затем дает вероятность выигрыша в каждом состоянии. Стратегия является отображением уровней значения капитала на ставки. Оптимальная стратегия максимизирует вероятность достижения цели. Обозначим p вероятность выпадения орла. Если она известна, то задача полностью определена и может быть решена, к примеру, итерацией по ценностям. На рис. 4.6 показано изменение функции ценности при последовательных прогонах итерации по ценностям и показана окончательная стратегия, найденная для случая $p = 0,4$.

Упражнение 4.7. Почему оптимальная стратегия для задачи игрока имеет такой необычный вид? В частности, согласно этой стратегии, при наличии капитала в 50\$ он весь ставится на одно бросание монетки, но в случае с капиталом в 51\$ такого не происходит. Почему такая стратегия выгодна?

Упражнение 4.8 [Программирование]. Реализуйте итерацию по ценостям для задачи игрока и решите эту задачу для $p = 0,25$ и $p = 0,55$. В программировании может оказаться удобным введение двух фиктивных состояний, соответствующих завершению игры при капитале, равном 0 и 100. Ценности этих состояний равны 0 и 1 соответственно. Постройте график полученных результатов, как на рис. 4.6. Будут ли получаемые результаты стабильными при $\theta \rightarrow 0$?

Упражнение 4.9. Каким будет аналог дублирования итерации по ценостям (4.10) для ценостей действий $Q_{k+1}(s, a)$?

4.5. Асинхронное динамическое программирование

Главный недостаток методов ДП, которые рассматривались до сих пор, состоит в том, что они включают в себя операции над всем множеством состояний МППР, т. е. они требуют прогонов для всего множества состояний. Если множество состояний велико, то даже единственный прогон может стать непозволительно трудоемким процессом. Например, игра в нарды имеет более 10^{20} состояний. Даже если бы удалось осуществить дублирование итерации по ценостям со скоростью миллион состояний в секунду, для завершения единственного прогона потребуется более миллиона лет.

Вместо таких алгоритмов можно предложить *асинхронные* алгоритмы ДП, которые также являются итерационными, но не основываются на систематических прогонах всего множества состояний. Данные алгоритмы выполняют дублирование ценостей состояний в любом порядке, используя любые доступные ценности других действий. Ценности некоторых действий могут быть скопированы несколько раз за то время, когда ценности других действий — только однажды. Тем не менее для хорошей сходимости асинхронный алгоритм должен провести дублирование всех состояний: он не может пренебречь каким-либо состоянием на некоторой стадии вычислений. Асинхронные алгоритмы ДП дают значительную свободу при выборе состояний, к которым будет применено дублирование.

Например, один из вариантов асинхронной итерации по ценостям производит дублирование ценности только для одного состояния s_k на каждом шаге k , используя дублирование итерации по ценостям (4.10). Если $0 \leq \gamma < 1$, то асимптотическая сходимость к функции V^* гарантирована при единственном условии, которое заключается в том, что все состояния в последовательности $\{s_k\}$ встречаются бесконечное число раз. (В случае неприведенной задачи, состоящей из эпизо-

дов, возможно существование некоторых последовательностей выполнения операций дублирования, не обеспечивающих сходимости, но этого сравнительно легко избежать.) Аналогичным образом можно сочетать дублирование для оценивания стратегии и для итерации по ценностям для получения некоторого варианта асинхронной усеченной итерации по стратегиям. Хотя детальное рассмотрение данного алгоритма, а также и еще более необычных алгоритмов ДП выходит за рамки данной книги, ясно, что несколько различных вариантов дублирования формируют своего рода стандартные блоки, которые можно достаточно широко применять в разнообразных алгоритмах ДП, не использующих прогон.

Конечно, тот факт, что в некотором алгоритме не используется прогон, не обязательно означает, что потребуется меньший объем вычислений. Это всего лишь значит, что алгоритм не должен защищаться на каком-либо безнадежно длинном прогоне, прежде чем он сможет добиться улучшения стратегии. Можно попробовать использовать преимущество такой гибкости алгоритма для ускорения его работы, выбирая состояния, к которым будет применяться дублирование. Можно регулировать процесс дублирования таким образом, чтобы информация распространялась от состояния к состоянию более рациональным образом. Возможно, ценности некоторых состояний не должны дублироваться так же часто, как и ценности других состояний. Иногда можно даже полностью пренебречь дублированием некоторых состояний, если они не имеют прямого отношения к оптимальной модели поведения. Некоторые из таких идей будут рассмотрены в гл. 9.

Кроме того, асинхронные алгоритмы позволяют легче сочетать вычисления с взаимодействием в режиме реального времени. Для вычисления заданного МППР итерационный алгоритм ДП задействуется *одновременно с выполнением агентом пробных действий для изучения данного МППР*. Опыт агента может быть использован для определения состояний, к которым алгоритм ДП применяет дублирование. В то же время вновь поступающая от алгоритма ДП информация о ценностях и стратегии может управлять процессом принятия решений агентом. Например, можно применять дублирование к состояниям, когда агент попадает в них. Это дает возможность *сфокусировать* используемое в алгоритме ДП дублирование на той части множества состояний, которая наиболее значима для агента. Такого типа фокусирование — одна из тем, которые постоянно обсуждаются в обучении с подкреплением.

4.6. Обобщенная итерация по стратегиям

Итерация по стратегиям состоит из двух одновременно протекающих взаимодействующих процессов, один из которых обеспечивает совместимость функции ценности с текущей стратегией (оценивание стратегии), а другой делает стратегию жадной по отношению к текущей функции ценности (улучшение стратегии). В итерации по стратегиям эти два процесса чередуются, один процесс начинается по завершении другого, хотя в этом и нет реальной необходимости. В итерации по ценностям, например между двумя улучшениями стратегии, имеет место только одна итерация оценивания стратегии. В асинхронных методах ДП процессы оценивания и улучшения разбиваются на еще более мелкие перемежающиеся составляющие. В некоторых случаях один процесс сменяет другой по завершении улучшения всего лишь одного состояния. Но каким бы образом оба процесса ни изменяли состояния, конечный результат обычно один и тот же — сходимость к оптимальной функции ценности и к оптимальной стратегии.

В термине *обобщенная итерация по стратегиям* (ОИС) подчеркивается общая идея взаимодействия процессов оценки и улучшения стратегии, вне зависимости от уровня структурированности и других характеристик, присущих двум данным процессам. Практически все методы обучения с подкреплением хорошо описываются с помощью ОИП, т. е. все эти методы имеют поддающиеся выявлению стратегии и функции ценности, со стратегиями, постоянно совершенствующимися относительно функций ценности и с функцией ценности, стремящейся к функции ценности для стратегии. Общая схема для ОИС приведена на рис. 4.7.

Легко заметить, что если процессы оценивания и улучшения стабилизируются, т. е. перестают порождать изменения, то полученные функция ценности и стратегия должны быть оптимальными. Функция ценности стабилизируется, только когда она согласуется с текущей стратегией, а стратегия стабилизируется, только когда она становится жадной по отношению к текущей функции ценности. Таким образом, оба процесса стабилизируются, только когда най-

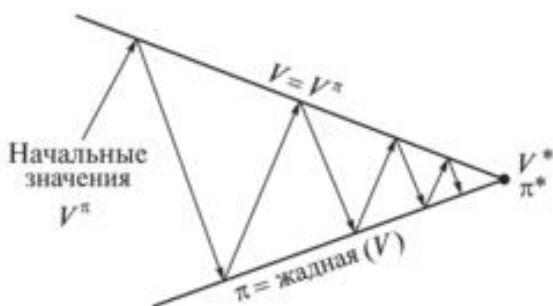


Рис. 4.7. Обобщенная итерация по стратегиям: функции ценности и стратегии взаимодействуют до тех пор, пока они не становятся оптимальными и, таким образом, совместимыми друг с другом

дена стратегия, жадная по отношению к своей собственной оценочной функции. При этом удовлетворяется уравнение оптимальности Беллмана (4.1), а стратегия и функция ценности являются оптимальными.

Процессы оценивания и улучшения в ОИС могут рассматриваться одновременно как конкурирующие и взаимодействующие. Они конкурируют в том смысле, что стремятся к изменениям в противоположных направлениях. При попытке сделать стратегию жадной по отношению к функции ценности обычно получается функция ценности, неадекватная изменившейся стратегии. Попытка согласовать функцию ценности со стратегией обычно оборачивается тем, что стратегия перестает быть жадной. Тем не менее эти два процесса взаимодействуют, находясь в конце концов единственное совместное решение: оптимальную функцию ценности и оптимальную стратегию.

Взаимодействие между процессами оценивания и улучшения в ОИС можно представлять посредством двух ограничивающих условий или целей, к примеру двух линий в двумерном пространстве:



Несмотря на то что в реальности геометрия намного сложнее, данный рисунок демонстрирует, что происходит в реальном случае. Каждый процесс подталкивает функцию ценности или стратегию к одной из линий, представляющих решение применительно к одной из двух поставленных целей. Цели взаимодействуют, так как эти две линии не ортогональны. Движение к одной из целей сопровождается удалением от другой. Тем не менее объединенный процесс неизбежно приводит к достижению общей цели — оптимальности стратегии и функции ценности. Стрелки на этой диаграмме соответствуют поведению итерации по стратегиям, при котором каждая итерация ведет систему к полному выполнению одной из двух поставленных целей. В ОИС возможно осуществление более мелких, неполных шагов по направлению к каждой цели. Так или иначе, два процесса совместно достигают общей оптимальности, при том что ни один из них не пытается достичь ее напрямую.

4.7. Эффективность динамического программирования

ДП может оказаться не очень практичным для решения очень больших задач, но в сравнении с другими методами вычисления МППР методы ДП достаточно эффективны. Если не принимать в расчет некоторые технические подробности, время (в наихудшем случае), необходимое некоторым методам ДП для нахождения оптимальной стратегии, оценивается полиномом, степень которого равна суммарному числу состояний и действий. Если n и m обозначают число состояний и действий, то количество вычислительных операций, необходимое методу ДП, меньше значения некоторой полиномиальной функции степени $(n+m)$. Метод ДП гарантированно найдет оптимальную стратегию за полиномиальное время, даже если общее число (детерминированных) стратегий равно m^n . В этом смысле ДП экспоненциально быстрее любого прямого поиска, возможного в пространстве стратегий, так как прямой поиск вынужден будет полностью исследовать каждую стратегию, чтобы дать аналогичные гарантии получения результата. Для работы с МППР можно также использовать методы линейного программирования, и в некоторых случаях гарантии их сходимости даже в наихудшем случае лучше, чем у методов ДП. Но методы линейного программирования оказываются непрактичными при числе состояний, намного меньшем (порядка 100), чем в случае с ДП. Для очень больших задач подходят только методы ДП.

Иногда считается, что область применения ДП ограничивает «проклятие размерности» (Беллман, 1957а), т. е. тот факт, что число возможных состояний растет экспоненциально с увеличением количества переменных, описывающих эти состояния. Большие множества состояний действительно создают некоторые трудности, но они являются неотъемлемым свойством решаемой задачи, а не особенностью именно методов ДП. Фактически методы ДП лучше подходят к работе с большими множествами состояний, чем конкурирующие методы, такие как прямой поиск или линейное программирование.

На практике, учитывая возможности современных компьютеров, методы ДП могут быть использованы для решения МППР с миллионами состояний. Широко применяется итерация по стратегиям, и итерация по ценностям, причем какой из этих методов лучше, в общем случае сказать нельзя. При решении прикладных задач эти методы обычно сходятся намного быстрее, чем в теоретически наихудших случаях для них, особенно если они начинаются с хороших начальных функций ценности и стратегий.

В задачах с большими множествами состояний зачастую предпочтительнее использовать *асинхронные* методы ДП. Для завершения даже одного прогона синхронному методу требуются вычисления и память для каждого из состояний. Для некоторых задач требующиеся в таком случае объемы вычислений и памяти непрактичны, хотя задача по-прежнему остается потенциально решаемой, поскольку с оптимальным решением связано лишь сравнительно небольшое число состояний. В подобных случаях можно применить асинхронные методы и другие разновидности ОИП, которые находят хорошие или оптимальные стратегии намного быстрее синхронных методов.

4.8. Итоги

В данной главе были рассмотрены основные идеи и алгоритмы динамического программирования в связи с финитными МППР. *Оценивание стратегии* обычно связано с итеративным вычислением функции ценности для заданной стратегии, а *улучшение стратегии* — с вычислением более приемлемой стратегии при заданной функции ценности для нее. Объединяя два этих вычислительных процесса, получают *итерацию по стратегиям* и *итерацию по ценностям* — два наиболее популярных метода в ДП. Каждый из них может быть использован как вполне надежное средство вычисления оптимальных стратегий и функций ценности для финитного МППР при наличии полного знания относительно МППР.

В основе классических методов ДП лежит прогон во множестве состояний, выполняющий *полное дублирование* каждого состояния. Каждое дублирование обновляет ценность одного состояния, основываясь на ценостях всех возможных последующих состояний и вероятностей их возникновения. Полное дублирование тесно связано с уравнениями Беллмана: размерность их несколько больше, чем при использовании формы записи с помощью операторов присваивания. Когда дублирование уже не приводит к каким-либо изменениям в значениях ценности, можно утверждать, что достигнута сходимость к ценостям, удовлетворяющая соответствующим уравнениям Беллмана. Так же как есть четыре основных функции ценности (V^π, V^*, Q^π и Q^*), существуют и четыре соответствующих уравнения Беллмана, а также четыре соответствующих варианта полного дублирования. Наглядной демонстрацией процесса дублирования является *диаграмма предшествующих состояний*.

Для понимания сути методов ДП и, фактически, всех методов обучения с подкреплением их рассматривают как *обобщенную итерацию по стратегиям* (ОИС). ОИС представляет собой объединяющую идею для двух взаимодействующих процессов, связанных с приближенной стратегией и приближенной функцией ценности. В одном из этих процессов стратегия считается заданной и выполняется оценивание стратегии за счет изменения функции ценности и приближения ее к истинной функции ценности для данной стратегии. В другом процессе предполагается, что функция ценности задана и осуществляется улучшение стратегии за счет изменения ее, пытаясь сделать лучше, полагая при этом, что функция ценности является функцией ценности данной стратегии. Несмотря на то что каждый процесс изменяет базис другого процесса, в целом вместе они работают на достижение совместного решения — получение стратегии и функции ценности, которые не изменяются ни одним из процессов и, следовательно, являются оптимальными. В некоторых случаях можно доказать сходимость ОИС, особенно для классических методов ДП, представленных в этой главе. В других случаях сходимость не доказана, но идея ОИП, тем не менее, помогает лучше понять данные методы.

В случае с методами ДП нет необходимости всегда использовать прогон по всему множеству состояний. Асинхронные методы ДП являются итерационными методами замещения, выполняющие дублирование состояний в произвольном порядке, возможно определенном случайно и используя устаревшую информацию. Многие из этих методов можно рассматривать как формы ОИП с разбиением итерационного процесса на достаточно мелкие фрагменты.

Отметим в заключение еще одно специфическое свойство методов ДП. Все они обновляют оценки ценностей состояний, основываясь на оценках ценностей последующих состояний, т. е. обновляют оценки, основываясь на других оценках. Такая идея называется *раскруткой*. Идея раскрутки используется во многих методах обучения с подкреплением, даже в тех, которые в отличие от ДП не требуют знания полной и точной модели окружающей среды. В следующей главе будут рассмотрены методы обучения с подкреплением, которые не требуют знания модели окружающей среды и не используют раскрутку. Эти ключевые признаки и свойства раздельны, но вместе они могут давать интересные комбинации.

4.9. Библиографические и исторические справки

Термин «динамическое программирование» был предложен Беллманом [Bellman (1957a)], который показал, как эти методы можно применять для решения широкого диапазона задач. Всестороннее рассмотрение ДП можно найти во многих работах, включая [Bertsekas (1995)], [Bertsekas and Tsitsiklis (1996)], [Dreyfus and Law (1977)], [Ross (1983)], [White (1969)] и [Whittle (1982, 1983)]. Наш интерес к ДП ограничивается его применением к рассмотрению МППР, но ДП можно использовать также и для решения других видов задач. В работе [Kumar and Kanal (1988)] предлагается более общая точка зрения на ДП.

Насколько нам известно, впервые ДП в связи с обучением с подкреплением было упомянуто в комментариях Марвина Мински (1961 г.) к программе Сэмюэля для игры в шашки. В примечании Мински отметил возможность применения ДП к задачам, в которых процесс дублирования, предложенный Сэмюэлем, можно реализовать в замкнутой аналитической форме. Это замечание, по-видимому, ввело в заблуждение исследователей в области искусственного интеллекта, которые предположили, что область применения ДП ограничивается аналитически решаемыми задачами и, следовательно, ДП совершенно неприменимо в задачах искусственного интеллекта. В работе [Andreae (1969b)] говорится о ДП в контексте обучения с подкреплением, а именно применительно к итерации по стратегиям, несмотря на то что ДП не связывается с алгоритмами обучения. В статье [Werbos (1977)] предложен метод приближенного ДП, получивший наименование «эвристическое динамическое программирование», в котором подчеркивалось значение методов градиентного спуска для задач с непрерывными состояниями (см. также [Werbos (1982, 1987, 1988, 1989, 1992)]). Эти методы тесно связаны с алгоритмами обучения с подкреплением, рассматриваемыми в данной книге. В работе [Watkins (1989)] в явном виде связывается обучение с подкреплением с ДП, при этом класс методов обучения с подкреплением характеризуется как «динамическое программирование в приращениях».

4.1–4. Данные разделы описывают хорошо известные алгоритмы ДП, которые содержатся в любой из вышеупомянутых работ общего характера по динамическому программированию. Теорема об улучшении стратегии и алгоритм итерации по стратегиям обязаны своим происхождением работам Беллмана [Bellman (1957a)] и Ховарда [Howard (1960)]. Влияние на наше изложение материала оказала точка зрения на способы улучшения стратегии, представленная в диссертации [Watkins (1989)]. Наше рассмотрение итерации по ценностям как усеченной формы итерации по стратегиям имеет в своей основе подход, предложенный в работе [Puterman and Shin (1978)], где введен класс алгоритмов, названный *модифицированной итерацией по стратегиям*, который включал в себя итерацию по стратегиям

и итерацию по ценностям в качестве частных случаев. Анализ, показывающий, как с помощью итерации по ценностям можно найти оптимальную стратегию, был представлен в [Bertsekas (1987)].

Итеративная оценка стратегии является примером классического алгоритма последовательных приближений для решения системы линейных уравнений. Вариант алгоритма, в котором используются два массива, один из которых содержит старые значения ценностей, а второй — их обновленные значения, часто называется алгоритмом *типа Якоби*, имея в виду классическое применение Якоби этого метода. Иногда его также называют *синхронным* алгоритмом, так как он может выполнять операции параллельно, используя работу нескольких процессоров, одновременно обновляющих значения ценностей отдельных состояний, получающих входные сигналы от других процессоров. Второй массив требуется для моделирования таких параллельных вычислений на последовательном процессоре. Вариант с замещением для данного алгоритма обычно называется алгоритмом *типа Гаусса—Зейделя*, поскольку он напоминает классический алгоритм Гаусса—Зейделя для решения систем линейных уравнений. Кроме итеративного оценивания стратегии, описанными выше методами могут быть реализованы и другие алгоритмы ДП. В работе [Bertsekas and Tsitsiklis (1989)] дается описание значительного числа разновидностей таких алгоритмов, а также рассматриваются различия в их характеристиках.

- 4.5. Асинхронные алгоритмы обязаны своим появлением статьям [Bertsekas (1982, 1983)], где они названы также распределенными алгоритмами ДП. Первоначально асинхронное ДП разрабатывалось для реализации с помощью мультипроцессорной системы с задержками в линиях связи между процессорами и без глобального генератора синхронизирующих импульсов. Данные алгоритмы интенсивно исследовались в работе [Bertsekas and Tsitsiklis (1989)]. Алгоритмы ДП типа Якоби и типа Гаусса—Зейделя являются частными случаями асинхронных алгоритмов. В статье [Williams and Baird (1990)] описаны алгоритмы ДП, которые асинхронны на более низком уровне, чем рассмотренные ранее: сами операции дублирования разбиваются на шаги, которые могут совершаться асинхронно.
- 4.6. В основе данного раздела, написанного при содействии Майкла Литтмана, лежит статья [Littman, Dean and Kaelbling (1995)].

5 Методы Монте-Карло

В этой главе вводятся в рассмотрение методы обучения для оценивания значений функций ценности и нахождения оптимальных стратегий. В отличие от предыдущей главы здесь не предполагается, что все полностью известно об окружающей среде. Методы Монте-Карло требуют только наличия *опытных данных* — последовательностей выборок состояний, действий и вознаграждений, получаемых путем непосредственного или смоделированного взаимодействия с окружающей средой. Обучение на основе опытных данных, получаемых и используемых *непосредственно* в процессе взаимодействия системы со средой дает поразительно хорошие результаты, поскольку не требует предварительного знания динамики окружающей среды, но, как и методы на основе модели среды, позволяет достигать оптимального поведения. Обучение на основе *смоделированных* опытных данных также достаточно эффективно. Несмотря на то что при таком подходе к обучению по-прежнему требуется модель, она необходима лишь для формирования переходов из одного состояния в другое, а не полного распределения вероятностей всех возможных переходов, которое требуется в методах динамического программирования (ДП). Существует удивительно много случаев, в которых достаточно легко сформировать элемент набора опытных данных согласно желаемым распределениям вероятностей, но невозможно получить такие распределения в явном виде.

Методы Монте-Карло позволяют решать задачу обучения с подкреплением, основываясь на выгоде, осредненной на некоторой выборке. Для обеспечения уверенности в том, что выгода будет вполне определенной, методы Монте-Карло используются только для заданий, состоящих из эпизодов, т. е. предполагается, что опытные данные разбиваются на эпизоды, а все эпизоды, в конечном счете, завершаются вне зависимости от выбираемых действий. Только по завершении эпизода происходит оценивание значения ценности и изменение стратегии. Таким образом, методы Монте-Карло по-прежнему реализуют процессы инкрементного типа, но на уровне эпизодов, а не шагов, как в методах ДП. Термин «Монте-Карло» часто трактуют расширительно, применяя его в отношении любого метода оценивания, включающего в себя значительную случайную составляющую. Здесь данный термин применяется к методам, осно-

вывающимся на усредненной полной выгодае (в противоположность методам, обучение в которых основано на частичной выгоде, они рассматриваются в следующей главе).

Несмотря на различия между методами Монте-Карло и методами ДП, наиболее важные идеи методов ДП распространяются и на методы Монте-Карло. Все эти методы не только вычисляют одну и ту же функцию ценности, но и по существу одинаковым образом достигают оптимальности. Как и в предыдущей главе, сначала рассмотрим оценивание стратегии, вычисление функций V^π и Q^π для произвольно выбранной стратегии π , затем улучшение стратегии и, наконец, обобщенную итерацию по стратегиям. Каждая из этих идей, взятая из ДП, распространяется на методы Монте-Карло, в которых доступны только выборочные опытные данные.

5.1. Оценка стратегии методами Монте-Карло

Вначале рассмотрим методы Монте-Карло (МК) для формирования функции оценивания состояния для заданной стратегии. Напомним, что ценностью состояния является ожидаемая выгода — ожидаемое накопленное будущее приведенное вознаграждение, — получаемая начиная с данного состояния. Тогда очевидным способом оценить выгоду, основываясь на опыте, будет простое усреднение значений выгоды, полученных после того, как было пройдено данное состояние. С ростом числа наблюдений, дающих значения выгоды, среднее значение выгоды должно сходиться к ожидаемой величине. Эта идея лежит в основе всех методов Монте-Карло.

Предположим, в частности, что требуется оценить величину $V^\pi(s)$, т. е. ценность состояния s при стратегии π , имея набор эпизодов, полученных при применении стратегии π и прохождении через состояние s . Каждое появление состояния s в эпизоде называется *посещением* s . *МК-метод всех посещений* оценивает $V^\pi(s)$ как среднее значение выгод, соответствующих всем посещениям s в некоторой совокупности эпизодов. В рамках заданного эпизода первое появление s именуется *первым посещением* s . *МК-метод первого посещения* усредняет выгоды, соответствующие только первым посещениям s . Два этих метода Монте-Карло очень похожи, но их теоретические свойства немного различаются. Лучше всего к настоящему времени изучен МК-метод первого посещения, развитие которого началось еще в 1940-х гг. Именно на этом методе мы сосредоточимся в данной главе, а также вернемся еще к нему в гл. 7. В процедурной форме МК-метод первого посещения показан на рис. 5.1.

Инициализировать:

$\pi \leftarrow$ оцениваемая стратегия

$V \leftarrow$ произвольная функция ценности состояний

$Returns(s) \leftarrow$ пустой список, для всех $s \in \mathcal{S}$

Повторять циклически:

(а) Сформировать эпизод, используя π

(б) Для каждого состояния s , появляющегося в эпизоде:

$R \leftarrow$ выгода, следующая за первым посещением s

Добавить R к $Returns(s)$

$V(s) \leftarrow$ среднее ($Returns(s)$)

Рис. 5.1. МК-метод первого посещения для оценки V^π

Результат, получаемый как с помощью МК-метода первого посещения, так и МК-метода всех посещений, сходится к ценности $V^\pi(s)$ при стремлении числа посещений (или первых посещений) состояния s к бесконечности. Это легко увидеть на примере МК-метода первого посещения. В этом случае все выгоды являются независимыми одинаково распределенными оценками значений ценности $V^\pi(s)$. По закону больших чисел последовательность средних значений этих оценок сходится к их математическому ожиданию. Каждое среднее значение представляет собой несмещенную оценку, стандартное отклонение которой определяется как $1/\sqrt{n}$, где n — это число усредненных выгод. МК-метод всех посещений несколько сложнее, но его оценки так же асимптотически сходятся к $V^\pi(s)$.

Использование методов Монте-Карло лучше всего проиллюстрировать на примере.

Пример 5.1. В казино популярна карточная игра «блэкджек» («два-дцать одно»). Цель игры — собрать карты так, чтобы сумма их очков была как можно больше, но не превышала 21. Все карты-фигуры (валет, дама, король) имеют значение 10, туз может принимать значения либо 1, либо 11, остальные карты — согласно их номиналам. Рассмотрим вариант игры, в которой каждый игрок состязается с крупье (игроком, сдающим карты) независимо от других участников. В начале игры и игроку, и крупье раздается по две карты. Одна из карт крупье располагается рубашкой вниз, а другая — рубашкой вверх. Если у игрока сразу выпадает 21 (туз и десятка), эта выигрышная комбинация, сданная в первых двух картах, называется «натуральное очко». В этом случае он выигрывает, если только у крупье тоже не выпала такая же комбинация. Если же и у крупье «натуральное очко», то результат ни-

чейный. Если у игрока сразу не выпало «натуральное очко», он просит одну за другой дополнительные карты («еще»), пока либо не остановится («хватит»), либо не получит больше 21 («перебор»). Если у игрока перебор, он проигрывает; если игрок останавливается, наступает очередь крупье. Крупье берет дополнительные карты или прекращает набор в соответствии со следующей фиксированной стратегией: при любой сумме, равной или превышающей 17, он прекращает набор, в остальных случаях берет дополнительные карты. Если у крупье перебор, то выигрывает игрок, в остальных случаях результат — выигрыш, проигрыш или ничья — зависит от того, чья сумма ближе к 21.

Игра в «блэкджек» естественным образом формулируется как эпизодный финитный МППР. Каждая партия данной игры является эпизодом. Выигрышу, ничьей и проигрышу соответствуют вознаграждения, равные +1, 0 и -1 соответственно. В рамках одной партии все вознаграждения равны нулю, а коэффициент приведения γ равен 1; таким образом, данные завершающие вознаграждения также являются и выгодами. Действиями являются: взять дополнительную карту или прекратить набор. Состояния зависят от карт игрока и от открытой карты крупье. Предполагается, что карты раздаются из бесконечной (т. е. обновляющейся) колоды, таким образом, отслеживание уже отыгранных карт не имеет смысла. Если у игрока на руках туз, который можно принять за 11, не выйдя за рамки 21, то этот туз называется *полезным*. В этом случае он всегда считается за 11, так как, принимая его за 1, получаем сумму, равную или меньшую 11, после чего игроку, очевидно, придется взять дополнительную карту. Соответственно, игрок принимает решение, учитывая значения трех переменных: сумма очков у него на руках (12–21), открытая карта крупье (туз–10) и наличие или отсутствие на руках полезного туза. Таким образом, в общей сложности имеется 200 состояний.

Рассмотрим следующую стратегию: прекращать набор, только если у игрока на руках сумма очков, равная 20 или 21, брать дополнительную карту в остальных случаях. Чтобы найти функцию ценности состояний для этой стратегии методом Монте-Карло, необходимо смоделировать большое количество игр, следя данной стратегии, а затем усреднить выгоды, отвечающие каждому состоянию. Отметим, что в данной задаче в рамках одного эпизода каждое состояние встречается не более одного раза. Таким образом, при такой постановке задачи отсутствует разница между МК-методами первого посещения и всех посещений. В данном случае получаем оценки значений функции ценности состоя-

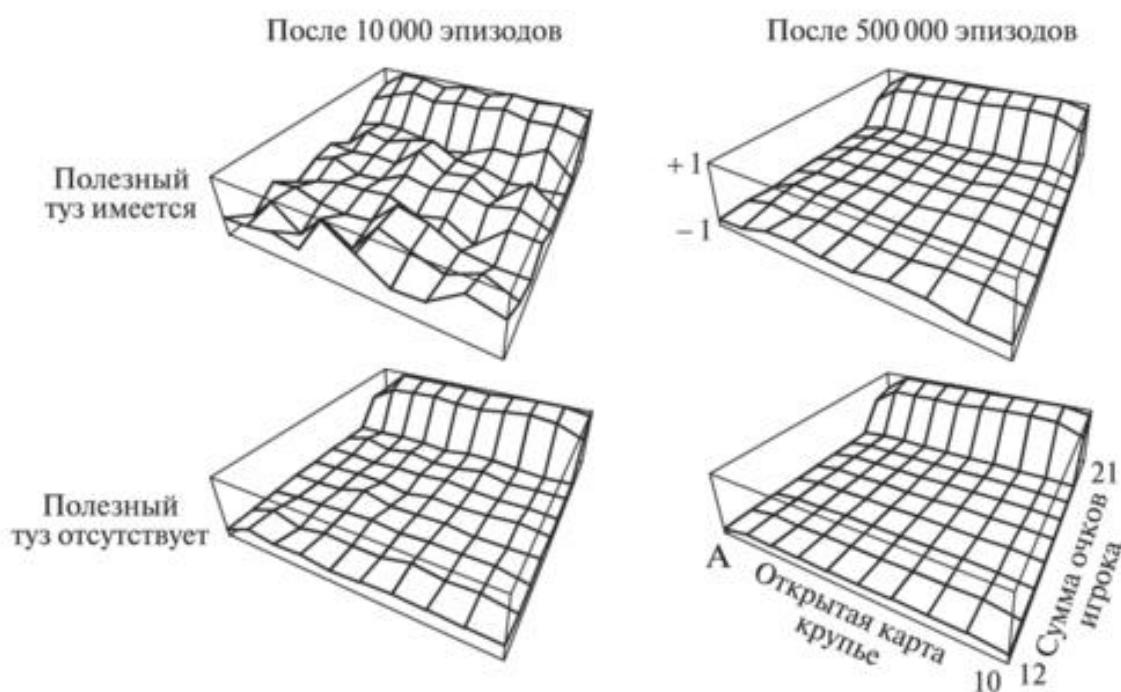


Рис. 5.2. Приближенные функции ценности состояний, рассчитанные методом Монте-Карло, для стратегии игры в блэкджек, которая останавливается только при числе очков 20 или 21

ний, показанные на рис. 5.2. Оценки состояний с полезным тузом менее определены и точны, поскольку такие состояния встречаются редко. Так или иначе, после 500 000 игр получается очень точное приближение функции ценности.

Хотя в этой задаче нам в точности известна модель окружающей среды, применить подход динамического программирования для вычисления функции ценности в данном случае не очень просто. Для методов ДП требуется знать распределения последующих событий, в частности необходимы величины $P_{ss'}^a$ и $R_{ss'}^a$, но применительно к игре в «двадцать одно» это вызывает затруднения. Предположим, например, что у игрока на руках сумма очков, равная 14, и он решает прекратить набор. Каково его ожидаемое вознаграждение как функция открытой карты крупье? Все ожидаемые вознаграждения и вероятности переходов должны быть рассчитаны до применения ДП, но такие расчеты обычно сложны и ненадежны. Напротив, разыгрывание партий рассматриваемой игры, требуемое методами Монте-Карло, не вызывает затруднений. Подобного рода ситуация встречается на удивление часто; способность методов Монте-Карло работать только с разыгрываемыми партиями может стать значительным преимуществом, даже если полностью известна модель динамики окружающей среды.

Можно ли распространить идею диаграммы предшествующих состояний на алгоритмы Монте-Карло? Основная идея такой диаграммы заключается в следующем: показать в верхней части диаграммы корневую вершину, которую требуется улучшить, а ниже изобразить все переходы и концевые вершины-листья, вознаграждения и расчетные ценности которых учитываются при улучшении. При оценивании функции V^π методом Монте-Карло корневой является вершина-состояние, а ниже располагается вся последовательность переходов в рамках конкретного эпизода, которая заканчивается терминальным состоянием, как показано на рис. 5.3. Здесь диаграмма ДП (рис. 3.4, а) показывает все возможные переходы, тогда как в диаграмме Монте-Карло представлены только переходы, осуществленные в одном эпизоде. При этом диаграмма ДП содержит только одношаговые переходы, а диаграмма Монте-Карло прослеживает весь путь до конца эпизода. Рассмотренные различия в диаграммах точно отражают фундаментальные различия между данными алгоритмами.

Важной особенностью методов Монте-Карло является независимость оценки каждого состояния. Оценка одного состояния не основывается на оценке какого-либо другого состояния, как это происходит в случае с ДП. Другими словами, в методах Монте-Карло нет того процесса «самонастройки», который описывался в предыдущей главе.

В частности, отметим, что требуемые для оценки значения ценности одного состояния вычислительные ресурсы не зависят от количества состояний. Это делает методы Монте-Карло особенно привлекательными, когда необходимо оценить ценность только для некоторого подмножества состояний. Можно сформировать большую выборку эпизодов, берущих начало в данных состояниях, усредняя выгоды только для них и игнорируя все остальные состояния. Это является третьим преимуществом методов Монте-Карло в сравнении с методами ДП (после способности обучаться на основе реальных и смоделированных опытных данных).

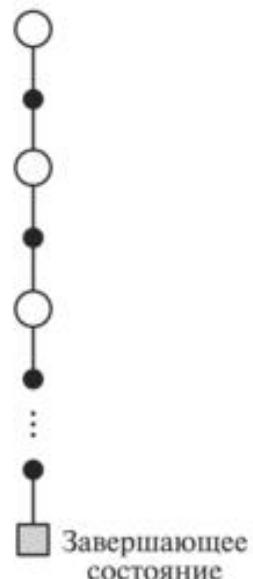


Рис. 5.3. Диаграмма предшествующих состояний для оценки функции V^π методом Монте-Карло

Пример 5.2 (Мыльный пузырь). Пусть проволочный каркас в виде замкнутой петли окунается в мыльную воду, образуя поверхность мыльного пузыря, границы которого определяются проволочной рамкой. Как можно рассчитать форму поверхности, если проволочная рамка имеет неправильную, но вместе с тем известную форму? Поверхность пленки обладает тем свойством, что в каждой точке поверхности суммарная сила, вызванная воздействием соседних точек, равна нулю (в противном случае форма поверхности изменится). Это означает, что высота поверхности в любой точке равна средней высоте точек поверхности, находящихся поблизости от данной точки. Кроме того, граница поверхности должна совпадать с проволочной рамкой. Обычно задачи такого типа решаются следующим способом: на область, ограниченную данной поверхностью, накладывается сетка и далее отыскивается высота каждой узловой точки сетки методом итераций. Узловые точки на границе совпадают с проволочной рамкой, высота всех остальных узловых точек считается равной средней высоте четырех ближайших точек. Затем данный процесс повторяется наподобие итеративного оценивания стратегии в ДП и, в конце концов, дает достаточно хорошее приближение к искомой поверхности.

Рассмотренный пример относится к тому типу задач, для решения которых изначально разрабатывались методы Монте-Карло. Вместо использования итерационного процесса, описанного выше, представьте, что вы находитесь на рассматриваемой поверхности и перемещаетесь по ней случайным образом за счет того, что произвольно с равной вероятностью делаете шаги от одной узловой точки к соседней, пока не достигнете границы. Оказывается, математическое ожидание высоты на границе является хорошим приближением для значения высоты искомой поверхности в начальной точке (фактически, это будет в точности значение, рассчитанное описанным выше методом итерационных исчислений). Таким образом, можно с достаточной точностью рассчитать высоту поверхности в точке, просто усреднив высоты точек границы, полученные при рассмотрении множества маршрутов от искомой точки до границы. Если необходимо рассчитать значение высоты только одной точки или небольшого заданного набора точек, то методы Монте-Карло могут оказаться намного более эффективными, чем итерационные методы, основанные на локальной проверке совпадения результатов с заданным диапазоном значений.

Упражнение 5.1. Рассмотрим диаграммы, показанные на рис. 5.2 справа. Почему в последних двух дальних рядах имеет место скачок

функции ценности? Почему она убывает на протяжении всего левого ряда? Почему значения передних рядов выше на верхней диаграмме?

5.2. Оценка ценности действия методом Монте-Карло

При отсутствии модели намного полезнее оценивать ценности *действий*, нежели *состояний*. Если модель имеется, то ценностей состояний вполне достаточно для определения стратегии: надо просто выполнить один шаг вперед и выбрать действие, которое приводит к наилучшей комбинации вознаграждения и следующего состояния, как это было описано в главе, посвященной ДП. Однако при отсутствии модели недостаточно знать только ценности состояний. Необходимо в явном виде оценить значение ценности каждого действия, чтобы можно было затем использовать полученные значения при выработке стратегии. В связи с этим одна из основных целей применения методов Монте-Карло в рассматриваемом случае — оценивание функции Q^* . Чтобы добиться этого, рассмотрим вначале еще одну задачу оценивания стратегии.

Задача оценивания стратегии на основе ценностей действий состоит в вычислении $Q^\pi(s, a)$ — ожидаемой выгоды при начальном состоянии s , выполнении действия a и следовании в дальнейшем стратегии π . В данном случае методы Монте-Карло по существу такие же, как и в только что рассмотренном случае с ценностями состояний. МК-метод всех посещений оценивает значение ценности пары состояние—действие как среднее значение выгоды, соответствующей посещению данного состояния, в котором выбиралось данное действие. МК-метод первого посещения находит среднее значение выгод, отвечающих первому посещению данного состояния и выполнению данного действия в каждом эпизоде. Эти методы, как и раньше, характеризуются квадратичной сходимостью к истинным значениям ожидаемых ценностей при стремлении к бесконечности количества посещений каждой пары состояние—действие.

Единственная сложность заключается в том, что многие значимые пары состояние—действие могут остаться без посещений. Если π — детерминированная стратегия, то, следуя ей, получим значение выгоды только одного из действий для каждого состояния. Поскольку набор значений выгоды для усреднения отсутствует, оценки, получаемые методом Монте-Карло для других действий, не будут улучшаться с накоплением опыта. Это серьезная проблема, поскольку формирование значений ценности действий осуществляется с целью обеспечить воз-

можность выбора одного из действий, доступных в каждом состоянии. Однако для того чтобы сравнивать возможные альтернативы, необходимо найти ценности всех действий в каждом состоянии, а не одного, пусть и наиболее предпочтительного в данный момент.

Это основная проблема *поддерживавшего изучения*, которая уже обсуждалась ранее применительно к задаче *n*-румого бандита в гл. 2. Чтобы оценивать стратегии на основе ценности действий, изучение необходимо осуществлять непрерывно. Добиться этого можно, к примеру, следующим способом: указать, что первый шаг каждого эпизода начинается *парой* состояния—действие и что каждая такая пара с ненулевой вероятностью может быть выбрана как стартовая. В пределе при числе эпизодов, стремящемся к бесконечности, это гарантирует посещение всех пар состояния—действие бесконечное число раз. Данное утверждение называется предположением об *изучающих стартах*.

Предположение об изучающих стартах иногда может оказаться полезным, но, конечно, на него нельзя полагаться в общем случае, особенно при обучении на основе непосредственного взаимодействия с окружающей средой. В этом случае начальные условия едва ли будут настолько полезны. Согласно наиболее распространенному альтернативному подходу, чтобы обеспечить появление всех пар состояния—действие, рассматриваются только стохастические стратегии с ненулевой вероятностью выбора всех действий. Две важных разновидности данного подхода будут рассмотрены в последующих разделах. Пока же примем предположение об изучающих стартах и завершим изложение материала, показывающего, каким образом метод Монте-Карло используется для решения задачи управления.

Упражнение 5.2. Какой будет диаграмма предшествующих состояний для оценивания функции Q^π с помощью метода Монте-Карло?

5.3. Формирование управления методом Монте-Карло

Теперь все подготовлено для того, чтобы рассмотреть, каким образом оценивание методом Монте-Карло можно применять для формирования управления, т. е. для аппроксимации оптимальных стратегий. Общая идея здесь состоит в том, чтобы действовать по той же схеме, что и в главе, посвященной динамическому программированию, т. е. опираться на понятие обобщенной итерации по стратегиям (ОИС). В концепции ОИС используется как приближенная стратегия, так и прибли-

женная функция ценности. Функция ценности постоянно корректируется, чтобы наилучшим образом отвечать текущей стратегии, а стратегия, в свою очередь, постоянно уточняется в зависимости от текущей функции ценности:



Эти два вида изменений в определенной степени работают друг против друга, смещая цели друг для друга, но, действуя совместно, они заставляют стратегию и функцию ценности стремиться к их оптимальным вариантам.

Для начала рассмотрим МК-версию классической итерации по стратегиям. В данном методе осуществляются попеременно полные шаги оценивания стратегии и улучшения стратегии, начиная с произвольной стратегии π_0 и заканчивая оптимальной стратегией и оптимальной функцией ценности:

$$\pi_0 \xrightarrow{E} Q^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} Q^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi^* \xrightarrow{E} Q^*,$$

где \xrightarrow{E} означает полное оценивание стратегии, а \xrightarrow{I} — полное улучшение стратегии. Оценивание стратегии осуществляется точно таким же образом, как это было описано в предыдущем разделе. Реализуется большое количество эпизодов с приближенной функцией ценности действия, асимптотически стремящейся к истинной функции. Примем пока, что действительно имело место бесконечное число эпизодов и, кроме того, что эпизоды формировались при помощи изучающих стартов. При таких предположениях методами Монте-Карло можно будет точно вычислить каждую функцию Q^{π_k} для произвольной стратегии π_k .

Улучшение стратегии осуществляется, делая искомую стратегию жадной по отношению к текущей функции ценности. В данном случае имеем функцию ценности *действия*, и, таким образом, для формирования жадной стратегии нам не требуется наличие модели. Для любой функции ценности действия Q соответствующей жадной стратегией является стратегия, которая для каждого состояния $s \in \mathcal{S}$ выбирает действие с максимальным значением Q :

$$\pi(s) = \arg \max_a Q(s, a). \quad (5.1)$$

Теперь улучшение стратегии может быть осуществлено за счет построения каждой стратегии π_{k+1} как жадной по отношению к функции Q^{π_k} . В этом случае к стратегиям π_k и π_{k+1} можно применить теорему об улучшении стратегии (разд. 4.2), так как для всех состояний $s \in \mathcal{S}$,

$$\begin{aligned} Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, \arg \max_a Q^{\pi_k}(s, a)) = \\ &= \max_a Q^{\pi_k}(s, a) \geq (s, \pi_k(s)) = V^{\pi_k}(s). \end{aligned}$$

Как уже было сказано в предыдущей главе, данная теорема гарантирует, что каждая стратегия π_{k+1} обязательно лучше, чем π_k , либо равна ей в случае, если обе стратегии оптимальны. Это, в свою очередь, гарантирует, что весь процесс сходится к оптимальной стратегии и оптимальной функции ценности. Таким образом, методы Монте-Карло можно использовать для отыскания оптимальной стратегии при наличии только некоторого набора эпизодов, при этом какие-либо знания о модели динамики окружающей среды не требуются.

Выше были заданы два трудновыполнимых условия, при которых гарантируется сходимость методов Монте-Карло. Согласно первому условию, каждый эпизод имеет изучающий старт, согласно второму условию, оценивание стратегии проводится при бесконечном числе эпизодов. Для получения реального алгоритма нам придется отказаться от этих условий. Первое из них будет рассмотрено позднее в этой главе.

Обратим пока внимание на условие, согласно которому оценивание стратегии осуществляется на бесконечном числе эпизодов. От этого условия достаточно легко избавиться. По сути та же самая проблема возникает даже в классических методах ДП, таких как итеративное оценивание стратегии, которое так же только асимптотически сходится к истинной функции ценности. Как в случае с ДП, так и в случае с методами Монте-Карло существует два способа решения этой проблемы. Согласно первому способу, необходимо строго придерживаться идеи аппроксимации функции Q^{π_k} при каждом оценивании стратегии. Вначале осуществляются измерения и выдвигаются предположения для того, чтобы получить граничные значения величины и вероятности ошибки оценивания, затем при каждом оценивании стратегии предпринимаются действия, в результате которых эти величины становятся достаточно малыми. Данный подход, по-видимому, можно довести до уровня, полностью удовлетворяющего нас в смысле гарантий корректной сходимости с требуемой степенью приближения. Тем не менее вполне возможно, что для его практического применения к любым реальным задачам, за ис-

ключением самых маленьких, потребуется слишком большое количество эпизодов.

Согласно второму способу, чтобы не иметь дело с бесконечным числом эпизодов, формально требующимся для оценки стратегии, не нужно предпринимать попытки завершать оценивание стратегии до перехода к улучшению стратегии. Каждый шаг оценивания приближает функцию ценности к Q^{π_k} , однако предполагается, что за небольшое количество шагов тесного сближения не произойдет. Данная идея уже использовалась при представлении идеи обобщенной итерации по стратегиям в разд. 4.6. Одним из предельных вариантов данной идеи является итерация по ценностям, когда между двумя соседними шагами по улучшению стратегии имеет место только одна итерация итеративного оценивания стратегии. Вариант итерации по ценностям, использующий замещение, является еще более крайней формой выражения рассматриваемой идеи, поскольку в нем осуществляется чередование шагов по улучшению и оценке для отдельных состояний.

В методах Монте-Карло обычно происходит чередование операций оценивания и улучшения от эпизода к эпизоду. По завершении каждого эпизода полученная выгода используется для оценивания стратегии, после чего данная стратегия улучшается во всех состояниях данного эпизода, посещение которых имело место. На рис. 5.4 приведен простой алгоритм, описывающий данную схему. Он получил название *ИС-алгоритма Монте-Карло*, т. е. алгоритма Монте-Карло с изучающими стартами.

Инициализировать для всех $s \in \mathcal{S}, a \in \mathcal{A}(s)$

$Q(s, a) \leftarrow$ произвольное значение

$\pi(s) \leftarrow$ произвольное значение

$Return(s, a) \leftarrow$ пустой список

Повторять циклически

(а) Сформировать эпизод, используя изучающие старты и π

(б) Для каждой пары s, a , появляющейся в эпизоде

$R \leftarrow$ выгода, следующая за первым посещением s, a

Добавить R к $Return(s, a)$

$Q(s, a) \leftarrow$ среднее ($Return(s, a)$)

(в) Для каждого s в эпизоде

$\pi(s) \leftarrow \arg \max_a Q(s, a)$

Рис. 5.4. ИС-алгоритм Монте-Карло: алгоритм управления Монте-Карло, использующий изучающие старты

В ИС-алгоритмах Монте-Карло все выгоды для каждой пары состояния—действие накапливаются и усредняются, независимо от того, какая стратегия имела место в момент получения данной выгоды. Легко заметить, что ИС-алгоритмы Монте-Карло не могут сходиться к какой бы то ни было субоптимальной стратегии. Если бы это могло произойти, то функция ценности непременно устремилась бы к функции ценности для данной стратегии, а это, в свою очередь, вызвало бы изменение стратегии. Стабильность достигается лишь в случае, когда и стратегия, и функция ценности являются оптимальными. Сходимость к этой оптимальной неподвижной точке представляется неизбежной, поскольку изменения функции ценности действия уменьшаются с течением времени, но формально это до сих пор не доказано. По нашему мнению, это одна из наиболее фундаментальных нерешенных проблем в обучении с подкреплением.

Пример 5.3 (Решение задачи для игры в блэкджек). Применим ИС-метод Монте-Карло к игре в блэкджек. Поскольку все эпизоды являются смоделированными играми, то достаточно легко организовать изу-

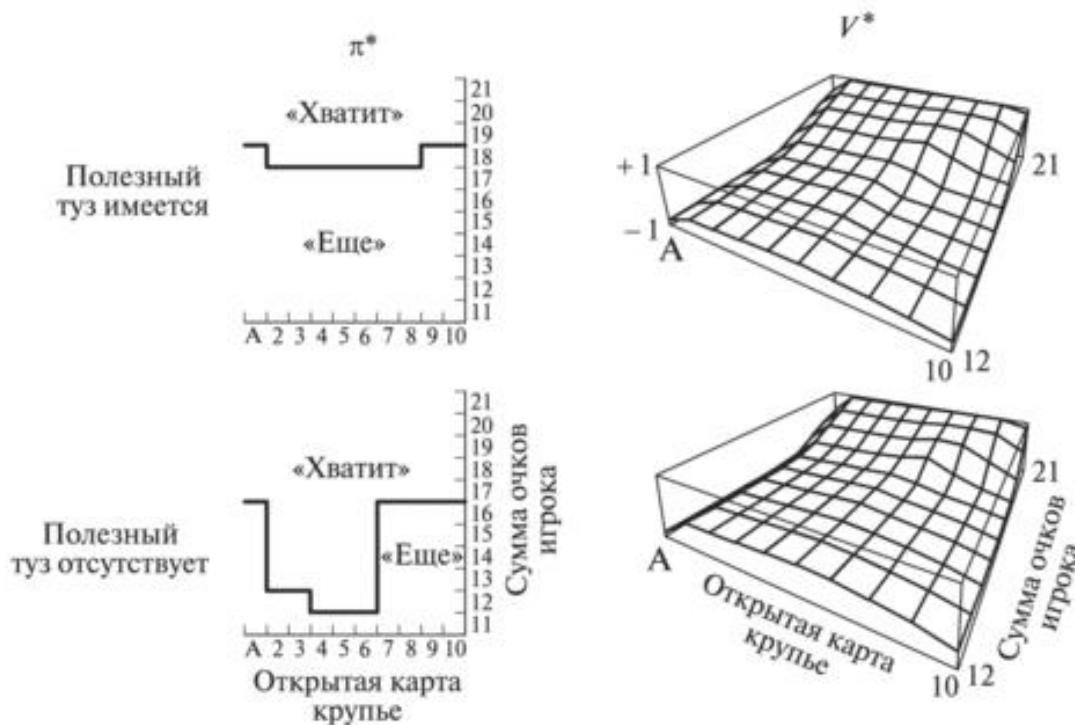


Рис. 5.5. Оптимальная стратегия и функция ценности действия для игры в блэкджек, найденные ИС-методом Монте-Карло (рис. 5.4). Представленная функция ценности состояния вычислена для функции ценности действия, найденной ИС-методом Монте-Карло

чающие старты, реализующие все возможности. В таком случае нужно только задать карты крупье, сумму очков для карт на руках у игрока и наличие или отсутствие у него на руках полезного туза — все это в любом порядке и с равной вероятностью. В качестве начальной стратегии возьмем стратегию из предыдущего примера с игрой в блэкджек, которая останавливается только когда число набранных очков равно 20 или 21. Начальная функция ценности действия может иметь нулевое значение для всех пар состояния—действие. На рис. 5.5 изображена оптимальная стратегия для игры в блэкджек, найденная ИС-методом Монте-Карло. Данная стратегия повторяет предложенную в работе Торпа [Thorp (1966)] «основную» стратегию за исключением ступеньки слева для стратегии с полезным тузом; в стратегии Торпа этой ступеньки нет. Нам непонятна причина данного несоответствия, но есть уверенность в том, что на представленных рисунках действительно изображена оптимальная стратегия для рассматриваемого варианта игры в блэкджек.

5.4. Управление по методу Монте-Карло с интегрированной оценкой ценности стратегий

Как можно избежать труднореализуемого предположения об изучающих стартах? Единственный достаточно общий способ обеспечить неограниченно частый выбор любого из возможных действий — заставить агента продолжать выбирать их. Имеется два подхода, реализующих этот способ, которые приводят к так называемым методам *с интегрированной оценкой ценности стратегий* и методам *с разделенной оценкой ценности стратегий*. Методы с интегрированной оценкой пытаются оценивать или улучшать стратегию непосредственно при использовании ее для принятия решений. В этом разделе представлен один из вариантов управления по методу Монте-Карло с интегрированной оценкой ценности стратегий (ИМК-метод), который иллюстрирует рассматриваемую идею.

В ИМК-методах управления стратегия, как правило, *гибкая*, т. е. неравенство $\pi(s, a) > 0$ выполняется для всех состояний $s \in \mathcal{S}$ и всех действий $a \in \mathcal{A}(s)$. Существует множество вариаций ИМК-методов. Например, можно постепенно изменять стратегию, пытаясь приблизить ее к детерминированной оптимальной стратегии. Многие из методов, рассмотренных в гл. 2, позволяют решать такую задачу. Описанный в этом разделе ИМК-метод использует ε -жадные стратегии. Это значит, что большую часть времени они выбирают действия с максимальными рас-

четными ценностями, а с вероятностью ε выбирают произвольные действия. При этом все нежадные действия имеют минимальную вероятность выбора $\varepsilon/|\mathcal{A}(s)|$, а во всех остальных случаях выбираются жадные действия, т. е. они имеют вероятность, определяемую соотношением

$$1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}(s)|}.$$

ε -жадные стратегии являются примером ε -гибких стратегий, определяемых как стратегии, для которых выполняется неравенство

$$\pi(s, a) \geq \frac{\varepsilon}{|\mathcal{A}(s)|}$$

для всех состояний и действий, для некоторого $\varepsilon > 0$. Среди всех ε -гибких стратегий ε -жадные стратегии — это стратегии, являющиеся в некотором смысле наиболее близкими к жадным.

Основная идея ИМК-метода управления все та же, что в случае с ОИС. Как и в ИС-методе Монте-Карло, здесь используются МК-методы первого посещения, чтобы оценить функцию ценности действия для текущей стратегии. Однако без предположения об изучающих стартах нам не удастся улучшить стратегию, просто сделав ее жадной по отношению к текущей функции ценности, так как это будет препятствовать дальнейшему исследованию нежадных действий. К счастью, ОИС не требует постоянно использовать жадную стратегию, но требует лишь постоянно двигаться *по направлению к* жадной стратегии. В рассматриваемом ИМК-методе будем осуществлять такое продвижение лишь к некоторой ε -жадной стратегии. Для любой ε -гибкой стратегии π любая ε -жадная по отношению к функции Q^π стратегия лучше, чем π , или равнозначна ей.

Согласно теореме об улучшении стратегии, любая ε -жадная по отношению к функции Q^π стратегия будет лучше, чем любая ε -гибкая стратегия π . Пусть π' — ε -жадная стратегия. Условия теоремы об улучшении стратегии выполняются, поскольку для любого состояния $s \in \mathcal{S}$

$$\begin{aligned} Q^\pi(s, \pi'(s)) &= \sum_a \pi'(s, a) Q^\pi(s, a) = \\ &= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s, a) + (1 - \varepsilon) \max_a Q^\pi(s, a) \geq \\ &\geq \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s, a) + \\ &\quad + (1 - \varepsilon) \sum_a \frac{\pi(s, a) - (\varepsilon/|\mathcal{A}(s)|)}{1 - \varepsilon} Q^\pi(s, a) \end{aligned} \tag{5.2}$$

(данная сумма является средневзвешенным значением с неотрицательными весами, в сумме дающими 1, и, следовательно, она должна быть меньше либо равна наибольшему из усредняемых чисел)

$$\begin{aligned} &= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s, a) - \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s, a) + \sum_a \pi(s, a) Q^\pi(s, a) = \\ &= V^\pi(s). \end{aligned}$$

Таким образом, по теореме об улучшении стратегии

$$\pi' \geq \pi$$

(т. е. $V^{\pi'}(s) \geq V^\pi(s)$ для всех состояний $s \in \mathcal{S}$). Докажем теперь, что равенство имеет место, только когда обе стратегии π' и π являются оптимальными среди ε -гибких стратегий, т. е. когда они лучше всех остальных ε -гибких стратегий или равнозначны им.

Рассмотрим новую окружающую среду, которая почти совпадает с первоначальной за исключением требования, чтобы стратегии были ε -гибкими и «введенными в среду». Новая окружающая среда имеет те же самые множества действий и состояний, что и первоначальная, при этом она ведет себя следующим образом. Если в состоянии s выбирается действие a , то с вероятностью $1 - \varepsilon$ новая окружающая среда ведет себя в точности так же, как и первоначальная. С вероятностью ε она повторно выбирает любое из возможных действий с одинаковой вероятностью и далее ведет себя как первоначальная окружающая среда с данным новым случайно выбранным действием. Наилучший результат, которого можно добиться в новой окружающей среде с обычными стратегиями, тот же самый, что и в случае с первоначальной окружающей средой и ε -гибкими стратегиями. Обозначим \tilde{V}^* оптимальную функцию ценности для новой окружающей среды. Тогда стратегия π будет оптимальной среди ε -гибких стратегий тогда и только тогда, когда

$$V^\pi = \tilde{V}^*.$$

Из определения функции \tilde{V}^* следует, что она является единственным решением уравнения

$$\begin{aligned} \tilde{V}^*(s) &= (1 - \varepsilon) \max_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma \tilde{V}^*(s')] + \\ &\quad + \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma \tilde{V}^*(s')]. \end{aligned}$$

Если имеет место равенство, а ε -гибкая стратегия больше не совершенствуется, то из соотношения (5.2) следует также, что

$$\begin{aligned}\tilde{V}^\pi(s) &= (1 - \varepsilon) \max_a Q^\pi(s, a) + \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s, a) = \\ &= (1 - \varepsilon) \max_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma \tilde{V}^*(s')] + \\ &\quad + \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma \tilde{V}^*(s')].\end{aligned}$$

Это соотношение практически совпадает с предыдущим за исключением того, что в одном из них фигурирует функция \tilde{V}^* , а в другом — функция V^π . Так как функция \tilde{V}^* является единственным решением, то очевидно, что $V^\pi = \tilde{V}^*$.

По сути, материал, представленный на нескольких последних страницах, показывает, что итерация по стратегиям работает и в случае с ε -гибкими стратегиями. Если использовать ранее введенное понятие жадной стратегии в случае с ε -гибкими стратегиями на каждом шаге, то улучшение гарантировано, кроме случаев, когда среди ε -гибких стратегий уже найдена наилучшая. Данный анализ не зависит от того, каким образом на каждом этапе определялись функции ценности действия, но предполагает, что эти функции вычисляются точно. Все это приводит примерно к тому же самому результату, что и был получен в предыдущем разделе. Разница в том, что получена наилучшая стратегия среди ε -гибких стратегий, но с другой стороны, удалось исключить предположение об изучающих стартах. Полный алгоритм приведен на рис. 5.6.

5.5. Оценивание одной стратегии при использовании другой

До сих пор рассматривались методы оценивания функции ценности для некоторой стратегии в случае, когда в нашем распоряжении имелось бесконечное число эпизодов, порождаемых данной стратегией. Примем теперь, что все, что у нас есть, — это эпизоды, порожденные *другой* стратегией, т. е. будем предполагать, что требуется оценить функции V^π или Q^π , используя только эпизоды, отвечающие стратегии π' , где $\pi' \neq \pi$. Можно ли получить функцию ценности для некоторой стратегии, располагая только опытными данными, полученными «вне» данной стратегии?

Инициализировать, для всех $s \in S, a \in \mathcal{A}(s)$

$Q(s, a) \leftarrow$ произвольное значение

$Return(s, a) \leftarrow$ пустой список

$\pi \leftarrow$ произвольная ε -гибкая стратегия

Повторять циклически

(а) Сформировать эпизод, используя π

(б) Для каждой пары s, a , появляющейся в эпизоде

$R \leftarrow$ выгода, следующая за первым посещением s, a

Добавить R к $Return(s, a)$

$Q(s, a) \leftarrow$ среднее ($Return(s, a)$)

(в) Для каждого s в эпизоде

$a^* \leftarrow \arg \max_a Q(s, a)$

Для всех $a \in \mathcal{A}(s)$

$$\pi(s, a) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon / |\mathcal{A}(s)| & \text{при } a = a^*, \\ \varepsilon / |\mathcal{A}(s)| & \text{при } a \neq a^*. \end{cases}$$

Рис. 5.6. ε -гибкое управление по методу Монте-Карло с интегрированной оценкой ценности стратегий

К счастью, во многих случаях это возможно. Конечно, чтобы использовать эпизоды из стратегии π' при оценке значений ценности для стратегии π , необходимо потребовать, чтобы каждое действие, осуществляемое при стратегии π , осуществлялось также, по крайней мере иногда, и при стратегии π' , т. е. чтобы из неравенства $\pi(s, a) > 0$ следовало $\pi'(s, a) > 0$. В эпизодах, полученных с использованием стратегии π' , рассмотрим i -е первое посещение состояния s и полную последовательность состояний и действий, следующих за этим посещением. Обозначим $p_i(s)$ и $p'_i(s)$ вероятности того, что данная полная последовательность будет иметь место при стратегиях π и π' соответственно и при начальном состоянии s . Через $R_i(s)$ обозначим выгоду, соответствующую состоянию s .

Чтобы усреднить эту выгоду для получения несмещенной оценки функции $V^\pi(s)$, требуется лишь взвесить каждую выгоду по ее относительной вероятности появления при стратегиях π и π' , т. е. по $p_i(s)/p'_i(s)$. Тогда искомая оценка Монте-Карло после получения n_s выгод от состояния s имеет вид

$$V(s) = \frac{\sum_{i=1}^{n_s} p_i(s)/p'_i(s) R_i(s)}{\sum_{i=1}^{n_s} p_i(s)/p'_i(s)}. \quad (5.3)$$

Это уравнение содержит вероятности $p_i(s)$ и $p'_i(s)$, которые при применении методов Монте-Карло обычно считаются неизвестными. К счастью, в данном случае необходимо знать только их отношение $p_i(s)/p'_i(s)$, которое *может* быть определено при отсутствии знаний о модели динамики окружающей среды. Обозначим $T_i(s)$ время завершения i -го эпизода, включающего состояние s . Тогда

$$p_i(s_t) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) \mathcal{P}_{s_k s_{k+1}}^{a_k}$$

и

$$\frac{p_i(s_t)}{p'_i(s_t)} = \frac{\prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) \mathcal{P}_{s_k s_{k+1}}^{a_k}}{\prod_{k=t}^{T_i(s)-1} \pi'(s_k, a_k) \mathcal{P}_{s_k s_{k+1}}^{a_k}} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)}.$$

Таким образом, вес $p_i(s)/p'_i(s)$, необходимый в равенстве (5.3), зависит только от двух рассматриваемых стратегий и не зависит от динамики окружающей среды.

Упражнение 5.3. Какой будет оценка Монте-Карло для ценностей действий, аналогичная (5.3), если имеются выгоды, полученные при использовании стратегии π' ?

5.6. Управление по методу Монте-Карло с разделенной оценкой ценности стратегий

Теперь можно представить пример второго класса методов формирования управления, рассматриваемых в этой книге, а именно управление по методу Монте-Карло с разделенной оценкой ценности стратегий (РМК-методы). Напомним, что отличительной особенностью ИМК-методов, т. е. управления по методу Монте-Карло с интегрированной оценкой ценности стратегий, является то, что они оценивают некоторую стратегию, используя ее при этом одновременно и для управления. В РМК-методах эти две функции разделены. Стратегия, используемая для формирования поведения, называемая *стратегией поведения*, может фактически не быть связанной со стратегией, называемой *стратегией оценивания*, которая оценивается и улучшается. Преимущество такого разделения заключается в том, что стратегия оценивания может быть детерминированной (например, жадной), в то время как стратегия поведения может продолжать испытывать все возможные действия.

В РМК-методах управления применяется описанная в предыдущем разделе методика вычисления функции ценности для одной стратегии,

Инициализировать, для всех $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s, a) \leftarrow$ произвольное значение

$N(s, a) \leftarrow 0$; числитель и

$D(s, a) \leftarrow 0$; знаменатель функции $Q(s, a)$

$\pi \leftarrow$ произвольная детерминированная стратегия

Повторять циклически:

(а) Выбрать стратегию π' и использовать ее для формирования эпизода

$s_0, a_0, r_1, s_1, a_1, r_2, \dots, r_T, s_{T-1}, a_{T-1}, s_T$,

(б) $\tau \leftarrow$ последний из моментов времени, в который $a_\tau \neq \pi(s_\tau)$

(в) Для каждой пары s, a , появляющейся в эпизоде в момент времени τ или позднее

$t \leftarrow$ время первого посещения s, a такое, что $t \geq \tau$

$$w \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k, a_k)}$$

$N(s, a) \leftarrow N(s, a) + wR_t$,

$D(s, a) \leftarrow D(s, a) + w$,

$$Q(s, a) \leftarrow \frac{N(s, a)}{D(s, a)},$$

(г) Для каждого $s \in \mathcal{S}$

$$\pi(s) \leftarrow \arg \max_a Q(s, a)$$

Рис. 5.7. РМК-алгоритм управления

когда используется некоторая другая стратегия. Данные методы следуют заданной стратегии поведения, изучая и улучшая при этом стратегию оценивания. Эта методика требует, чтобы стратегия поведения имела ненулевую вероятность выбора всех действий, которые могут быть использованы стратегией оценивания. Для изучения всех возможностей потребуем, чтобы стратегия поведения была гибкой.

На рис. 5.7 представлен РМК-алгоритм управления, в основе которого лежит ОИС для расчета функции Q^* . Стратегия поведения π' остается произвольной гибкой стратегией. Стратегия оценивания π является жадной стратегией по отношению к функции Q , оценке функции Q^π . Выбранная на шаге (а) стратегия поведения может быть какой угодно, но для того чтобы гарантировать сходимость стратегии π к оптимальной, для каждой пары состояния—действие необходимо получить бесконечное число значений выгоды, пригодных для использования на шаге (в). Это достигается тщательным выбором стратегии поведения. Подойдет, например, любая ϵ -гибкая стратегия.

Проблема, которая может возникнуть при использовании такого метода, состоит в том, что при обучении учитывается только заключительная часть эпизода, начинающаяся после последнего нежадного действия. Если нежадные действия встречаются часто, то обучение будет

медленным, особенно для состояний, возникающих в ранней части продолжительного эпизода. Потенциально это может очень сильно замедлить обучение. На данный момент опыта использования РМК-методов недостаточно, чтобы оценить, насколько серьезной является данная проблема.

Упражнение 5.4 [Гоночный трек (программирование)]. Рассмотрим управление гоночным автомобилем при прохождении поворотов, представленных на рис. 5.8. Необходимо ехать настолько быстро, насколько это возможно, оставаясь в пределах трека. В нашей упрощенной модели трека местоположение автомобиля ограничивается дискретным набором позиций, представленных ячейками на схеме. Скорость, выраженная количеством ячеек сетки в горизонтальном и вертикальном направлении за один временной шаг, также дискретна. Действиями являются приращения компонент скорости. Каждая компонента на одном временном шаге может изменяться на $+1$, -1 или 0 , в совокупности образуя девять действий. Значения обеих компонент ограничены интервалом от 0 до 5 , кроме того, они не могут одновременно обращаться в нуль. Каждый эпизод начинается одним из случайно выбранных начальных состояний и заканчивается, когда машина пересекает финишную черту. Вознаграждение составляет -1 за каждый шаг в пределах трека и -5 за попытку агента выехать за пределы трека. Вообще говоря, выезд за пределы трека не разрешен, но автомобиль на каждом шаге продвигается, по крайней мере, на одну ячейку в вертикальном или горизонтальном направлении. Рассматривая при данных ограничениях только

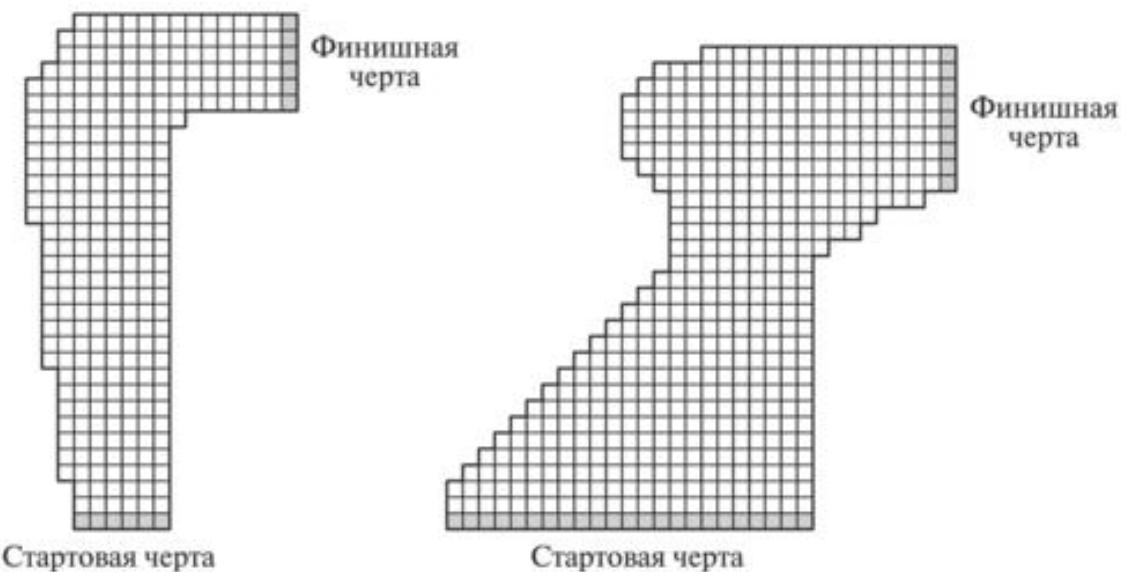


Рис. 5.8. Пара правых поворотов для задачи с гоночным треком

правые повороты, можно гарантировать, что все эпизоды непременно завершатся; кроме того, маловероятно, что оптимальная стратегия не будет найдена. Для усложнения задачи предположим, что в половине случаев на временном шаге позиция смещается на одну дополнительную ячейку вверх или вправо помимо перемещений, определяемых скоростью. Примените в этой задаче управление по методу Монте-Карло с интегрированной оценкой ценности стратегий, чтобы вычислить оптимальную стратегию для каждого из начальных состояний. Покажите несколько траекторий, соответствующих оптимальной стратегии.

5.7. Пошаговая реализация

Методы Монте-Карло могут реализовываться пошаговым образом, путем перехода от эпизода к эпизоду, для чего нужно обобщить методы, рассмотренные в гл. 2. Следует использовать усредненные *выгоды* наподобие того, как в некоторых методах решения задачи об *n*-руком бандите, описанных в гл. 2, используются усредненные *вознаграждения*. Методики из разд. 2.5 и 2.6 можно непосредственно применить и к случаю методов Монте-Карло. Они позволяют этим методам обрабатывать каждую новую выгоду на пошаговой основе, не требуя при этом увеличения объема вычислений и памяти с ростом числа эпизодов.

Есть два различия между подходами, используемыми в случае метода Монте-Карло и в случае с задачей о бандите. Во-первых, в случае с методом Монте-Карло обычно используется множество ситуаций, т. е. различным состояниям соответствуют различные процессы усреднения, тогда как в задаче о бандите имеется только одно состояние (по крайней мере, в простейшем случае, рассмотренном в гл. 2). Во-вторых, в задаче о бандите распределения вознаграждений обычно стационарные, тогда как в методах Монте-Карло распределения выгод обычно нестационарные. Это обусловлено зависимостью выгод от стратегии, стратегия же обычно меняется и совершенствуется со временем.

Пошаговая реализация, описанная в разд. 2.5, имеет дело со случаем простого или арифметического усреднения, в котором каждая выгода имеет одинаковый вес. Предположим, что вместо этого необходимо реализовать *взвешенное* усреднение, в котором каждая выгода R_n имеет вес ω_n , и требуется вычислить величину

$$V_n = \sum_{k=1}^n w_k R_k / \sum_{k=1}^n w_k. \quad (5.4)$$

Например, описанный в разд. 5.5 метод оценки одной стратегии при исследовании другой использует веса $w_n(s) = p_n(s)/p'_n(s)$. Взвешенные средние значения также подчиняются простому правилу пошаговой корректировки. Кроме того, отслеживая значения V_n , для каждого состояния нам необходимо сохранять совокупную сумму весов W_n , заданную для первых n выгод. Правило корректировки для величины V_n следующее:

$$V_{n+1} = V_n + \frac{w_{n+1}}{W_{n+1}} [R_{n+1} - V_n] \quad (5.5)$$

и

$$W_{n+1} = W_n + w_{n+1},$$

где $V_0 = W_0 = 0$.

Упражнение 5.5. Измените алгоритм для оценивания стратегии с помощью МК-метода первого посещения (рис. 5.1) так, чтобы в нем применялась пошаговая реализация для стационарных средних значений, описанная в разд. 2.5.

Упражнение 5.6. Выведите правило корректировки средневзвешенных значений (5.5) из соотношения (5.4). Используйте схему вывода правила для невзвешенных значений (2.4) из соотношения (2.1).

Упражнение 5.7. Измените РМК-алгоритм управления (рис. 5.7) таким образом, чтобы в нем применялся описанный выше метод пошагового расчета средневзвешенных значений.

5.8. Итоги

Представленные в этой главе методы Монте-Карло формируют функции ценности и оптимальные стратегии, используя опытные данные в форме *совокупности эпизодов*. Это дает, по крайней мере, три вида преимуществ по сравнению с методами динамического программирования. Во-первых, методы Монте-Карло могут применяться для выработки оптимального поведения непосредственно из взаимодействия с окружающей средой, при этом не требуется знания модели динамики окружающей среды. Во-вторых, эти методы можно использовать в сочетании с методами моделирования, позволяющими получить *смоделированные совокупности эпизодов*. Оказывается, в большинстве случаев можно легко смоделировать требуемую совокупность эпизодов, даже если трудно получить явную модель вероятностей переходов, которая требуется методам ДП. В-третьих, методы Монте-Карло могут легко и эффективно *фокусироваться* на небольшом подмножестве состояний.

В этом случае можно точно оценить небольшую область, представляющую интерес, не затрачивая ресурсов на точную оценку остального множества (такая ситуация рассматривается в гл. 9).

Четвертым преимуществом методов Монте-Карло, которое будет рассмотрено позже, является их меньшая чувствительность к нарушению марковского свойства. Это является следствием того, что в методах Монте-Карло при корректировке значений ценности состояний не используются значения ценности последующих состояний. Другими словами, методы Монте-Карло не используют самонастройку.

Конструирование МК-методов управления осуществлялось согласно полной схеме *обобщенной итерации по стратегиям* (ОИС), представленной в гл. 4. ОИС включает в себя взаимодействующие процессы оценки и улучшения стратегии. Методы Монте-Карло предлагают альтернативный процесс оценивания стратегии. Вместо того чтобы пользоваться моделью при вычислении ценности каждого состояния, они просто усредняют множество выгод, ставших следствием данного состояния. Так как ценностью состояния является ожидаемая выгода, такое усреднение может дать хорошее приближение к истинной ценности. В методах управления особый интерес представляют аппроксимирующие функции ценности действия, так как они могут быть использованы для улучшения стратегии при отсутствии модели динамики переходов в окружающей среде. Методы Монте-Карло чередуют шаги по оцениванию и улучшению стратегии на поэпизодной основе, они могут быть реализованы пошаговым образом, эпизод за эпизодом.

Поддержание достаточного уровня изучения представляет собой важную проблему для МК-методов управления. Недостаточно просто выбирать действия, считающиеся лучшими на данный момент, так как в этом случае будут неизвестны выгоды для альтернативных действий, а более выгодные действия могут так и остаться неизученными. Согласно одному из подходов, можно пренебречь данной проблемой, полагая, что эпизоды начинаются парами состояния—действие, выбираемыми произвольным образом и охватывающими все возможности. Такого типа *изучающие старты* могут использоваться в задачах, в которых эпизоды смоделированы, но они маловероятны в обучении на основе реального опыта. Вместо этого можно использовать один из двух подходов общего вида. В управлении по методу Монте-Карло с интегрированной оценкой ценности стратегий агент постоянно осуществляет действия исследовательского характера и пытается найти наилучшую стратегию, которая будет продолжать эти действия. В управлении по методу Монте-Карло с разделенной оценкой ценности стратегий агент

также выполняет исследовательские действия, но вырабатывает детерминированную оптимальную стратегию, которая может быть не связана с текущей стратегией. Примеры таких методов представлены в следующей главе.

Методы Монте-Карло для обучения с подкреплением были подробно описаны сравнительно недавно. Их свойства сходимости пока не ясны, и их практическая эффективность мало исследована. В настоящий момент основным их преимуществом является простота, а также их взаимосвязь с другими методами.

Методы Монте-Карло от методов динамического программирования отличаются в двух отношениях. Во-первых, они оперируют с непосредственно получаемыми опытными данными, в силу чего могут применяться для непосредственного обучения без использования модели. Во-вторых, в них не производится самонастройка, т. е. они не корректируют значения ценности, основываясь на каких-то других значениях ценности. Эти два различия не связаны друг с другом неразрывно, так что их можно разделить. В следующей главе будут рассмотрены методы, принимающие опыт за основу обучения, наподобие методов Монте-Карло, и в которых, кроме того, происходит самонастройка, как в методах ДП.

5.9. Библиографические и исторические справки

Термин «Монте-Карло» появился в 1940-х годах, когда физики в Лос-Аламосе придумывали азартные игры, изучавшиеся для лучшего понимания физических явлений, связанных с атомной бомбой. В такой форме описание методов Монте-Карло может быть найдено в различных учебниках (см., например, [Kalos and Whitlock (1986)], [Rubinstein (1981)]).

Одно из самых первых применений методов Монте-Карло для вычисления ценности действий в обучении с подкреплением описано в статье [Michie and Chambers (1968)]. В задаче балансировки стержня (пример 3.4) авторы данной статьи использовали усредненные продолжительности эпизодов для оценки ценности (ожидаемой «продолжительности жизни» для балансируемого стержня) каждого возможного действия в каждом состоянии, а затем использовали эту оценку для управления процессом выбора действий. Их метод, по сути, аналогичен ИС-методу Монте-Карло. Пользуясь нашими терминами, можно сказать, что они использовали МК-метод всех посещений. В работе [Narendra and Wheeler (1986)] был изучен метод Монте-Карло для эргодических финитных цепей Маркова, в которых выгода, накопленная за период от одного посещения состояния до другого, используется как вознаграждение за корректировку вероятностей действий обучающего автомата.

В статье [Barto and Duff (1994)] рассматривается оценивание стратегии в контексте классических алгоритмов Монте-Карло для решения систем линейных уравнений. При этом используются результаты исследования из работы [Curtiss (1954)], чтобы показать вычислительные преимущества оценивания стратегии методами Монте-Карло для больших задач. В работе [Singh and Sutton (1996)] выделен МК-метод всех посещений и МК-метод первого посещения, а также обоснованы результаты, связывающие данные методы с алгоритмами обучения с подкреплением.

Пример игры в блэкджек основан на материале из статьи [Widrow, Gupta and Maitra (1973)]. Пример с мыльной пленкой является классической задачей Дирихле, решение которой методами Монте-Карло было впервые предложено в работе [Kakutani (1945)]; см. также [Hersh and Griego (1969)], [Doyle and Snell (1984)]. Упражнение с гоночным треком взято из работ [Barto, Bradtke and Singh (1995)] и [Gardner (1973)].

6 Обучение на основе временных различий

Из числа современных идей в обучении с подкреплением одной из наиболее важных, несомненно, является обучение на основе *временных различий* (TD – Temporal-Difference), которое будем обозначать далее как TD-обучение. Обучение на основе временных различий представляет собой сочетание идей метода Монте-Карло и динамического программирования (ДП). Как и в методах Монте-Карло, в TD-методах процесс обучения может основываться непосредственно на получаемом опыте без предварительных знаний о модели поведения окружающей среды. Как и в динамическом программировании, TD-методы обновляют расчетные оценки, основанные отчасти на других полученных оценках, не дожидаясь окончательного результата (они самонастраиваются). Тема взаимосвязи между TD-методами, динамическим программированием и методами Монте-Карло многократно встречается в теории обучения с подкреплением. В данной главе будет начато ее изучение. Мы увидим, как эти методы взаимодействуют и сочетаются друг с другом множеством различных способов. В частности, в гл. 7 будет рассмотрен алгоритм TD(λ), который объединяет в себе ДП и методы Монте-Карло.

Как обычно, вначале мы обратимся к оцениванию стратегии или задаче *предсказания*, т. е. оцениванию функции ценности V^π для данной стратегии π . В случае с задачей *управления* (нахождение оптимальной стратегии) TD-методы, ДП и методы Монте-Карло используют некоторую разновидность обобщенной итерации по стратегиям (ОИС). Основными различиями в перечисленных методах являются их подходы к решению задачи предсказания.

6.1. Предсказание на основе временных различий

Как TD-методы, так и методы Монте-Карло для решения задачи предсказания используют имеющийся опыт. При наличии некоторого опыта следования стратегии π оба указанных метода корректируют свои оценки V функции V^π . Если имеет место посещение нетерминального состояния s_t в момент времени t , то оба метода корректируют свои оценки $V(s_t)$, основываясь на том, что случилось после этого посещения. Грубо

говоря, метод Монте-Карло ждет момента, когда станет известна выгода от посещения, а затем использует эту выгоду как целевое значение, к которому должна стремиться оценка $V(s_t)$. Простой МК-метод всех посещений, пригодный для случая нестационарной окружающей среды, имеет вид:

$$V_{s_t} \leftarrow V(s_t) + \alpha[R_t - V(s_t)], \quad (6.1)$$

где R_t — фактическая выгода, полученная после момента времени t , а α — постоянная длина шага (см. выражение (2.5)). Назовем это *МК-методом с постоянной α* . В то время как методы Монте-Карло должны ждать окончания эпизода для определения приращения к $V(s_t)$ (только тогда известна выгода R_t), в случае с TD-методами необходимо дождаться только следующего временного шага. Непосредственно в момент времени $t+1$ формируется целевое значение оценки, после чего производится необходимая корректировка с учетом уже имеющегося вознаграждения r_{t+1} и оценки $V(s_{t+1})$. Простейший TD-метод, известный как TD(0), имеет следующий вид:

$$V_{s_t} \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]. \quad (6.2)$$

Фактически, при корректировке методом Монте-Карло целью является выгода R_t , а при корректировке TD-методом целью будет величина

$$r_{t+1} + \gamma V_t(s_{t+1}).$$

Вследствие того что в своих корректировках TD-метод частично основывается на существующих оценках, будем считать этот метод *самонастраивающимся*, как и метод ДП. Из гл. 3 известно, что

$$V^\pi(s) = E_\pi\{R_t \mid s_t = s\} = \quad (6.3)$$

$$= E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\} =$$

$$= E_\pi\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s\right\} =$$

$$= E_\pi\{r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s\}. \quad (6.4)$$

Упрощенно говоря, в качестве цели методы Монте-Карло используют оценку (6.3), тогда как методы ДП — оценку (6.4). Целевым значением для метода Монте-Карло является некоторая оценка, так как математическое ожидание в соотношении (6.3) неизвестно; вместо действительной ожидаемой выгода используется смоделированная. Целевое значение в методах представляет собой некоторую оценку не из-за

Инициализировать $V(s)$ произвольно, π — оцениваемая стратегия
Повторять (для каждого эпизода):

Инициализировать s

Повторять (для каждого шага эпизода):

$a \leftarrow$ действие, задаваемое стратегией π для s

Выполнить действие a ; найти вознаграждение r
и следующее состояние s'

$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$

$s \leftarrow s'$

Пока s не станет завершающим состоянием

Рис. 6.1. Схема TD(0)-метода оценивания функции V^π

имеющегося ожидаемого значения, которое полностью определяется моделью окружающей среды, но потому что неизвестна функция $V^\pi(s_{t+1})$ и вместо нее используется текущая оценка $V_t(s_{t+1})$. Данная оценка является некоторой целью для TD-метода по двум причинам: он проверяет ожидаемое значение в соотношении (6.4) и использует текущую оценку V_t вместо истинной функции V^π . Таким образом, TD-метод сочетает выборки метода Монте-Карло с самонастройкой метода ДП. Как будет видно далее, сочетая воображение с осторожностью, можно использовать преимущества как методов Монте-Карло, так и TD-методов.

На рис. 6.1 полностью определен метод TD(0), представленный в процедурной форме, а на рис. 6.2 показана диаграмма предшествующих состояний для него. Оценка значения для вершины-состояния в верхней части диаграммы корректируется на основе одного выборочного перехода от данной вершины непосредственно к следующему состоянию.



Рис. 6.2. Диаграмма предшествующих состояний для TD(0)

Корректировки методом Монте-Карло и TD-методом будем рассматривать как выборочное дублирование, так как они вначале рассматривают выборочное последующее состояние (или пару состояния—действие), учитывая его значение и соответствующее вознаграждение при вычислении скопированного резервного значения, а затем на основе этого изменяют соответствующее значение исходного состояния (или значение пары состояния—действие).

Отличие выборочного дублирования от полного дублирования, используемого в методах ДП, состоит в том, что первое из них основывается на единственном последующем элементе, тогда как второе учитывает полное распределение всех возможных последующих состояний.

Таблица 6.1

Состояние	Предсказанное полное время (мин.)	Затраченное время (мин.)	Предсказанное оставшееся время (мин.)
Выход из офиса, пятница, 18:00	0	30	30
Машина, дождь	5	35	40
Съезд с автомагистрали	20	15	35
Второстепенная дорога, позади грузовика	30	10	40
Поворот на нужную улицу	40	3	43
Прибытие домой	43	0	43

Пример 6.1 (Поездка с работы домой). Когда вы едете домой с работы, то стараетесь предугадать, как долго продлится поездка. Покидая офис, вы обращаете внимание на время отъезда, день недели и все, что может оказаться существенным в данном случае. Например, в эту пятницу вы уходите ровно в 18 часов и полагаете, что ваш путь займет примерно 30 минут. Вы подходите к машине в 18:05 и замечаете, что начинается дождь. Движение на дорогах в дождливую погоду обычно замедляется, поэтому теперь вы отводите на оставшийся путь 35 минут, и, следовательно, в сумме получается уже 40 минут. За следующие пятнадцать минут вам удается преодолеть участок пути, пролегающего по автомагистрали. Когда вы поворачиваете на второстепенную дорогу, то уже оцениваете полное время своего пути в 35 минут. К несчастью, в этот момент вы застреваете позади медленно едущего грузовика, а дорога в этом месте слишком узкая для выполнения обгона. Вам приходится следовать за грузовиком до самой улицы, на которой вы живете и на которую вы, наконец, поворачиваете в 18:40. Три минуты спустя вы дома. Последовательность состояний, реальных и предполагаемых промежутков времени дана ниже.

Вознаграждениями в этом примере является время, затраченное на каждом этапе пути¹⁾. В данном случае коэффициент приведения γ равен единице, и, таким образом, выгодой для каждого состояния является оставшееся время пути. Ценностью каждого состояния является

¹⁾ Если бы перед нами стояла задача управления с целью минимизации времени, то, конечно, вознаграждениям были бы присвоены *отрицательные* значения затраченного времени. Но поскольку в данном случае нас интересует лишь предсказания (оценивание стратегии), можно упростить задачу, используя положительные значения.



Рис. 6.3. Изменения, рекомендованные методами Монте-Карло, в примере с поездкой с работы домой

ожидаемое время пути. Во второй колонке цифр представлены текущие оценки времени для каждого из рассматриваемых состояний.

Проще всего наблюдать действие методов Монте-Карло, если изобразить значения предсказанного полного времени пути (последняя колонка цифр в приведенной выше таблице) в виде графика, представленного на рис. 6.3. Стрелками показаны изменения в предсказаниях, рекомендованные методом Монте-Карло с постоянной α (6.1), для $\alpha = 1$. Эти изменения в точности совпадают с расхождениями между оцениваемым значением (предсказанным временем) в каждом состоянии и действительной выгодой (действительным временем пути). Например, когда вы съезжали с автомагистрали, то рассчитывали добраться до дома за 15 минут, но на самом деле это заняло у вас 23 минуты. В этом месте применяется уравнение (6.1), и с его помощью определяется приращение оценки времени пути после съезда с шоссе. Ошибка $R_t - V_t(s_t)$ в данном случае составляет восемь минут. Примем параметр длины шага равным $\alpha = 1/2$. Как результат, предсказанное время в пути после съезда с автомагистрали будет увеличено на четыре минуты. Возможно, в этом случае это слишком большое изменение; вероятно, появление грузовика было достаточно случайным обстоятельством. В любом случае, такое изменение может быть сделано лишь задним числом, т. е. после того, как вы приехали домой. Только тогда вы можете знать о том, какова фактическая выгода.

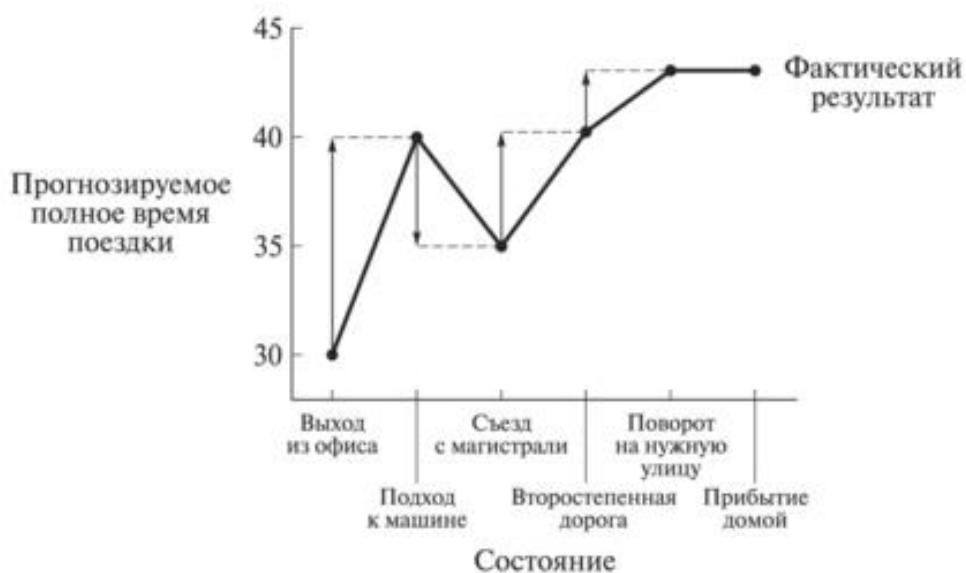


Рис. 6.4. Изменения, рекомендованные TD-методами, в примере с поездкой с работы домой

Необходимо ли ждать окончательного результата, чтобы начать обучение? Предположим, что на следующий день, покидая офис, вы снова предполагаете, что доберетесь до дома за 30 минут, но затем застреваете в большой пробке. Спустя двадцать пять минут после выхода из офиса вы все еще едва двигаетесь в пробке на шоссе. Теперь вы предполагаете, что будете дома через 25 минут с этого момента, т. е. через 50 минут после выхода из офиса. Стоя в пробке, вы уже понимаете, что ваши изначальные предположения о 30 минутах были слишком оптимистичными. Следует ли ждать прибытия домой, чтобы повысить ваше оценочное значение для начального состояния? Согласно методу Монте-Карло, вы действительно должны дожидаться этого момента, так как еще не знаете истинного значения выгоды.

С другой стороны, согласно TD-методу, вы имеете возможность сразу начинать обучение, изменяя свою начальную оценку с 30 минут до 50. Фактически каждая оценка будет меняться в сторону оценки, непосредственно за ней следующей. Вернемся к нашему первому дню. Рис. 6.4 демонстрирует те же предсказания, что и рис. 6.3, за исключением изменений, рекомендованных TD-правилом (6.2) (это изменения, сделанные при условии $\alpha = 1$). Каждая ошибка пропорциональна изменению предсказания во времени, т. е. *временному различию* в предсказании.

Обучаться, основываясь на текущих предсказаниях, предпочтительнее, нежели дожидаться завершения ситуации, когда вы будете знать действительную выгоду. Кроме того, что вы сможете как-то скротать

время нахождения в пробке, тому есть и другие причины, связанные с процессом вычислений. Обсудим их кратко.

6.2. Преимущества TD-методов предсказания

TD-методы строят свои оценки, частично основываясь на других оценках. Они делают предположения, основываясь на предположениях — они *самонастраиваются*. Хорошо ли это? Какими преимуществами обладают TD-методы в сравнении с методами Монте-Карло и ДП? Полный ответ на этот вопрос займет всю оставшуюся часть книги и не только. В этом разделе дается краткий обзор некоторых из возможных ответов.

Очевидно, что одним из преимуществ TD-методов над методами ДП является то, что первые не требуют знания модели окружающей среды с ее вознаграждениями и вероятностным распределением последующих состояний.

Другим очевидным преимуществом TD-методов над методами Монте-Карло является то, что первые по своей сути являются интерактивными, полностью инкрементными методами. В случае с методами Монте-Карло каждый раз необходимо ждать завершения эпизода, так как только тогда становится известна выгода, тогда как в случае с TD-методами необходимо дождаться лишь следующего временного шага. Достаточно часто данная особенность имеет решающее значение. В некоторых ситуациях эпизоды могут быть настолько продолжительными, что задержки процесса обучения, связанные с необходимостью завершения эпизодов, будут слишком велики. В других ситуациях имеются непрерывные задачи, а эпизоды отсутствуют как таковые. Наконец, как было отмечено в предыдущей главе, некоторые методы Монте-Карло должны игнорировать эпизоды или снижать значимость эпизодов, в которых предпринимаются экспериментальные действия, что может сильно замедлить обучение. TD-методы не так чувствительны к такого рода проблемам, поскольку обучаются на основе каждого перехода, вне зависимости от осуществляемых в дальнейшем действий.

Однако надежны ли TD-методы? Несомненно, удобно строить одно предположение на основе другого, не дожидаясь реального результата, но можно ли гарантировать сходимость к верному решению? К счастью, ответ на этот вопрос — да. Доказано, что для любой заданной стратегии π описанный выше TD-алгоритм сходится к функции V^π в среднем при некотором постоянном значении длины шага, если тот достаточно мал, и с вероятностью 1, если параметр длины шага уменьшается в соответствии с обычными условиями стохастической аппроксимации (2.8).

Большинство доказательств сходимости применимо только к случаю с табличным алгоритмом (6.2), представленным выше, но некоторые также применимы и к случаю линейной аппроксимации функции. При более общих предположениях эти результаты рассматриваются в следующих двух главах.

Если и TD-метод, и метод Монте-Карло оба асимптотически сходятся к верным предсказаниям, то возникает естественный вопрос: «Какой из методов первым достигнет цели?» Другими словами, какой из методов обучается быстрее? Какой из методов наиболее эффективно использует имеющуюся ограниченную информацию? В настоящее время этот вопрос остается открытым в том смысле, что до сих пор никто еще математически не доказал, что один из методов сходится быстрее другого. На самом деле, пока даже не ясно, какой должна быть формальная постановка этого вопроса! На практике, тем не менее, обычно оказывается, что TD-методы сходятся быстрее методов Монте-Карло с постоянной α в стохастических задачах. Покажем это на следующем примере.

Пример 6.2 (Случайные блуждания). В этом примере эмпирически сравниваются предсказания метода TD(0) и метода Монте-Карло с постоянной α , примененных к небольшому марковскому процессу, показанному на рис. 6.5. Все эпизоды начинаются из центрального состояния C и продолжаются вправо или влево на одно состояние, на каждом шаге с равной вероятностью. Такое поведение предположительно обусловлено сочетанием фиксированной стратегии с вероятностями состояний переходов окружающей среды. Само это сочетание нас мало интересует; интерес представляют лишь предсказываемые выгоды вне зависимости от того, каким способом они порождаются. Эпизоды завершаются либо на правом, либо на левом конце цепочки. Когда эпизод завершается справа, следует вознаграждение +1; все остальные вознаграждения равны 0. Например, последовательность состояний и вознаграждений может иметь следующий вид: $C, 0, B, 0, C, 0, D, 0, E, 1$. Так как эта задача неприведенная и эпизодная, истинной ценностью каждого состояния является вероятность того, что при старте из данного состояния завершение эпизода произойдет справа. Таким образом, истинной ценностью

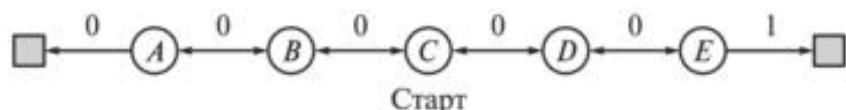


Рис. 6.5. Небольшой марковский процесс для генерации случайных блужданий

центрального состояния является величина $V^\pi(C) = 0,5$. Истинные ценности всех состояний от A до E следующие: $1/6, 2/6, 3/6, 4/6$ и $5/6$ соответственно. На рис. 6.6 показано, как с увеличением количества пройденных эпизодов значения ценностей, получаемых TD(0)-методом, стремятся к истинным значениям. Выполнив усреднение по множеству последовательностей эпизодов, получим результат, изображенный на рис. 6.7, который показывает среднюю ошибку в предсказаниях TD(0)-метода и метода Монте-Карло с постоянной α для различных значений α , являющуюся функцией количества эпизодов. Во всех случаях начальное значение приближенной функции ценности $V(s) = 0,5$ для всех состояний s . В данной задаче и с данным количеством эпизодов TD-метод всегда оказывается лучше метода Монте-Карло.

Упражнение 6.1. Это упражнение должно помочь пониманию того, почему TD-методы часто эффективнее методов Монте-Карло. Рассмотрим пример поездки на автомобиле домой с привлечением как TD-метода, так и метода Монте-Карло. Можете ли вы придумать сценарий, при котором корректировки TD-метода будут лучше в среднем корректировок метода Монте-Карло? Приведите пример такого сценария — описание прошлого опыта и текущее состояние. Подсказка: предположите, что у вас накопился большой опыт поездок из офиса домой. Затем вы переехали в новое здание с новой парковкой (но вы по-прежнему выезжаете на автомагистраль в том же месте, что и раньше). Теперь вы начинаете строить свои предсказания для вашего нового местоположения. Понятно ли вам теперь, что в этом случае корректировки TD-метода, скорее всего, окажутся намного лучше, по крайней мере в начале? Может ли тоже самое произойти в первоначальной задаче?

Упражнение 6.2. Из рис. 6.6 видно, что исходом первого эпизода является лишь изменение значения $V(A)$. Что это говорит вам о том, что происходило в первом эпизоде? Почему изменилась оценка только одного этого состояния? Насколько именно она изменилась?

Упражнение 6.3. Как вы считаете, мог ли выбор какого-либо другого значения параметра α (при том, что этот параметр остается постоянным) существенно улучшить результаты алгоритма по сравнению с результатами, показанными на рис. 6.7? Каким образом? Почему?

Упражнение 6.4. Как видно из рис. 6.7, среднеквадратическая ошибка TD-метода сначала убывает, но потом опять возрастает, особенно при больших α . С чем это связано? Всегда ли это будет иметь место, или это может быть связано с тем, как изначально задается функция ценности?

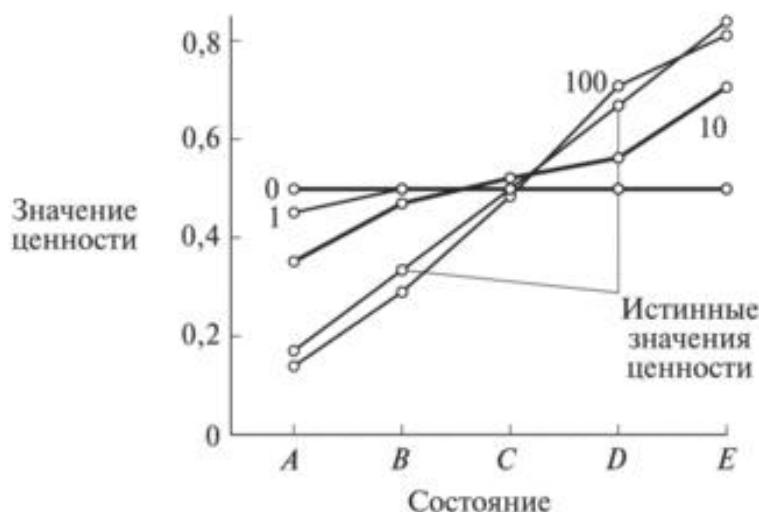


Рис. 6.6. Значения ценностей, полученные методом TD(0) при различном количестве эпизодов в примере с порождением случайных блужданий. Завершающая оценка является наиболее близкой к истинному значению. При постоянной длине шага (в этом примере $\alpha = 0,1$) значения ценностей безостановочно изменяются колебательным образом как реакция на исходы ближайших эпизодов

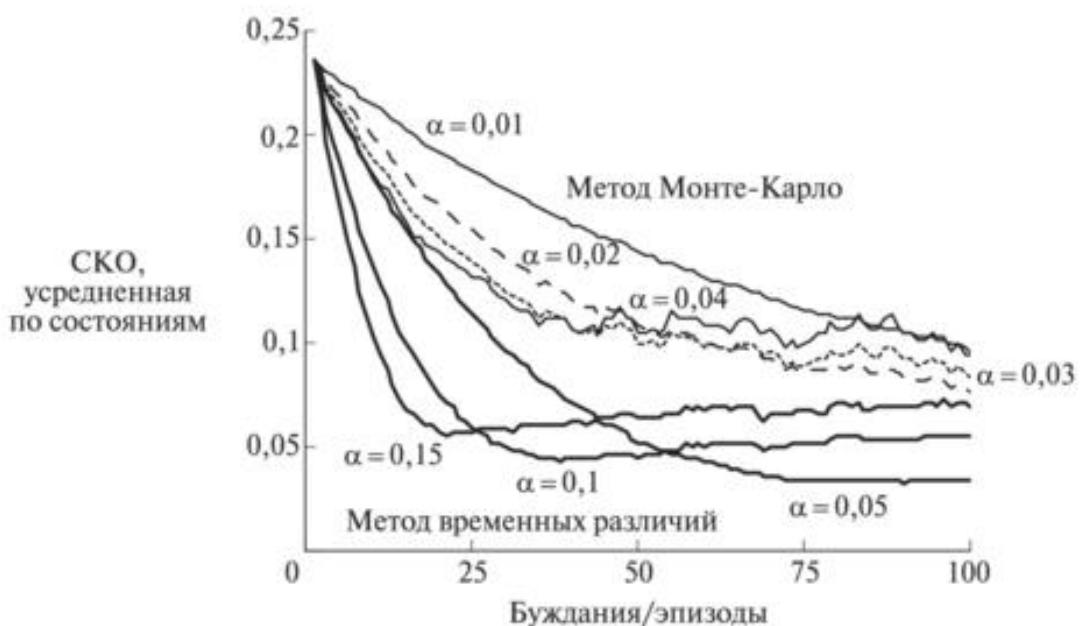


Рис. 6.7. Кривые обучения для метода TD(0) и метода Монте-Карло с параметром α для различных значений α в задаче о случайных блужданиях. Изображена среднеквадратическая ошибка (СКО), сформированная для изучаемой и истинной функций ценности, усредненная по пяти состояниям. Эти данные являются результатом усреднения по 100 различным последовательностям эпизодов

Упражнение 6.5. Выше были установлены истинные значения произвольных шагов, которые равны $1/6$, $2/6$, $3/6$, $4/6$ и $5/6$ для состояний с A по E соответственно. Приведите по крайней мере два различных способа, с помощью которых можно получить данные результаты. Как вы думаете, какой из них мы использовали? Почему?

6.3. Оптимальность метода TD(0)

Пусть нам доступны только опытные данные ограниченного объема, например 10 эпизодов или 100 временных шагов. В такой ситуации обычным подходом в случае с пошаговыми методами обучения является многократное повторное использование имеющегося опыта до тех пор, пока не будет достигнута сходимость к решению. При заданной приближенной функции ценности V приращения, определяемые соотношениями (6.1) или (6.2), вычисляются на каждом временном шаге t , на котором имеет место посещение нетерминального состояния, но функция ценности меняется только один раз, при суммировании всех приращений. Затем весь доступный опыт опять обрабатывается уже с учетом новой функции ценности для получения нового общего приращения и т. д., пока функция ценности не сойдется. Такой процесс называется *групповой корректировкой*, так как корректировка происходит только после обработки группы данных обучения.

При групповой корректировке метод TD(0) сходится детерминированно к единственному решению, вне зависимости от размера шага α до тех пор, пока значение α выбирается достаточно малым. Метод Монте-Карло с параметром α также сходится детерминированно при тех же условиях, но к другому решению. Понимание этих двух решений поможет нам понять различие между двумя данными методами. Нельзя сказать, что при нормальной корректировке данные методы постоянно стремятся к соответствующим групповым решениям, однако можно утверждать, что они делают шаги в данном направлении. Прежде чем попытаться понять оба эти решения в общем для всех возможных задач, рассмотрим несколько примеров.

Пример 6.3 (Случайные блуждания при групповой корректировке). Версии TD-метода и метода Монте-Карло с параметром α в версии с групповой корректировкой были применены к задаче о случайных блужданиях (пример 6.2) следующим образом. После каждого нового эпизода все имевшие место до этого эпизоды считались группой. Они многократно предъялялись алгоритму TD(0) или методу Монте-Карло с параметром α , где значение α достаточно мало, чтобы функция

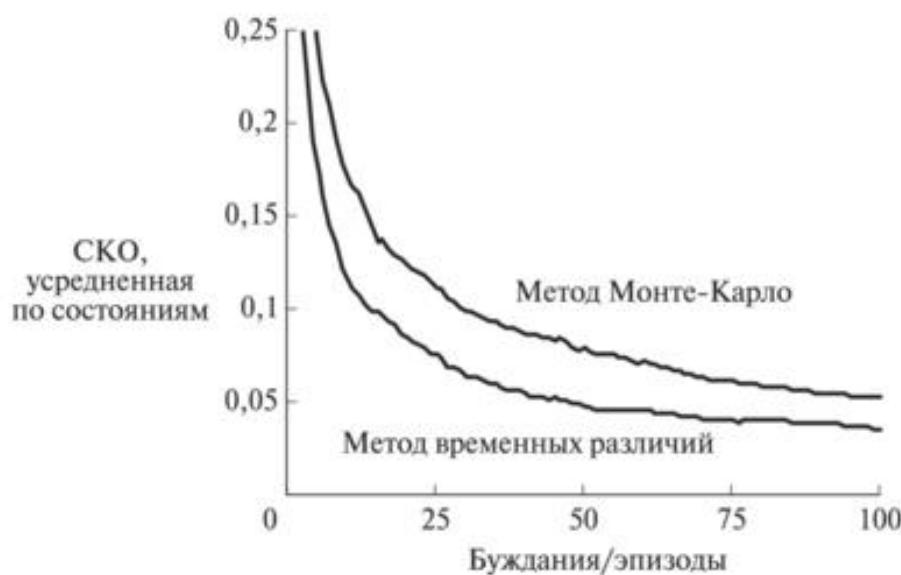


Рис. 6.8. Результаты применения метода TD(0) и метода Монте-Карло с параметром α к задаче с произвольными шагами при групповом обучении

ценности сходилась. Полученная функция ценности затем сравнивалась с V^π , а среднее значение среднеквадратической ошибки пяти состояний (на протяжении 100 независимых повторений всего эксперимента) наносилось на график. Таким образом были получены кривые обучения, изображенные на рис. 6.8. Обратите внимание, что групповой TD-метод всегда оказывался лучше группового метода Монте-Карло.

При групповом обучении метод Монте-Карло с параметром α сходится к значениям ценности $V(s)$, которые являются средними выборочными действительных выгод, последовавших за посещением каждого состояния s . Эти оценки оптимальны в том смысле, что они минимизируют среднеквадратическую ошибку оценки реальных выгод в серии обучений. В связи с этим достаточно неожиданно, что групповой TD-метод проявил себя лучше с точки зрения среднеквадратических ошибок, как видно из рис. 6.8. Каким образом получилось, что групповой TD-метод проявил себя лучше оптимального метода? Ответ заключается в том, что оптимальность метода Монте-Карло до некоторой степени ограничена, и что TD-метод является оптимальным в том смысле, что он лучше предсказывает выгоду. Но для начала проясним наше понимание различных типов оптимальности с помощью следующего примера.

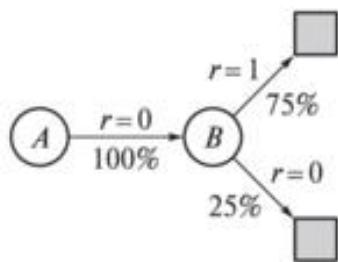
Пример 6.4 (Вы — предсказатель). Представьте теперь себя в роли предсказателя выгод неизвестного марковского процесса получения воз-

награждений. Предположим, имели место следующие восемь эпизодов:

$A, 0$	$B, 0$	$B, 1$
$B, 1$		$B, 1$
$B, 1$		$B, 1$
$B, 1$		$B, 0$

Это означает, что первый эпизод начался в состоянии A , перешел в B с вознаграждением 0 и затем завершился в B с вознаграждением 0. Следующие семь эпизодов были еще короче. Начавшись в B , они немедленно завершались. Располагая такой группой данных, что бы вы могли сказать об оптимальных предсказаниях, о наилучших значениях оценок $V(B)$ и $V(A)$? Наверное, все согласятся с тем, что оптимальным значением для $V(B)$ является $3/4$, так как шесть из восьми раз в состоянии B процесс немедленно завершался с выгодой 1, и остальные два раза процесс немедленно завершался с выгодой 0.

Но каким будет оптимальное значение оценки $V(A)$ при имеющихся данных? Есть два разумных ответа. Первым ответ будет следующим: поскольку в 100% случаев наблюдался незамедлительный переход из состояния A в состояние B (с вознаграждением 0) и поскольку уже определено, что B имеет ценность $3/4$, то и A также должно иметь ценность $3/4$. Для понимания этого ответа рассмотрим первую модель марковского процесса для данного случая:



и затем рассчитаем правильную оценку, предлагаемую моделью, которая на самом деле имеет вид $V(A) = 3/4$. Такой же ответ дает и групповой TD(0) метод.

Согласно логике второго ответа, состояние A наблюдалось только однажды, и последовавшая за ним выгода равна была 0; следовательно можно оценить значение $V(A)$ также равным 0. Именно такой ответ дает групповой метод Монте-Карло. Заметьте, что этот ответ дает минимальную квадратичную ошибку на обучающих данных. По сути, он дает нулевую ошибку на этих данных. Тем не менее предполагается, что первый ответ предпочтительнее. Если процесс марковский, ожидается, что

первый ответ повлечет за собой меньшую ошибку для будущих данных, несмотря на то что ответ метода Монте-Карло лучше при существующих данных.

Описанный выше пример иллюстрирует основное различие между оценками группового метода TD(0) и группового метода Монте-Карло. Групповые методы Монте-Карло всегда находят такие оценки, которые минимизируют среднеквадратичную ошибку обучающей последовательности, тогда как групповой метод TD(0) всегда находит оценки, которые были бы точными применительно к модели максимального правдоподобия для марковского процесса. В общем случае *оценкой по методу максимального правдоподобия* для некоторого параметра представляет собой такое значение параметра, при котором вероятность порождения данных результатов максимальна. В данном случае оценка по методу максимального правдоподобия дает модель марковского процесса, формируемого очевидным образом из имевших место эпизодов: оценкой вероятности перехода из i в j является та часть всех имевших место переходов, которая содержит только переходы из i в j . Ожидаемое вознаграждение в этом случае представляет собой среднее по вознаграждениям, соответствующим данным переходам. Имея такую модель, можно рассчитать оценочное значение функции ценности, которое будет точным в той степени, в какой точна сама модель. Это называется *стохастически эквивалентной оценкой*, поскольку она эквивалентно предположению о том, что оценка основного процесса не является приближенной в обычном смысле, а лишь известна с некоторой степенью уверенности. В общем случае групповой метод TD(0) сходится к стохастически эквивалентной оценке.

Теперь понятно, почему TD-методы сходятся быстрее методов Монте-Карло. В групповой форме метод TD(0) быстрее метода Монте-Карло вследствие того что вычисляет истинную стохастически эквивалентную оценку. Это объясняет преимущество метода TD(0), которое видно из результатов для примера со случайными блужданиями (рис. 6.8). Связь со стохастически эквивалентной оценкой можно также отчасти объяснить преимущество в скорости негруппового метода TD(0) (рис. 6.7). Несмотря на то что негрупповые методы не достигают ни стохастической эквивалентности, ни оценок с минимальной квадратичной ошибкой, можно сказать, что они ориентированы примерно в этих направлениях. Негрупповой метод TD(0) может оказаться быстрее метода Монте-Карло с параметром α , так как двигается в сторону более верных оценок, даже несмотря на то, что не всегда действует подобным образом. В настоящее время нельзя сказать ничего более

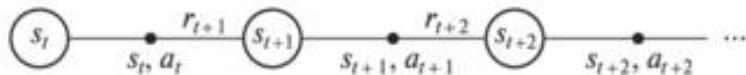
определенного про относительную эффективность TD-метода и метода Монте-Карло, работающих в оперативном режиме.

В конце концов, даже несмотря на то, что стохастически эквивалентная оценка является в некотором смысле оптимальным решением, в действительности практически невозможно вычислить ее напрямую. Если число состояний равно N , то только на формирование оценки по методу максимального правдоподобия может потребоваться объем памяти порядка N^2 , а вычисление соответствующей функции ценности потребует порядка N^3 шагов. В сравнении с этим действительно выдающимся выглядит TD-метод, которому для получения тех же решений требуется объем памяти не более N и который повторяет вычисления на одном и том же обучающем множестве. В задачах с большим множеством состояний TD-методы могут быть единственным реальным способом приблизиться к стохастически эквивалентному решению.

6.4. SARSA: управление по TD-методу с интегрированной оценкой ценности стратегий

Обратимся теперь к применению TD-методов предсказания к задаче управления. Как обычно, будем использовать модель обобщенной итерации по стратегиям (ОИС), только на этот раз с применением TD-методов в оценочной или предсказательной части. Как и в случае с методами Монте-Карло, перед нами встает необходимость попеременного использования операций изучения и применения, и снова подходы к решению разделяются на два основных класса: с интегрированной оценкой ценности стратегий и с разделенной оценкой ценности стратегий. В этом разделе представлен TD-метод управления с интегрированной оценкой ценности стратегий.

Первый шаг заключается в изучении функции ценности действий. В частности, для метода с интегрированной оценкой ценности стратегий нам необходимо оценить функцию $Q^\pi(s, a)$ для текущей стратегии π и для всех состояний s и действий a . Это можно сделать, используя, по сути, тот же самый TD-метод для изучения функции V^π , который был описан выше. Напомним, что эпизод состоит из последовательности перемежающихся состояний и пар состояния—действие:



В предыдущем разделе мы рассмотрели переходы из состояния в состояние и изучили ценности состояний. Теперь рассмотрим переходы

```

Инициализировать  $Q(s, a)$  произвольно
Повторять (для каждого эпизода):
    Инициализировать  $s$ 
    Выбрать  $a$  по  $s$ , используя стратегию, полученную из  $Q$ 
        (например,  $\epsilon$ -жадную)
    Повторять (для каждого шага эпизода):
        Выполнить действие  $a$ , найти  $r, s'$ 
        Найти  $a'$  по  $s'$ , используя стратегию, полученную из  $Q$ 
            (например,  $\epsilon$ -жадную)
         $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$ 
         $s \leftarrow s', a \leftarrow a'$ 
    Пока  $s$  не станет завершающим состоянием

```

Рис. 6.9. SARSA: управление по TD-методу с интегрированной оценкой ценности стратегий

из пары состояние—действие в пару состояние—действие и изучим ценности этих пар. Формально эти случаи аналогичны: они оба являются марковскими цепями с процессом вознаграждения. Теоремы, гарантирующие сходимость ценностей состояний при использовании метода TD(0), применимы и к соответствующему алгоритму для ценностей действий:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]. \quad (6.5)$$

Данная корректировка имеет место после каждого перехода из нетерминального состояния s_t . Если состояние s_{t+1} является терминальным (заключительным), то значение $Q(s_{t+1}, a_{t+1})$ полагается равным нулю. Это правило использует каждый элемент из пятерки событий $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$, из которых состоит переход от одной пары состояние—действие к другой. Эта пятерка дала название SARSA данному алгоритму¹⁾.

Не представляет большой сложности создать алгоритм с интегрированной оценкой ценности стратегий управления, основанный на методе предсказания SARSA. Как и во всех методах с интегрированной оценкой ценности стратегий, мы все время оцениваем функцию Q^π для стратегии поведения π и в то же время стратегия π делается более жадной по отношению к Q^π . Общая форма алгоритма управления SARSA дана на рис. 6.9.

¹⁾ То есть SARSA (State-Action-Reward-State-Action) — это аббревиатура, составленная из букв, входящих в данную пятерку, где State — состояние, Action — действие, Reward — вознаграждение. Впервые такое наименование для данного алгоритма было введено в работе [Rummery, Niranjan (1994)]. — Прим. ред.

Свойства сходимости алгоритма SARSA зависят от характера зависимости стратегии от функции Q . Например, можно использовать ε -жадную и ε -гибкую стратегии. Алгоритм SARSA сходится с вероятностью 1 к оптимальной стратегии и функции ценности действия при условии, что все пары состояния—действие посещались бесконечное число раз, и стратегия сходится в пределе к жадной стратегии (что может быть осуществлено, например, при помощи ε -жадных стратегий с $\varepsilon = 1/t$).

Пример 6.5 (Сетка с ветром). На рис. 6.10 изображена стандартная сетка с начальными и целевыми состояниями, но с одним отличием: внутри сетки присутствует направленный вверх ветер. Имеется стандартный набор из 4 действий — up (вверх), down (вниз), right (вправо), left (влево) — но в центральной области сетки очередные результирующие состояния перемещаются вверх «по ветру», сила которого меняется от колонки к колонке. Сила ветра задана под каждой колонкой количеством клеток, на которые происходит сдвиг вверх. Например, если вы находитесь в одной клетке справа от цели, то действие left (влево) переместит вас в клетку непосредственно над целью. Будем рассматривать эту задачу как неприведенную и эпизодную с постоянным вознаграждением, равным -1 , до тех пор, пока не достигнута цель. На рис. 6.11 приведен результат применения к этой задаче ε -жадного метода SARSA с $\varepsilon = 0,1$, $\alpha = 0,1$ и начальными ценностями $Q(s, a) = 0$ для всех пар состояния—действие s, a . Возрастающий угол наклона кривой говорит о том, что цель с течением времени достигается все быстрее. К тому времени как прошло 8000 временных шагов, жадная стратегия (изображена на врезке) уже давно являлась оптимальной; продолженное ε -жадное изучение дает 17 шагов в качестве средней

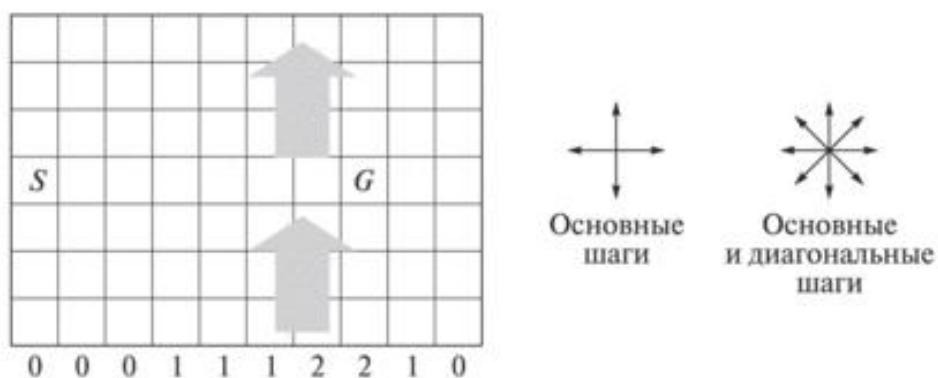


Рис. 6.10. Сетка, в которой перемещения изменяются под воздействием ветра, зависящего от местоположения и направленного вверх

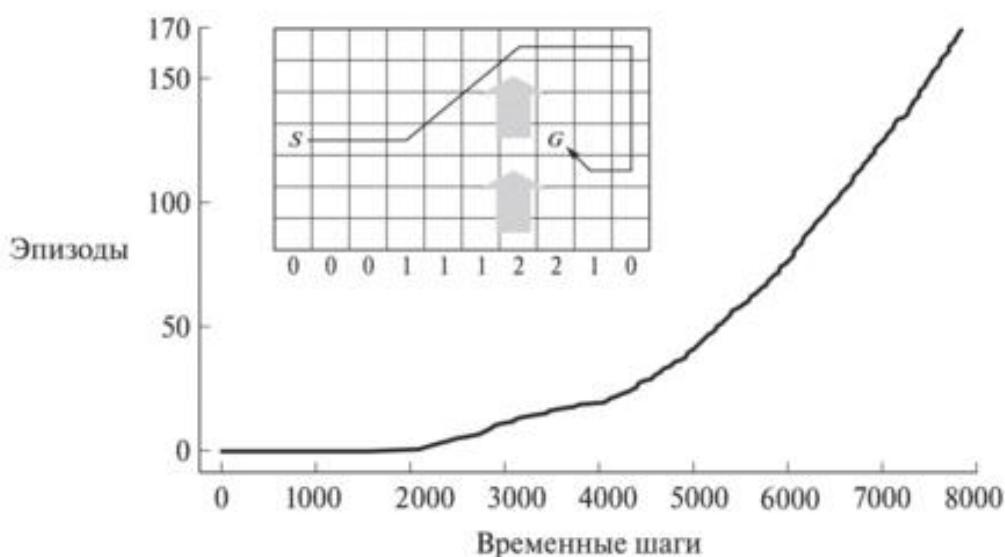


Рис. 6.11. Результаты применения алгоритма SARSA к задаче о сетке с ветром

длины эпизода, что на два шага отличается от минимальных 15. Заметьте, что методы Монте-Карло нельзя так же легко применить к этой задаче, поскольку завершение не гарантировано для всех стратегий. Если какая-либо из стратегий приведет к тому, что агент будет оставаться в одном и том же состоянии, то следующий эпизод никогда не завершится. Пошаговые методы наподобие алгоритма SARSA не сталкиваются с подобными проблемами, так как они быстро обнаруживают в течение эпизода, что такая стратегия неэффективна, и переключаются на что-нибудь другое.

Упражнение 6.6 [Дополнительные шаги в сетке с ветром (программирование)]. Решите задачу о сетке с ветром, в которой помимо основных четырех шагов есть возможность совершать дополнительные четыре шага по диагонали. Насколько лучше будут результаты с этими дополнительными шагами? Сможете ли вы еще улучшить результаты, введя девятое действие, которое не будет вызывать вообще никакого перемещения за исключением тех, которые вызваны действием ветра?

Упражнение 6.7 [Случайный ветер (программирование)]. Решите задачу о сетке с ветром с возможностью диагональных шагов, предполагая, что влияние ветра, если оно имеется, является случайным, иногда отклоняясь на 1 от среднего значения, заданного для каждой колонки, т. е. треть всего времени вы двигаетесь согласно заданным значениям, как и в предыдущем примере, но также третья всего

времени смещаетесь на одну клетку выше этого значения, и треть — на одну клетку ниже. Например, если вы в одной клетке справа от цели и осуществляете движение `left` (влево), то в одной трети случаев вы перемещаетесь в клетку непосредственно над целью, еще в одной трети случаев — на две клетки выше цели, и в оставшейся одной трети случаев вы попадаете в цель.

Упражнение 6.8. Как будет выглядеть диаграмма предшествующих состояний для алгоритма SARSA?

6.5. *Q*-обучение: управление по TD-методу с разделенной оценкой ценности стратегий

Одним из наиболее важных достижений в обучении с подкреплением стало развитие управления по TD-методу с разделенной оценкой ценности стратегий, известного как *Q*-обучение [Watkins (1989)]. Его простейшая форма, одноступенчатое *Q*-обучение, определяется следующим образом:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]. \quad (6.6)$$

В этом случае искомая функция ценности действия Q , непосредственно аппроксимирует Q^* , оптимальную функцию ценности действия, независимо от применяющейся стратегии. Это значительно упрощает анализ алгоритма и сохраняет в силе предложенные ранее доказательства сходимости. Стратегия по-прежнему определяет то, какие пары состояния—действие посещаются и корректируются. Тем не менее для обеспечения сходимости необходимо лишь, чтобы все пары продолжали корректироваться. Как было показано в гл. 5, это является минимальным требованием в том смысле, что каждый метод, гарантированно находящий оптимальную линию поведения, в общем случае должен удовлетворять данному условию. При таком условии и при таком варианте обычных условий стохастической аппроксимации для последовательности значений длины шага показано, что функция Q_t сходится к Q^* с вероятностью 1. Алгоритм *Q*-обучения в процедурной форме показан на рис. 6.12.

Что собой представляет диаграмма предшествующих состояний для *Q*-обучения? Правило (6.6) корректирует пару состояния—действие таким образом, что верхняя вершина, основание диаграммы, должна быть маленькой закрашенной вершиной-действием. Восстановление предшествующих состояний также происходит из вершин-действий, исходя из максимума по всем действиям, которые возможны в следующем состоя-

```

Инициализировать  $Q(s, a)$  произвольно
Повторять (для каждого эпизода):
    Инициализировать  $s$ 
    Повторять (для каждого шага эпизода):
        Найти  $a$  по  $s$ , используя стратегию, полученную из  $Q$ 
        (например,  $\varepsilon$ -жадную)
        Выполнить действие  $a$ , найти  $r, s'$ 
         $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
         $s \leftarrow s'$ ,
    Пока  $s$  не станет завершающим состоянием

```

Рис. 6.12. Q-обучение: алгоритм управления по TD-методу с разделенной оценкой ценности стратегий

нии. Таким образом, концевые вершины диаграммы должны быть всеми этими вершинами-действиями. Вспомните теперь, что, отыскивая максимумы для всех этих вершин «следующего действия», мы помечали их, соединяя дугой, как это показано на рис. 3.7. Можете ли вы теперь догадаться, какой будет требуемая диаграмма? Если да, то непременно сформулируйте свое предположение до того, как обратитесь к ответу на рис. 6.13.

Пример 6.6 (Прогулка у пропасти). В этом примере с сеткой сравним SARSA и Q-обучение и продемонстрируем различия между управлением по TD-методу с интегрированной оценкой ценности стратегий (SARSA) и управлением по TD-методу с разделенной оценкой ценности стратегий (Q-обучение). Рассмотрим сетку, изображенную в верхней части рис. 6.14. Это стандартная неприведенная, эпизодная задача с начальными и целевыми состояниями и стандартным набором действий: вверх, вниз, вправо и влево. Вознаграждение равно -1 за все переходы, кроме переходов в область обозначенную «Пропасть». Попадание в эту область влечет за собой вознаграждение -100 и немедленно отправляет агента обратно в начальную позицию. На нижней части рисунка изображено поведение метода SARSA и метода Q-обучения с ε -жадным выбором действий при $\varepsilon = 0.1$. После начального переходного этапа метод Q-обучения старается найти значения, соответствующие оптимальной стратегии, согласно которой путь проходит по краю пропасти. К сожалению, это приводит к периодическим падениям в пропасть



Рис. 6.13. Диаграмма предшествующих состояний для Q-обучения

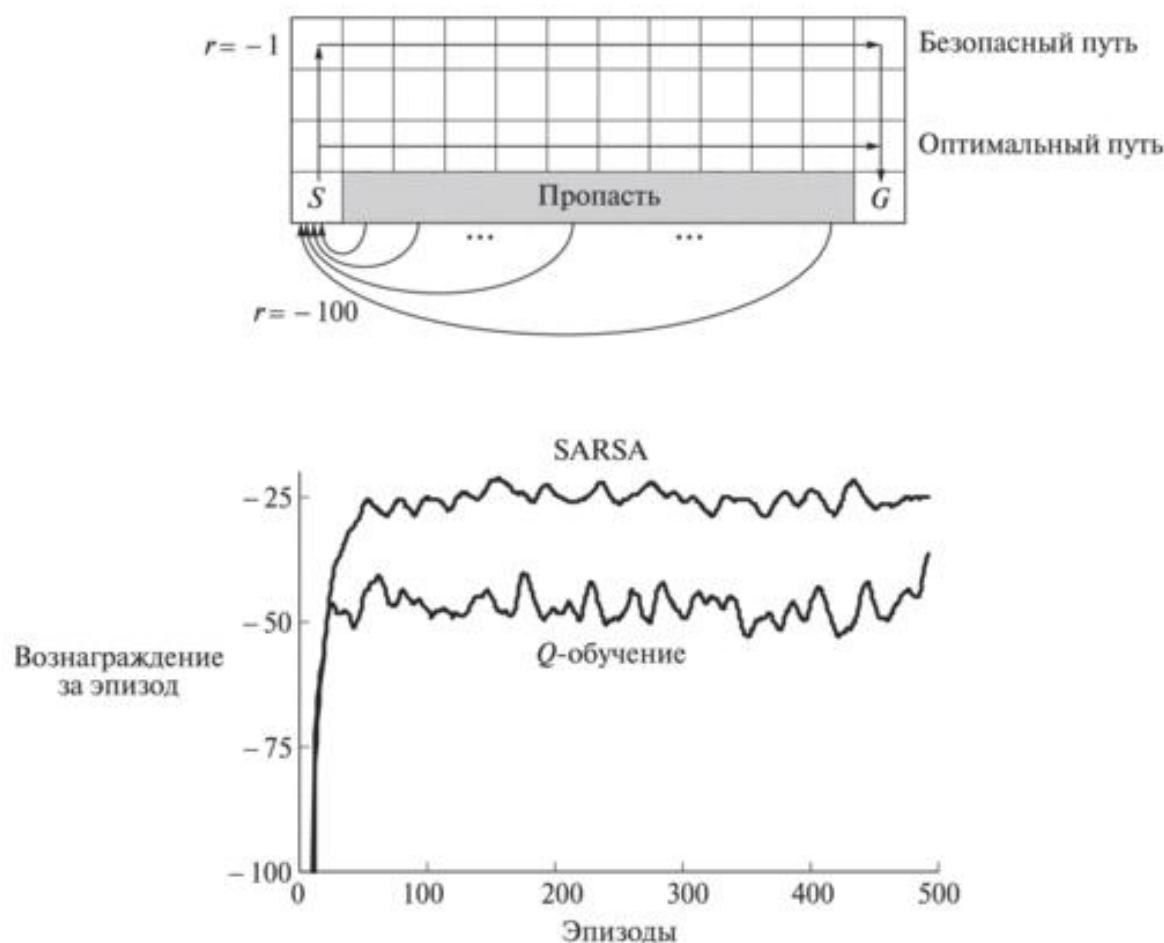


Рис. 6.14. Задача о пропасти. Сглаженные результаты, полученные для одного цикла

из-за ϵ -жадного выбора действий. Метод SARSA, с другой стороны, учитывает выбор действий и пытается найти более длинный, но и более безопасный путь, проходящий по верхней части сетки. Несмотря на то что метод Q -обучения в действительности находит значения, соответствующие оптимальной стратегии, показываемые им в реальном времени результаты хуже, чем у метода SARSA, который находит обходной путь. Конечно, если бы значение ϵ постепенно уменьшалось, то оба метода асимптотически сошлись бы к оптимальной стратегии.

Упражнение 6.9. Почему Q -обучение принято считать управлением по TD-методу с разделенной оценкой ценности стратегий?

Упражнение 6.10. Рассмотрим алгоритм обучения, аналогичный Q -обучению за исключением того, что вместо максимального значения для следующей пары состояние—действие в нем используется

ожидаемая ценность, с учетом того, насколько вероятно каждое действие в текущей стратегии, т. е. рассмотрим аналогичный Q -обучению алгоритм с правилом корректировки

$$\begin{aligned} Q(s_t, a_t) &\leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma E\{Q(s_{t+1}, a_{t+1}) \mid s_t\} - Q(s_t, a_t)] \leftarrow \\ &\leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \sum_a \pi(s_t, a) Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \end{aligned}$$

Является ли этот новый подход управлением по TD-методу с интегрированной оценкой ценности стратегий или управлением по TD-методу с разделенной оценкой ценности стратегий? Какой будет диаграмма предшествующих состояний этого алгоритма? При том же количестве имеющегося опыта окажется ли этот метод лучше или хуже метода SARSA? Что еще стоит учитывать при сравнении этого метода и SARSA?

*6.6. Методы исполнитель—критик

Методы исполнитель—критик — это TD-методы, имеющие раздельную структуру памяти для явного представления стратегии независимо от функции ценности. Структура, отвечающая за стратегию, называется *исполнитель*, так как она используется для выбора действий, а оцениваемая функция ценности называется *критик*, так как она критикует действия, осуществляемые исполнителем. Обучение всегда является управлением по TD-методу с интегрированной оценкой ценности стратегий: критик должен узнавать и критиковать любую стратегию, используемую в данный момент исполнителем. Критика принимает форму TD-ошибки. Этот скалярный сигнал, являющийся единственным сигналом от критика, регулирует все обучение как критика, так и исполнителя, как показано на рис. 6.15.

Методы исполнитель—критик являются естественным расширением методов сравнения с подкреплением (разд. 2.8) на TD-обучение и на всю задачу обучения с подкреплением. Обычно критик является функцией ценности состояния. После каждого выбора действия критик производит оценку нового состояния на предмет улучшения или ухудшения положения вещей по сравнению с тем, что ожидалось. Эта оценка является TD-ошибкой:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t),$$

где V — текущая функция ценности, реализованная критиком. Эта TD-ошибка может быть использована для оценки только что выбранного действия a_t , осуществленного в состоянии s_t . Если TD-ошибка по-



Рис. 6.15. Схема исполнитель–критик

ложительная, то это означает, что в будущем тенденция выбора действия a_t должна быть усиlena, тогда как, если TD-ошибка отрицательная, то данная тенденция должна быть ослаблена. Предположим, действия порождаются softmax-методом Гиббса:

$$\pi_t(s, a) = \Pr\{a_t = a \mid s_t = s\} = \frac{e^{p(s,a)}}{\sum_b e^{p(s,a)}},$$

где $p(a, s)$ — значение в момент времени t для модифицируемого параметра стратегии исполнителя, который показывает тенденцию к выбору (*предпочтение*) каждого действия a в каждом из состояний s . Тогда описанное выше усиление или ослабление может быть реализовано посредством увеличения или уменьшения параметра $p(a_t, s_t)$, например:

$$p(s_t, a_t) \leftarrow p(s_t, a_t) + \beta \delta_t,$$

где β является еще одним положительным параметром, определяющим длину шага.

Это лишь один пример метода исполнитель–критик. Другие его варианты выбирают действия другими способами или используют следы приемлемости, которые описаны в следующей главе. Другим распространенным способом варьирования, как и в методе сравнения с подкреплением, является учет дополнительных факторов, в зависимости от которых изменяется степень доверия к осуществляющему действию a_t . Например, один из наиболее общих факторов такого типа обратно пропорционален вероятности выбора действия a_t , что отражено в правиле

корректировки:

$$p(s_t, a_t) \leftarrow p(s_t, a_t) + \beta \delta_t (1 - \pi_t(s_t, a_t)).$$

Такой подход исследовался уже достаточно давно, особенно для случая с немедленным вознаграждением (см. работы [Sutton (1984); Williams (1992)], и не вполне удовлетворяет современным требованиям.

Многие ранние системы обучения с подкреплением, которые использовали TD-методы, были основаны на схеме исполнитель—критик (см. работы [Witten (1977); Barto, Sutton and Anderson (1983)]). С тех пор внимание в основном уделялось методам, которые пытаются найти функцию ценности действия и определяют стратегию исключительно на основании рассчитанных значений (к числу таких методов относятся SARSA и Q -обучение). По-видимому, это так просто сложилось исторически. Например, можно представить себе и промежуточные структуры, в которых будет формироваться как функция ценности действия, так и независимая стратегия. В любом случае, методы исполнитель—критик в настоящее время вызывают большой интерес, по всей видимости, вследствие следующих двух значительных и очевидных преимуществ, присущих им:

- Они требуют минимального объема вычислений для выбора действий. Рассмотрим случай, когда число возможный действий бесконечно — например, в случае, когда действие выбирается из некоторой непрерывной области. Любой метод, вычисляющий только ценность действий, чтобы выбрать какое-либо действие должен будет осуществить поиск среди всего бесконечного множества действий. Если стратегия хранится в явном виде, то проводить столь громоздкие вычисления при каждом выборе действия может и не потребоваться.
- Они могут выработать явным образом заданную стохастическую стратегию; т. е. они могут отыскать оптимальные вероятности выбора всевозможных действий. Это свойство оказывается полезным в случае с немарковскими задачами, в которых присутствует фактор конкуренции (см. работу [Singh, Jaakkola, and Jordan (1994)]).

Кроме того, обособление исполнителя в методах исполнитель—критик делает эти методы более привлекательными для использования в психологических и биологических моделях. В некоторых случаях может оказаться полезным введение специфических для данного случая ограничений на множество допускаемых стратегий.

*6.7. R-обучение для неприведенных продолжающихся задач

R-обучение является управлением по TD-методу с разделенной оценкой ценности стратегий для усовершенствованного варианта задачи обучения с подкреплением, в которой не происходит ни приведения, ни разбиения опыта на отдельные эпизоды с конечной выгодой. Данный метод стремится получить максимальное вознаграждение за временной шаг. Функции ценности для стратегии π определяются относительно среднего ожидаемого вознаграждения за временной шаг при данной стратегии:

$$\rho^\pi = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n E_\pi\{r_t\},$$

в предположении, что процесс является эргодическим (ненулевая вероятность перехода в любое состояние из любого другого состояния при любой стратегии) и, таким образом, вознаграждение ρ^π не зависит от начального состояния. В любом состоянии в течение продолжительного промежутка времени среднее вознаграждение одинаково, но имеет место неустойчивость. За некоторыми состояниями следует вознаграждение выше среднего, а за другими — ниже среднего. Эта неустойчивость и определяет ценность действия:

$$\tilde{V}^\pi(s) = \sum_{k=1}^{\infty} E_\pi\{r_{t+k} - \rho^\pi \mid s_t = s\}.$$

Аналогичным образом, ценностью пары состояние—действие является неустойчивое различие в вознаграждениях при начале из данного состояния и при осуществлении данного действия:

$$\tilde{Q}^\pi(s, a) = \sum_{k=1}^{\infty} E_\pi\{r_{t+k} - \rho^\pi \mid s_t = s, a_t = a\}.$$

Эти величины называются *относительными ценностями*, так как рассчитываются относительно среднего вознаграждения при данной стратегии.

Существуют тонкие различия между различными видами оптимальности в случае с неприведенной продолжающейся задачей. Тем не менее в большинстве практических задач достаточно просто упорядочить стратегии в соответствии с их средним вознаграждением за временной шаг, другими словами, в соответствии с их значениями ρ^π . Таким образом, будем считать все стратегии, которые достигают максимального значения ρ^π , оптимальными.

```

Инициализировать  $\rho$  и  $Q(s, a)$  произвольно, для всех  $s, a$ 
Повторять циклически:
     $s \leftarrow$  текущее состояние
    Выбрать действие  $a$  в состоянии  $s$ , используя стратегию поведения
        (например,  $\varepsilon$ -жадную)
    Выполнить действие  $a$ , найти  $r, s'$ 
     $Q(s, a) \leftarrow Q(s, a) + \alpha[r - \rho + \max_{a'} Q(s', a') - Q(s, a)]$ 
    Если  $Q(s, a) = \max_a Q(s, a)$ , то
         $\rho \leftarrow \rho + \beta[r - \rho + \max_{a'} Q(s', a') - \max_a Q(s, a)]$ 

```

Рис. 6.16. R-обучение: управление по TD-методу с разделенной оценкой ценности стратегий для неприведенной продолжающейся задачи. Скалярные величины α и β являются параметрами, определяющими размер шага

Помимо использования в случае с относительными ценностями, R-обучение является стандартным TD-методом управления, основывающимся на разделенной оценке ценности для обобщенной итерации по стратегиям (ОИС), наподобие Q-обучения. Этот метод использует две стратегии — стратегию поведения и стратегию оценивания, а также функцию ценности действия и предполагаемое среднее вознаграждение. Стратегия поведения используется для порождения опытных данных; это может быть ε -жадная стратегия по отношению к функции ценности действия. Стратегия оценивания та же самая, что и в случае с ОИС. Обычно это жадная стратегия по отношению к функции ценности действия. Если π является стратегией оценивания, то функция ценности действия Q является аппроксимацией \tilde{Q}^π , а среднее вознаграждение ρ является аппроксимацией ρ^π . Полный алгоритм представлен на рис. 6.16. Этот метод еще недостаточно хорошо изучен и поэтому должен рассматриваться как экспериментальный.

Пример 6.7 (Задача управления очередностью доступа). Это задача принятия решений, включающая в себя управление доступом к набору из n серверов. Клиенты четырех различных категорий, определяемых присвоенным им приоритетом, направляются в одну единственную очередь. Получая доступ к серверу, клиент платит вознаграждение, равное 1, 2, 4 или 8, в зависимости от его категории, и чем выше категория (т. е. чем выше приоритет пользователя), тем выше вознаграждение. На каждом временном шаге клиент из начала очереди либо принимается (получает доступ к одному из серверов), либо отвергается (удаляется из данной очереди). В обоих случаях на следующем временном шаге рассматривается уже следующий клиент из очереди. Очередь никогда

не исчерпывается, и доля (полученная случайным образом) клиентов с высоким приоритетом в очереди равна h . Очевидно, что клиент может быть обслужен, только если имеется свободный сервер. Каждый сервер освобождается с вероятностью r на каждом временном шаге. Для определенности будем полагать, что статистика поступления и ухода клиентов известна. Задача состоит в том, чтобы на каждом шаге решить, принять или отвергнуть очередного клиента на основании его категории и количества свободных серверов, таким образом, чтобы максимизировать долгосрочное вознаграждение без приведения. На рис. 6.17 изображено решение, найденное при помощи метода R -обучения при $n = 10$,

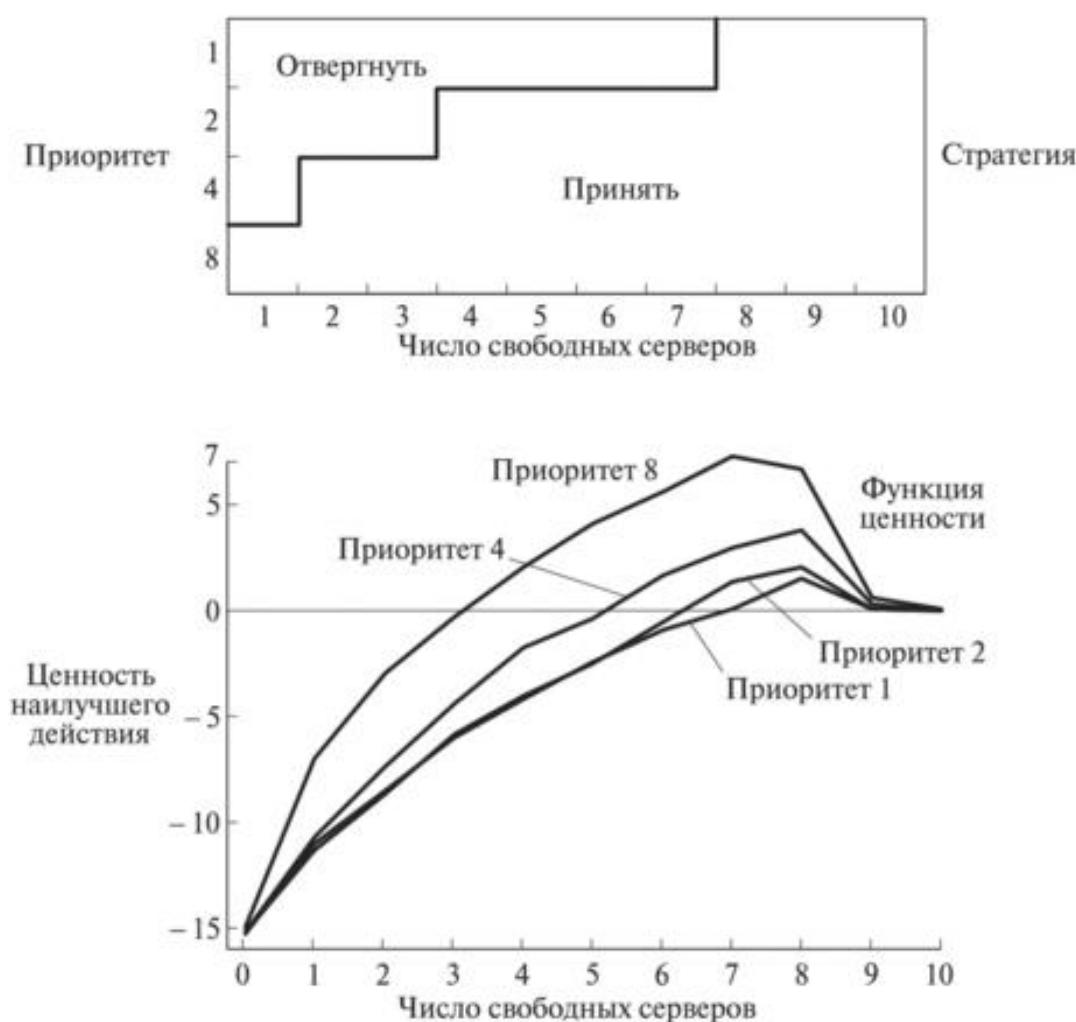


Рис. 6.17. Стратегия и функция ценности, найденные методом R -обучения в задаче управления доступом после двух миллионов шагов. Убывание функций в правой части, по-видимому, вызвано недостаточным количеством данных; многие из этих состояний никогда не испытывались. Найденное значение ρ было равным примерно 2,73

$h = 0,5$ и $p = 0,06$. Параметры R -обучения следующие:

$$\alpha = 0,1, \quad \beta = 0,01 \quad \text{и} \quad \varepsilon = 0,1.$$

Начальные ценности действий, а также ρ , были равны нулю.

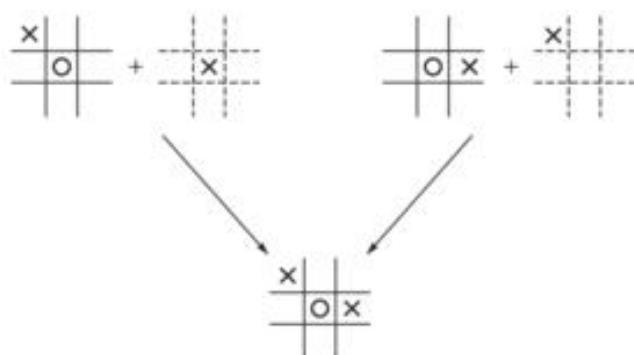
Упражнение 6.11*. Придумайте TD-метод управления с интегрированной оценкой ценности стратегий метод для неприведенной продолжающейся задачи.

6.8. Игры, послесостояния и другие особые случаи

В этой книге предпринимается попытка представить единый подход к решению широкого класса задач, но, естественно, всегда присутствуют особые задачи, которые лучше решать специальными методами. Например, наш общий подход включает в себя изучение функции ценности *действия*, но в гл. 1 был представлен TD-метод обучения игре в крестики-нолики, в котором формировалось нечто более похожее на функцию ценности *состояния*. При более детальном рассмотрении этого примера становится очевидно, что изучаемая в нем функция не являлась ни функцией ценности действия, ни функцией ценности состояния в обычном понимании. Традиционная функция ценности состояния оценивает состояния, в которых агент имеет право выбирать действия, но функция ценности состояния в примере с игрой в крестики-нолики оценивает позицию в игре *после* того, как агент совершил ход. Назовем такие элементы *послесостояниями*¹⁾, а функцию ценности — *функцией ценности послесостояний*. Послесостояния оказываются полезными в том случае, если известна некоторая начальная часть динамики окружающей среды, но не обязательно полностью вся динамика. Например, в играх обычно известен непосредственный результат наших ходов. В шахматах нам известно, какая сложится позиция на доске после нашего хода, но неизвестно, как на наш ход ответит противник. Функции ценности послесостояний являются естественным способом использовать преимущества такого типа знаний и, таким образом, вырабатывать более эффективные методы обучения.

Причина, по которой алгоритмы на основе послесостояний являются более эффективными, явно видна на примере с игрой в крестики-нолики. Традиционная функция ценности действия производила бы оценку ценности, исходя из позиций и ходов. Но многие пары позиция—ход приводят к однаковому результату, как это показано на примере выше.

¹⁾ По аналогии с последействием, послесвечением и т. п. — Прим. ред.



В таких случаях пары позиция—ход различны, но приводят к одинаковым «последозициям», и, таким образом, они должны иметь одинаковую ценность. Традиционной функции ценности действия пришлось бы отдельно производить оценку каждой из пар, тогда как функции ценности послесостояний сразу же оценит обе пары одинаково. Любая оценка правой пары позиция—ход автоматически будет перенесена на левую пару.

Послесостояния возникают во многих задачах, не только в играх. Например, в задачах управления очередью присутствуют действия, такие как допуск клиента к серверам, отказ клиенту или отбрасывание информации. В таких случаях действия фактически определяются на основании непосредственного эффекта от них, который полностью известен. Например, в описанном в предыдущем разделе примере с управлением очередностью доступа к серверам более эффективный метод обучения мог бы быть получен разделением динамики окружающей среды на непосредственный эффект от действия, который детерминирован и полностью известен, и на неизвестный случайный процесс, связанный с прибытием и уходом клиентов. Послесостоянием было бы число свободных серверов после действия, но перед тем, как случайный процесс сформирует следующее обычное состояние. Формирование функции ценности послесостояний на основе рассматриваемых послесостояний позволило бы объединить опыт всех действий, результатом выполнения которых является одно и то же количество свободных серверов. Это привело бы к значительному сокращению времени обучения.

Практически невозможно описать все многообразие задач специального вида и соответствующие им специфические алгоритмы обучения. Тем не менее основные принципы, представленные в этой книге, могут применяться достаточно широко. Например, методы послесостояний надлежащим образом описываются при помощи обобщенной итерации по стратегиям, со стратегией и функцией ценности (последующих со-

стояний) взаимодействующими, по сути дела, таким же способом. Во многих случаях по-прежнему будет возникать проблема выбора между управлением по TD-методу с интегрированной оценкой ценности стратегий и управлением с разделенной оценкой ценности стратегий, чтобы выявить потребность в изучающих действиях.

Упражнение 6.12. Опишите, каким образом задача о компании «Автомобили напрокат» (пример 4.2) могла бы быть скорректирована с применением послесостояний. Почему в случае с данной конкретной задачей такая корректировка, скорее всего, приведет к более быстрой сходимости?

6.9. Итоги

В этой главе был описан новый метод обучения, а именно, обучение на основе временных различий (TD-обучение), а также показано, как его можно применить к решению задачи обучения с подкреплением. Как обычно, мы разбивали всю задачу на задачу предсказания и задачу управления. TD-методы являются альтернативой методам Монте-Карло при решении задачи предсказания. В обоих случаях задача управления решалась на основе идеи обобщенной итерации по стратегиям (ОИС), которая была позаимствована из динамического программирования. Данная идея заключается в том, что приближенные стратегии и функции ценности должны взаимодействовать таким образом, чтобы каждая из них стремилась к своему оптимальному значению.

Один из двух процессов, имеющих место в ОИС, приближает функцию ценности к точному предсказанию выгод для текущей стратегии; это является задачей предсказания. Следствием другого процесса является локальное улучшение стратегии (фактически, стратегия является ε -жадной) по отношению к текущей функции ценности. Когда первый процесс основывается на опыте, возникает сложность, касающаяся поддержания достаточного уровня изучения. Как и в гл. 5, TD-методы управления группировались в зависимости от того, решали ли они эту проблему при помощи управления по TD-методу с разделенной оценкой ценности стратегий или управления по TD-методу с интегрированной оценкой ценности стратегий. Методы SARSA и исполнитель—критик являются методами с интегрированной оценкой ценности стратегий, *Q*-обучение и *R*-обучение — методами с разделенной оценкой.

Представленные в данной главе методы являются на сегодняшний день наиболее широко применяемыми методами обучения с подкреплением. Возможно, во многом это обусловлено их простотой: они могут

с минимальным количеством вычислений применяться в оперативном режиме, используя опыт, полученный от взаимодействия с окружающей средой; их можно практически полностью описать одним уравнением, которое решается небольшой компьютерной программой. В нескольких следующих главах эти алгоритмы будут расширены, что сделает их немного более сложными и при этом значительно более мощными. В основе всех новых алгоритмов будут лежать алгоритмы, представленные здесь: они смогут оперативно обрабатывать имеющийся опыт при сравнительно небольшом количестве вычислений, и в своей деятельности будут основываться на TD-ошибках. Представленные в данной главе частные случаи TD-методов должны, строго говоря, называться *одношаговыми, табличными, безмодельными* TD-методами. В следующих трех главах они будут расширены до многошаговой формы (связь с методами Монте-Карло), формы, использующей приближенное представление функции вместо таблиц (связь с искусственными нейронными сетями), и формы, которая включает в себя модель окружающей среды (связь с планированием и динамическим программированием).

Наконец, в данной главе TD-методы были рассмотрены исключительно в контексте задач обучения с подкреплением. Однако TD-методы, вообще говоря, являются более общими методами. Они позволяют осуществлять обучение долгосрочным предсказаниям относительно динамических систем. Например, TD-методы уместно применять для прогнозирования финансовых данных, продолжительности жизни, результатов выборов, погоды, поведения животных, энергопотребления или потребительского спроса. Только после того как TD-методы были проанализированы как методы предсказания в чистом виде, независимо от их использования в обучении с подкреплением, удалось добиться хорошего понимания их теоретических свойств. Но даже после этого приведенные выше возможные варианты их применения до сих пор не исследованы в должной мере.

6.10. Библиографические и исторические справки

Как отмечалось в гл. 1, идея TD-обучения изначально появилась при изучении психологии животных и в исследованиях по искусственноному интеллекту, в особенности в работах Сэмюэля [Samuel (1959)] и Клонфа [Klopff (1972)]. Работа Сэмюэля описана в учебном примере в разд. 11.2. С TD-обучением также связаны ранние идеи из работ Холланда [Holland (1975, 1976)] о согласованности предсказаний ценности. Эти идеи оказали влия-

ние на одного из авторов данной книги (Барто), который был аспирантом с 1970 по 1975 год в Мичиганском Университете, где преподавал Холланд. Идеи Холланда привели к созданию TD-подобных систем, включая работу Букера [Booker (1982)], а также цепочку приборов с зарядовой связью из работы Холланда [Holland (1986)], связанную с методом SARSA, как будет рассмотрено ниже.

6.1–2. Большая часть материала этих разделов взята из статьи Саттона [Sutton (1988)], включая алгоритм TD(0), пример с произвольными шагами и термин «обучение на основе временных различий». На способ анализа взаимосвязей между динамическим программированием и методами Монте-Карло повлияли работы [Watkins (1989)], [Werbos (1987)] и др. Использование диаграмм предшествующих состояний в данной и в других главах описывается впервые. Авторство примера 6.4, который никогда до этого не публиковался, принадлежит Саттону.

Сходимость табличного метода TD(0) в среднем была доказана в статье [Sutton (1988)], а сходимость с вероятностью 1 — в статье [Dayan (1992)]. В основу данных доказательств легла работа [Watkins and Dayan (1992)]. Данные результаты были подтверждены и дополнены в работах [Jaakkola, Jordan and Singh (1994)] и [Tsitsiklis (1994)] с использованием существующей мощной теории стохастической аппроксимации. Другие дополнения и обобщения рассматриваются в следующих двух главах.

6.3. Оптимальность TD-алгоритма при групповом обучении была установлена в статье Саттона [Sutton (1988)]. Термин «стохастически эквивалентная оценка» взят из литературы по адаптивному управлению [Goodwin and Sin (1984)]. Пониманию природы полученных результатов способствует работа [Barnard (1993)], в которой TD-алгоритм рассматривается как комбинация одного шага пошагового метода формирования модели марковской цепи и одного шага вычислительного процесса для получения предсказаний на основе данной модели.

6.4. Алгоритм SARSA был впервые исследован в работе [Rummery and Niranjan (1994)], где назван *модифицированным Q-обучением*. Название SARSA было предложено в работе [Sutton (1996)]. Сходимость одношагового табличного метода SARSA (частный случай метода, рассмотренный в данной главе) была доказана в готовящейся к публикации работе [Singh, Jaakkola, Littman and Szepesvari]. Пример «сетки с ветром» был предложен Томом Колтом (Tom Kalt).

Идея статьи Холланда [Holland (1986)] с цепочкой приборов с зарядовой связью (ПЗС) развилаась в алгоритм, тесно связанный с алгоритмом SARSA. Первоначальная идея о цепочки ПЗС содержала цепочки правил, инициировавших друг друга; она фокусировалась на передаче данных от текущего правила обратно к правилам, инициировавшим данное правило. Со временем идея цепочки ПЗС стала больше походить на TD-обучение, передавая данные обратно любому предшествующему во времени правилу,

а не только тем правилам, которые инициировали текущее правило. Современная форма цепочки ПЗС, упрощенная различными способами, практически идентична одношаговому алгоритму SARSA; этот вопрос подробно изложен в статье [Wilson (1994)].

- 6.5. *Q*-обучение было представлено в работе [Watkins (1989)]; приведенная в ней схема доказательства сходимости была позднее реализована в строгом доказательстве в работе [Watkins and Dayan, (1992)]. Более общие результаты, относящиеся к сходимости, получены в статьях [Jaakkola, Jordan, and Singh (1994)] и [Tsitsiklis (1994)].
 - 6.6. Схемы типа исполнитель—критик, использующие TD-обучение были впервые изучены в статье [Witten (1977)], а затем в работах [Barto, Sutton and Anderson (1983); Sutton (1984)], где и было предложено данное использование терминов «исполнитель» и «критик». В работах [Sutton (1984)] и [Williams (1992)] разработаны условия приемлемости, упомянутые в данном разделе. В работах [Barto (1995a)] и [Houk, Adams and Barto (1995)] описана модель того, как схема исполнитель—критик может реализовываться в мозге.
 - 6.7. *R*-обучение обязано своим происхождением статье [Schwartz (1993)]. В работах [Mahadevan (1996)], [Tadepalli and Ok (1994)] и [Bertsekas and Tsitsiklis (1996)] рассматривается обучение с подкреплением для неприведенных продолжающихся задач. В литературе неприведенная продолжающаяся задача часто называется задачей максимизации «среднего вознаграждения за временной шаг» или «задачей среднего вознаграждения». Название *R*-обучение, возможно, было выбрано просто как следующее по алфавиту за *Q*-обучением, но принято считать, что под этим подразумевается изучение *относительных* (relative) ценностей. Пример с задачей управления очередностью доступа к серверам был предложен в статье [Carlström and Nordström (1997)].
-

III

Единый подход

До настоящего момента рассматривались три класса методов решения задачи обучения с подкреплением: динамическое программирование, методы Монте-Карло и обучение на основе временных различий. Несмотря на то, что эти методы отличаются друг от друга, нельзя однозначно сказать, что какой-либо один метод предпочтительнее другого. Очень удобно и часто желательно применять несколько методов разного типа одновременно, т. е. применять комбинированный метод, сочетающий в себе разные подходы к решению. Для различных задач или различных частей одной задачи один метод может быть предпочтительнее другого. Нет причин, препятствующих такому выбору одного из методов, но осуществлять его следует на стадии их применения, а не разработки. В части III с единой точки зрения будут рассмотрены три типа первичных методов решения, описанных в части II.

Единый подход в данной части книги не сводится просто к поиску аналогий между этими методами. Будут разработаны отдельные алгоритмы, объединяющие в себе ключевые идеи одного или нескольких первичных методов решения. Вначале будет представлен механизм следов приемлемости, объединяющий методы Монте-Карло и методы временных различий. Затем рассмотрим аппроксимацию функции, реализующую операцию обобщения для совокупности состояний и действий. В заключение будут заново введены модели окружающей среды, позволяющие реализовать достоинства динамического программирования и эвристического поиска. Все они могут быть использованы совместно как составные части комбинированных методов.

7 Следы приемлемости

Следы приемлемости — один из основных механизмов обучения с подкреплением. Например, в известном алгоритме TD(λ) параметр λ указывает на использование следа приемлемости. Почти любой метод обучения на основе временных различий (TD-обучение), такой как Q -обучение или SARSA, может быть объединен со следами приемлемости, в результате чего получается более общий метод, реализующий более эффективное обучение.

Существует две точки зрения на следы приемлемости. С более общей теоретической точки зрения, которой здесь будет уделено особое внимание, они служат мостиком от TD-методов к методам Монте-Карло. Когда TD-методы дополняются следами приемлемости, образуется целое семейство новых методов, реализующий широкий спектр подходов. На одном конце этого спектра находятся методы Монте-Карло, а на другом — TD-методы. Между ними располагаются промежуточные варианты методов, которые часто оказываются лучше, чем «крайние» варианты. В этом смысле, следы приемлемости весьма эффективно объединяют TD-методы и методы Монте-Карло.

Другая точка зрения на следы приемлемости более механистична. Согласно ей, след приемлемости является временной записью, отражающей факт наступления события, такого как посещение некоторого состояния или осуществление какого-либо действия. Этот след помечает параметры памяти, соответствующие событию, как приемлемые для изменений, предпринимаемых в процессе обучения. Когда возникает TD-ошибка, то лишь приемлемые состояния или действия рассматриваются в качестве причины возникновения данной ошибки с присвоением им соответствующих значений показателей доверия или порицания. Таким образом, следы приемлемости помогают преодолеть разрыв между происходящими событиями и обучающей информацией. Как и сами TD-методы, следы приемлемости являются базисным механизмом присваивания значений временным показателям доверия.

По причинам, которые станут более ясными позднее, подход более теоретического характера к следам приемлемости называется *прямым*, а более механистического характера — *обратным*. Прямой подход наиболее удобен для понимания того, что именно вычисля-

ется при использовании следов приемлемости, тогда как обратный больше подходит для понимания самих алгоритмов. В данной главе приводятся оба подхода, а затем анализируется вопрос о том, в каком смысле их можно считать эквивалентными, т. е. когда их можно считать одним и тем же алгоритмом, рассматриваемым с двух разных точек зрения. Как обычно, вначале рассматривается задача прогнозирования, а затем задача управления. Это означает, что вначале рассматривается, каким образом следы приемлемости можно использовать для предсказания выгоды как функции состояния для фиксированной стратегии (т. е. для оценивания функции V^π). Только после исследования этих двух подходов к следам приемлемости для прогнозирования данные идеи будут распространены на ценности действий и методы управления.

7.1. *n*-шаговое TD-прогнозирование

Что представляет собой пространство методов между методами Монте-Карло и TD-методами? Рассмотрим оценивание функции V^π для выборки эпизодов, сформированной при стратегии π . Методы Монте-Карло осуществляют дублирование для каждого состояния, основываясь на всей последовательности имевших место вознаграждений в рамках данного состояния до конца эпизода. Дублирование простых TD-методов, с другой стороны, основывается всего на одном следующем вознаграждении, используя ценность состояния одним шагом позже в качестве заменителя остальных вознаграждений. В таком случае, один из типов промежуточных методов должен осуществлять дублирование, основываясь на промежуточном количестве вознаграждений: больше, чем одно, но меньше, чем все до завершения эпизода. Например, двухшаговое дублирование будет основываться на первых двух вознаграждениях и на расчетной ценности состояния двумя шагами позже. Аналогичным образом будут действовать трехшаговое дублирование, четырехшаговое и т. д. Рис. 7.1 иллюстрирует диапазон *n*-шаговых вариантов дублирования для функции V^π от одношагового дублирования в простом TD-методе слева до дублирования в методе Монте-Карло, действующего до завершения эпизода, справа.

Методы с *n*-шаговым дублированием являются все еще TD-методами, так как они по-прежнему изменяют более раннюю оценку, основываясь на том, как она отличается от более поздней оценки. В этом случае более поздняя оценка имеет место не одним, а *n* шагами позже. Методы, в которых временное различие имеет протяженность в *n* шагов, называются *n*-шаговыми TD-методами. Все TD-методы, представленные

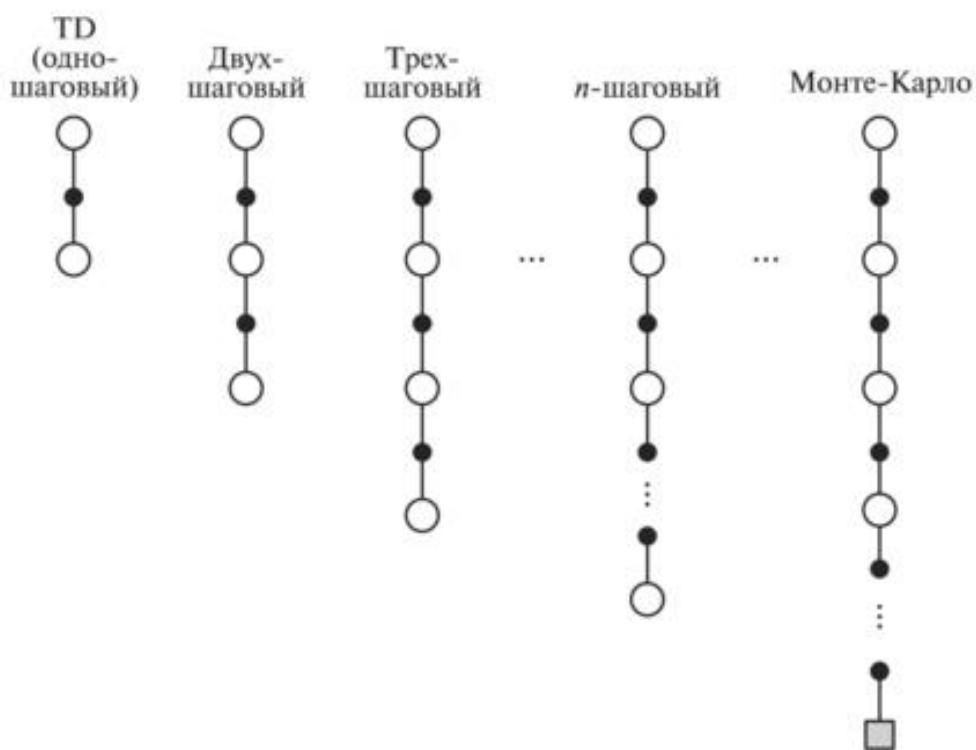


Рис. 7.1. Диапазон вариантов дублирования от одношагового дублирования в простом TD-методе до дублирования метода Монте-Карло, действующего до завершения эпизода. Между ними располагаются n -шаговые варианты дублирования, основывающиеся на n шагах с реальными вознаграждениями и на расчетной ценности n -го следующего состояния, все с учетом соответствующих коэффициентов приведения

в предыдущей главе, использовали одношаговое дублирование, и в дальнейшем они будут называться *одношаговыми TD-методами*.

Подходя более формально, рассмотрим применение дублирования к состоянию s_t как результат последовательности состояния-вознаграждение $s_t, r_{t+1}, s_{t+1}, r_{t+2}, \dots, r_T, s_T$ (пренебрегая действиями, чтобы упростить задачу). Известно, что в дублировании методов Монте-Карло оценка $V_t(s_t)$ ценности $V^\pi(s_t)$ корректируется в направлении значения полной выгода:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots + \gamma^{T-t-1} r_T,$$

где T является последним временным шагом в рамках эпизода. Назовем эту величину *целью* дублирования. В то время как целью дублирования Монте-Карло является полная выгода, целью одношагового дублирования является первое вознаграждение, увеличенное на неприведенную расчетную ценность следующего состояния:

$$R_t^{(1)} = r_{t+1} + \gamma V_t(s_{t+1}).$$

Это имеет смысл, так как величина $\gamma V_t(s_{t+1})$ занимает место оставшихся слагаемых

$$\gamma r_{t+2} + \gamma^2 r_{t+3} + \dots + \gamma^{T-t-1} r_T,$$

как это было описано в предыдущей главе. Суть в том, что данная идея имеет такой же смысл после двух шагов, как и после одного. При двух шагах цель имеет следующий вид:

$$R_t^{(2)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 V_t(s_{t+2}),$$

где величина $\gamma^2 V_t(s_{t+2})$ теперь занимает место слагаемых

$$\gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots + \gamma^{T-t-1} r_T.$$

В общем случае при n шагах цель имеет вид

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n}). \quad (7.1)$$

Данная величина иногда называется «уточненной n -шаговой усеченной выгодой», так как является выгодой, ограниченной n шагами, с введенной затем поправкой на данное ограничение путем добавления расчетной ценности n -го следующего состояния. Этот термин правильно описывает суть, но он слишком длинный. Поэтому $R_t^{(n)}$ называют просто n -шаговой выгодой в момент времени t .

Конечно, если эпизод завершается раньше чем через n шагов, то усечение n -шаговой выгоды происходит по окончании эпизода, результатом чего является обычная полная выгода. Другими словами, если $T - t \leq n$, то $R_t^{(n)} = R_t^{(T-t)} = R_t$. Таким образом, последние n вычисленных n -шаговых выгод всегда будут полными выгодами, и бесконечношаговая выгода всегда будет полной выгода. Такое определение позволяет рассматривать методы Монте-Карло как частный случай бесконечношаговых выгод. Все это согласуется с теми приемами эквивалентного подхода к эпизодным и продолжающимся задачам, которые были описаны в разд. 3.4. Там заключительное состояние рассматривалось как состояние, которое всегда переходит само в себя с нулевым вознаграждением. С учетом этого приема все n -шаговые выгоды, которые делятся до самого завершения или после него, имеют ту же ценность, что и полная выгода.

n-шаговое дублирование определяется как дублирование до n -шаговой выгоды. В случае с ценностью состояния в табличной форме прращение величины $V_t(s_t)$ (оценки ценности $V^\pi(s_t)$ в момент времени t) в результате n -шагового дублирования определяется как

$$\Delta V_t(s_t) = \alpha [R_t^{(n)} - V_t(s_t)],$$

где α , как обычно, является положительным параметром, определяющим размер шага. Приращения оценок ценностей других состояний $\Delta V_t(s)$ равны 0 для всех состояний $s \neq s_t$. Здесь n -шаговое дублирование определяется через приращение, а не через прямое правило корректировки, как в предыдущей главе, для того чтобы различать два несовпадающих способа осуществления корректировки. При *оперативной корректировке* уточнение происходит в течение эпизода, сразу же, как только вычислено приращение. В этом случае

$$V_{t+1}(s) = V_t(s) + \Delta V_t(s)$$

для всех состояний $s \in \mathcal{S}$. Данный случай рассмотрен в предыдущей главе. При независимой (автономной) корректировке, с другой стороны, приращения накапливаются «на стороне» и не используются для изменения оценок ценностей до завершения эпизода. В этом случае величина $V_t(s)$ постоянна на протяжении эпизода для всех s . Если в данном эпизоде она равна $V(s)$, то ее новое значение в следующем эпизоде будет составлять

$$V(s) + \sum_{t=0}^{T-1} \Delta V_t(s).$$

Ожидаемые значения всех n -шаговых выгод гарантированно улучшаются определенным образом на основе текущей функции ценности как аппроксимация истинной функции ценности. Для любой оценки V ожидаемое значение n -шаговой выгоды, использующей V , будет гарантированно лучшей оценкой функции V^π , чем сама функция V , относительно наихудшего состояния, т. е. наихудшая ошибка при новой оценке будет гарантированно меньше или равна аналогичной ошибке при V , умноженной на γ^n :

$$\max_s |E_\pi\{R_t^{(n)} \mid s_t = s\} - V^\pi(s)| \leq \gamma^n \max_s |V(s) - V^\pi(s)|. \quad (7.2)$$

Это называется *свойством уменьшения ошибки* n -шаговой выгоды. Учитывая свойство уменьшения ошибки, можно формально показать, что оперативные и автономные TD-методы прогнозирования, использующие n -шаговое дублирование, сходятся к верным прогнозам при соответствующих технических условиях. Таким образом, n -шаговые TD-методы образуют семейство эффективных методов с одношаговыми TD-методами и методами Монте-Карло в качестве крайних случаев.

Пример 7.1 (n -шаговые TD-методы в задаче о случайном блуждании). Рассмотрим применение n -шаговых TD-методов к задаче о случайных блужданиях, описанной в примере 6.2 и проиллюстрированной

рис. 6.5. Предположим, что первый эпизод развивается прямо из центрального состояния **C** направо, через **D** и **E**, а затем завершается справа с выгодаю, равной 1. Напомним, что расчетные ценности всех состояний вначале имеют среднее значение $V_0(s) = 0,5$. В результате полученного опыта одиношаговый метод изменит только оценку последнего состояния $V(E)$, которая получит значение 1, т. е. значение полученной выгоды. Двухшаговый метод, в свою очередь, увеличит ценности двух состояний, предшествующих завершению: оба значения $V(D)$ и $V(E)$ будут равны 1. Трехшаговый метод или любой *n*-шаговый метод с $n > 2$ повысит ценности всех трех состояний до 1, т. е. до одинакового значения. Какое значение *n* лучше? На рис. 7.2 представлены результаты простой эмпирической оценки большего процесса случайных блужданий с 19 состояниями (с результатом -1 слева и с нулевой начальной ценностью всех состояний). Показана среднеквадратическая ошибка в прогнозах в конце эпизода, усредненная по состояниям, по 10 эпизодам и по 100 повторениям всего эксперимента (для всех методов использовались одинаковые наборы шагов). Результаты показаны для оперативных и автономных *n*-шаговых TD-методов с различными значениями *n* и α . Исходя из опыта, оперативные методы со средними значениями *n* работают в данной задаче наилучшим образом. Это показывает, каким образом обобщение TD-методов и методов Монте-Карло до *n*-шаговых методов может оказаться потенциально более эффективным, чем любой из двух предельных вариантов методов.

Тем не менее *n*-шаговые TD-методы редко используются, поскольку они неудобны в вычислительном отношении. Для расчета *n*-шаговой выгоды необходимо дождаться прохождения *n* шагов для получения результирующих вознаграждений и состояний. Для больших *n* это может оказаться проблематичным, особенно при решении задач управления. *n*-шаговые TD-методы цепны с точки зрения теории и понимания родственных методов, которые более удобны при вычислениях. В нескольких следующих разделах идея *n*-шаговых TD-методов используется для объяснения и обоснования методов, основанных на следах приемлемости.

Упражнение 7.1. Как вы думаете, почему в примерах данной главы использовалась задача о случайных блужданиях большей размерности (19 состояний вместо 5)? Могло ли уменьшение количества состояний повлиять на преимущества использования различных значений *n*? Что если изменить результат левой части с 0 на -1 ? Повлияло ли бы это каким-то образом на выбор наилучшего значения *n*?

Упражнение 7.2. Как вы думаете, почему оперативные методы работают лучше автономных в представленном выше примере?

Упражнение 7.3. Обратите внимание на нижнюю часть рис. 7.2: график, соответствующий значению $n = 3$, отличается от других, сильно убывая при значениях α , намного меньших, чем у других аналогичных методов. На самом деле, то же самое наблюдалось также для $n = 5$, $n = 7$ и $n = 9$. Можете ли вы объяснить, с чем это могло быть связано? В действительности авторы данной книги сами не имеют точного ответа на данный вопрос.

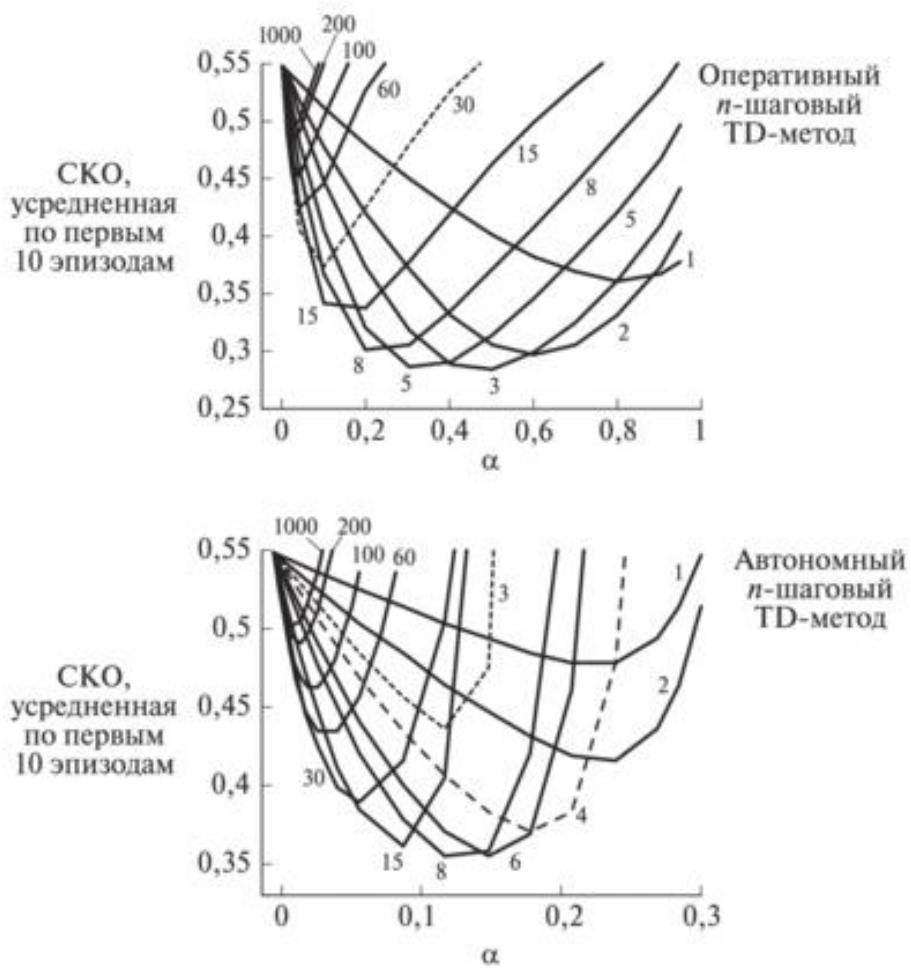


Рис. 7.2. Результаты применения n -шаговых TD-методов как функций от α при различных значениях n к решению задачи о случайных блужданиях с 19 состояниями. Показанный показатель качества представляет собой среднеквадратическую ошибку, определенную через разницу между истинными ценностями состояний и ценностями, найденными методами обучения, усредненную по 19 состояниям, по 10 первым попыткам и по 100 различным последовательностям шагов

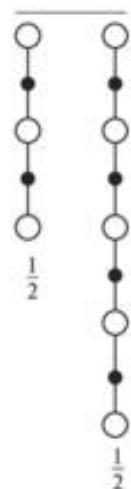
7.2. Прямой подход к методам $TD(\lambda)$

Методы дублирования могут применяться не только для n -шаговой выгода, но и для любого *усреднения* n -шаговых выгод. Например, дублирование может осуществляться для выгода, которая равна полусумме двухшаговой и четырехшаговой выгод:

$$R_t^{ave} = \frac{1}{2} R_t^{(2)} + \frac{1}{2} R_t^{(4)}.$$

Любая, даже бесконечная, последовательность выгод может быть усреднена таким образом при условии, что веса компонент положительны и в сумме дают 1. Так же как и отдельная n -шаговая выгода (7.2), общая выгода обладает свойством уменьшения ошибки и может быть использована для построения вариантов дублирования, обладающих свойством гарантированной сходимости. Данное усреднение порождает отдельную новую совокупность алгоритмов. Например, можно усреднить одношаговое и бесконечношаговое дублирования для получения еще одного метода, связывающего метод Монте-Карло с TD-методом. Можно даже усреднить дублирования, основывающиеся на опыте и основывающиеся на имеющейся модели, и получить простую комбинацию методов, основывающихся и на опыте, и на модели (см. гл. 9).

Дублирование, являющееся результатом усреднения более простых дублирований, называется *сложным дублированием*. Диаграмма предшествующих состояний для сложного дублирования состоит из диаграмм каждой из простых компонент с горизонтальной чертой наверху и с весовыми компонентами внизу. Например, упоминавшееся выше сложное дублирование, объединяющее половину двухшагового и половину четырехшагового дублирования, имеет следующую диаграмму:



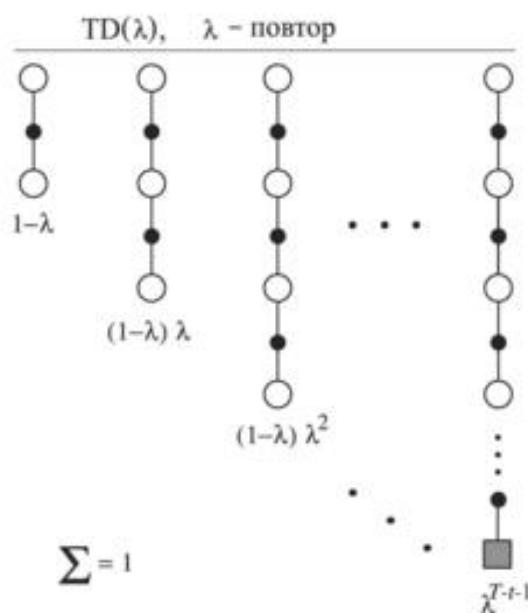


Рис. 7.3. Диаграмма предшествующих состояний для TD(λ). При $\lambda = 0$ все дублирование сокращается до одной, первой компоненты, представляющей собой одношаговое TD-дублирование. Однако при $\lambda = 1$ этой единственной компонентой является последняя компонента, а именно дублирование Монте-Карло

Алгоритм TD(λ) может быть представлен как один из конкретных способов усреднения n -шаговых дублирований. Данное усреднение содержит n -шаговые дублирования, взятые с весом λ^{n-1} , где $0 \leq \lambda \leq 1$ (рис. 7.3). Нормировочный множитель $1 - \lambda$ гарантирует, что сумма весов будет равна 1. Результирующее дублирование осуществляется для достижения выгоды, называемой λ -выгодой, которая определяется соотношением

$$R_t^\lambda = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}.$$

На рис. 7.4 изображена данная взвешивающая последовательность. Одношаговой выгода присваивается самый большой вес $1 - \lambda$; двухшаговой выгода присваивается второй по величине вес $(1 - \lambda)\lambda$; трехшаговой выгода присваивается вес $(1 - \lambda)\lambda^2$ и т. д. На каждом шаге вес уменьшается в λ раз. По достижении заключительного состояния все последующие n -шаговые выгоды приравниваются к R_t . Если необходимо, можно отделить эти члены от основной суммы, записав

$$R_t^\lambda = (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t. \quad (7.3)$$

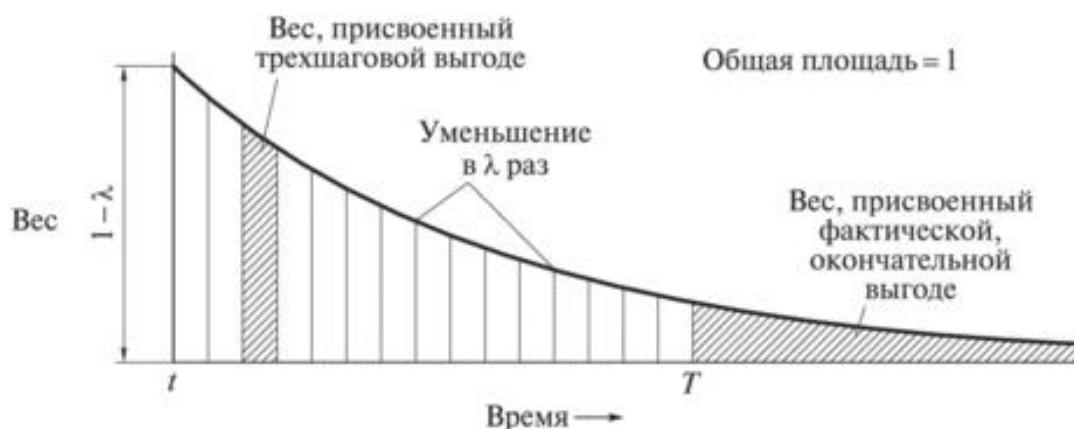


Рис. 7.4. Взвешивание в случае λ -выгоды для каждой из n -шаговых выгод

При помощи данного соотношения легко понять, что происходит при $\lambda = 1$. В этом случае член уравнения, содержащий сумму, обращающуюся в нуль, а остаточный член оказывается равным обычной выгоде R_t .

Таким образом, для $\lambda = 1$ дублирование для λ -выгоды совпадает с алгоритмом Монте-Карло, который в предыдущей главе назывался алгоритмом Монте-Карло с постоянной α (6.1). С другой стороны, если $\lambda = 0$, то λ -выгода сокращается до $R_t^{(1)}$, т. е. до одношаговой выгоды. Таким образом, при $\lambda = 0$ дублирование до λ -выгоды представляет собой то же самое, что и одношаговый TD-метод, именуемый TD(0).

λ -выгодный алгоритм определяется как алгоритм, который осуществляет дублирование, используя λ -выгоду. На каждом шаге t он рассчитывает приращение $\Delta V_t(s_t)$ ценности состояния, возникающего на данном шаге:

$$\Delta V_t(s_t) = \alpha [R_t^\lambda - V_t(s_t)]. \quad (7.4)$$

(Для всех других состояний $s \neq s_t$ приращения $\Delta V_t(s) = 0$.) Как и в случае с n -шаговыми TD-методами, корректировка может быть как оперативной, так и автономной.

Подход к изучению обучающих алгоритмов, рассматривавшийся до сих пор, называется теоретическим или *прямым*. В случае с каждым посещаемым состоянием происходит обзор всех будущих (имеющих место *впереди* во времени) вознаграждений и принимается решение, как наилучшим образом их скомбинировать. Представьте себе, что вы находитесь в потоке состояний и из каждого состояния смотрите вперед, определяя корректировку данного состояния, как это показано на рис. 7.5.

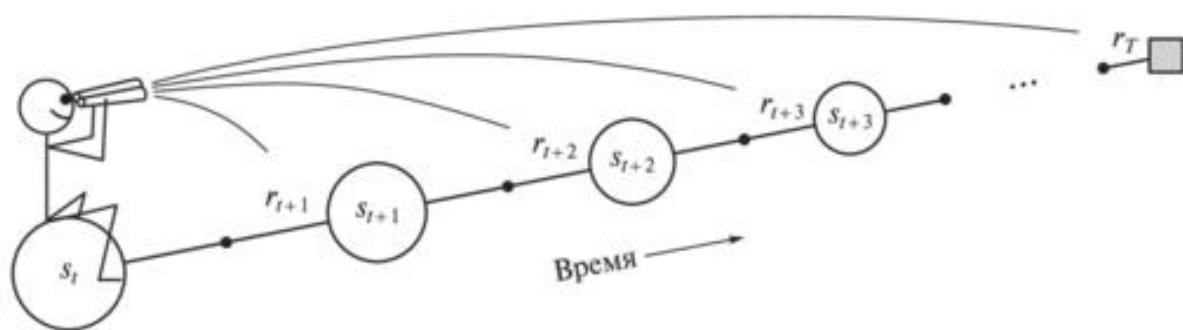


Рис. 7.5. Прямое или теоретическое представление. Корректировка каждого состояния определяется, исходя из состояний и вознаграждений, которые будут иметь место в будущем

После корректировки данного состояния вы перемещаетесь в следующее и больше никогда не сталкиваетесь с оставленным вами состоянием. В свою очередь будущие состояния многократно оцениваются и обрабатываются из каждой выгодной точки, предшествующей данному состоянию.

Пример 7.2 (λ -выгодный алгоритм в задаче о случайных блужданиях). На рис. 7.6 показаны результаты применения автономного λ -выгодного алгоритма в задаче о случайных блужданиях с 19 состояниями, который использовался с n -шаговыми TD-методами в примере 7.1. Данный эксперимент аналогичен эксперименту в n -шаговом случае за исключением того, что здесь вариируется λ вместо n . Заметьте, что наилучшие результаты были получены при среднем значении λ .

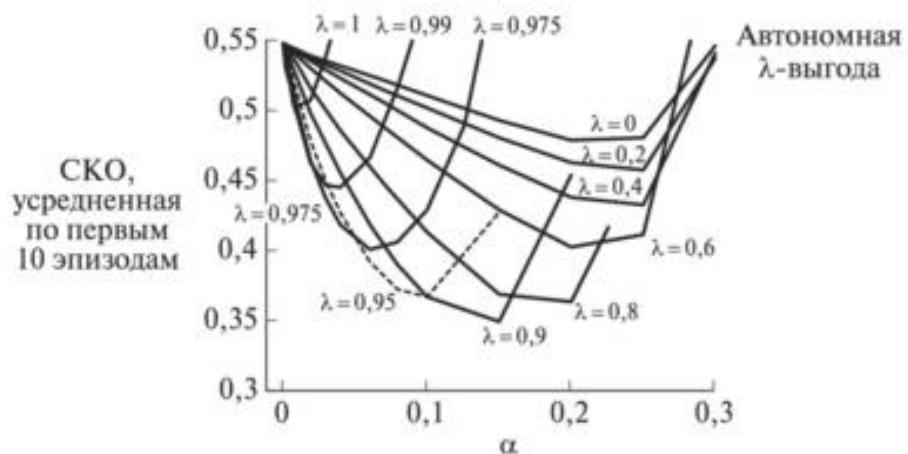


Рис. 7.6. Результаты действия автономного λ -выгодного алгоритма в задаче о случайных блужданиях с 19 состояниями

Как и $TD(\lambda)$ алгоритм, рассматриваемый λ -выгодный алгоритм является основой прямого подхода к следам приемлемости. На самом деле, в следующем разделе будет показано, что в автономном случае λ -выгодный алгоритм является $TD(\lambda)$ -алгоритмом. λ -выгодный и $TD(\lambda)$ методы используют параметр λ , чтобы переходить от одношаговых TD-методов к методам Монте-Карло. Данный способ перехода интересен сам по себе, но не очевидно, лучше он или хуже способа перехода, предлагаемого n -шаговыми методами, когда варьируется значение n . В конечном счете, наиболее веской причиной использовать λ -способ сочетания n -шаговых дублирований является возможность применения простого алгоритма — $TD(\lambda)$. Это скорее прикладной вопрос, чем теоретический. В следующих разделах мы рассмотрим механистический или обратный подход к следам приемлемости в случае с $TD(\lambda)$ -алгоритмом.

Упражнение 7.4. Параметр λ характеризует то, насколько быстро убывает экспоненциальная зависимость взвешивания на рис. 7.4 и, таким образом, насколько далеко в будущее распространяется λ -выгодный алгоритм, определяя параметры своего дублирования. Однако в ряде случаев использовать такой фактор, как λ , для определения скорости убывания неудобно. В некоторых случаях удобнее задать временную константу или период полуспада. Каким будет уравнение, связывающее λ и период полуспада, т. е. время, за которое последовательность весов уменьшится до половины своего начального значения?

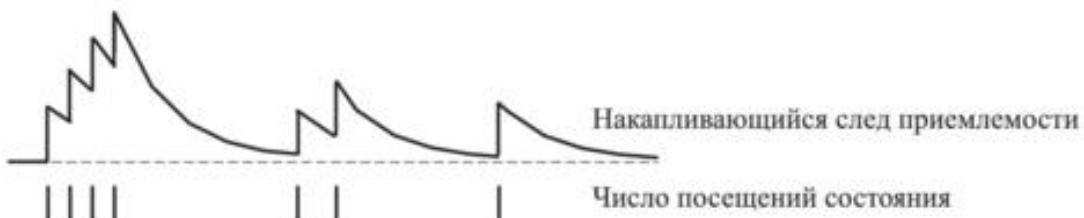
7.3. Обратный подход к методам $TD(\lambda)$

В предыдущем разделе мы рассмотрели прямой или теоретический подход к табличному $TD(\lambda)$ -алгоритму как способу сочетания дублирований, параметры которых изменяются от параметров, соответствующих TD-методу, до параметров, соответствующих методу Монте-Карло. В данном разделе будет определен $TD(\lambda)$ -алгоритм с механистической точки зрения, а в следующем разделе будет показано, что такой подход хорошо согласуется с прямым. Механистическая или обратная трактовка алгоритма $TD(\lambda)$ удобна, поскольку проста как концептуально, так и с вычислительной точки зрения. В частности, прямой подход в чистом виде не реализуется, так как вследствие использования в нем знаний о том, что произойдет на много шагов позже, в нем нарушается причинно-следственная связь. Обратный подход обеспечивает причинно-следственный инкрементный механизм для аппроксимации прямого подхода и для точного его исполнения в автономном случае.

В обратном подходе к алгоритму TD(λ) имеется дополнительная переменная памяти, соответствующая каждому состоянию, а именно его *след приемлемости*. След приемлемости для состояния s в момент времени t обозначается $e^t(s) \in R^+$. На каждом шаге следы приемлемости для всех состояний убывают с коэффициентом $\gamma\lambda$, а след приемлемости для посещаемого на данном шаге состояния увеличивается на 1:

$$e^t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{при } s \neq s_t, \\ \gamma\lambda e_{t-1}(s) + 1 & \text{при } s = s_t \end{cases} \quad (7.5)$$

для всех состояний $s \in \mathcal{S}$, где γ — коэффициент приведения, а λ — параметр, заданный в предыдущем разделе. С этого момента λ будет называться *параметром затухания следа*. Данный вид следа приемлемости называется *накапливающимся*, так как он накапливается каждый раз при посещении состояния и затем постепенно убывает, когда состояние не посещается, как это показано ниже:



След приемлемости все время регистрирует, посещение каких состояний имело место недавно, где смысл понятия «недавно» определяется с помощью коэффициента $\gamma\lambda$. Следы показывают степень приемлемости каждого состояния при происходящих изменениях в обучении, если возникает подкрепляющее событие. Представляющими интерес подкрепляющими событиями являются одношаговые TD-ошибки, возникающие в каждый момент. Например, TD-ошибка при прогнозировании ценности состояния имеет вид

$$\delta_t r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t). \quad (7.6)$$

В обратном подходе к алгоритму TD(λ) общий сигнал TD-ошибки инициирует пропорциональную корректировку всех состояний, которые недавно посещались, о чем сигнализируют их следы:

$$\Delta V_t(s) = \alpha \delta_t e_t(s) \quad \text{для всех } s \in \mathcal{S}. \quad (7.7)$$

Как всегда, приращения могут осуществляться на каждом шаге, формируя оперативный алгоритм, или накапливаться до окончания эпизода, формируя автономный алгоритм. В обоих случаях уравнения (7.5)–(7.7)

Инициализировать $V(s)$ произвольно, $e(s) = 0$, для всех $s \in \mathcal{S}$
Повторять (для каждого эпизода):
 Инициализировать s
 Повторять (для каждого шага эпизода):
 $a \leftarrow$ действие, задаваемое стратегией π для состояния s
 Выполнить действие a , найти вознаграждение r и следующее состояние s'
 $\delta \leftarrow r + \gamma V(s') - V(s)$
 $e(s) \leftarrow e(s) + 1$
 Для всех s :
 $V(s) \leftarrow V(s) + \alpha \delta e(s)$
 $e(s) \leftarrow \gamma \lambda e(s)$
 $s \leftarrow s'$
Пока s не станет завершающим состоянием

Рис. 7.7. Оперативный табличный алгоритм TD(λ)

дают механистическое определение TD(λ)-алгоритма. Запись оперативного алгоритма TD(λ) приведена на рис. 7.7.

Обратный подход к алгоритму TD(λ) ориентирован назад во времени. В каждый момент наблюдается текущая TD-ошибка, и она присваивается каждому предыдущему состоянию в соответствии со следом приемлемости данного состояния в тот момент времени. Можете представить, что вы находитесь в потоке состояний, рассчитываете TD-ошибки, и сообщаете их обратно состояниям, которые вы перед этим посещали, как это показано на рис. 7.8. Там, где встречаются TD-ошибки и следы, происходит корректировка согласно формуле (7.7).

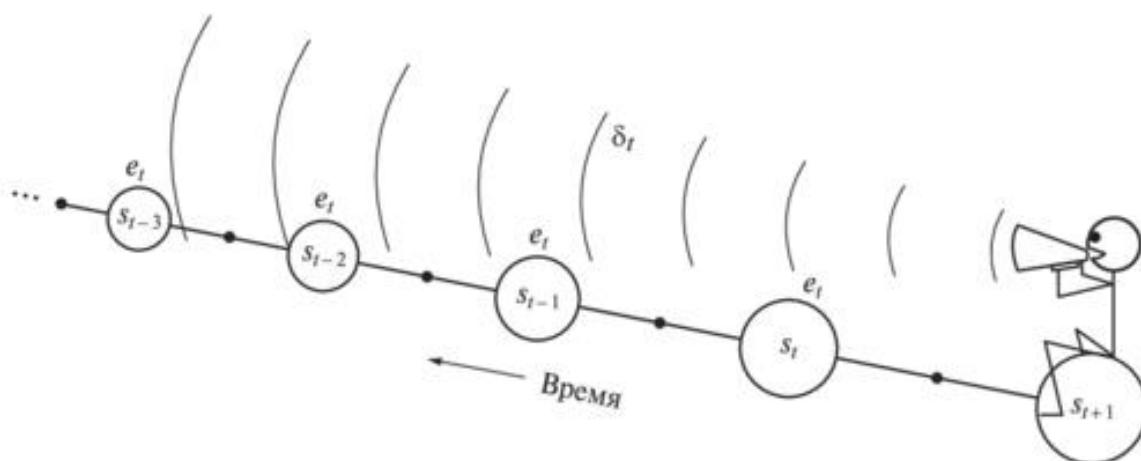


Рис. 7.8. Обратное или механистическое представление. Каждая корректировка зависит от текущей TD-ошибки в сочетании со следами предшествующих событий

Чтобы освоиться с обратным подходом, рассмотрим, что происходит при различных значениях λ . Если $\lambda = 0$, то, согласно (7.5), в момент времени t все следы равны нулю, за исключением следа, соответствующего состоянию s_t . Таким образом, TD(λ)-корректировка (7.7) превращается в простое TD-правило (6.2), TD(0). Согласно рис. 7.8, TD(0) представляет собой случай, когда только одно состояние, предшествующее текущему, меняется TD-ошибкой. При больших значениях λ , но все еще при $\lambda < 1$, изменяется большее количество предыдущих состояний, но чем раньше имело место состояние, тем меньше оно изменяется, так как его след приемлемости все меньше, как это показано на рис. 7.8. Говорят, что более ранние состояния имеют меньший коэффициент доверия для TD-ошибки.

Если $\lambda = 1$, коэффициент доверия, присваиваемый предыдущим состояниям, убывает только с коэффициентом γ на каждом шаге. Данная стратегия позволяет достичь модели поведения Монте-Карло. Стоит напомнить, к примеру, что TD-ошибка δ_t содержит в себе неприведенное слагаемое r_{t+1} . При возвращении обратно на k шагов данный период должен быть приведен с коэффициентом γ^k , как и любое вознаграждение, составляющее выгоду, что является как раз тем значением, которого достигают убывающие следы приемлемости. Если $\lambda = 1$ и $\gamma = 1$, то следы приемлемости со временем не убывают. В этом случае метод ведет себя как метод Монте-Карло для неприведенной эпизодной задачи. При $\lambda = 1$ алгоритм также называют TD(1).

TD(1) является более общим способом реализации алгоритмов Монте-Карло, чем способы, представленные ранее. Кроме того, данный способ значительно расширяет возможности применения данных методов. В то время как применение методов Монте-Карло ограничено только эпизодными задачами, TD(1) может также применяться в приведенных продолжающихся задачах и, более того, может быть реализован в пошаговой и оперативной форме. Одним из недостатков методов Монте-Карло является то, что они не могут обучаться, пока не завершится эпизод. Например, если метод Монте-Карло осуществляет некоторое действие, результатом которого является очень маленькое вознаграждение, но при этом эпизод еще не завершился, то направление деятельности агента никак не изменится в течение эпизода. Оперативный TD(1)-метод, в свою очередь, обучается n -шаговым TD-способом в течение всего эпизода, где n является числом предшествующих шагов. Если во время эпизода происходит очень негативное или очень позитивное событие, методы управления, основанные на TD(1), немедленно вносят изменения в модель своего поведения в данном эпизоде.

7.4. Эквивалентность прямого и обратного представлений

В данном разделе мы покажем, что автономный алгоритм $\text{TD}(\lambda)$, описанный выше с механистической точки зрения, достигает тех же весовых корректировок, что и автономный λ -выгодный алгоритм. Таким образом прямая (теоретическая) и обратная (механистическая) точки зрения на $\text{TD}(\lambda)$ -алгоритм эквивалентны. Обозначим как $\Delta V_t^\lambda(s_t)$ корректировку значения $V(s_t)$ в момент времени t в соответствии с λ -выгодным алгоритмом (7.4), а через $\Delta V_t^{\text{TD}}(s)$ — корректировку в момент времени t состояния s в соответствии с механистическим определением $\text{TD}(\lambda)$ -алгоритма (7.7). Основная цель — показать, что сумма всех корректировок в течение эпизода одинакова для обоих алгоритмов:

$$\sum_{t=0}^{T-1} \Delta V_t^{\text{TD}}(s) = \sum_{t=0}^{T-1} \Delta V_t^\lambda(s_t) \mathcal{I}_{ss_t} \quad \text{для всех } s \in \mathcal{S}, \quad (7.8)$$

где \mathcal{I}_{ss_t} — индикатор тождественности, равный 1, при $s = s_t$ и 0 во всех остальных случаях. Заметьте, что накапливающийся след приемлемости может быть задан явно (нерекуррентно):

$$e_t(s) = \sum_{k=0}^t (\gamma\lambda)^{t-k} \mathcal{I}_{ss_k}.$$

Таким образом, левую часть уравнения (7.8) можно записать следующим образом:

$$\begin{aligned} \sum_{t=0}^{T-1} \Delta V_t^{\text{TD}}(s) &= \sum_{t=0}^{T-1} \alpha \delta_t \sum_{k=0}^t (\gamma\lambda)^{t-k} \mathcal{I}_{ss_k} = \sum_{k=0}^{T-1} \alpha \sum_{t=0}^k (\gamma\lambda)^{k-t} \mathcal{I}_{ss_t} \delta_k = \\ &= \sum_{t=0}^{T-1} \alpha \sum_{k=t}^{T-1} (\gamma\lambda)^{k-t} \mathcal{I}_{ss_t} \delta_k = \sum_{t=0}^{T-1} \alpha \mathcal{I}_{ss_t} \sum_{k=t}^{T-1} (\gamma\lambda)^{k-t} \delta_k. \end{aligned} \quad (7.9)$$

Далее обратимся к правой части соотношения (7.8). Рассмотрим отдельную корректировку в λ -выгодном алгоритме:

$$\begin{aligned} \frac{1}{\alpha} \Delta V_t^\lambda(s_t) &= R_t^\lambda - V_t(s_t) = \\ &= -V_t(s_t) + (1-\lambda)\lambda^0[r_{t+1} + \gamma V_t(s_{t+1})] + \\ &\quad + (1-\lambda)\lambda^1[r_{t+1} + \gamma r_{t+2} + \gamma^2 V_t(s_{t+2})] + \\ &\quad + (1-\lambda)\lambda^2[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V_t(s_{t+3})] + \\ &\quad \vdots \quad \vdots \quad \vdots \quad \ddots \end{aligned}$$

Исследуем первую колонку в скобках — все слагаемые r_{t+1} с их весами $(1 - \lambda)$, умноженными на λ в соответствующей степени. Оказывается, что все веса в сумме дают 1. Таким образом, можно вместо всей первой колонки записать просто неприведенное слагаемое r_{t+1} . Аналогичную операцию можно проделать со второй колонкой, начиная со второго ряда, получив сумму слагаемых, равную $\gamma\lambda r_{t+2}$. Проделывая данную операцию со всеми колонками, получаем:

$$\begin{aligned} \frac{1}{\alpha} \Delta V_t^\lambda(s_t) &= -V_t(s_t) + (\gamma\lambda)^0 [r_{t+1} + \gamma V_t(s_{t+1}) - \gamma\lambda V_t(s_{t+1})] + \\ &\quad + (\gamma\lambda)^1 [r_{t+2} + \gamma V_t(s_{t+2}) - \gamma\lambda V_t(s_{t+2})] + \\ &\quad + (\gamma\lambda)^2 [r_{t+3} + \gamma V_t(s_{t+3}) - \gamma\lambda V_t(s_{t+3})] + \\ &\quad \vdots \\ &= (\gamma\lambda)^0 [r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)] + \\ &\quad + (\gamma\lambda)^1 [r_{t+2} + \gamma V_t(s_{t+2}) - V_t(s_{t+1})] + \\ &\quad + (\gamma\lambda)^2 [r_{t+3} + \gamma V_t(s_{t+3}) - V_t(s_{t+2})] + \\ &\quad \vdots \\ &\approx \sum_{k=t}^{\infty} (\gamma\lambda)^{k-t} \delta_k = \sum_{k=t}^{T-1} (\gamma\lambda)^{k-t} \delta_k. \end{aligned}$$

Полученная выше аппроксимация является точной в случае с автономной корректировкой. В этом случае значение V_t одинаково для всех t . Последний шаг является точным (не аппроксимацией), так как все опущенные слагаемые δ_t относятся к фиктивным шагам «после» достижения заключительного состояния. Все эти шаги имеют нулевое вознаграждение и нулевую ценность; таким образом, все их ошибки δ_t также равны нулю. Таким образом, мы показали, что в автономном случае правая часть выражения (7.8) может быть записана следующим образом:

$$\sum_{t=0}^{T-1} \Delta V_t^\lambda(s_t) \mathcal{I}_{ss_t} = \sum_{t=0}^{T-1} \alpha \mathcal{I}_{ss_t} \sum_{k=t}^{T-1} (\lambda\gamma)^{k-t} \delta_k,$$

что, по сути, является соотношением (7.9). Итак, соотношение (7.8) доказано.

В случае автономной корректировки проделанная выше аппроксимация будет тем точнее, чем меньше параметр α и, таким образом, чем меньше меняется V_t внутри эпизода. Даже в оперативном случае можно

ожидать, что корректировки $\text{TD}(\lambda)$ и λ -выгодным алгоритмами будут схожими.

Предположим, что в течение эпизода приращения достаточно малы и что по ходу эпизода оперативный $\text{TD}(\lambda)$ -метод осуществляет, по сути, такую же корректировку, что и λ -выгодный алгоритм. Интересные вопросы по-прежнему остаются по поводу того, что происходит *в течение* эпизода. Рассмотрим корректировку ценности состояния s_t в середине эпизода, в момент времени $t+k$. При действии $\text{TD}(\lambda)$ -метода результат в рассматриваемый момент времени $t+k$ будет таким же, как если бы проводилась λ -выгодная корректировка и при этом наблюдавшееся последним состояние считалось бы заключительным состоянием эпизода с ненулевой заключительной ценностью, равной его текущей расчетной ценности. Такая взаимосвязь сохраняется от шага к шагу по мере возникновения новых состояний.

Пример 7.3 (Случайные блуждания и метод $\text{TD}(\lambda)$). Так как автономный метод $\text{TD}(\lambda)$ эквивалентен λ -выгодному алгоритму, у нас уже имеются результаты применения автономного метода $\text{TD}(\lambda)$ к задаче о случайных блужданиях, содержащей 19 состояний; результаты приведены на рис. 7.6. Сравнительные результаты для оперативного метода $\text{TD}(\lambda)$ приведены на рис. 7.9. Обратите внимание на то, что оперативный алгоритм работает лучше при более широком диапазоне параметров. Это часто является причиной применения оперативных методов.

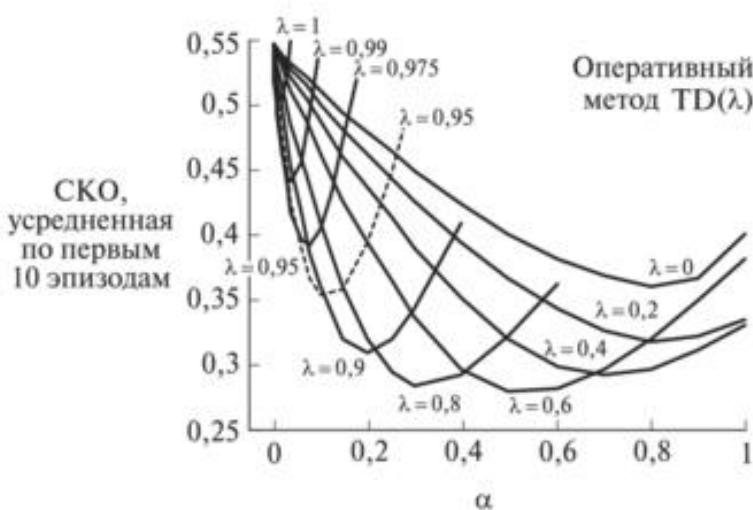


Рис. 7.9. Результаты действия оперативного метода $\text{TD}(\lambda)$ в задаче о случайных блужданиях с 19 состояниями

Упражнение 7.5*. Несмотря на то что оперативный алгоритм TD(λ) является лишь аппроксимацией λ -выгодного алгоритма, может быть, существует несколько отличающийся TD-метод, который будет полностью эквивалентен даже в оперативном случае. Например, можно определить TD-ошибку как

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_{t-1}(s_t),$$

а n -шаговую выгоду как

$$R_t^{(n)} = r_{t+1} + \dots + \gamma^{n-1} r_{t+n} + \gamma^n V_{t+n-1}(s_{t+n}).$$

Покажите, что модифицированный таким образом TD(λ)-алгоритм достигнет в точности корректировки

$$\Delta V_t(s_t) = \alpha [R_t^\lambda - V_{t-1}(s_t)],$$

даже в случае с оперативной корректировкой с большими значениями α . В каких случаях такой модифицированный метод может быть лучше или хуже, чем описанный до этого алгоритм? Смоделируйте эксперимент, по результатам которого можно было бы оценить относительные качества двух данных алгоритмов.

7.5. SARSA(λ)

Каким образом можно применить следы приемлемости не только к прогнозированию, как в TD(λ)-методе, но и к управлению? Как обычно, основной идеей одного из популярных подходов является просто изучение ценностей действий $Q_t(s, a)$ вместо ценностей состояний $V_t(s)$. В данном разделе мы покажем, каким образом следы приемлемости могут сочетаться непосредственно с методом SARSA для получения оперативного TD-метода управления. Вариант SARSA со следами приемлемости будет называться *SARSA(λ)*, а первоначальная версия, представленная в предыдущей главе, будет тогда называться *одношаговый алгоритм SARSA*.

Идея алгоритма SARSA(λ) состоит в применении TD(λ)-метода прогнозирования к парам состояние—действие вместо просто действий. Очевидно, в этом случае необходим след для каждой пары состояния—действие, а не только для каждого состояния. Обозначим как $e_t(s, a)$ след для каждой пары состояния—действие a, s . За исключением замен, связанных с введением пары состояния—действие — $V_t(s)$ на $Q_t(s, a)$ и $e_t(s)$ на $e_t(s, a)$, данный метод, по сути повторяет метод TD(λ):

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha \delta_t e_t(s, a) \quad \text{для всех } s, a,$$

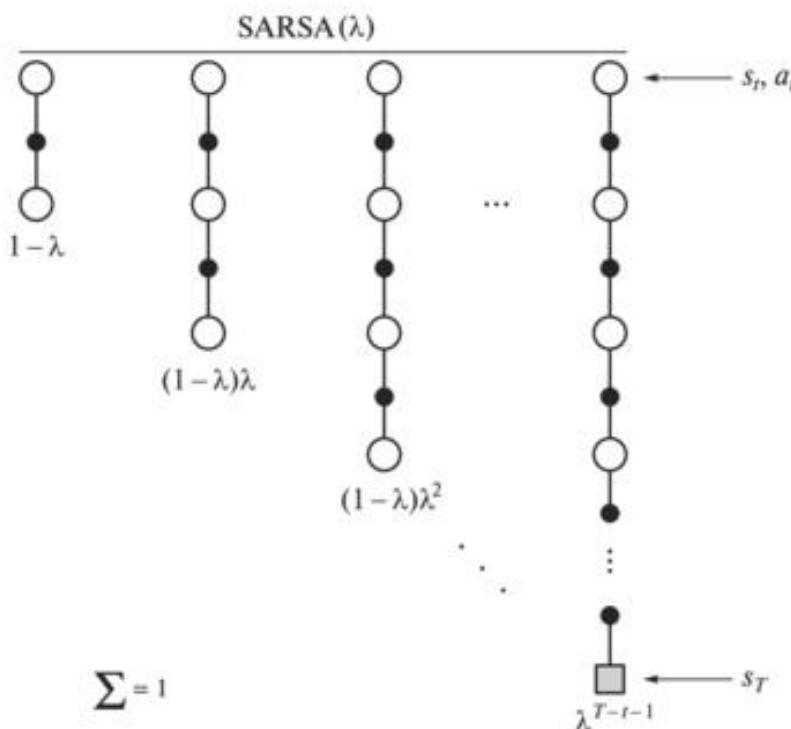


Рис. 7.10. Диаграмма предшествующих состояний метода SARSA(λ)

где

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)$$

и

$$e_t(s, a) = \begin{cases} \gamma \lambda e_{t-1}(s, a) & \text{при } s = s_t \text{ и } a = a_t; \\ \gamma \lambda e_{t-1}(s, a) & \text{в остальных случаях;} \end{cases} \quad \text{при всех } s, a. \quad (7.10)$$

На рис. 7.10 приведена диаграмма предшествующих состояний для метода SARSA(λ). Обратите внимание на схожесть данной диаграммы с диаграммой алгоритма TD(λ) (рис. 7.3). Первое дублирование распространяется на один полный шаг вперед, на следующую пару состояние—действие. Второе дублирование распространяется на два шага, и т. д. Последнее дублирование основывается на полной выгоде. Взвешивание каждого дублирования точно такое же, как и в TD(λ) и λ -выгодном алгоритмах.

Одношаговые SARSA и SARSA(λ) являются алгоритмами с интегрированной оценкой ценности стратегий в том смысле, что они оценивают ценности действий $Q^\pi(s, a)$ при текущей стратегии π и затем постепенно улучшают стратегию, основываясь на приближенных ценностях для текущей стратегии. Как уже демонстрировалось в данной книге, улучшение стратегии может осуществляться множеством различных способов. Например, простейшим способом является использование ϵ -жадной

```

Инициализировать  $Q(s, a)$  произвольно,  $e(s) = 0$ , для всех  $s, a$ 
Повторять (для каждого эпизода):
    Инициализировать  $s, a$ 
    Повторять (для каждого шага эпизода):
        Выполнить действие  $a$ , найти  $r$  и  $s'$ 
        Выбрать  $a'$  по  $s'$ , используя стратегию, полученную из  $Q$ 
        ( $\epsilon$ -жадную)
         $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$ 
         $e(s, a) \leftarrow e(s, a) + 1$ 
        Для всех  $s, a$ 
             $Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a)$ 
             $e(s, a) \leftarrow \gamma \lambda e(s, a)$ 
             $s \leftarrow s', a \leftarrow a'$ 
    Пока  $s$  не станет завершающим состоянием

```

Рис. 7.11. Табличный метод SARSA(λ)

стратегии по отношению к текущим расчетным ценностям действий. На рис. 7.11 приведен полный алгоритм SARSA(λ) для данного случая.

Пример 7.4 (Следы на сетке). Использование следов приемлемости может существенно повысить эффективность алгоритмов управления. Причина этого иллюстрируется примером с сеткой на рис. 7.12. В левой части показан путь, который выбирается агентом в одном эпизоде и который заканчивается в точке с высоким вознаграждением, отмеченной символом *. В данном примере все начальные значения были равны 0, и все вознаграждения были нулевыми, за исключением положительного вознаграждения за точку *. Стрелки в двух других частях рисунка показывают, какие ценности действий были увеличены в результате прохождения данного пути одношаговыми методами SARSA



Рис. 7.12. Пример с сеткой, иллюстрирующий увеличение скорости изучения стратегии благодаря использованию следов приемлемости

и SARSA(λ). Одношаговый метод увеличивает ценность только последнего во всей последовательности действия, которое повлекло высокое вознаграждение, тогда как метод следов увеличивает ценности многих действий в последовательности. Степень увеличения (показана размежеванием стрелок) падает (с коэффициентом $\gamma\lambda$) по мере удаления от вознаграждения. В данном примере $\gamma = 1$, а $\lambda = 0,9$.

7.6. Метод $Q(\lambda)$

Были предложены два метода, которые объединяют в себе следы приемлемости и Q -обучение: метод $Q(\lambda)$ Уоткинса и метод $Q(\lambda)$ Пенга, в названии которых фигурируют имена их разработчиков (Watkins, Peng). Вначале рассмотрим метод $Q(\lambda)$ Уоткинса.

Напомним, что Q -обучение является методом с разделенной оценкой ценности стратегий; это означает, что изучаемая стратегия и стратегия, согласно которой осуществляются действия, не обязательно должны совпадать. В частности, Q -обучение изучает жадную стратегию, однако следует, как правило, стратегии, включающей исследовательские действия — периодически выбираются квазиоптимальные с точки зрения Q_t действия. Вследствие этого требуется особо аккуратно обращаться со следами приемлемости.

Предположим, что в момент времени t копируется пара состояния—действие s_t, a_t . Пусть в течение следующих двух шагов агент выбирает жадное действие, но на третьем шаге в момент времени $t+3$ агент выбирает исследовательское, нежадное действие. В процессе изучения ценности жадной стратегии при s_t, a_t можно использовать полученный опыт только до тех пор, пока агент следует жадной стратегии. Таким образом, в данном случае можно использовать одношаговую и двухшаговую выгоды, но трехшаговую выгоду использовать уже нельзя. n -шаговые выгоды для всех $n \geq 3$ уже не имеют отношения к жадной стратегии.

Таким образом, в отличие от методов TD(λ) и SARSA(λ), метод $Q(\lambda)$ Уоткинса не принимает в расчет все будущие ситуации вплоть до конца эпизода при выполнении дублирования. Он принимает в расчет только те будущие действия, которые предшествуют следующему исследовательскому действию. Тем не менее, за исключением данного различия, метод $Q(\lambda)$ Уоткинса очень похож на TD(λ) и SARSA(λ). Обзор будущих событий последних двух методов заканчивается в конце эпизода, тогда как обзор будущих событий метода $Q(\lambda)$ заканчивается на первом исследовательском действии или в конце эпизода, если до этого больше нет исследовательских действий. Говоря более точно, метод $Q(\lambda)$ Уоткинса

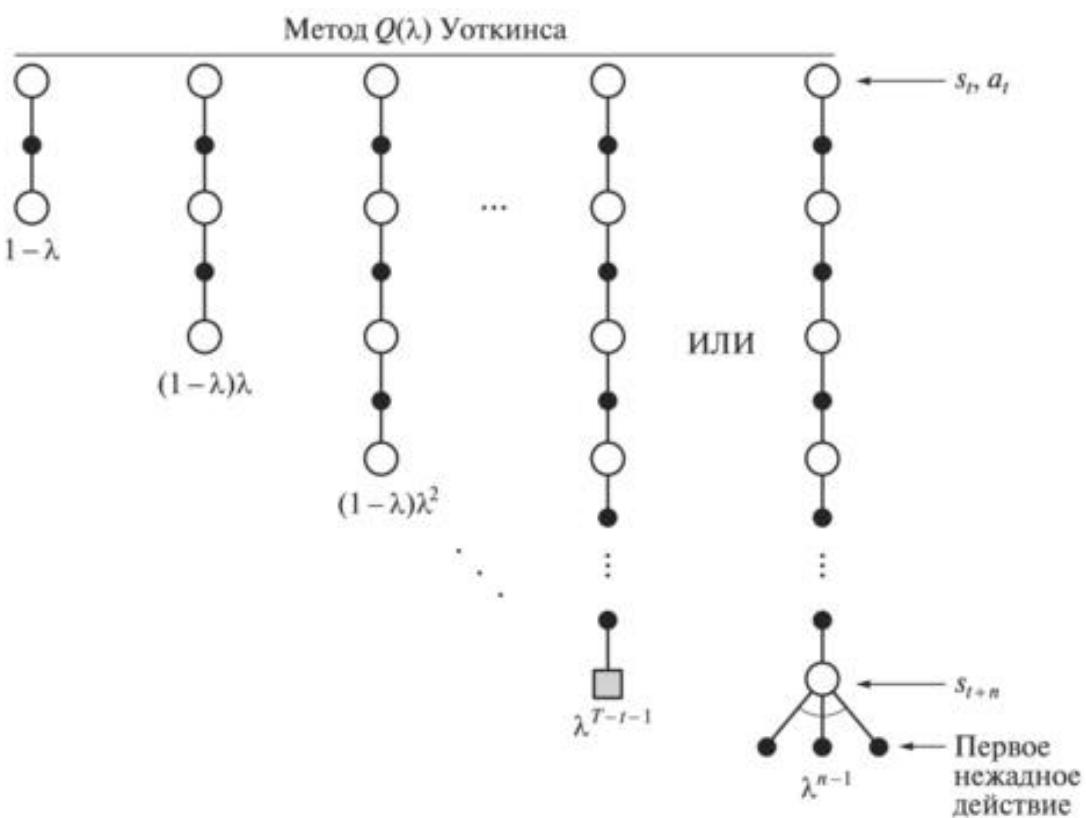


Рис. 7.13. Диаграмма предшествующих состояний для метода $Q(\lambda)$ Уоткинса. Серия компонент дублирования заканчивается либо с окончанием эпизода, либо с появлением первого нежадного действия

рассматривает одно действие *после* исследования, используя имеющиеся знания о ценностях действий. Например, предположим, что первое действие a_{t+1} является исследовательским. Метод $Q(\lambda)$ Уоткинса выполнит, тем не менее, одношаговую корректировку $Q_t(s_t, a_t)$, получив $r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a)$. В общем случае, если a_{t+n} является первым исследовательским действием, то наиболее длинное дублирование осуществляется для

$$r_{t+1} + \gamma r_{t+2} + \dots + \gamma^{n-1} r_{t+n} + \gamma^n \max_a Q_t(s_{t+n}, a),$$

причем здесь предполагается автономная корректировка. Диаграмма предшествующих состояний на рис. 7.13, которая содержит все компоненты дублирования, иллюстрирует прямой подход к методу $Q(\lambda)$ Уоткинса.

Механистический или обратный подход к методу $Q(\lambda)$ Уоткинса также достаточно прост. Следы приемлемости используются так же, как и в методе SARSA(λ), за исключением того, что они принимаются равными нулю, когда осуществляется исследовательское (нежадное)

```

Инициализировать  $Q(s, a)$  произвольно,  $e(s, a) = 0$  для всех  $s, a$ 
Повторять (для каждого эпизода):
    Инициализировать  $s, a$ 
    Повторять (для каждого шага эпизода):
        Выполнить действие  $a$ , найти  $r$  и  $s'$ 
        Выбрать  $a'$  по  $s'$ , используя стратегию, полученную из  $Q$ 
        (например,  $\varepsilon$ -жадную)
         $a^* \leftarrow \arg \max_b Q(s', b)$  (если  $a'$  дает максимум, то  $a^* \leftarrow a'$ )
         $\delta \leftarrow r + \gamma Q(s', a^*) - Q(s, a)$ 
         $e(s, a) \leftarrow e(s, a) + 1$ 
        Для всех  $s, a$ :
             $Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a)$ 
            Если  $a' = a^*$ , то  $e(s, a) \leftarrow \gamma \lambda e(s, a)$ 
            иначе  $e(s, a) \leftarrow 0$ 
             $s \leftarrow s' a \leftarrow a'$ 
    Пока  $s$  не станет завершающим состоянием

```

Рис. 7.14. Табличный вариант алгоритма $Q(\lambda)$ Уоткинса

действие. Корректировку следов проще всего представить состоящей из двух шагов. На первом шаге следы всех пар состояния—действие либо убывают с коэффициентом $\gamma\lambda$, либо, если было осуществлено исследовательское действие, приравниваются к 0. На втором шаге след, соответствующий текущему состоянию и действию, получает приращение 1. Окончательный результат имеет вид:

$$e_t(s, a) = \mathcal{I}_{ss_t} \cdot \mathcal{I}_{aa_t} + \begin{cases} \gamma \lambda e_{t-1}(s, a) & \text{при } Q_{t-1}(s_t, a_t) = \max_a Q_{t-1}(s_t, a); \\ \gamma \lambda e_{t-1}(s, a) & \text{в остальных случаях,} \end{cases}$$

где \mathcal{I}_{xy} , как и раньше — индикатор тождественности, равный 1, если $x = y$, и 0 в остальных случаях. Оставшаяся часть алгоритма определяется как

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha \delta_t e_t(s, a),$$

где

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t)$$

На рис. 7.14 приведен полный алгоритм в символьном виде.

К сожалению, прерывание следов приемлемости каждый раз, когда осуществляется исследовательское действие, ведет к потере значительной части преимуществ следов приемлемости. Если исследовательские действия осуществляются часто, что обычно имеет место на ранней стадии обучения, то лишь немногие дублирования будут длиннее одного или двух шагов, и обучение может происходить лишь немного быстрее

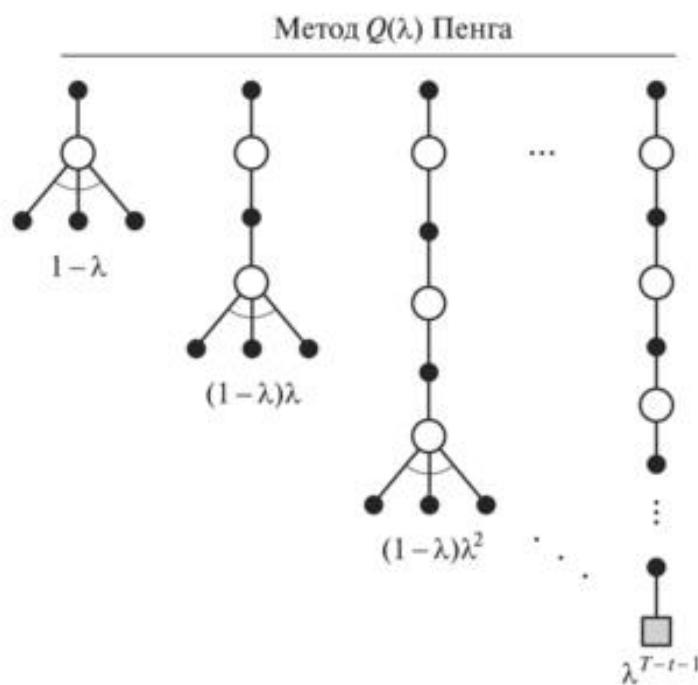


Рис. 7.15. Диаграмма предшествующих состояний метода $Q(\lambda)$ Пенга

одношагового Q -обучения. Метод $Q(\lambda)$ Пенга является очередной версией $Q(\lambda)$, призванной устранить данный недостаток. Метод $Q(\lambda)$ Пенга можно представить как гибрид методов SARSA(λ) и $Q(\lambda)$ Уоткинса.

По сути, в методе $Q(\lambda)$ Пенга используется совокупность дублирований, показанная на рис. 7.15. В отличие от Q -обучения здесь не делается различия между исследовательскими и жадными действиями. Каждое составляющее дублирование включает в себя множество исследовательских шагов, и все они, кроме последнего, претерпевают финальную максимизацию действий. Таким образом, составляющие дублирования не являются ни методами с интегрированной оценкой ценности стратегий, ни методами с разделенной оценкой ценности стратегий. Ранние переходы в каждом из них реализуют методы с интегрированной оценкой, тогда как последний (фиктивный) переход использует жадную политику. Как следствие, для фиксированной нежадной политики функция Q_t в методе $Q(\lambda)$ Пенга не сходится ни к Q^π ни к Q^* в отдельности, но сходится к некоторому гибриду данных ценностей. Тем не менее, если постепенно делать стратегию более жадной, то метод может все же сходиться к Q^* . На момент написания этих слов данный факт еще не был доказан. Тем не менее на практике метод себя показал с хорошей стороны. В большинстве процессов обучения данный метод ведет себя значительно лучше метода $Q(\lambda)$ Уоткинса и почти так же хорошо, как метод SARSA(λ).

С другой стороны, метод $Q(\lambda)$ Пенга не может быть так же просто реализован, как и метод $Q(\lambda)$ Уоткинса. Полное описание реализации метода можно найти в работах Пенга и Уильямса [Peng and Williams (1994, 1996)]. Тем не менее можно представить себе третью версию метода $Q(\lambda)$, назовем его *простым* методом $Q(\lambda)$, который аналогичен методу $Q(\lambda)$ Уоткинса, за исключением того, что следы исследовательских действий ненулевые. Такой метод мог бы обладать некоторыми из преимуществ метода $Q(\lambda)$ Пенга и при этом не требовать сложной реализации. В данный момент отсутствует опыт работы с этим методом, однако он может оказаться не таким уж и простым, как можно было бы предположить.

*7.7. Следы приемлемости для методов типа исполнитель—критик

В данном разделе описывается, как можно расширить возможности методов исполнитель—критик, представленных в разд. 6.6, используя следы приемлемости. Это делается достаточно очевидным образом. Критик в методе исполнитель—критик является, по сути, формирование V^π путем обучения с разделенной оценкой ценности стратегий. Для этого можно использовать $Q(\lambda)$ алгоритм с одним следом приемлемости для каждого состояния. В случае с исполнителем необходимо использовать след приемлемости для каждой пары состояние—действие. Таким образом, для метода исполнитель—критик необходимо иметь два набора следов приемлемости: один для каждого состояния и один для каждой пары состояние—действие.

Напомним, что одношаговый метод исполнитель—критик корректирует исполнителя следующим образом:

$$p_{t+1}(s, a) = \begin{cases} p_t(s, a) + \alpha \delta_t, & \text{если } a = a_t \text{ и } s = s_t; \\ p_t(s, a) & \text{в остальных случаях,} \end{cases}$$

где δ_t является TD-ошибкой (7.6), а $p_t(s, a)$ является предпочтением для применения действия a в момент времени t в состоянии s . Предпочтения определяют стратегию посредством, например, softmax-метода (разд. 2.3). Для использования следов приемлемости предыдущее уравнение обобщается следующим образом:

$$p_{t+1}(s, a) = p_t(s, a) + \alpha \delta_t e_t(s, a), \quad (7.11)$$

где $e_t(s, a)$ обозначает след в момент времени t для пары состояние—действие s, a . В простейшем случае, описанном выше, след может корректироваться как в методе SARSA(λ).

В разд. 6.6 рассматривался также более сложный метод исполнитель—критик, в котором применялась корректировка вида

$$p_{t+1}(s, a) = \begin{cases} p_t(s, a) + \alpha \delta_t [1 - \pi_t(s, a)], & \text{если } a = a_t \text{ и } s = s_t; \\ p_t(s, a) & \text{в остальных случаях.} \end{cases}$$

Чтобы учесть в данном соотношении следы приемлемости, можно применить корректировку (7.11) с немного измененным следом. Вместо приращения 1 каждый раз при возникновении пары состояние—действие след получает приращение $1 - \pi_t(s_t, a_t)$:

$$e_t(s, a) = \begin{cases} \gamma \lambda e_{t-1}(s, a) + 1 - \pi_t(s_t, a_t) & \text{при } s = s_t \text{ и } a = a_t; \\ \gamma \lambda e_{t-1}(s, a) & \text{в остальных случаях,} \end{cases} \quad (7.12)$$

для всех пар s, a .

7.8. Замещающие следы

В некоторых случаях можно добиться значительно лучших результатов, используя немного модифицированный вид следов, который получил наименование *замещающие следы*. Предположим, что имеет место посещение некоторого состояния, а затем это состояние посещается повторно, до того как след, соответствующий первому посещению, полностью обратился в нуль. При накапливающихся следах (7.5) повторное посещение станет причиной приращения следа, значение которого, таким образом, будет больше 1, тогда как в случае с замещающими следами, значение следа не превышает 1. На рис. 7.16 сравниваются два типа следов.

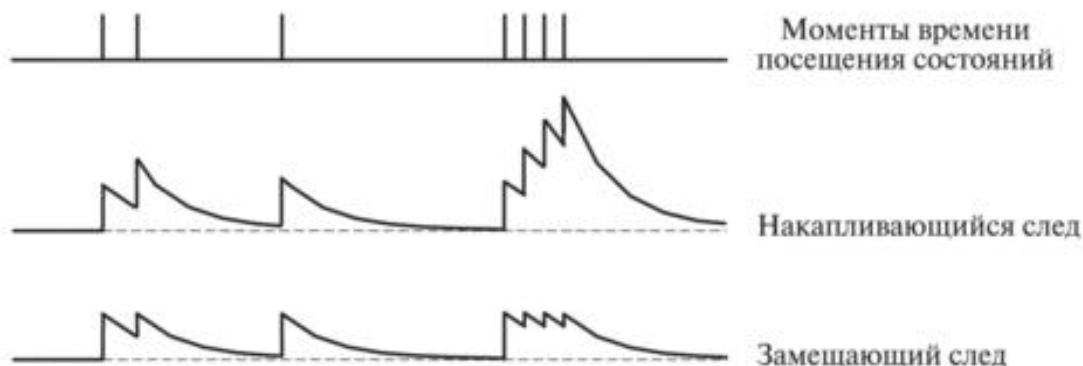


Рис. 7.16. Замещающие и накапливающиеся следы

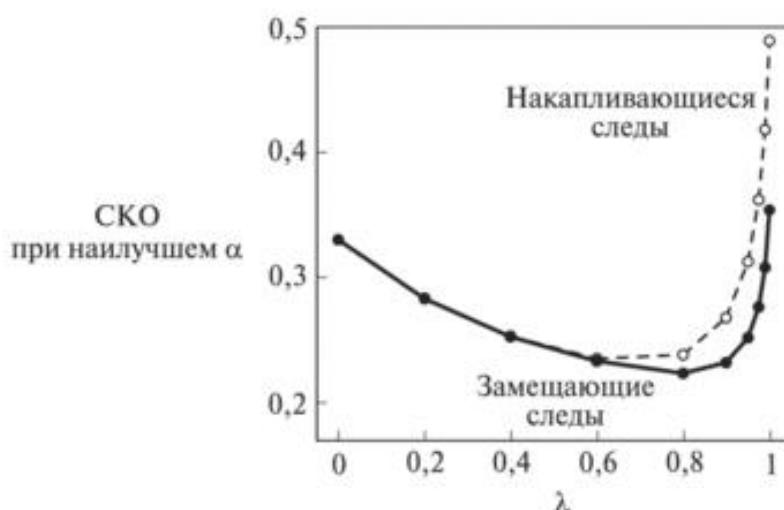


Рис. 7.17. Ошибка как функция от λ в задаче прогнозирования о случайных блужданиях, использующей 19 состояний

Формально, замещающий след для дискретного состояния s определяется следующим образом:

$$e_t(s, a) = \begin{cases} \gamma \lambda e_{t-1}(s) & \text{при } s \neq s_t; \\ 1 & \text{при } s = s_t. \end{cases} \quad (7.13)$$

Алгоритмы прогнозирования или управления, использующие замещающие следы, обычно называются методами *замещения следов*. Несмотря на то что замещающие следы лишь ненамного отличаются от накапливающихся следов, они могут значительно поднять уровень обучения. На рис. 7.17 показаны результаты действия обычного метода TD(λ) и метода TD(λ) замещения следов в задаче прогнозирования для случайных блужданий, содержащей 19 состояний. Другие примеры для более общего случая представлены на рис. 8.15 в следующей главе.

Пример 7.5. На рис. 7.18 показан пример задачи, сложной для метода управления, использующего накапливающиеся следы приемлемости. Все вознаграждения равны нулю, за исключением вступления в заклю-

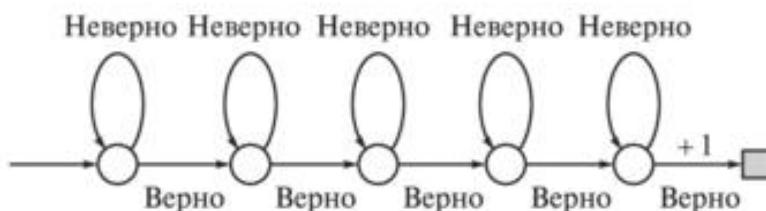


Рис. 7.18. Простая задача, вызывающая затруднения у метода управления, использующего накапливающиеся следы

чительное состояние, когда вознаграждение равно +1. В каждом состоянии при выборе «верного» (*right*) действия агент становится на один шаг ближе к заключительному вознаграждению, тогда как «неверное» (*wrong*) действие (вверху) оставляет агента в том же состоянии. Вся последовательность состояний достаточно длинная для того, чтобы можно было использовать длинные шаги для наиболее быстрого обучения. Тем не менее при использовании длинных следов возникают проблемы. Предположим, что в первом эпизоде в некотором состоянии s агент случайно несколько раз выбирает «неверное» действие, прежде чем выбрать «верное». После перехода в следующее состояние след $e(s, \text{wrong})$, скорее всего, будет больше следа $e(s, \text{right})$. Верное действие было последним, но неверное действие выбиралось большее количество раз. Когда, в конце концов, агент получает вознаграждение, ценность неверного действия возрастает больше, чем ценность верного действия. В следующем эпизоде агент с еще большей вероятностью будет несколько раз выбирать неверное действие, прежде чем выбрать верное, таким образом, еще увеличивая след неверного действия. В конце концов, все это будет исправлено, но обучение значительно замедлится. С замещающими следами такой проблемы никогда не возникает. Вне зависимости от того, сколько раз выбиралось неверное действие, его след приемлемости будет всегда меньше следа верного действия, после того, верное действие будет выбрано.

Существует интересная взаимосвязь между методами замещения следов и методами Монте-Карло в случае с неприведенной задачей. Так же как обычный метод TD(1) связан с алгоритмом Монте-Карло, метод TD(1) замещения шагов связан с МК-алгоритмом первого посещения. В частности, автономный вариант метода TD(1)-замещения шагов формально идентичен МК-алгоритму первого посещения (см. работу [Singh and Sutton, (1996)]). Таким образом можно распространить данные методы и результаты на случай с неприведенной задачей и можно ли вообще это сделать, пока неизвестно.

Существует несколько возможных способов обобщения замещающих следов приемлемости для использования их в методах управления. Очевидно, что, когда происходит повторное посещение состояния и выбирается новое действие, след данного действия необходимо положить равным 1. Но как насчет остальных действий данного состояния? Согласно подходу, рекомендованному в работе [Singh and Sutton (1996)], следы всех остальных действий данного состояния полагаются равными 0. В таком случае правило корректировки следов пары состояния—

действие вместо (7.10) принимает следующий вид:

$$e_t(s, a) = \begin{cases} 1 & \text{при } s = s_t \text{ и } a = a_t; \\ 0 & \text{при } s = s_t \text{ и } a \neq a_t; \\ \gamma \lambda e_{t-1}(s, a) & \text{при } s \neq s_t \end{cases} \quad \text{при всех } s, a. \quad (7.14)$$

Отметим, что данный вариант замещающих следов работает даже лучше, чем первоначальные замещающие следы в задаче из рассмотренного примера. Как только выбирается правильное действие, неправильное действие остается вообще без следа. Результаты, показанные на рис. 8.15, были получены при использовании данного типа замещающих следов.

Упражнение 7.6. Предположим, что в примере 7.5 в состоянии s неправильное действие было выбрано дважды, прежде чем было выбрано правильное действие. Если применяются накапливающиеся следы, то насколько большим должен быть параметр следа λ , чтобы по окончании эпизода неправильное действие имело след, больший, чем у правильного действия?

Упражнение 7.7 [Программирование]. Напишите программу для решения задачи 7.5 и на ее основе сравните версии метода SARSA(λ) с накоплением следов и с замещением следов для $\lambda = 0,9$ и различных значений α . Можете ли вы на основе данного опыта продемонстрировать заявленные преимущества замещающих следов?

Упражнение 7.8*. Нарисуйте диаграмму предшествующих состояний для метода SARSA(λ) с замещающими следами.

7.9. Проблемы реализации

Может показаться, что методы, использующие следы приемлемости, намного сложнее одношаговых методов. При реализации таких методов для каждого состояния (пары состояние—действие) требуется проводить коррекцию не только его ценности, но и его следов приемлемости на каждом шаге. Реализация подобного алгоритма не является проблемой для параллельных компьютеров с одним потоком команд и множеством потоков данных (SIMD-архитектура) или при реализации данного алгоритма с использованием адекватных нейросетевых структур, но для обычных компьютеров с последовательной архитектурой это может стать слишком сложной задачей. К счастью, для наиболее распространенных значений λ и γ следы приемлемости практически всех состояний почти всегда близки к нулю; только те состояния, которые недавно

посещались, будут иметь следы, значительно большие нуля. Необходимо проводить корректировку только этих нескольких состояний, так как корректировка остальных состояний существенного значения не имеет.

На практике при работе на обычных компьютерах отслеживаются и корректируются лишь несколько состояний с ненулевыми следами. При использовании данного приема объемы вычислений при применении следов обычно лишь в несколько раз превышают объемы вычислений в случае с одношаговым методом. Точная структура вычислений зависит, конечно, от параметров λ и γ и от объема остальных вычислений. В работе [Cichosz (1995)] продемонстрирована методика вычислений, которая снижала их объем до некоторого постоянного значения, не зависящего от параметров λ и γ . В заключение необходимо отметить, что табличный случай является, в некотором смысле, наихудшим с точки зрения сложности вычислений. При использовании аппроксимации функции (гл. 8), преимущество малого объема вычислений, в случае неиспользования следов, как правило, снижается. Например, если применяются искусственные нейронные сети и обратное распространение ошибки обучения, то использование следов потребует, как правило, увеличения памяти и объема вычислений лишь в два раза.

Упражнение 7.9. Реализуйте в символьном виде алгоритм TD(λ), который корректирует ценности только тех состояний, следы которых больше некоторой небольшой положительной константы.

*7.10. Переменный параметр λ

λ -выгоду можно значительно обобщить, если позволить параметру λ изменяться от шага к шагу, т. е. если переопределить корректировку следов следующим образом:

$$e_t(s, a) = \begin{cases} \gamma \lambda_t e_{t-1}(s) & \text{при } s \neq s_t, \\ \gamma \lambda_t e_{t-1}(s) & \text{при } s = s_t, \end{cases}$$

где через λ_t обозначено значение параметра λ в момент времени t . Это дополнительная тема, рассчитанная на перспективу, поскольку введенное выше обобщение никогда ранее не использовалось в практических задачах, но оно интересно с теоретической точки зрения, и, возможно, еще окажется полезным. Например, согласно одной из идей, параметр λ можно представить в виде функции состояния: $\lambda_t = \lambda(s_t)$. Если предполагается, что оценка ценности состояния известна с высокой точностью, то имеет смысл полностью использовать ее, не принимая во внимание состояния и вознаграждения, следующие за данным состоянием. Это

соответствует тому, что, как только достигается данное состояние, все следы отсекаются, т. е. для определенного состояния параметр λ полагается равным нулю или очень маленькой величине. Аналогично, состояниям, оценки ценности которых очень неточны, например, из-за недостоверной ценности состояния, может быть поставлено в соответствие значение λ_s , близкое к 1. Вследствие этого, их оценки ценности не оказывают большого влияния на корректировку. Они «пропускаются» до тех пор, пока не появится состояние, которое известно в большей степени. Некоторые из этих идей формально использовались в работе [Sutton and Singh (1994)].

Приведенное выше соотношение для следов приемлемости реализует обратный подход к определению переменной λ_s . Соответствующий прямой подход дает более общее определение λ -выгоды:

$$R_t^\lambda = \sum_{n=1}^{\infty} R_t^{(n)} (1 - \lambda_{t+n}) \prod_{i=t+1}^{t+n-1} \lambda_i = \sum_{k=t+1}^{T-1} R_t^{(k-t)} (1 - \lambda_k) \prod_{i=t+1}^{k-1} \lambda_i + R_t \prod_{i=t+1}^{T-1} \lambda_i.$$

Упражнение 7.10*. Докажите, что прямой и обратный подходы к автономному методу TD(λ) остаются эквивалентными при их новом определении с переменным параметром λ , рассмотренным в данном разделе. Следуйте примеру доказательства из разд. 7.4.

7.11. Итоги

Следы приемлемости вместе с TD-ошибками дают эффективный инкрементный способ перемещения и выбора между методом Монте-Карло и TD-методом. Следы могут использоваться и без TD-ошибок для достижения того же эффекта, но это часто бывает затруднительным. Такой метод, как TD(λ), позволяет добиться этого на основе частичного опыта с небольшими затратами памяти и с незначительным разбросом в прогнозировании.

Как отмечалось в гл. 5, методы Монте-Карло могут иметь преимущества в немарковских задачах, так как они не самонастраиваются. Поскольку следы приемлемости делают TD-методы похожими на методы Монте-Карло, они также могут иметь преимущества в таких задачах. Если необходимо использовать TD-методы вследствие других имеющихся у них преимуществ, но задача хотя бы отчасти немарковская, то рекомендуется использовать метод со следами приемлемости. Следы приемлемости являются широко распространенным методом решения как проблем отложенных вознаграждений, так и немарковских задач.

Подбирая различные значения λ , можно перемещать методы со следами приемлемости между Монте-Карло и одношаговыми TD-методами в пространстве методов. В каком же месте их необходимо поместить? Пока на этот вопрос не существует приемлемого теоретического ответа, тогда как эмпирический ответ можно сформулировать достаточно точно. Опыт показывает, что в задачах с множеством шагов в каждом эпизоде или с множеством шагов до момента полуспада ценностей значительно выгоднее использовать следы приемлемости (см. рис. 8.15). С другой стороны, если следы настолько длинные, что в результате образуется метод Монте-Карло в чистом виде или близкий к нему метод, результаты резко ухудшаются. Оптимальным выбором является некоторая промежуточная композиция данных методов. Следы приемлемости призваны приблизить рассматриваемый метод к методу Монте-Карло, но не слишком близко. Вероятно, в будущем появится возможность более гибкого компромисса между методами Монте-Карло и TD-методами, с использованием переменного параметра λ , но на сегодняшний день пока не ясно, каким образом это можно реализовать с достаточной степенью надежности и полезности.

Методы, использующие следы приемлемости, требуют большего количества вычислений, чем одношаговые методы, но взамен они предлагают значительно более быстрое обучение, особенно когда вознаграждения отложены на большое число шагов. Таким образом, часто имеет смысл использовать следы приемлемости, при недостаточном количестве данных и при невозможности их повторной обработки, что часто наблюдается в оперативных задачах. С другой стороны, в автономных задачах, в которых результаты можно сгенерировать сравнительно легко, например, с помощью несложных моделирующих программ, использовать следы приемлемости обычно невыгодно. В таких случаях целью является не получение как можно большей выгоды от имеющегося небольшого количества результатов, но быстрое формирование как можно большего количества результатов. В таких случаях выигрыш в скорости получения данных, который дают следы приемлемости, обычно не стоит затрат на расчеты, и одношаговые методы оказываются предпочтительнее.

7.12. Библиографические и исторические справки

7.1–2. Передовой подход к следам приемлемости на основе n -шаговых выгод и λ -выгод предложен в диссертации Уоткинса (1989), где также впер-

вые обсуждалось свойство уменьшения ошибки n -шаговой выгода. Представление в данной книге основано на немного модифицированном подходе из статьи [Jaakkola, Jordan, and Singh (1994)]. Результаты в примерах со случайными блужданиями, приведенные в тексте, основывались на работе Саттона (1988) и совместной работе Саттона и [Singh] (1994). Диаграммы предшествующих состояний для описания представленных в данной главе и других алгоритмов применялись впервые, так же как и впервые применялись термины «прямой подход» и «обратный подход».

В статье [Dayan] (1992) доказана сходимость метода $TD(\lambda)$ в среднем, а сходимость с вероятностью 1 была доказана во многих работах: Пенг (1993), [Dayan and Sejnowski (1994)] и [Tsitsiklis (1994)]. В статье [Jaakkola, Jordan, and Singh (1994)] впервые доказана сходимость алгоритма $TD(\lambda)$ при независимой (on-line) корректировке. В работе [Gurvits, Lin, and Hanson (1994)] доказана сходимость более общего класса методов со следами приемлемости.

7.3. Идея о том, что стимул к возникновению необходимых для обучения последствий в нервной системе, далеко не нова. Еще в таких старых работах, как Павлов (1927) и [Hull] (1943, 1952), использованы эти идеи. Однако следы стимулов в таких теориях больше похожи на отображения переходных состояний, чем на введенные нами следы приемлемости: они могут соответствовать действиям, тогда как след приемлемости может использоваться только для присвоения коэффициента доверия. Идея следов стимулов, специально предназначенных для присваивания коэффициента доверия, очевидно, впервые появилась в докладе [Kloph] (1972), где была выдвинута гипотеза, что при определенных условиях нейронные синапсы будут «приемлемы» из-за последующих модификаций, которые позднее в процессе закрепления будут внесены в нейрон. Применение следов приемлемости в данной книге основывалось на работе [Kloph] (см. также Саттон, 1978а, 1978б, 1978с; Барто и Саттон, 1981а, 1981б; Саттон и Барто, 1981а; Барто, Саттон и Андерсон, 1983; Саттон, 1984). $TD(\lambda)$ алгоритм был предложен в работе Саттона (1988).

7.4. Эквивалентность прямого и обратного подходов и связь с методами Монте-Карло были доказаны в работе Саттона (1989) для неприведенной эпизодической задачи, затем распространены Уоткинсом на общий случай. Использованная в примере 7.5 идея предложена впервые.

7.5. Метод SARSA(λ) был впервые рассмотрен как метод управления в работах [Rummery and Niranjan (1994)] и [Rummery (1995)].

7.6. Метод $Q(\lambda)$ Уоткинса предложен в диссертации Уоткинса (1989). Метод $Q(\lambda)$ Пенга предложен в работах (Пенг, 1993; Пенг и Уильямс, 1994, 1996). В диссертации [Rummery] (1995) проведено обширное сравнительное изучения данных алгоритмов.

Для $0 < \lambda < 1$ сходимость ни одного из методов управления доказана не была.

- 7.7. Методы исполнитель—критик были среди первых методов, в которых использовались следы приемлемости (см. работы Барто, Саттон и Андерсон, 1983; Саттон, 1984). Рассмотренный в данной главе специальный алгоритм никогда еще не применялся на практике.
- 7.8. Замещающие следы предложены в статье [Singh] и Саттон (1996). Показанные на рис. 7.17 результаты взяты из статей данных авторов. Задача на рис. 7.18 была использована в диссертации Саттона (1984) для демонстрации слабости накапливающих шагов. Взаимосвязь обоих типов методов со специальными методами Монте-Карло была исследована в статье [Singh] и Саттон (1996).
- 7.9–10. Идеи данных двух разделов в целом были известны уже давно, но, за исключением цитируемых здесь источников, данный текст, возможно, является первым подробным описанием этих идей. Вероятно, впервые изменяемый параметр λ рассматривался в диссертации Уоткинса (1989), где отмечается, что прекращение последовательности дублирований (рис. 7.13) в алгоритме $Q(\lambda)$, когда выбирается нежадное действие, можно реализовать, временно приравняв λ к 0.
-

8

Обобщение и аппроксимация функций

До сих пор предполагалось, что оценки функций ценности представлены в виде таблицы с одной позицией для каждого состояния или каждой пары состояния—действие. Это особенно ясный и наглядный случай, но, естественно, к нему относятся только задачи с небольшим количеством состояний и действий. Проблема заключается не только в объемах памяти, требующейся для больших таблиц, но и в том количестве времени и данных, которое необходимы для точного заполнения этих таблиц. Другими словами, ключевым вопросом является вопрос *обобщения*. Каким образом можно обобщить опытные данные, имеющиеся для ограниченного подмножества в пространстве состояний, чтобы получить хорошую аппроксимацию в случае более обширного подмножества?

Это серьезная проблема. Во многих задачах, к которым можно было бы применить обучение с подкреплением, большинство встречающихся состояний никогда не будут повторяться в точности. Почти всегда в этих случаях пространства состояний или действий будут включать в себя непрерывные переменные или сложные ощущения, такие как визуальное изображение. Единственным способом построить обучение в таких случаях является обобщение опыта предыдущих состояний и его распространение на последующие состояния, которые еще никогда не происходили.

К счастью, обобщение на наборах примеров исследуется уже достаточно долго и поэтому нет необходимости изобретать полностью оригинальные методы для их использования в обучении с подкреплением. По большому счету, необходимо лишь комбинировать методы обучения с подкреплением с существующими методами обобщения. Рассматриваемый тип обобщения часто называется *аппроксимацией функций*, так как в нем берутся примеры из искомой функции (т. е. функции ценности) и делаются попытки их обобщения для построения аппроксимации всей функции. Аппроксимация функций является примером *обучения с учителем*, одной из главных тем при изучении машинного обучения, искусственных нейронных сетей, распознавания образов и статистической аппроксимации кривых. В принципе, любые из применяемых в данных областях методов могут быть использованы в обучении с подкреплением, как это описано в данной главе.

8.1. Прогнозирование ценности при помощи аппроксимации функции

Как обычно, вначале рассматривается задача прогнозирования для оценки функции ценности состояния V^π на основе накопленного при следовании стратегии π опыта. Нововведением данной главы является то, что аппроксимация функции ценности в момент времени t , V_t , представляется не в виде таблицы, но в форме параметризованного функционала с вектором параметров θ_t . Это означает, что функция ценности V_t полностью зависит от θ_t , изменяясь от шага к шагу только при изменении θ_t . Например, V_t может быть функцией, рассчитанной искусственной нейронной сетью, с вектором θ_t промежуточных весовых коэффициентов. Подбором весов с помощью сети можно реализовать любую из множества различных функций V_t . Кроме того, V_t может быть функцией, рассчитанной с помощью дерева решений, где вектор θ_t содержит все параметры, определяющие разветвления и ценности листьев дерева. Обычно число параметров (число компонент вектора θ_t) значительно меньше числа состояний, а изменение одного параметра влечет за собой изменение оценок ценностей многих состояний. Вследствие этого, когда выполняется дублирование отдельно взятого состояния, изменение этого состояния отражается на ценностях многих других состояний.

Все методы прогнозирования, упомянутые в данной книге, рассматривались как виды дублирования, т. е. как корректировки оценок функции ценности, которые меняют ее значение в конкретных состояниях, основываясь на «дублированной ценности» данного состояния. Обозначим отдельно взятое дублирование как $s \mapsto v$, где s — это дублированное состояние, а v — дублированная ценность, или цель, к которой стремится расчетная ценность состояния s . Например, TD-дублирование для прогнозирования ценности имеет вид

$$s \mapsto E_\pi\{r_{t+1} + \gamma V_t(s_{t+1}) | s_t = s\},$$

дублирование Монте-Карло — вид $s_t \mapsto R_t$, а TD(0) дублирование — вид

$$s_t \mapsto r_{t+1} + \gamma V_t(s_{t+1}),$$

и общее TD(λ)-дублирование представляется в виде $s_t \rightarrow R_t^\lambda$. В случае с ДП дублируется произвольное состояние s , тогда как в остальных случаях дублируется состояние s_t , которому соответствует имеющийся (возможно, смоделированный) опыт.

Обычно каждое дублирование интерпретируют как действие, определяющее пример требуемого вход-выходного поведения оценки функ-

ции ценности. В известном смысле, дублирование $s \rightarrow v$ означает, что оценка ценности состояния s должна быть ближе к v . До сих пор фактическая корректировка, выполняющая дублирование, была тривиальной: элемент таблицы, соответствующий расчетной ценности состояния s , просто смещался на некоторое расстояние по направлению к v . Теперь же для осуществления дублирования допускается применение методов аппроксимации функции произвольной сложности. Стандартными входящими данными для таких методов являются примеры требуемого вход-выходного поведения аппроксимируемой функции. При использовании данных методов для прогнозирования ценности в качестве обучающего примера на вход просто подается $s \mapsto v$ для каждого дублирования. Полученная с их помощью аппроксимированная функция считается оценкой функции ценности.

Рассматривая каждое дублирование как обычный обучающий пример, мы получаем возможность использовать любой из множества существующих методов аппроксимации функций для прогнозирования ценности. Фактически для обучения с учителем на примерах можно использовать любой метод, включая искусственные нейронные сети, деревья решений и различные виды многомерных регрессий. Тем не менее не все методы аппроксимации функции одинаково хорошо подходят для решения задачи обучения с подкреплением. Наиболее развитые нейросетевые модели и статистические методы предполагают наличие статистического обучающего набора, на основе которого пробуются различные пути решения. В обучении с подкреплением, однако, важно, чтобы обучение проходило оперативно, в процессе взаимодействия с окружающей средой или моделью окружающей среды. Это требует наличия методов, которые обеспечивают эффективное обучение на основе постепенно приобретаемой информации. Кроме того, в общем случае обучение с подкреплением требует наличия методов аппроксимации функций, способных оперировать нестационарной целевой функцией (целевая функция, которая меняется во времени). Например, в ОИС методах управления часто отыскивается функция Q^π , в то время как сама стратегия π меняется. Даже в случае, когда стратегия остается неизменной, искомые значения обучающих примеров будут нестационарными, если они сгенерированы самонастраивающимися методами (например, ДП и TD-методами). Методы, которые не могут оперировать такой нестационарностью, хуже подходят к обучению с подкреплением.

Какие критерии качества соответствуют методам аппроксимации функции ценности? Большинство методов обучения с учителем построены на минимизации среднеквадратической ошибки (СКО; MSE — Mean-

Squared Error) при некотором распределении P входных данных. В нашей задаче прогнозирования ценности входными данными являются состояния, а целевой функцией является истинная функция ценности V^π . Таким образом, используя параметр $\vec{\theta}_t$, СКО для аппроксимации V_t имеет вид

$$\text{MSE}(\vec{\theta}_t) = \sum_{s \in \mathcal{S}} P(s) [V^\pi(s) - V_t(s)]^2, \quad (8.1)$$

где P — распределение весов ошибок различных состояний. Данное распределение имеет большое значение, так как обычно невозможно добиться нулевой ошибки во всех состояниях. В конце концов, в общем случае количество состояний много больше количества компонент вектора θ_t . Таким образом, гибкость аппроксиматора функции сильно ограничена. Улучшение аппроксимации в одних состояниях в общем случае может быть достигнуто только за счет ухудшения ее в других. Распределение P определяет, каким образом осуществить данный компромисс.

Кроме того, именно из распределения P обычно выбираются состояния в обучающих примерах, и, таким образом, оно является распределением состояний, к которым применяется дублирование. Если необходимо минимизировать ошибку в некотором распределении состояний, то имеет смысл обучать аппроксиматор функции на примерах, взятых из того же распределения. Если же необходимо иметь одинаковый уровень ошибки на протяжении всего набора состояний, то имеет смысл осуществлять обучение с дублированиями, распределенными равномерно по всему набору состояний так же, как и при полной прогонке в некоторых методах ДП. С этого момента будем считать, что распределение состояний, в которых осуществлялось дублирование, и распределение весов ошибок P — это одно и то же распределение.

Особенно интересно распределение, описывающее частоту возникновения состояния, когда агент взаимодействует с окружающей средой и осуществляет действия в соответствии со стратегией π , функция ценности которой является объектом аппроксимации. Оно называется *распределением с интегрированной оценкой ценности стратегий*, отчасти потому, что является распределением дублирований в методах с интегрированной оценкой ценности стратегий управления. Минимизация ошибки в распределении с интегрированной оценкой ценности стратегий позволяет сконцентрировать ресурсы метода аппроксимации функции на тех состояниях, которые действительно возникают при следовании заданной стратегии, и пропускать состояния, которые никогда не возникают. Кроме того, для распределения с интегрированной оценкой легче всего получить обучающие примеры, используя Монте-Карло или

TD-методы. Данные методы генерируют дублирования на основе выборочного опыта, используя стратегию π . Вследствие того что дублирование формируется для каждого состояния, опыт встречи с которым уже имеется, доступные обучающие примеры распределены в соответствии с распределением с интегрированной оценкой. Позднее будет показано, что при распределении с интегрированной оценкой имеют место более сильные результаты сходимости, чем при других распределениях.

Нет полной ясности в том, что необходимо добиваться минимизации СКО. По сути, задача прогнозирования ценности состоит не в этом, так как конечной целью является использование прогнозирования как вспомогательного средства отыскания лучшей стратегии. Наилучшие прогнозы для данной цели могут не соответствовать наилучшим прогнозам для минимизации СКО. Тем не менее до сих пор не ясно, что может являться более полезной альтернативной целью для прогнозирования ценности. В данный момент основное внимание будет уделяться СКО.

Идеальной целью с точки зрения СКО было бы отыскание *глобального оптимума*, т. е. вектора параметров $\vec{\theta}^*$, для которого неравенство $MSE(\vec{\theta}^*) \leq MSE(\vec{\theta})$ выполняется для всех возможных векторов $\vec{\theta}$. Иногда этой цели могут достигать простые, например линейные, аппроксиматоры функций, но это почти невозможно в случае со сложными аппроксиматорами, такими как искусственные нейронные сети и деревья решений. Целью сложного аппроксиматора функции может оказаться сходимость к *локальному оптимуму*, т. е. вектору параметров $\vec{\theta}^*$, для которого неравенство $MSE(\vec{\theta}^*) \leq MSE(\vec{\theta})$ выполняется для всех векторов $\vec{\theta}$ в некоторой окрестности вектора $\vec{\theta}^*$. Несмотря на то что такое свойство немного обнадеживает, в большинстве случаев это самое большое, что может дать нелинейный аппроксиматор функции. Во многих случаях, представляющих интерес с точки зрения обучения с подкреплением, сходимость к оптимуму или даже истинная сходимость не наблюдается. Тем не менее сходимость по СКО в небольшой окрестности оптимума может быть достигнута с помощью некоторых методов. Другие методы могут в действительности расходиться, а их СКО в пределе может уходить на бесконечность.

В данном разделе мы рассмотрели в общих чертах возможности сочетания различных методов обучения с подкреплением, применяющихся для прогнозирования ценностей, с различными методами аппроксимации функций с применением дублирования первых для генерирования обучающих примеров для вторых. Кроме того, был рассмотрен ряд среднеквадратических показателей качества, которыми руководствуются данные методы. Количество всевозможных методов слишком велико,

чтобы охватить их все, но в любом случае про большую часть из них известно слишком мало, чтобы можно было надежно оценить эти методы или дать какие-либо рекомендации. Таким образом, здесь рассматриваются только некоторые возможные варианты. В конце данной главы основное внимание будет сосредоточено на методах аппроксимации функций, основанных на градиентных принципах, и в особенности на линейных методах наискорейшего спуска. Основное внимание уделяется данным методам отчасти потому, что они считаются наиболее перспективными, и потому, что в них проявляются основные теоретические аспекты. Кроме того, эти методы просты, а объем данной книги ограничен. Если бы аппроксимации функций была посвящена еще одна глава, были бы также рассмотрены, по крайней мере, методы, основывающиеся на памяти, а также методы деревьев решений.

8.2. Методы наискорейшего спуска

Рассмотрим подробно класс методов аппроксимации функций, в основе которого лежит наискорейший (градиентный) спуск. Среди всех методов аппроксимации функций методы наискорейшего спуска являются одними из наиболее часто применяемых. Они хорошо подходят для решения задачи обучения с подкреплением.

В методах наискорейшего спуска вектор параметров является вектором-столбцом с действительными компонентами,

$$\vec{\theta}_t = (\theta_t(1), \theta_t(2), \dots, \theta_t(n))^T$$

(Т здесь обозначает транспонирование), а $V_t(s)$ является гладкой дифференцируемой функцией от $\vec{\theta}_t$ для всех состояний $s \in \mathcal{S}$. С этого момента предполагается, что на каждом временном шаге t имеет место новый пример $s_t \mapsto V^\pi(s_t)$. Данные состояния могут быть последовательными состояниями взаимодействия с окружающей средой, однако мы не требуем выполнения этого условия. Даже несмотря на то, что заданы правильные и точные ценности $V^\pi(s_t)$ для каждого состояния s_t , задача все еще остается сложной, так как имеющийся аппроксиматор функции имеет ограниченные ресурсы и, таким образом, ограниченную точность получаемого решения. В частности, вектор $\vec{\theta}_t$ в большинстве случаев не может точно описать все состояния или даже все примеры. Кроме того, необходимо обобщить все остальные состояния, которые не встретились в примерах.

Предполагается, что состояния в примерах возникают согласно одному и тому же распределению P , применительно к которому миними-

зируется СКО в соответствии с соотношением (8.1). Хорошей стратегией в данном случае является минимизация ошибки на предъявленных примерах. Методы наискорейшего спуска добиваются этого, изменяя после каждого примера параметры вектора на небольшую величину в сторону наибольшего уменьшения ошибки в данном примере:

$$\begin{aligned}\vec{\theta}_{t+1} &= \vec{\theta}_t - \frac{1}{2} \alpha \nabla_{\vec{\theta}_t} [V^\pi(s_t) - V_t(s_t)]^2 = \\ &= \vec{\theta}_t + \alpha [V^\pi(s_t) - V_t(s_t)] \nabla_{\vec{\theta}_t} V_t(s_t),\end{aligned}\quad (8.2)$$

где α — положительный параметр, определяющий размер шага, а $\nabla_{\vec{\theta}_t} f(\vec{\theta}_t)$ для любой функции f обозначает вектор частных производных

$$\left(\frac{\partial f(\vec{\theta}_t)}{\partial \theta_t(1)}, \frac{\partial f(\vec{\theta}_t)}{\partial \theta_t(2)}, \dots, \frac{\partial f(\vec{\theta}_t)}{\partial \theta_t(2)} \right)^T.$$

Данный вектор частных производных является градиентом функции f по вектору $\vec{\theta}_t$. Методы данного типа называются *методами наискорейшего спуска*, так как полный шаг в $\vec{\theta}_t$ пропорционален отрицательному градиенту квадратичной ошибки в примере. Это направление, в котором ошибка убывает наиболее быстро.

Может показаться неясным, почему в направлении градиента делаются только небольшие шаги. Нельзя ли двигаться в данном направлении как можно дальше и полностью избавиться от ошибки в примере? Так можно поступить во многих случаях, но обычно это нежелательный сценарий. Помните, что основной целью является не поиск функции ценности с нулевой ошибкой во всех состояниях, но поиск аппроксимации, которая сбалансирует ошибку в различных состояниях. Если полностью избавиться от ошибки в примере на одном шаге, то такой баланс не будет найден. Фактически, сходимость методов наискорейшего спуска подразумевает, что размер шага убывает со временем. Если он убывает, удовлетворяя нормальным условиям стохастической аппроксимации (2.8), то метод наискорейшего спуска (8.2) гарантированно сходится к локальному оптимуму.

Теперь вернемся к случаю, когда заданная выходная величина $v_t t$ -го обучающего примера $s_t \rightarrow v_t$ не является истинным значением $V^\pi(s_t)$, но некоторой его аппроксимацией. Например, v_t может быть вариантом $V^\pi(s_t)$ с шумовыми искажениями, или она может быть одним из скопированных значений, упомянутых в предыдущем разделе. В таких случаях невозможно осуществить в точности корректировку (8.2), так как значение $V^\pi(s_t)$ неизвестно, тем не менее можно его аппроксимировать, заменив на v_t . Это приводит к общему методу наискорейшего спуска для

прогнозирования ценности состояния:

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha [v_t - V_t(s_t)] \nabla_{\vec{\theta}_t} V_t(s_t). \quad (8.3)$$

Если v_t является *несмешенной* оценкой, т. е. если $E\{s_t\} = V^\pi(s_t)$, для каждого t , то вектор $\vec{\theta}_t$ гарантированно сходится к локальному оптимуму при обычных условиях стохастической аппроксимации (2.8) и при убывающем размере шага α .

Предположим, например, что действия в данном примере сформированы на основе взаимодействия (или смоделированного взаимодействия) с окружающей средой, при следовании стратегии π . Обозначим R_t выгода каждого состояния s_t . Так как истинной ценностью состояния является ожидаемая выгода данного состояния, цель Монте-Карло $v_t = R_t$ является по определению несмешенной оценкой $V^\pi(s_t)$. При таком выборе общий метод наискорейшего спуска (8.3) сходится к локальной оптимальной аппроксимации $V^\pi(s_t)$. Таким образом, вариант метода Монте-Карло для прогнозирования ценности состояния, основанный на наискорейшем спуске, гарантированно находит локально оптимальное решение.

Аналогичным образом можно использовать n -шаговые TD-выгоды и их средние значения для v_t . Например, вариант метода TD(λ), основанный на наискорейшем спуске, использует λ -выгоду $v_t = R_t^\lambda$ как аппроксимацию $V^\pi(s_t)$, результатом чего является корректировка, соответствующая прямому представлению:

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha [R_t^\lambda - V_t(s_t)] \nabla_{\vec{\theta}_t} V_t(s_t). \quad (8.4)$$

Однако при $\lambda < 1$ величина R_t^λ не является несмешенной оценкой $V^\pi(s_t)$, и, таким образом, данный метод не сходится к локальному оптимуму. Ситуация аналогична, когда используются TD-цели, такие как

$$v_t = E_\pi\{r_{t+1} + V_t(s_{t+1})|s_t\}.$$

Тем не менее такие самонастраивающиеся методы могут работать достаточно эффективно и, кроме того, могут позволять решать задачи для некоторых важных частных случаев, которые будут рассмотрены позднее в данной главе. Сейчас необходимо подчеркнуть связь этих методов с общей формой методов наискорейшего спуска (8.3). Несмотря на то что приращения в соотношении (8.4) не являются градиентами, удобно рассматривать данный метод как метод наискорейшего спуска (8.3) с самонастраивающейся аппроксимацией вместо искомых результатов вычислений $V^\pi(s_t)$.

```

Инициализировать  $\vec{\theta}$  произвольно,  $\vec{e} = \vec{0}$ 
Повторять (для каждого эпизода):
     $s \leftarrow$  начальное состояние эпизода
    Повторять (для каждого шага эпизода):
         $a \leftarrow$  действие, задаваемое стратегией  $\pi$  для  $s$ 
        Выполнить действие  $a$ , найти вознаграждение  $r$  и следующее состояние  $s'$ 
         $\delta \leftarrow r + \gamma V(s') - V(s)$ 
         $\vec{e} \leftarrow \gamma \lambda \vec{e} + \nabla_{\vec{\theta}} V(s)$ 
         $\vec{\theta} \leftarrow \vec{\theta} + \alpha \delta \vec{e}$ 
         $s \leftarrow s'$ 
    Пока  $s$  не станет завершающим состоянием

```

Рис. 8.1. Оперативный метод TD(λ) на основе наискорейшего спуска для оценивания функции V^π . Приближенная функция ценности V является неявной функцией от вектора θ_t

Так же как прямое представление метода TD(λ) наискорейшего спуска описывается формулой (8.4), обратное представление описывается следующим соотношением:

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha \delta_t \vec{e}_t, \quad (8.5)$$

где δ_t — это обычная TD-ошибка,

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t), \quad (8.6)$$

а \vec{e}_t — это вектор-столбец следов приемлемости, содержащий компоненты вектора $\vec{\theta}_t$ и скорректированный по правилу

$$\vec{e}_t = \gamma \lambda \vec{e}_{t-1} + \nabla_{\vec{\theta}_t} V_t(s_t), \quad (8.7)$$

причем $\vec{e}_0 = \vec{0}$. Полный алгоритм оперативного метода TD(λ) на основе наискорейшего спуска приведен на рис. 8.1.

В обучении с подкреплением широко применяются два метода аппроксимации функций, основанные на наискорейшем спуске. Первый представляет собой многоуровневые искусственные нейронные сети, использующие алгоритм обратного распространения ошибки. Это соответствует приведенным выше уравнениям и алгоритмам, в которых процесс обратного распространения является способом вычисления градиентов. Второй широко применяемой разновидностью является линейная форма, которая подробно рассматривается в следующем разделе.

Упражнение 8.1. Покажите, что табличный TD(λ)-поиск является частным случаем общего TD(λ)-метода, заданного выражениями (8.5)–(8.7).

Упражнение 8.2. Объединение состояний является простой формой аппроксимации обобщающих функций, когда состояния группируются вместе, с одной позицией в таблице (расчетной ценностью), используемой для каждой группы. Каждый раз, когда возникает состояние из данной группы, соответствующая группе позиция используется для определения ценности состояния, а когда корректируется состояние, корректируется и соответствующая группе позиция. Покажите, что такой тип объединения состояний является частным случаем градиентного метода, такого как (8.3).

Упражнение 8.3. Приведенные в этом разделе уравнения описывают оперативную версию метода TD(λ) на основе наискорейшего спуска. Какими будут уравнения для автономного варианта? Приведите полное описание, задающее новую приближенную функцию ценности V' в конце эпизода на основе приближенной функции ценности V , используемой в течение эпизода. Начните с изменения соотношения (8.4), описывающего прямой подход метода TD(λ).

Упражнение 8.4*. Покажите, что в случае с автономной корректировкой, выражения (8.5)–(8.7) описывают такую же корректировку, как и (8.4).

8.3. Линейные методы

Одним из наиболее важных частных случаев аппроксимации функций на основе метода наискорейшего спуска является случай, в котором приближенная функция V_t представляет собой линейную функцию вектора параметров $\vec{\theta}_t$. Каждому состоянию соответствует вектор-столбец его свойств $\vec{\phi}_s = (\phi_s(1), \phi_s(2), \dots, \phi_s(n))^T$ с таким же числом компонент, что и у вектора $\vec{\theta}_t$. Свойства могут быть заданы множеством способов; ниже рассматриваются некоторые из них. Однако если эти свойства заданы, приближенная функция ценности состояния имеет вид

$$V_t(s) = \vec{\theta}_t^T \vec{\phi}_s = \sum_{i=1}^n \theta_t(i) \phi_s(i). \quad (8.8)$$

В данном случае говорят, что приближенная функция ценности *линейна по параметрам*, или просто *линейна*.

Удобно использовать корректировки методом наискорейшего спуска совместно с линейной аппроксимацией функций. В этом случае градиент приближенной функции ценности относительно θ_t имеет вид

$$\nabla_{\vec{\theta}_t} V_t(s) = \vec{\phi}_s.$$

Таким образом, в линейном случае общая корректировка методом наискорейшего спуска (8.3) приобретает очень простую форму. Кроме того, в линейном случае имеет место только один оптимум $\vec{\theta}^*$ (или один набор одинаковых оптимумов в вырожденном случае). Таким образом, любой метод, гарантированно сходящийся к локальному оптимуму или в его окрестность, гарантированно сходится к глобальному оптимуму или в его окрестность. Вследствие своей простоты линейный вариант на основе наискорейшего спуска — один из самых удобных для изучения с точки зрения математического анализа. Почти все важные результаты сходимости для систем обучения всех видов получены для линейных (или более простых) методов аппроксимации функций.

В частности, сходимость алгоритма TD(λ), основанного на наискорейшем спуске, рассмотренного в предыдущем разделе (рис. 8.1), была доказана для линейного случая при условии, что размер шага убывает со временем в соответствии с обычными условиями (2.8). Имеет место сходимость не к вектору параметров с минимальной ошибкой $\vec{\theta}^*$, но к близкому вектору параметров $\vec{\theta}_\infty$, ошибка которого будет ограниченной согласно соотношению

$$\text{MSE}(\vec{\theta}_\infty) \leq \frac{10\gamma\lambda}{1-\gamma} \text{ MSE}(\vec{\theta}^*). \quad (8.9)$$

То есть асимптотическая ошибка не превышает значения наименьшей возможной ошибки, умноженного на $(1 - \gamma\lambda)/(1 - \gamma)$. При стремлении λ к 1 данная граница приближается к минимальной ошибке. Аналогичная граница применяется в случае с другими самонастраивающимися методами с интегрированной оценкой ценности стратегий. Например, линейные TD-корректировки на основе наискорейшего спуска (8.3) с распределением на основе интегрированной оценкой ценности стратегий будут сходиться к тому же результату, что и TD(0). Формально, данная граница применима только в случае с приведенными продолжительными задачами, но похожий результат предположительно достигается и в случае с эпизодными задачами. Должны выполняться еще несколько формальных условий, накладываемых на вознаграждения, свойства и убывание размера шага, которые здесь не затрагиваются. Все подробности можно найти в работе ([Tsitsiklis and Van Roy (1997a)]), где эти вопросы рассматривались впервые.

Для рассмотренных выше результатов строго необходимо, чтобы состояния дублировались в соответствии с распределением на основе интегрированной оценки. Для других распределений дублирования самонастраивающиеся методы, использующие аппроксимацию функции, могут,

вообще говоря, расходиться на бесконечности. Такие примеры и возможные методы решений рассмотрены в разд. 8.5.

Помимо рассмотренных теоретических результатов линейные методы обучения интересны тем, что могут быть очень эффективными на практике как при оперировании с данными, так и с точки зрения вычислений. Так это будет или нет, больше всего зависит от того, каким образом представлены состояния и их свойства. Выбор свойств, адекватных задаче, является важным способом задания априорных знаний в данной области для систем обучения с подкреплением. Очевидно, что эти свойства должны соответствовать начальным свойствам задачи, на основе которых строится обобщение. Если оцениваются геометрические объекты, то необходимо, например, иметь свойства каждой возможной формы, цвета, размера или выполняемой функции. Если оцениваются состояния мобильного робота, то необходимо иметь свойства месторасположения, степень зарядки элементов питания, недавние показания системы звуковой локации и т. д.

В общем случае необходимо также иметь свойства комбинаций данных характеристик, поскольку линейные варианты алгоритмов не допускают представления взаимосвязей между свойствами. Примером такой взаимосвязи может служить положительный эффект от свойства i , который имеет место только при отсутствии свойства j . Скажем, в задаче о балансировке стержня (пример 3.4) высокая угловая скорость может иметь как положительный, так и отрицательный эффект в зависимости от имеющегося отклонения от вертикали. Если угол большой, то высокая угловая скорость означает неизбежное падение, негативное состояние, тогда как если угол небольшой, то высокая угловая скорость означает, что стержень будет выправлять свое положение, т. е. положительное состояние. В случае таких взаимосвязей при использовании линейных методов аппроксимации функции, необходимо вводить свойства сочетаний ценностей свойств. Далее мы рассмотрим некоторые общие пути решения данной проблемы.

Упражнение 8.5. Каким образом можно преобразовать табличный случай, используя линейный подход?

Упражнение 8.6. Каким образом можно преобразовать случай с объединением состояний (см. упражнение 8.2), используя линейный подход?

Грубое кодирование

Рассмотрим задачу с непрерывным двумерным множеством состояний. Состояние в этом случае представляет собой точку в двумерном про-

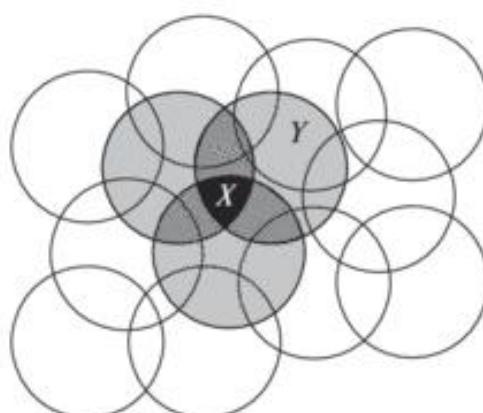


Рис. 8.2. Грубое кодирование. Обобщение между состояниями X и Y зависит от количества их свойств, рецептивные поля которых (в данном случае окружности) пересекаются. Данные состояния имеют одно общее свойство, таким образом, обобщение между ними будет небольшим

пространстве, которой соответствует вектор с двумя компонентами. Одним из типов свойств в этом случае будут свойства, соответствующие окружностям в пространстве состояний, как показано на рис. 8.2. Если состояние внутри окружности, то соответствующее свойство имеет значение 1, и говорят, что оно *присутствует*; в противном случае свойство равно 0, и говорят, что оно *отсутствует*. Свойства такого типа называются *бинарными*. При заданном состоянии то, какие бинарные свойства присутствуют, показывает, в каких окружностях лежит состояние, и, таким образом, приближенно программирует его положение. Задание состояния с помощью перекрывающихся таким образом свойств (которые не обязательно должны быть окружностями или бинарными свойствами) называется *грубым кодированием*.

Рассмотрим влияние размера и плотности окружностей в случае с линейным методом аппроксимации функций на основе наискорейшего спуска. Каждой окружности соответствует один параметр (компоненты вектора $\vec{\theta}_t$), который меняется в процессе обучения. Если обучение происходит в одной точке (состоянии) X , то влияние будет оказываться на параметры всех окружностей, содержащих X . Таким образом, в соответствии с соотношением (8.8) приближенная функция ценности будет изменяться во всех точках внутри общей площади окружностей, с тем большим эффектом, чем больше данная окружность «имеет общего» с состоянием X , как показано на рис. 8.2. Если окружности небольшие, то обобщение будет иметь место на небольших расстояниях, как на рис. 8.3, а, если окружности большие, то соответствующая дистанция будет больше, как на рис. 8.3, б. Кроме того, природу обобщения



Рис. 8.3. Обобщение в линейных методах аппроксимации функции определяется размерами и формой рецептивных полей, соответствующих свойствам. Во всех трех случаях имеется примерно одинаковое количество и плотность свойств

определяет форму свойств. Например, если форма не является строго окружностью, но вытянута в некотором направлении, то обобщение будет соответствующим, как показано на рис. 8.3, б.

Свойства с большими рецептивными полями дают более широкое обобщение, но могут также ограничивать искомую функцию на уровне грубой аппроксимации, проводя разграничения не точнее, чем это позволяет ширина рецептивных полей. Однако в данном случае это не так. Начальное обобщение двух точек действительно контролируется размером и формой рецептивного поля, но четкость, наиболее точное возможное разграничение, в большей степени контролируется общим числом свойств.

Пример 8.1 (Степень грубоści для грубого кодирования). Дан-
ный пример иллюстрирует влияние размеров рецептивных полей на обу-
чение в грубом кодировании. Для исследования одномерной функции
прямоугольных импульсов (показана в верхней части рис. 8.4) была ис-
пользована линейная аппроксимация функции, основанная на грубом
кодировании и методе (8.3). Значения данной функции были использо-
ваны в качестве целей v_t . В одномерном пространстве рецептивные поля
вместо окружностей представляли собой интервалы. Обучение проводи-
лось с тремя различными размерами интервалов: маленьким, средним и
большим, как показано в нижней части рисунка. Во всех трех случаях
имела место одинаковая плотность свойств, около 50 на исследуемую об-
ласть функции. Обучающие примеры генерировались равномерно слу-
чайным образом в этой области. Использовался размер шага $\alpha = 0,2/m$,

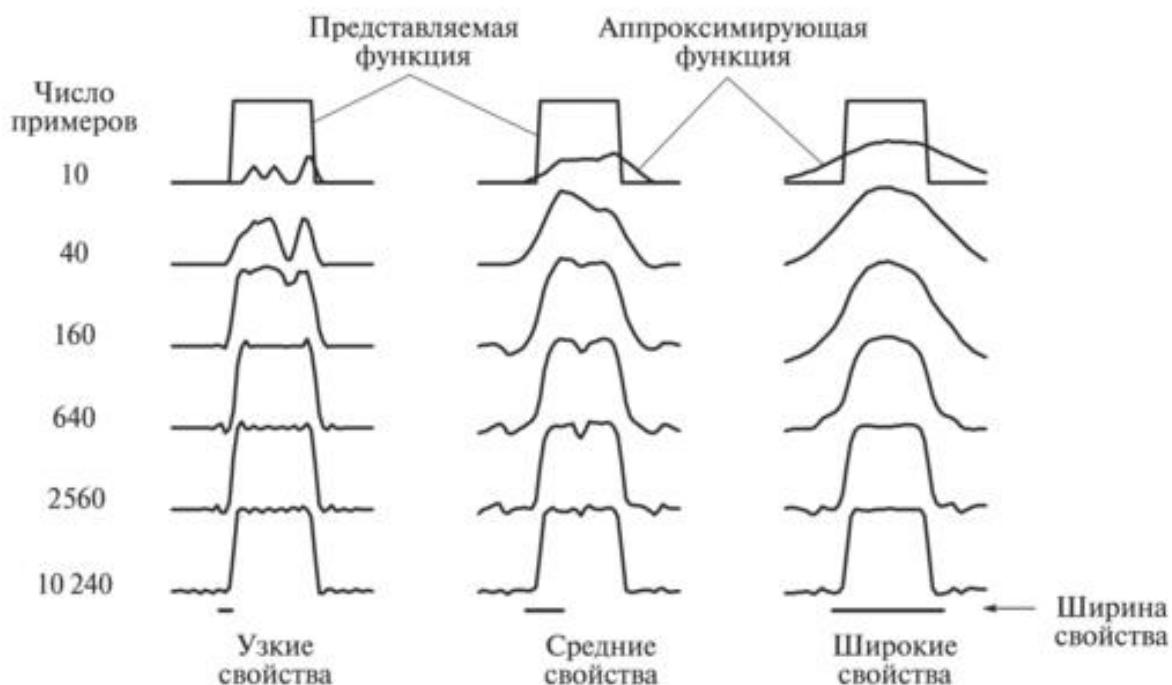


Рис. 8.4. Пример значительного влияния ширины свойств на начальное обобщение (первый ряд) и незначительного влияния на асимптотическую точность (последний ряд)

где t — число представленных одновременно свойств. На рис. 8.4 показаны функции, найденные во всех трех случаях в течение обучения. Заметьте, что ширина свойства имеет большое значение в начале обучения. С широкими свойствами обобщение также стремится к расширению; с узкими свойствами изменялись только располагавшиеся рядом области, делая исследуемую функцию более бугристой. Тем не менее на окончательную функцию ширина свойств оказала незначительное влияние. Рецептивные поля оказывали сильное влияние на обобщение, но влияние на качество асимптотического решения было незначительным.

Мозаичное кодирование

Мозаичное (фрагментное) кодирование является формой грубого кодирования, которая особенно хорошо подходит для цифровых компьютеров с жесткой последовательностью операций и для эффективного оперативного обучения. В мозаичном кодировании рецептивные поля свойств группируются при помощи исчерпывающего разбиения пространства входов. Каждое такое разбиение называется *фрагментацией изображения*, и каждый элемент разбиения называется *фрагментом*.

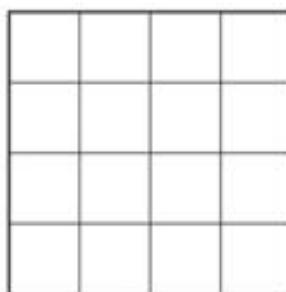
том. Каждому фрагменту рецептивного поля соответствует одно свойство.

Очевидным преимуществом мозаичного кодирования является то, что общее количество свойств, представляемых одновременно, строго контролируется и не зависит от состояния входа.

В каждом фрагменте представлено в точности одно свойство, и таким образом, общее число представленных свойств всегда совпадает с числом фрагментов. Это позволяет очень просто задавать размер шага. Например, выбор $\alpha = 1/m$, где m — число фрагментов, приводит к обучению с одной попыткой. Если получен пример $s_t \rightarrow v_t$, то при любом предыдущем значении $V_t(s_t)$ новое значение будет $V_{t+1}(s_t) = v_t$. Обычно необходимы более медленные изменения, позволяющие обобщать и случайным образом варьировать выходные параметры. Например, можно выбрать $\alpha = 1/10m$. В этом случае за одну корректировку будет проходить одна десятая пути до цели.

Вследствие того что мозаичное кодирование использует исключительно бинарные (0 или 1) значения свойств, взвешенная сумма, определяющая приближенную функцию ценности (8.8), вычисляется тривиальным образом. Вместо того чтобы выполнять n раз операции умножения и сложения, можно просто рассчитать индексы $m \ll n$ представленных свойств и затем прибавить m соответствующих компонент вектора параметров. Расчет следов приемлемости (8.7) также упрощается, так как компоненты градиента $\nabla_{\vec{\theta}} V_t(s_t)$ обычно или равны 0, или 1.

Расчет индексов свойств особенно удобно проводить с использованием структуры фрагментов в виде сетки. Идеи и методы в данном случае лучше всего иллюстрируются на примерах. Предположим, что в задаче имеются два непрерывных параметра состояния. Тогда наиболее простым способом разбиения пространства на фрагменты будет использование равномерной двумерной сетки:



При заданных координатах x и y точки в пространстве не составляет большого труда определить индекс фрагмента, в котором находится данная точка. При использовании нескольких смешанных друг

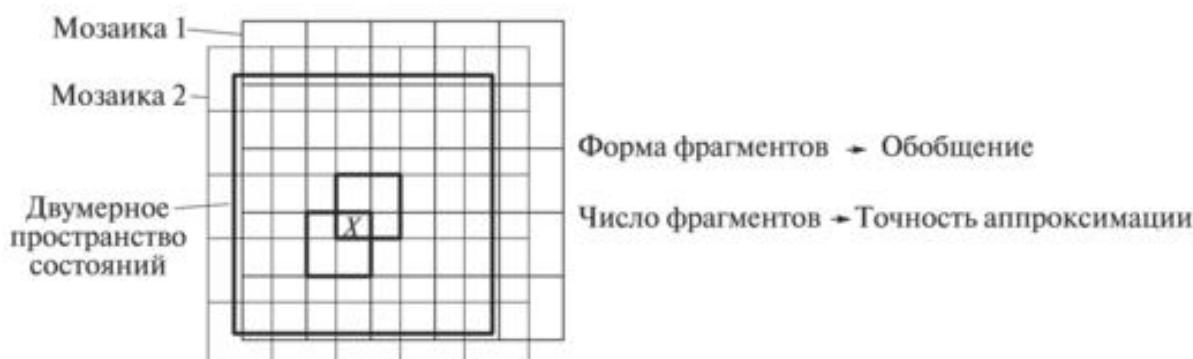


Рис. 8.5. Мозаичное разбиение сеточного типа с перекрывающимися прямоугольными фрагментами

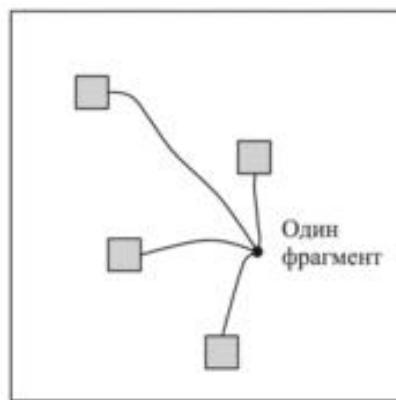
относительно дуга мозаичных разбиений каждое из них разбивает пространство по-своему. В примере, изображенном на рис. 8.5, к сетке были добавлены дополнительные колонка и ряд таким образом, чтобы не осталось неохваченных точек. На рисунке выделены два фрагмента, которые соответствуют состоянию, обозначенному X . Разные разбиения могут быть смешены друг относительно друга произвольно или в соответствии с четко продуманными детерминированными стратегиями (известно, что простое смещение разбиений на одинаковое расстояние не является эффективным методом). Показанные на рис. 8.3 и 8.4 влияния различных факторов на обобщение и асимптотическую точность также имеют место в данном случае. Ширина и форма фрагментов должны выбираться в соответствии с шириной обобщения, которое необходимо в данном случае. Количество фрагментов должно выбираться в соответствии с их требуемой плотностью. Чем больше будет плотность фрагментов, тем точнее может быть аппроксимирована искомая функция, и тем больше потребуется вычислений.

Важно отметить, что мозаичное разбиение может быть произвольным и не обязательно должно иметь вид равномерной сетки. Кроме того, что фрагменты могут иметь какую угодно форму, как на рис. 8.6, *a*, их форма и распределение могут определяться конкретным видом обобщения. Например, изображенное на рис. 8.6, *б*, разбиение полосами обеспечивает обобщение по вертикали и ограничение по горизонтали, особенно слева. Изображенное на рис. 8.6, *в*, диагональное разбиение обеспечивает обобщение по диагонали. В случае с пространствами с большим количеством измерений параллельные оси полосы разбиения соответствуют игнорированию некоторых измерений в некоторых фрагментах, т. е. гиперплоским фрагментам.



Рис. 8.6. Мозаичные разбиения

Еще одним важным способом уменьшения затрат памяти является **хеширование** — последовательное псевдослучайное дробление больших мозаичных разбиений на значительно более мелкие множества фрагментов. Хеширование приводит к возникновению фрагментов, состоящих из прерывистых непересекающихся областей, случайным образом распределенных в пространстве состояний, но, тем не менее, формирующих полное мозаичных разбиение. Например, один фрагмент может состоять из подфрагментов, как показано ниже:



При использовании хеширования затраты памяти часто снижаются в значительной степени при незначительных снижениях производительности. Это оказывается возможным вследствие того, что высокая точность необходима лишь в небольших областях пространства состояний. Хеширование позволяет избежать сложностей, связанных с размерностью пространства в том смысле, что требуемое количество памяти не должно экспоненциально возрастать с увеличением размерности, но должно лишь соответствовать конкретным требованиям задачи.

В настоящее время программные реализации данных алгоритмов, в том числе и хеширование, широко доступны для использования.

Упражнение 8.7. Предположим, что одно из двух измерений пространства состояний оказывает большее влияние на функцию ценности, чем другое, и что обобщение должно происходить преимущественно по-перек этого измерения, а не вдоль него. Какой тип разбиения можно использовать, чтобы учесть это обстоятельство?

Радиально-базисные функции

Радиально-базисные функции (РБФ) являются естественным обобщением грубого кодирования на свойства с непрерывными значениями. Теперь свойства могут принимать не только значения 1 либо 0, но любые значения из интервала $[0, 1]$, что отражает различные *уровни* представленных свойств. Типичное РБФ-свойство, обозначаемое как i , имеет гауссовский (колоколообразный) отклик, зависящий только от расстояния между состоянием s и состоянием-прототипом или центральным состоянием c_i свойства, а также от ширины свойства σ_i :

$$\phi_s(i) = \exp\left\{-\frac{\|s - c_i\|^2}{2\sigma_i^2}\right\}.$$

Очевидно, что норма или метрика расстояния здесь может быть выбрана любым способом, удобным в конкретной задаче с конкретными состояниями. На рис. 8.7 изображен одномерный пример с евклидовым расстоянием.

РБФ-сеть — это линейный аппроксиматор функций, использующий РБФ для представления своих свойств. Обучение описывается соотношениями (8.3) и (8.8), в точности как и в случае с другими линейными аппроксиматорами функции. Основное преимущество РБФ над бинарными свойствами заключается в том, что РБФ дают плавно изменяющиеся и дифференцируемые приближенные функции. Кроме того, некоторые методы обучения для РБФ-сетей меняют центр и ширину свойства.

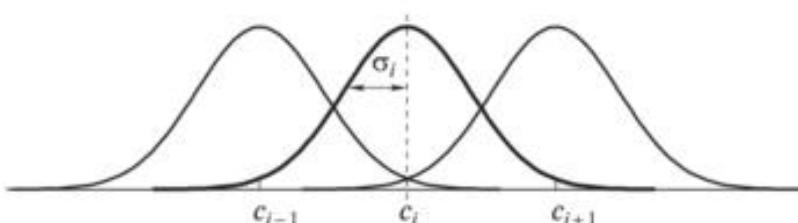


Рис. 8.7. Одномерные радиально-базисные функции

С помощью таких нелинейных методов можно получать искомую функцию с более высокой точностью. К недостаткам РБФ-сетей и особенно нелинейных РБФ-сетей можно отнести присущую им большую сложность вычислений и часто более длительную ручную настройку прежде, чем обучение становится устойчивым и эффективным.

Кодирование по методу Канервы

В задачах высоких размерностей, например с сотнями измерений, мозаичное кодирование и РБФ-сети неприменимы. Сложность вычислений любого из этих методов возрастает экспоненциально с увеличением размерности пространства. Существуют некоторые приемы, позволяющие замедлить этот рост (например, хеширование), но и они оказываются бесполезными уже при наличии всего нескольких десятков измерений.

С другой стороны, некоторые общие идеи, лежащие в основе данных методов, могут использоваться и в случае многомерных задач. В частности, идея представления состояний в виде набора свойств с последующим линейным отображением данных свойств на некоторую аппроксимацию может с успехом применяться в больших задачах. В данном случае важно не допустить резкого увеличения числа свойств. Есть ли основания полагать, что это возможно?

Для начала необходимо определиться с ожидаемым результатом. Упрощенно говоря, аппроксиматор функций заданной сложности может хорошо аппроксимировать искомые функции лишь сравнимой сложности. Однако с ростом числа измерений размер пространства состояний, вообще говоря, возрастает экспоненциально. Разумно предположить, что в наихудшем случае сложность искомой функции возрастает в соответствии с ростом размерности пространства состояний. Таким образом, если рассматривать наихудший случай, то решения не существует, т. е. не существует способа добиться хорошей аппроксимации в задачах с многомерными пространствами, не прибегая к ресурсам, экспоненциально возрастающим с ростом размерности пространства.

Более действенный способ решения данной проблемы — рассмотреть сложность искомой функции в отрыве от величины области состояний и размерности пространства состояний. Величина области состояний может задавать верхнюю границу сложности, но ниже данной верхней границы сложность и размерность могут быть никак не связаны. Например, в многомерной задаче из тысячи измерений пространства важную роль может играть лишь одно. Задаваясь уровнем сложности, необходимо уметь хорошо аппроксимировать любую искомую функцию равной

или меньшей сложности. С ростом уровня сложности необходимо пропорциональное увеличение доступных вычислительных ресурсов.

При таком подходе точкой отсчета становится уже не размерность пространства, но сложность искомой функции или хорошей ее аппроксимации. Таким образом, добавление размерностей в задаче, например путем введения новых свойств или новых сенсоров, не должно иметь никаких последствий, пока сложность требуемой аппроксимации остается неизменной. Новые размерности могут даже облегчить решение задачи, если искомая функция может быть достаточно просто выражена через них. К сожалению, такие методы, как фрагментное кодирование или РБФ-кодирование, не работают в этом случае. Их сложность возрастает экспоненциально с ростом размерности, даже если сложность искомой функции остается неизменной. Для этих методов размерность по-прежнему является проблемой. Необходимы методы, сложность которых не зависит непосредственно от размерности пространства, но зависит лишь от сложности того, что они аппроксимируют.

Один из таких методов, удовлетворяющий данным критериям и называемый *Канерва-кодированием*, предлагает выбирать свойства, которые соответствуют конкретным *состояниям-прототипам*. Для определенности будем считать, что данные прототипы случайным образом выбираются из пространства состояний. Рецептивным полем такого свойства являются все состояния, находящиеся в достаточной близости к прототипу. В Канерва-кодировании используется метрика расстояния, отличная от той, что использовалась в мозаичном кодировании и РБФ. Для определенности будем рассматривать *бинарное* пространство состояний и *хэмминговское расстояние*, которые отличаются числом разрядов, используемых для представления состояний. Состояния считаются одинаковыми, если они совпадают в достаточном количестве измерений пространства, даже если в остальных измерениях они совершенно различные.

Преимущество Канерва-кодирования заключается в том, что сложность изучаемых функций, полностью зависит от количества свойств, что не означает обязательной связи с размерностью задачи. Количество свойств может быть как больше, так и меньше количества размерностей задачи. Экспоненциальная зависимость от количества размерностей имеет место только в наихудшем случае. Таким образом, размерность сама по себе больше не является проблемой. Проблемой, как это и должно быть, по-прежнему остаются сложные функции. Для решения более сложных задач методу Канерва-кодирования просто требуется наличие большего числа свойств. На данный момент еще не накоплено

значительного опыта использования таких систем, но есть основания полагать, что их возможности возрастают пропорционально имеющимся вычислительным ресурсам. На настоящее время данная область является объектом исследования, и значительные улучшения существующих методов еще предстоит осуществить.

8.4. Управление с аппроксимацией функции

Следуя схеме ОИС, расширим методы прогнозирования ценности до методов управления при использовании аппроксимации функций. Вначале мы расширим методы прогнозирования ценности состояний до методов прогнозирования действий, а затем дополним их методиками улучшения стратегии и выбора действий. Как обычно, проблема достаточной доли изучающих действий решается применением либо управления по методу Монте-Карло с интегрированной, либо с разделенной оценкой ценности стратегий.

Процесс расширения рассматриваемого метода до прогнозирования ценности действий достаточно очевиден. В этом случае в параметрической функциональной форме с вектором параметров θ_t представляется функция ценности действий $Q_t \approx Q_\pi$. В то время как раньше обучающие примеры имели вид $s_t \mapsto v_t$, сейчас они принимают вид $s_t, a_t \mapsto v_t$. Выход v_t может быть любой аппроксимацией функции $Q_\pi(s_t, a_t)$, включая обычные дублированные значения, такие как полная выгода R_t по методу Монте-Карло или выгода по типу одношагового метода SARSA

$$r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}).$$

Общая корректировка на основе наискорейшего спуска для прогнозирования ценности действий имеет вид

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha [v_t - Q_t(s_t, a_t)] \nabla_{\vec{\theta}_t} Q_t(s_t, a_t).$$

Например, обратный подход к методу ценности действий, аналогичному TD(λ), дает:

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha \delta_t \vec{e}_t,$$

где

$$\delta_t = r_{t+1} + \gamma Q_t Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)$$

и

$$\vec{e}_t = \gamma \lambda \vec{e}_{t-1} + \nabla_{\vec{\theta}_t} Q_t(s_t, a_t),$$

причем $\vec{e}_0 = \vec{0}$. Данный метод называется *методом SARSA(λ) на основе наискорейшего спуска*, особенно когда он имеет законченный вид

метода управления. При постоянной стратегии данный метод сходится таким же образом, что и метод $\text{TD}(\lambda)$, с аналогичным типом границы ошибки (8.9).

Для формирования метода управления необходимо соединить методы прогнозирования ценности действия с методиками улучшения стратегии и выбора действий. Подходящие методики, имеющие дело с непрерывными действиями или с действиями, выбираемыми из больших дискретных множеств, в настоящий момент являются объектом исследования с неясным пока результатом. С другой стороны, если набор действий дискретный и не очень большой, то можно использовать методики, уже рассмотренные в предыдущих главах, т. е. для каждого возможного действия a , доступного в текущем состоянии s_t , можно рассчитать значение $Q_t(s_t, a)$ и затем найти жадное действие $a_t^* = \arg \max_a Q_t(s_t, a)$. Совершенствование стратегии осуществляется заменой расчетной стратегии на жадную стратегию (управление с разделенной оценкой ценности стратегий) или на гибкую аппроксимацию жадной стратегии, например ε -жадную стратегию (в методах с интегрированной оценкой ценности стратегий). Действия выбираются согласно этой самой стратегии в методах с интегрированной оценкой ценности стратегий или согласно произвольной стратегии в методах с разделенной оценкой ценности стратегий.

На рис. 8.8 и 8.9 показаны примеры методов управления с интегрированной оценкой ценности стратегий ($\text{SARSA}(\lambda)$) и методов с разделенной оценкой ценности стратегий (метод $Q(\lambda)$ Уоткинса), использующих аппроксимацию функций. Оба метода используют линейную аппроксимацию на основе наискорейшего спуска для функций с бинарными свойствами, такими как в мозаичном кодировании и в Канервакодировании. Оба метода используют ε -жадную стратегию для выбора действий, так же как метод $\text{SARSA}(\lambda)$ использует ее для ОИС. Оба вычисляют множества имеющихся свойств \mathcal{F}_a , соответствующих текущему состоянию и всем возможным действиям a . Если функция ценности для каждого действия представляет собой отдельную линейную функцию тех же свойств (обычный случай), то индексы множества \mathcal{F}_a для каждого действия, по сути, одни и те же, что значительно упрощает процедуру вычислений.

Все рассмотренные выше методы используют *накапливающиеся* следы приемлемости. Несмотря на то что замещающие следы (см. разд. 7.8) имеют преимущества в случае с табличными методами, они не могут быть непосредственно использованы в аппроксимации функций. Напомним, что идея замещающих следов заключается в том, что каждый раз,

```

Инициализировать  $\vec{\theta}$  произвольно,  $\vec{e} = \vec{0}$ 
Повторять (для каждого эпизода):
     $s \leftarrow$  начальное состояние эпизода
    Для всех  $a \in \mathcal{A}(s)$ :
         $\mathcal{F}_a \leftarrow$  набор свойств для  $s, a$ 
         $Q_a \leftarrow \sum_{i \in \mathcal{F}_a} \theta(i)$ 
         $a \leftarrow \arg \max_a Q_a$ 
        С вероятностью  $\varepsilon$ :  $a \leftarrow$  случайное действие  $\in \mathcal{A}(s)$ 
    Повторять (для каждого шага эпизода):
         $\vec{e} \leftarrow \gamma \lambda \vec{e}$ 
        Для всех  $\bar{a} \neq a$ : (вспомогательный блок для замещающих следов)
            Для всех  $i \in \mathcal{F}_{\bar{a}}$ 
                 $e(i) \leftarrow 0$ 
            Для всех  $i \in \mathcal{F}_a$ 
                 $e(i) \leftarrow e(i) + 1$  (накапливающиеся следы)
                 $e(i) \leftarrow 1$  (замещающие следы)
            Выполнить действие  $a$ , найти вознаграждение  $r$  и следующее состояние  $s'$ 
             $\delta \leftarrow r - Q_a$ 
            Для всех  $a \in \mathcal{A}(s')$ 
                 $\mathcal{F}_a \leftarrow$  набор свойств для  $s', a$ 
                 $Q_a \leftarrow \sum_{i \in \mathcal{F}_a} \theta(i)$ 
                 $a' \leftarrow \arg \max_a Q_a$ 
            С вероятностью  $\varepsilon$ :  $a' \leftarrow$  случайное действие  $\in \mathcal{A}(s')$ 
             $\delta \leftarrow \delta + \gamma Q_{a'}$ 
             $\vec{\theta} \leftarrow \vec{\theta} + \alpha \delta \vec{e}$ 
             $a \leftarrow a'$ 
        Пока  $s'$  не станет завершающим состоянием
    
```

Рис. 8.8. Линейный метод SARSA(λ), основанный на наискорейшем спуске, с бинарными свойствами и ε -жадной стратегией. В данном случае задана корректировка как для накапливающих, так и для замещающих следов, включая вариант (при использовании замещающих следов) стирания следов невыбранных действий

когда происходит посещение данного состояния, вместо увеличения значения его следа на 1 это значение обращается в 1. В случае с аппроксимацией функций не существует отдельного соответствующего некоторому состоянию следа, но существует след для каждой из компонент вектора $\vec{\theta}_t$, который соответствует многим состояниям. Согласно одному из подходов, который хорошо работает с линейными аппроксимациями на основе наискорейшего спуска для функций с бинарными свойствами, необходимо рассматривать свойства как состояния, которые являются целью для замещающих следов, т. е. каждый раз, когда встречается со-

```

Инициализировать  $\vec{\theta}$  произвольно,  $\vec{e} = \vec{0}$ 
Повторять (для каждого эпизода):
     $s \leftarrow$  начальное состояние эпизода
    Для всех  $a \in \mathcal{A}(s)$ :
         $\mathcal{F}_a \leftarrow$  набор свойств для  $s, a$ 
         $Q_a \leftarrow \sum_{i \in \mathcal{F}_a} \theta(i)$ 
    Повторять (для каждого шага эпизода):
        С вероятностью  $1 - \varepsilon$ :
             $a \leftarrow \arg \max_a Q_a$ 
             $\vec{e} \leftarrow \gamma \lambda \vec{e}$ 
        иначе
             $a \leftarrow$  случайное действие  $\in \mathcal{A}(s)$ 
             $\vec{e} \leftarrow \vec{0}$ 
        Для всех  $i \in \mathcal{F}_a$   $e(i) \leftarrow e(i) + 1$ 
        Выполнить действие  $a$ , найти вознаграждение  $r$  и следующее состояние  $s'$ 
         $\delta \leftarrow r - Q_a$ 
        Для всех  $a \in \mathcal{A}(s')$ :
             $\mathcal{F}_a \leftarrow$  набор свойств для  $s', a$ 
             $Q_a \leftarrow \sum_{i \in \mathcal{F}_a} \theta(i)$ 
             $a' \leftarrow \arg \max_a Q_a$ 
             $\delta \leftarrow \delta + \gamma Q_{a'}$ 
             $\vec{\theta} \leftarrow \vec{\theta} + \alpha \delta \vec{e}$ 
    Пока  $s'$  не станет завершающим состоянием

```

Рис. 8.9. Линейный вариант метода $Q(\lambda)$ Уоткинса, основанный на наискорейшем спуске, с бинарными свойствами, ε -жадной стратегией и накапливающимиися следами

стояние, обладающее свойством i , след свойства i приравнивается к 1 вместо того, чтобы получить приращение на 1, как это было бы в случае с накапливающимися следами.

При работе со следами пар состояния—действие может оказаться удобным стирание (приравнивание к нулю) следов всех невыбранных действий во встретившихся состояниях (см. разд. 7.8). Данная идея может быть также распространена на случай с линейной аппроксимацией функций с бинарными свойствами. Для каждого встретившегося состояния вначале стираются следы всех свойств данного состояния и невыбранных действий, затем приравниваются к 1 следы свойств состояния и выбранного действия. Как уже отмечалось для табличного случая, данный способ может оказаться наиболее эффективным при использовании замещающих следов, но может и не оказаться таковым. Процедур-

ное описание обоих типов шагов, включая вариант со стиранием следов невыбранных действий, приведено для алгоритма SARSA(λ) на рис. 8.8.

Пример 8.2 (Задача с машиной на горе). Рассмотрим задачу о движении машины с недостатком мощности по крутой горной дороге, как показано на рис. 8.10 слева вверху. Проблема в том, что гравитация в данном случае сильнее двигателя машины, и даже при полностью открытой дроссельной заслонке машина не может одолеть этот крутой подъем. Единственное решение данной проблемы состоит в том, чтобы подняться на противоположный склон и затем при полностью открытой дроссельной заслонке достичь необходимой инерции для преодоления правого подъема, несмотря на то что машина будет замедляться на протяжении всего подъема. Это простой пример продолжающейся задачи управления, в которой результаты вначале становятся хуже (машина отдаляется от цели) для того, чтобы в конце улучшиться. Многие методики управления испытывают большие сложности, сталкиваясь с задачами такого типа, до тех пор пока не будет самим разработчиком подсказан в явном виде ход решения.

Вознаграждение в данной задаче равно -1 на каждом временном шаге до тех пор, пока машина не преодолеет подъем и не окажется на вершине горы. В этом случае эпизод заканчивается. Существуют три воз-

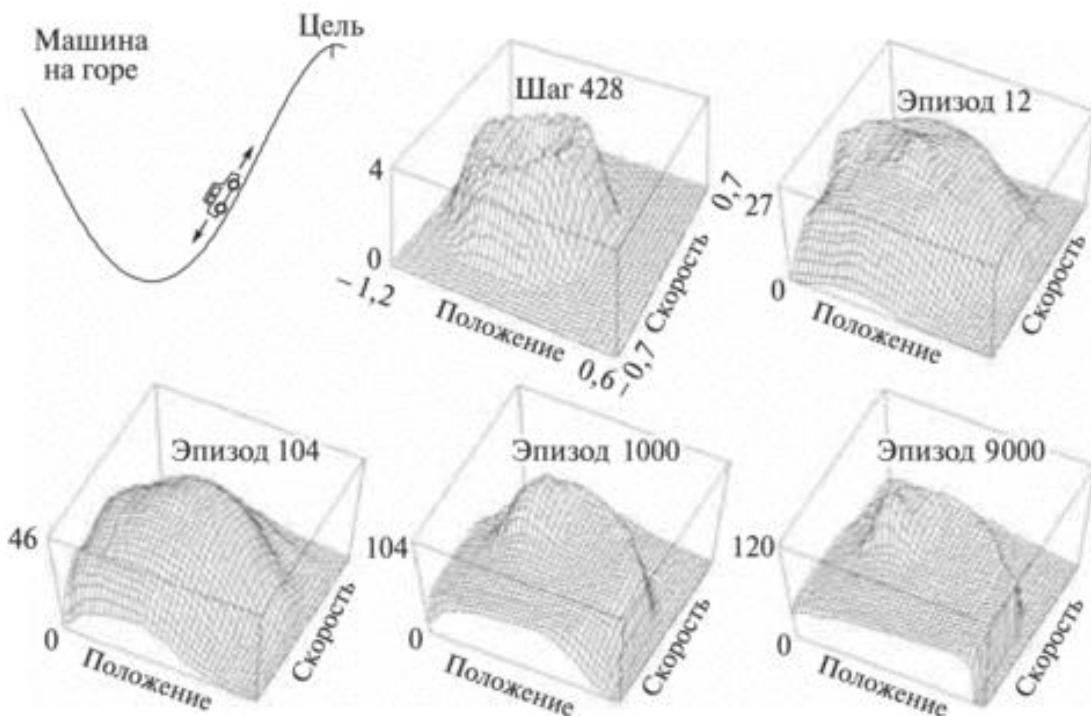


Рис. 8.10. Задача с машиной на горе (левое верхнее изображение) и функция затрат на движение ($-\max_a Q_t(s, a)$), найденная в процессе обучения

можных варианта действия: полный газ вперед (+1), полный газ назад (-1) и закрытая дроссельная заслонка (0). Машина движется в соответствии с простейшими законами физики. Ее положение x_t и скорость $d/dt(x_t)$ корректируются следующим образом:

$$\begin{aligned}x_{t+1} &= \text{bound}[x_t + \dot{x}_{t+1}], \\ \dot{x}_{t+1} &= \text{bound}[\dot{x}_t + 0,001a_t + (-0,0025 \cos(3x_t))],\end{aligned}$$

где оператор bound обеспечивает выполнение неравенств

$$-1,2 \leq x_{t+1} \leq 0,5 \quad \text{и} \quad -0,07 \leq \dot{x}_{t+1} \leq 0,07.$$

Когда положение машины x_{t+1} достигало левой границы, скорость $d/dt(x_{t+1})$ приравнивалась нулю. Когда положение достигало правой границы, цель считалась достигнутой, и эпизод завершался. Каждый эпизод начинался с произвольного положения и скорости машины, которые равномерно выбирались из заданных интервалов. Чтобы перевести непрерывные переменные состояния в бинарные свойства, была использована фрагментная сетка, как показано на рис. 8.5. Были использованы десять фрагментов 9×9 , каждый из которых смешен на некоторую произвольную часть ширины фрагмента.

Изображенный на рис. 8.8 алгоритм SARSA(λ) (использовавший замещающие следы и стирание следов) достаточно просто справился с этой задачей, отыскав близкую к оптимальной функцию в течение 100 эпизодов. На рис. 8.10 показана отрицательная функция ценности (функция затрат на движение), найденная за один прогон, использующая параметры $\lambda = 0,9$, $\varepsilon = 0$ и $\alpha = 0,05$ ($0,1/m$). Все начальные ценности действий были равны нулю, что было оптимистичной оценкой (все истинные значения в данной задаче отрицательные), вызвавшей активный процесс изучения, даже несмотря на то, что параметр изучения ε был равен 0. Это можно наблюдать на средней верхней части рисунка, обозначенной «Шаг 428». В то время, когда еще не был завершен ни один эпизод, машина многократно двигалась во впадине вверх и вниз, следуя круговым траекториям в пространстве состояний. Все часто посещаемые состояния имели худшую ценность по сравнению с неисследованными состояниями, так как получаемые вознаграждения были хуже ожидаемых (не совпадавших с реальными). Это постоянно уводило агента от его начального положения, заставляя исследовать новые состояния до того момента, пока не будет найдено решение. На рис. 8.11 изображены результаты подробного изучения влияния параметров α и λ и типа следов на скорость обучения в данной задаче.

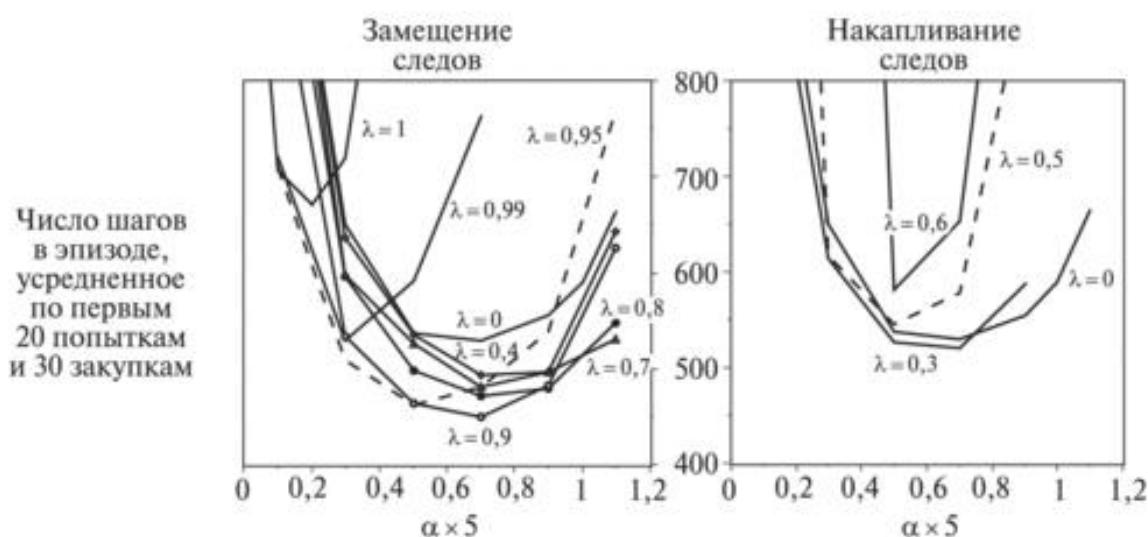


Рис. 8.11. Влияние параметров α , λ и вида следов на результаты в ранней стадии в задаче с машиной на горе. В данном обучении использовались пять фрагментов 9×9

Упражнение 8.8*. Опишите, каким образом можно сочетать метод управления исполнитель–критик с аппроксимацией функций на основе метода наискорейшего спуска.

8.5. Самонастройка с разделенной оценкой ценности стратегий

Вернемся к задаче прогнозирования и рассмотрим более подробно взаимодействие между самонастройкой, аппроксимацией функции и распределением с интегрированной оценкой ценности стратегий. Под *самонастройкой* понимается корректировка оценок ценностей на основе других оценок ценностей. TD-методы, как и методы ДП, содержат в себе самонастройку, тогда как методы Монте Карло не содержат. $TD(\lambda)$ представляет собой самонастраивающийся метод при $\lambda < 1$. По принятому соглашению, при $\lambda = 1$ данный метод считается *не* самонастраивающимся. Несмотря на то что $TD(1)$ использует самонастройку в рамках отдельно взятого эпизода, ее влияние на весь эпизод такое же, как и влияние корректировки не самонастраивающегося метода Монте Карло.

Самонастраивающиеся методы намного сложнее объединить с аппроксимацией функций, чем не самонастраивающиеся. Например, рассмотрим случай прогнозирования ценности с линейной аппроксимацией функций на основе наискорейшего спуска. В этом случае не самонастраивающиеся методы находят решения с минимальной СКО (8.1) для

любого распределения обучающих примеров P , тогда как самонастраивающиеся методы находят лишь решения с приблизительно минимальной СКО (8.9) и только для распределения с интегрированной оценкой ценности стратегий. Более того, точность СКО для метода TD(λ) уменьшается по мере того, как λ отдаляется от 1, т. е. чем больше метод отдаляется от своей несамонастраивающейся формы.

Ограничение на сходимость для самонастраивающихся методов, связанное с распределением с интегрированной оценкой ценности стратегий, вызывает наибольшее беспокойство. Данное ограничение не является проблемой для методов с интегрированной оценкой, таких как SARSA и метод исполнитель—критик, но оно создает определенные проблемы для методов с разделенной оценкой ценности стратегий, таких как Q -обучение и методы ДП. Методы управления с разделенной оценкой ценности стратегий не производят дублирование состояний (или пар состояния—действие) по тому же распределению, согласно которому состояния будут появляться в дальнейшем при следовании стратегии оценивания (стратегии, чью функцию ценности они оценивают). Многие методы ДП, например, равномерно дублируют все состояния. Q -обучение может дублировать состояния в соответствии со случайным распределением. Однако чаще всего оно дублирует состояния в соответствии с распределением, сформированным при взаимодействии с окружающей средой и при использовании гибкой стратегии, близкой к жадной стратегии оценивания. Будем использовать термин «самонастройка с разделенной оценкой ценности стратегий» применительно к любому типу самонастройки, использующей распределение дублирования, отличное от распределения с интегрированной оценкой ценности стратегий. Как ни странно, самонастройка с разделенной оценкой ценности стратегий в сочетании с аппроксимацией функции может привести к расходимости и к бесконечной СКО.

Пример 8.3 (Контрпример Бэрда). Рассмотрим эпизодный марковский процесс из шести состояний, изображенный на рис. 8.12. Эпизод начинается в одном из пяти верхних состояний, немедленно переходит в нижнее состояние и совершает в нем некоторое количество шагов до завершения. Вознаграждение равно нулю за все переходы, таким образом, истинная функция ценности $V^\pi(s)$ равна 0 для всех s . Приближенная функция ценности выражена соотношениями, вписанными в каждое из состояний. Отметим, что общая функция линейна и что состояний меньше, чем компонент вектора $\vec{\theta}_t$. Более того, множество векторов свойств $\{\vec{\phi}_s: s \in \mathcal{S}\}$, соответствующих данной функции, явля-

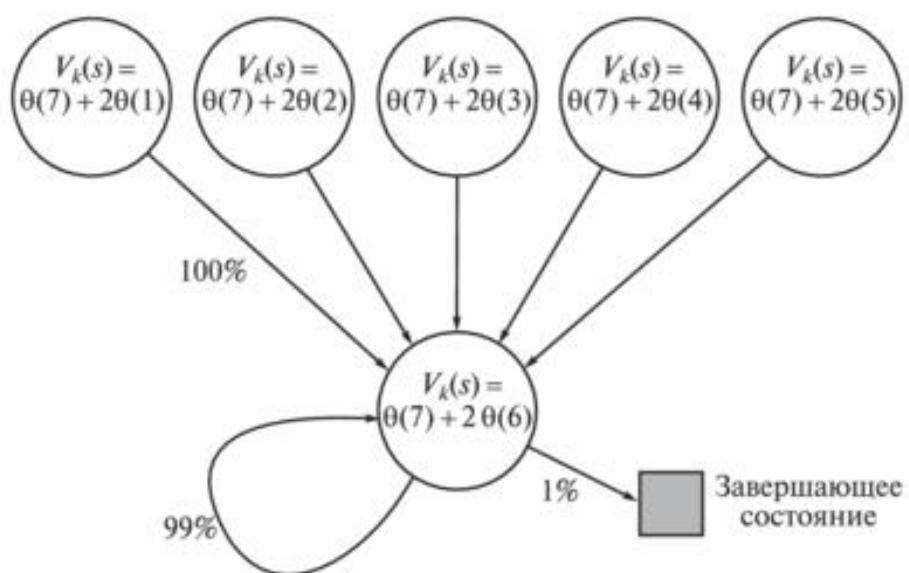


Рис. 8.12. Контрпример Бэрда. Форма приближенной функции ценности данного марковского процесса задана линейными выражениями, вписанными внутрь каждого состояния. Вознаграждение всегда равно нулю

ется линейно независимым множеством, а истинную функцию ценности легко получить, если положить $\vec{\theta}_t = \vec{0}$. В любом случае, в данной задаче уместно применять линейную аппроксимацию функций.

Используемый в данной задаче метод прогнозирования является линейной формой подхода ДП к оценке стратегии, основанной на наискорейшем спуске. Вектор параметров $\vec{\theta}_t$ корректируется прогонкой в пространстве состояний, выполняя синхронное дублирование на основе наискорейшего спуска в каждом состоянии s , используя аппарат ДП (полное дублирование):

$$\vec{\theta}_{k+1} = \vec{\theta}_k + \alpha \sum_s [E\{r_{t+1} + \gamma V_t(s_{t+1}) \mid s_t = s\} - V_k(s)] \nabla_{\vec{\theta}_k} V_k(s).$$

Как и большинство методов ДП, данный метод использует равномерное распределение при дублировании, т. е. одно из самых простых распределений с разделенной оценкой ценности стратегий. В остальном этот случай является идеальной моделью. В нем нет хаотичности и асинхронности. Каждое состояние корректируется строго один раз за прогонку, в соответствии с классическим ДП-дублированием. Метод полностью стандартный, за исключением того, что в нем используется аппроксимация функций на основе наискорейшего спуска. Тем не менее при некоторых начальных значениях параметров система становится неустойчивой, как показано на рис. 8.13.

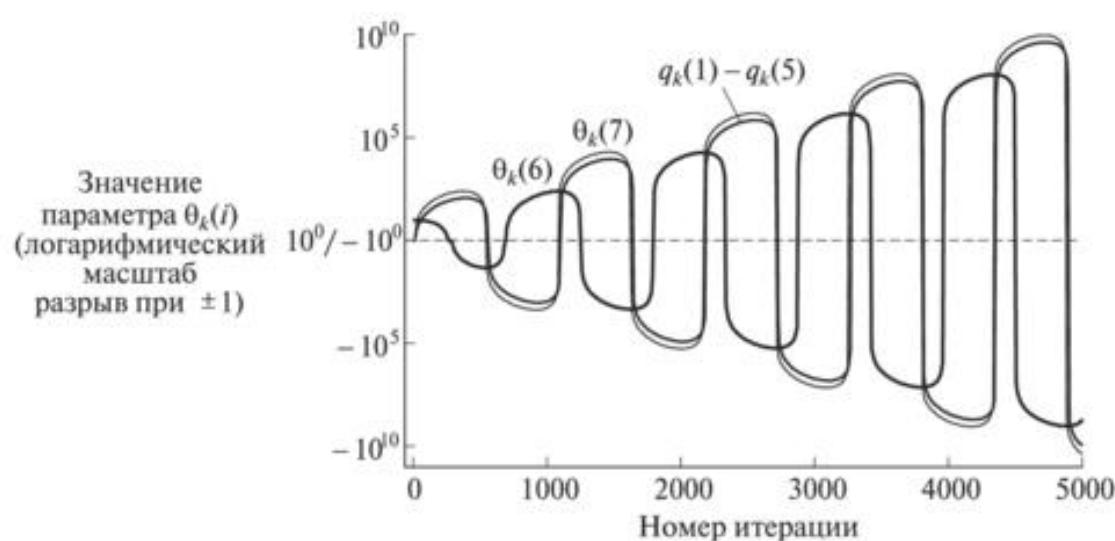


Рис. 8.13. Экспериментальная демонстрация нестабильности ДП-прогнозирования ценности с линейной аппроксимацией функций в контрпримере Бэрда. Использовались следующие параметры: $\gamma = 0,99$, $\alpha = 0,01$ и $\theta_0 = (1, 1, 1, 1, 1, 10, 1)^T$

Если в контрпримере Бэрда изменить только распределение ДП-дублирования, с равномерного на распределение с интегрированной оценкой ценности стратегий (что в общем случае потребует асинхронной корректировки), это гарантирует сходимость к решению с ошибкой, ограниченной неравенством (8.9) для $\lambda = 0$. Результаты данного примера достаточно неожиданны, так как используемый метод ДП является, по-видимому, наиболее простым и наиболее понятным самонастраивающимся методом, а используемый линейный метод, основанный на наиболее скорейшем спуске, также является, вероятно, наиболее простой и наиболее понятной формой аппроксимации функции. Данный пример показывает, что даже простейшая комбинация самонастройки и аппроксимации функции может быть неустойчивой, если дублирования осуществляются не в соответствии с распределением с интегрированной оценкой ценности стратегий.

Существуют также контрпримеры, показывающие расходимость Q -обучения. Это повод для беспокойства, так как в остальных случаях Q -обучение имеет наилучшие гарантии сходимости среди всех методов управления. Были приложены значительные усилия, направленные на решение данной проблемы или хотя бы получение более слабых, но при этом приемлемых, гарантий сходимости. Например, можно гарантировать сходимость Q -обучения до тех пор, пока стратегия поведения (стратегия, используемая при выборе действий) достаточно близка

к стратегии оценивания (стратегии, используемой в ОИС), например, если это ε -жадная стратегия. Насколько известно на данный момент, Q -обучение в этом случае никогда не расходится, однако теоретического анализа не проводилось. В оставшейся части данного раздела представлены еще несколько из исследовавшихся идей.

Предположим, что вместо того чтобы делать шаг навстречу ожидаемой одношаговой выгоде при каждой итерации, как это происходило в контрипримере Бэрда, изменяется функция ценности, приближаясь к наилучшей, наиболее гладкой аппроксимации. Решило бы это проблему неустойчивости? Очевидно, что решило бы, если векторы свойств $\{\phi_s : s \in \mathcal{S}\}$ образуют линейно независимое множество, как это происходит в контрипримере Бэрда, так как в этом случае точная аппроксимация возможна при каждой итерации, и метод вырождается в стандартный табличный метод ДП. Очевидно, однако, что интерес представляет задача, в которой точного решения достичь *невозможно*. В этом случае устойчивость не гарантирована, даже при осуществлении наилучшей аппроксимации при каждой итерации, как показано в следующем примере.

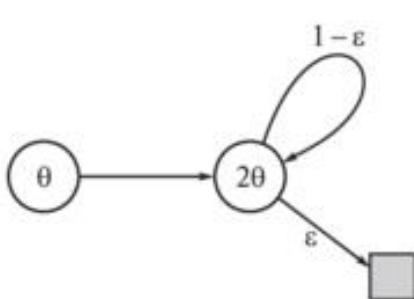


Рис. 8.14. Контрипримеры Цициклиса и Ван-Роя для ДП-оценки стратегии с наиболее гладкой линейной аппроксимацией функции

обоих состояний равны нулю, что в точности описывается равенством $\theta_k = 0$. Если на каждом шаге значение θ_{k+1} задается таким образом, чтобы минимизировать СКО между оценкой ценности и ожидаемой одношаговой выгодой, то имеет место соотношение

$$\begin{aligned} \theta_{k+1} &= \arg \min_{\theta \in \mathfrak{R}} \sum_{s \in \mathcal{S}} [V_\theta(s) - E_\pi \{r_{t+1} + \gamma V_{\theta_k}(s_{t+1}) \mid s_t = s\}]^2 = \\ &= \arg \min_{\theta \in \mathfrak{R}} [\theta - \gamma 2\theta_k]^2 + [2\theta - (1 - \varepsilon)\gamma 2\theta_k]^2 = \frac{6 - 4\varepsilon}{5} \gamma \theta_k, \end{aligned} \quad (8.10)$$

где через V_θ обозначена функция ценности при заданном значении θ . Последовательность $\{\theta_k\}$ расходится при $\gamma > 5/(6 - 4\varepsilon)$ и $\theta_0 \neq 0$.

Пример 8.4 (Контрипример Цициклиса и Ван-Роя). Простейший контрипример линейной наиболее гладкой ДП-аппроксимации функций показан на рис. 8.14. В данном случае имеются только два нетерминальных состояния, и изменяемый параметр θ_k является скаляром. Оценка ценности первого состояния равна θ_k , оценка ценности второго состояния равна $2\theta_k$. Вознаграждение за все переходы равно нулю, так что истинные ценности

Одним из способов избежать неустойчивости является использование специальных методов для аппроксимации функций. В частности, устойчивость гарантируют методы аппроксимации функций, которые не экстраполируют уже пройденные цели. Такие методы, называемые *схемами усреднения*, включают в себя ближайшие соседние методы и локально взвешенную регрессию. Но они не включают в себя такие распространенные методы, как мозаичное кодирование и обратное распространение ошибки.

Другим подходом является попытка минимизации не среднеквадратичной ошибки функции ценности (8.1), а среднеквадратичной ошибки ожидаемой одношаговой выгоды. Обычно ошибку, измеряемую таким образом, называют среднеквадратичной беллмановской ошибкой:

$$\sum_s P(s) [E_\pi\{r_{t+1} + \gamma V_t(s_{t+1}) \mid s_t = s\} - V_t(s)]^2. \quad (8.11)$$

В этом случае предлагается следующая процедура наискорейшего спуска:

$$\begin{aligned} \vec{\theta}_{t+1} &= \vec{\theta}_t - \frac{1}{2} \alpha \nabla_{\vec{\theta}_t} [E_\pi\{r_{t+1} + \gamma V_t(s_{t+1})\} - V_t(s_t)]^2 = \\ &= \vec{\theta}_t + \alpha [E_\pi\{r_{t+1} + \gamma V_t(s_{t+1})\} - V_t(s_t)] \times \\ &\quad \times [\nabla_{\vec{\theta}_t} V_t(s_t) - E_\pi\{\nabla_{\vec{\theta}_t} V_t(s_{t+1})\}], \end{aligned}$$

где ожидаемые ценности в неявной форме зависят от s_t . Данная корректировка гарантированно сходится к минимуму среднеквадратичной беллмановской ошибки при стандартных требованиях к размеру шага. Тем не менее данный метод подходит только для детерминированных систем или для задач с заданной моделью окружающей среды. Проблема заключается в том, что приведенная выше корректировка содержит следующее состояние s_{t+1} , которое появляется в двух ожидаемых ценностях, умножаемых друг на друга. Чтобы иметь объективную выборку результатов, необходимо иметь две независимые выборки следующего состояния, однако в течение нормального взаимодействия с окружающей средой можно получить лишь одну. Вследствие этого практическое применение метода, вероятно, ограничено случаями, когда имеется модель окружающей среды (что позволяет получить вторую выборку). На практике данный метод иногда также сходится очень медленно. Для решения данной проблемы в статье Бэрда [Baird (1995)] предлагается параметрически объединить данный метод с традиционными TD-методами.

Упражнение 8.9 [Программирование]. Найдите в Интернете статью Бэрда [Baird (1995)] и изучите контрипример, относящийся к Q -обу-

чению. Реализуйте его алгоритм и продемонстрируйте расходимость.

8.6. Нужна ли самонастройка?

Вы уже, вероятно, задавались вопросом, зачем вообще нужны все эти самонастраивающиеся методы. Несамонастраивающиеся методы более надежны при использовании совместно с аппроксимацией функций и при более широком спектре условий. Несамонастраивающиеся методы достигают меньшей асимптотической ошибки, чем самонастраивающиеся методы, даже когда дублирование осуществляется в соответствии с распределением с интегрированной оценкой ценности стратегий. При использовании следов приемлемости и параметра $\lambda = 1$ возможна даже оперативная реализация несамонастраивающихся методов пошаговым способом. Несмотря на все это, на практике обычно предпочтение отдается самонастраивающимся методам.

При эмпирическом сравнении самонастраивающиеся методы обычно показывают себя намного лучше несамонастраивающихся методов. Удобным способом получения таких сравнений является использование TD-метода со следами приемлемости и с параметром λ , изменяемым от 0 (чистая самонастройка) до 1 (полное отсутствие самонастройки). На рис. 8.15 приведены результаты данных экспериментов. Во всех случаях по мере приближения λ к 1, т. е. к отсутствию самонастройки, результаты резко ухудшаются. Особенно четко это видно на правом верхнем рисунке. В данном случае имеет место задача оценивания стратегии (прогнозирования), а критерием эффективности метода служит корень из СКО (в конце каждого эпизода, усредненной по 20 эпизодам). Асимптотически, в соответствии с данным критерием, случай с $\lambda = 1$ должен быть наилучшим, но в данном случае вблизи единицы можно наблюдать наихудшие результаты.

В настоящее время не ясно, почему обладающие некоторой самонастройкой методы показывают себя намного лучше, чем абсолютно несамонастраивающиеся методы. Возможно, самонастраивающиеся методы быстрее обучаются, или, например, чему-то они обучаются лучше несамонастраивающихся методов. Имеющиеся результаты показывают, что несамонастраивающиеся методы лучше самонастраивающихся справляются с уменьшением СКО при определении истинной функции ценности, однако это уменьшение не всегда является основной целью. Например, если прибавить 1000 к истинной функции ценности действия в каждой паре состояния—действие, то получится совсем неудовлетворитель-

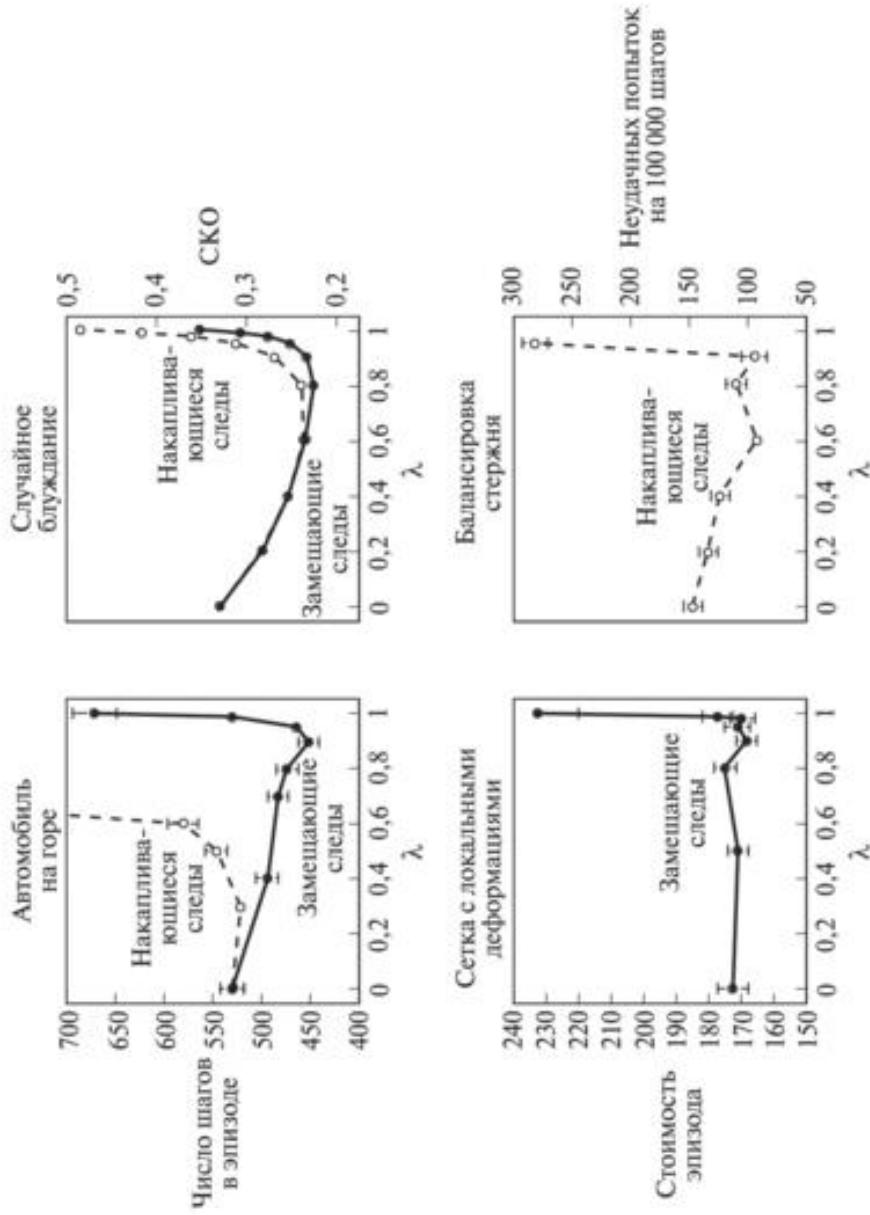


Рис. 8.15. Влияние параметра λ на процесс обучения с подкреплением. Во всех случаях чем лучше результаты, тем ниже располагается кривая. Два левых изображения представляют собой результаты применения к задаче управления с непрерывными состояниями для алгоритма SARSA(λ) и мозаичного кодирования либо с заменщиками, либо с накапливающимися следами [Sutton (1996)]. Верхнее правое изображение относится к оцениванию стратегии в задаче о случайных блужданиях с использованием TD(λ) ([Singh and Sutton (1996)]). Нижнее правое изображение соответствует неопубликованным результатам решения задачи о балансировке стержня (пример 3.4), относящейся к более ранним исследованиям [Sutton (1984)]

ная СКО, но, тем не менее, по-прежнему будет найдена оптимальная стратегия. С самонастраивающимися методами все далеко не так просто, однако, по всей видимости, они что-то делают правильно. Ожидается, что по мере развития исследований наше понимание данного вопроса будет улучшаться.

8.7. Итоги

Системы обучения с подкреплением, применяемые к искусственному интеллекту или к большим инженерным задачам, должны поддаваться *обобщению*. Для того чтобы этого добиться, при использовании любого из существующих методов, применяемых для *обучения с учителем аппроксимации функций*, каждое дублирование можно рассматривать как обучающий пример. *Методы наискорейшего спуска*, в частности, дают возможность расширить все рассмотренные в предыдущих главах методики, включая следы приемлемости, до аппроксимации функций. *Линейные методы наискорейшего спуска* особенно интересны с теоретической точки зрения и хорошо работают на практике при наличии подходящих свойств. Выбор свойств является одним из важнейших способов добавления априорных знаний в конкретной области в системы обучения с подкреплением. Линейные методы включают в себя радиально-базисные функции, мозаичное кодирование и Канерва-кодирование. Методы с обратным распространением ошибки обучения для многослойных нейронных сетей применяются к *нелинейной аппроксимации функций* на основе наискорейшего спуска.

В большинстве случаев расширение методов обучения с подкреплением применительно к задачам прогнозирования и управления до форм, основанных на наискорейшем спуске, тривиально. Тем не менее имеет место интересное взаимодействие между аппроксимацией функций, самонастройкой и использованием интегрированной/разделенной оценки ценности стратегий. Самонастраивающиеся методы, такие как ДП и $TD(\lambda)$ для $\lambda < 1$, работают надежно в совокупности с аппроксимацией функций при более узком диапазоне условий, чем несамонастраивающиеся методы. Вследствие того что случай управления еще не подвергался теоретическому анализу, исследования концентрировались на задаче прогнозирования ценности. В этом случае самонастраивающиеся методы с интегрированной оценкой ценности стратегий в связке с линейной аппроксимацией функции, основанной на наискорейшем спуске, надежно сходятся к решению со среднеквадратичной ошибкой, ограниченной минимально возможной оценкой, умноженной на коэффициент

$(1 - \gamma\lambda)/(1 - \gamma)$. Самонастраивающиеся методы с разделенной оценкой ценности стратегий, в свою очередь, могут приводить к бесконечной ошибке. Было предложено несколько подходов к решению проблемы, связанной с работой самонастраивающихся методов с разделенной оценкой ценности стратегий совместно с аппроксимацией функций, однако данный вопрос все еще остается открытым. Несмотря на теоретически ограниченные гарантии корректного функционирования, самонастраивающиеся методы в обучении с подкреплением вызывают неослабевающий интерес, поскольку на практике они обычно работают значительно лучше несамонастраивающихся методов.

8.8. Библиографические и исторические справки

Несмотря на то что в данной книге обобщение и аппроксимация рассматриваются ближе к концу книги, они всегда были неотъемлемой частью обучения с подкреплением. Только в последние десять или даже меньше лет исследования в данной области в основном концентрировались на табличном случае, что отражено в первых семи главах. В книге [Bertsekas and Tsitsiklis (1996)] заложены основы современного положения дел с аппроксимацией функций в обучении с подкреплением. Кроме того, интерес представляют статьи [Boyan, Moore and Sutton (1995)]. Некоторые ранние исследования по аппроксимации функций в обучении с подкреплением рассматриваются в конце данного раздела.

- 8.2. Методы наискорейшего спуска для минимизации среднеквадратичной ошибки в обучении с учителем широко известны. В статье [Widrow and Hoff (1960)] построен алгоритм, основанный на методе наименьших квадратов, представляющий собой первоначальный вариант пошагового метода наискорейшего спуска. Все это достаточно подробно, включая рассмотрение различных родственных алгоритмов, описано во множестве работ (см., например, [Widrow and Stearns (1985); Bishop (1995); Duda and Hart (1973)]). Анализ наискорейшего спуска в TD-обучении был проведен в работе [Sutton (1988)]. Методы более сложные, чем обычный наискорейший спуск, о которых упоминалось в данном разделе, также изучались в контексте обучения с подкреплением, например, квазиньютоновские методы [Werbos (1990)] и рекурсивные методы наименьших квадратов [Bradtko (1993, 1994); Bradtko and Barto (1996); Bradtko, Ydstie, and Barto (1994)]. В книге [Bertsekas and Tsitsiklis (1996)] проведено детальное исследование таких методов. Самым ранним применением объединения состояний в обучении с подкреплением является, по-видимому, система BOXES, описанная в статье [Michie and Chambers (1968)]. Теория объединения состояний в обучении с подкреплением была изложена в работах [Singh, Jaakkola and Jordan (1995)] и [Tsitsiklis and Van Roy (1996)].

8.3. TD(λ) алгоритм с линейной аппроксимацией функций, основанной на наискорейшем спуске, был впервые исследован в работах [Sutton (1984, 1988)], где доказана сходимость TD(0) в среднем к решению с минимальной СКО для случая, когда векторы параметров $\{\vec{\phi}: s \in S\}$ линейно независимы. Сходимость с вероятностью 1 для параметра λ общего вида была доказана несколькими исследователями приблизительно в одно время (см. [Peng (1993); Dayan and Sejnowski (1994); Tsitsiklis (1994); Gurvits, Lin and Hanson (1994)]). Кроме этого, в статье [Jaakkola, Jordan and Singh (1994)] доказана сходимость при оперативной корректировке. Все эти результаты предполагают линейно независимыми векторы свойств, которые содержат по крайней мере столько компонент вектора θ_t , сколько имеется состояний. Сходимость линейного TD(λ) в более интересном общем случае (с зависимыми векторами свойств) была показана в работе [Dayan (1992)]. Значительное обобщение и усиление этих результатов были осуществлены в статье [Tsitsiklis and Van Roy (1997a)]. Доказаны основные результаты, представленные в разд. 8.3, найдена граница асимптотической ошибки метода TD(λ) и других самонастраивающихся методов. Недавно результаты этого анализа были также применены к случаю неприведенной продолжающейся задачи [Tsitsiklis and Van Roy (1997b)].

Представленный в данной книге ряд возможностей линейной аппроксимации функции базируется на статье [Barto (1990)]. Термин «грубое кодирование» введен в работе [Hinton (1984)], из которой взята основа рис. 8.2. В статье [Waltz and Fu (1965)] предложен один из первых примеров для данного типа аппроксимации функций в системах обучения с подкреплением.

Мозаичное кодирование, в том числе и хеширование, было предложено в работах Альбуса [Albus (1971, 1981)]. Он описал предложенный подход в виде модели CMAC (Cerebellar Model Articulation Controller), которая интерпретировалась как модель мозжечка. Под таким названием мозаичное кодирование чаще всего встречается в литературе. Термин «мозаичное кодирование» впервые использован в данной книге, хотя идея описания CMAC в подобных терминах позаимствована в диссертации [Watkins (1989)]. Мозаичное кодирование применяется во многих системах обучения с подкреплением [Shewchuk and Dean (1990); Lin and Kim (1991); Miller, Scalera and Kim (1994); Sofge and White (1992); Tham (1994); Sutton (1996); Watkins (1989)], так же как и в других обучающихся системах управления [Kraft and Campagna (1990); Kraft, Miller and Dietz (1992)]. После того как в нейронных сетях были применены аппроксимации функций, использующие радиально-базисную функцию (РБФ) [Broomhead and Lowe (1988)], они стали вызывать повышенный интерес. В работе [Powell (1987)] рассмотрены более ранние применения РБФ, а в статьях [Poggio and Girosi (1989, 1990)] данный подход был значительно усовершенствован. То, что называлось выше Канерва-кодированием, было предложено в книге [Kanerva (1988)] как часть более общей идеи *мозаично распределенной памяти*. По-

дробнее данная идея, а также родственные модели памяти рассмотрены в статье [Kanerva (1993)]. Среди прочих данный подход был исследован также в работах [Gallant (1993)] и [Sutton and Whitehead (1993)].

- 8.4. Метод $Q(\lambda)$ с аппроксимацией функций впервые был изучен в диссертации [Watkins (1989)]. Метод SARSA(λ) с аппроксимацией функций первоначально исследовался в работе [Rummery and Niranjan (1994)]. Пример с машиной на горе основан на аналогичной задаче, которая исследовалась в диссертации [Moore (1990)]. Представленные в данной книге результаты решения данной задачи взяты из работ [Sutton (1996)] и [Singh and Sutton (1996)]. Сходимость методов управления, представленных в данном разделе, не доказывалась (более того, учитывая результаты, представленные в разд. 8.5, для $Q(\lambda)$ сходимость маловероятна). Результаты сходимости методов управления с объединением состояний и другими частными видами аппроксимации функций были доказаны в работах [Tsitsiklis and Van Roy (1996)], [Singh, Jaakkola and Jordan (1995)] и [Gordon (1995)].
- 8.5. Контрпример Бэрда предложен в статье [Baird (1995)]. Контрпримеры Цициклиса и Ван-Роя были предложены в статье [Tsitsiklis and Van Roy (1997a)]. Методы усреднения для аппроксимации функций были описаны в работах [Gordon (1995, 1996)]. Методы наискорейшего спуска для минимизации беллмановской ошибки предложены Бэрдом, который назвал их *методами остаточного градиента*. Другие примеры нестабильности при применении TD-методов с разделенной оценкой ценности стратегий и более сложных методов аппроксимации функций представлены в работе [Boyan and Moore (1995)]. В статье [Bradtko (1993)] приведен пример, в котором Q -обучение, использовавшее линейную аппроксимацию функции в линейно-квадратичной задаче регулирования, сходилось к дестабилизирующей стратегии.

Первые применения аппроксимации функций в обучении с подкреплением имели место в ранних работах по нейронным сетям (см. [Farley and Clark (1954); Clark and Farley (1955)]), где обучение с подкреплением использовалось для подстройки параметров линейных пороговых функций, через которые выражались стратегии. Самый ранний из известных примеров, в котором методы аппроксимации функций использовались для поиска функций ценности, является программа для игры в шашки, описанная в статьях [Samuel (1959, 1967)]. Сэмюэль использовал предположение из статьи Шеннона [Shannon (1950)] о том, что использующаяся для выбора ходов в игре функция ценности не обязательно должна быть точной, а может быть аппроксимирована линейной комбинацией свойств. Кроме линейной аппроксимации функций Сэмюэль экспериментировал с таблицами поиска и с иерархическими таблицами поиска, называемыми сигнатурными таблицами [Griffith (1966, 1974); Page (1977); Biermann, Fairfield and Beres (1982)].

Примерно в одно время с опубликованием работ Сэмюэля в статье Беллмана и Дрейфуса [Bellman and Dreyfus (1959)] было предложено использовать

метод аппроксимации совместно с ДП. (Естественным выглядит предположение о том, что Беллман и Сэмюэль оказывали друг на друга какое-то влияние, однако в их работах подтверждения этому отсутствуют.) В настоящее время существует действительно большое количество литературы по методам аппроксимации функции и ДП, таким как многосеточные методы и методы, использующие сплайны и ортогональные полиномы [Bellman and Dreyfus (1959); Bellman, Kalaba and Kotkin (1973); Daniel (1976); Whitt (1978); Reetz (1977); Schweitzer and Seidmann (1985); Chow and Tsitsiklis (1991); Kushner and Dupuis (1992); Rust (1996)].

В системе классификаторов из статьи Холланда [Holland (1986)] использована избирательная методика сопоставления свойств для обобщения оценочной информации по парам состояния—действие. Каждый классификатор, имея заданные ценности для подмножества свойств, сопоставлял подмножество состояний с оставшимися свойствами, имеющими случайные ценности («случайное событие»). Эти подмножества затем использовались, согласно традиционному подходу объединения состояний, для аппроксимации функции. Идея Холланда заключалась в том, чтобы использовать формирующий алгоритм для расширения множества классификаторов, которые совместно реализуют приемлемую функцию ценности состояния. Идеи Холланда повлияли на авторов ранних исследований в обучении с подкреплением, но в данной книге основное внимание уделяется другим подходам к аппроксимации функций. Так же как и аппроксиматоры функций, классификаторы имеют несколько ограничений. Во-первых, они относятся к методам объединения состояний с сопутствующими ограничениями на масштабирование и эффективное представление гладких функций. Кроме того, правила согласования для классификаторов могут реализовывать границы объединения, только параллельные осям свойств. Вероятно, самое главное ограничение традиционных систем классификаторов заключается в том, что они обучаются при помощи формирующего алгоритма, т. е. эволюционным методом. Как уже отмечалось в гл. 1, в процессе обучения доступно намного больше подробной информации о том, как обучаться, чем может быть учтено в эволюционных методах. Все это привело нас к необходимости адаптировать методы обучения с учителем к обучению с подкреплением, особенно методы наискорейшего спуска и нейронные сети. Данные различия между подходом Холланда и подходом, описанным в данной книге, не должны вызывать удивления, так как идеи Холланда разрабатывались в период, когда нейронные сети считались слишком слабым инструментом с точки зрения вычислительных возможностей, тогда как работа авторов этой книги происходила в период, когда данная общепринятая точка зрения уже не казалась столь уж очевидно верной. Остается также много возможностей для комбинирования идей этих двух различных подходов.

Необходимо отметить некоторые не обсуждавшиеся ранее исследования обучения с подкреплением, использующие методы аппроксимации функ-

ций. В статьях [Barto, Sutton and Brouwer (1981)] и [Barto and Sutton (1981b)] идея сети ассоциативной памяти из работ [Kohonen (1977); Anderson, Silverstein, Ritz and Jones (1977)] была распространена на обучение с подкреплением. В работах [Hampson (1983, 1989)] предлагается применять многослойные нейронные сети для нахождения функций ценности. В работах Андерсона [Anderson (1986, 1987)] TD-алгоритм соединен с алгоритмом обратного распространения ошибки для нахождения функции ценности. В статье [Barto and Anandan (1985)] предлагается стохастическая версия алгоритма избирательной самонастройки из статьи [Widrow, Gupta and Maitra (1973)], где он назван алгоритмом ассоциативных вознаграждений-штрафов (A_{P-R}). В работах Уильямса [Williams (1986, 1987, 1988, 1992)] данный тип алгоритмов расширен до общего класса REINFORCE алгоритмов подкрепления и показано, что они демонстрируют стохастическое увеличение градиента в предполагаемом подкреплении. В статьях Уильямса и в [Gullapalli (1990)] разработаны алгоритмы нахождения обобщающих стратегий для случая с непрерывными действиями. В статье [Phansalkar and Thathachar (1995)] доказана теорема как о локальной, так и о глобальной сходимости модифицированной версии алгоритмов подкрепления REINFORCE. В работе [Christensen and Korf (1986)] описаны эксперименты с методами регрессии для модифицируемых коэффициентов линейных аппроксимаций функций ценности в игре в шахматы. В работах [Chapman and Kaelbling (1991)] и [Tan (1991)] адаптированы методы на основе дерева решений для функций ценности. Методы обучения, основывающиеся на объяснении, нетребовательные к ресурсам памяти, также были адаптированы применительно к задаче нахождения функций ценности (см. [Yee, Saxena, Utgoff and Barto (1990); Dietterich and Flann (1995)]).

9 Планирование и обучение

В данной главе с единой точки зрения будут рассмотрены методы, которые требуют наличия модели окружающей среды, такие как динамическое программирование и эвристический поиск, а также методы, которые можно применять без модели, такие как метод Монте-Карло и методы временных различий. Первые методы будем называть методами *планирования*, а вторые — методами *обучения*. Несмотря на то что между этими методами существуют значительные различия, существует также и значительное сходство. В частности, суть каждого из этих методов заключается в нахождении функции ценности. Более того, в основе всех этих методов лежит один принцип: обзор будущих событий, расчет дублированной ценности и дальнейшее ее использование для корректировки приближенной функции ценности. Ранее в этой книге были описаны методы Монте-Карло и методы временных различий в качестве альтернативы друг другу, а затем было показано, как они могут плавно переходить один в другой при использовании следов приемлемости, как, например, в методе $\text{TD}(\lambda)$. Целью данной главы является аналогичное объединение методов планирования и обучения. В предыдущих главах данные методы были охарактеризованы как отличные друг от друга, рассмотрим теперь, каким образом можно их объединить.

9.1. Модели и планирование

Под *моделью* окружающей среды понимается все, что может быть использовано агентом для прогнозирования реакции окружающей среды на его действия. При заданном состоянии и действии модель выдает прогноз относительно следующего состояния и вознаграждения. Если модель стохастическая, то существует несколько возможных следующих состояний и вознаграждений, имеющих одинаковую вероятность возникновения. Некоторые модели выдают описание всех возможностей и их вероятностей; все они называются *моделями распределения*. Другие модели выдают только одну из всех возможностей, выбранную согласно вероятности; такие модели называются *моделями выборки*. Например, рассмотрим моделирование суммы очков, выпавших на десяти иг-

ральных костях. Модель распределения будет выдавать все возможные суммы и вероятности их выпадения, тогда как модель выборки будет выдавать отдельно взятую выпавшую сумму в соответствии с данным распределением вероятности. Тип модели, используемой в динамическом программировании, — расчеты вероятностей переходов состояний и ожидаемые вознаграждения $\mathcal{P}_{ss'}^a$ и $\mathcal{R}_{ss'}^a$ — это модель распределения. Модели распределения более универсальны, чем модели выборки, в том смысле, что всегда могут быть использованы для получения выборки. Тем не менее в большом количестве случаев значительно проще использовать модели выборки, нежели модели распределения.

Модели могут использоваться для имитации опыта. При заданных начальном состоянии и действии модель выборки выдает возможный переход, а модель распределения генерирует все возможные переходы, взвешенные в соответствии с вероятностью их появления. При заданном начальном состоянии и заданной стратегии модель выборки может выдать весь эпизод, а модель распределения может сформировать все возможные эпизоды и их вероятности. В обоих случаях говорят, что модель используется для *имитации* окружающей среды и выработки *имитируемого опыта*.

Слово «планирование» имеет различные значения в различных областях. Здесь оно применяется для обозначения любого вычислительного процесса, который использует модель в качестве источника входных данных и выдает или совершенствует стратегию взаимодействия с моделируемой окружающей средой:

$$\text{модель} \xrightarrow{\text{планирование}} \text{стратегия}.$$

В области искусственного интеллекта существуют два различных подхода к планированию в соответствии с данным определением. В *планировании в пространстве состояний*, когда применяется описанный в данной книге подход, планирование рассматривается в основном в качестве средства поиска оптимальной стратегии или пути к цели в пространстве состояний. Действия вызывают переходы из состояния в состояние, а функции ценности вычисляются, исходя из полученных состояний. В том, что называется *планированием в пространстве планов*, планирование рассматривается как поиск в пространстве планов. С помощью операторов один план преобразуется в другой, а функции ценности, если таковые существуют, определяются в пространстве планов. Планирование в пространстве планов включает в себя поэтапные методы и *планирование частичного порядка*, распространенный в области искусственного интеллекта вид планирования, когда порядок шагов

на всех этапах планирования определен не полностью. Методы планирования в пространстве планов не подходят для эффективного решения задач управления со стохастическим оптимумом, на которых сфокусировано основное внимание в обучении с подкреплением и не будут рассматриваться нами в дальнейшем (см., однако, разд. 11.6, в котором идет речь о применении планирования в пространстве планов в обучении с подкреплением).

Описываемая в данной главе единая точка зрения предполагает, что все методы планирования в пространстве состояний имеют одинаковую структуру, присущую представленным в данной книге методам обучения. Мы будем придерживаться этой точки зрения в оставшейся части данной главы, развивая при этом две основные идеи:

- (1) все методы планирования в пространстве состояний включают в себя вычисляемые функции ценности в качестве ключевого промежуточного шага по направлению к улучшению стратегии, и
- (2) они рассчитывают соответствующие функции ценности с помощью операций дублирования, применяемых к имитируемому опыту.

Данная общая структура может быть схематично представлена следующим образом:



Очевидно, что методы динамического программирования соответствуют данной структуре: они выполняют прогонку в пространстве состояний, генерируя для каждого состояния распределение возможных переходов. Затем каждое распределение используется для расчета дублированного значения ценности и для корректировки оценки ценности состояния. В данной главе будет показано, что множество других методов планирования в пространстве состояний также соответствуют данной структуре, причем отдельные методы отличаются только типом выполняемого дублирования, порядком его выполнения и тем сроком, в течение которого хранится дублированная информация.

Такой подход к методам планирования подчеркивает их связь с методами обучения, которые рассматриваются в данной книге. В основе обоих типов методов лежит оценивание функции ценности с помощью операции дублирования. Различие заключается в следующем: в то время как при планировании используется имитируемый моделью опыт, методы обучения используют реальный опыт, выработанный при взаимодействии с окружающей средой. Очевидно, что данное различие рождает ряд других различий, например, в том, как оценивается эф-

Выполнять циклически:

1. Выбрать случайным образом состояние $s \in \mathcal{S}$ и действие $a \in \mathcal{A}(s)$
2. Передать s, a в модель выборки, получить следующее выборочное состояние s' и вознаграждение r
3. Применить одношаговое табличное Q -обучение к s, a, s', r :

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

Рис. 9.1. Одношаговое табличное Q -планирование со случайной выборкой

фективность и насколько гибко может генерироваться опыт. Тем не менее, наличие общей структуры означает, что многие идеи и алгоритмы могут переноситься с обучения на планирование и наоборот. В частности, во многих случаях алгоритм обучения может быть заменен ключевым шагом дублирования метода планирования. Методы обучения требуют лишь наличие опыта в качестве входных данных, и во многих случаях наряду с реальным опытом в них может быть использован имитируемый опыт. На рис. 9.1 показан простой пример метода планирования, в основе которого лежит одношаговое табличное Q -обучение и случайные выборки из модели выборки. Данный метод, который называется *одношаговым табличным Q -планированием со случайной выборкой*, сходится к оптимальной стратегии для модели при тех же условиях, при которых одношаговое табличное Q -обучение сходится к оптимальной стратегии для реальной окружающей среды (каждая пара состояние—действие должна выбираться бесконечное число раз на шаге 1, а параметр α должен убывать соответствующим образом во времени).

Кроме выработки единого подхода к методам планирования и обучения в данной главе будет рассмотрено преимущество планирования при помощи небольших шагов. Такой способ позволяет прерывать планирование и изменять его направление в любое время с небольшими потерями на вычисления, что является ключевым требованием к эффективности сочетания планирования с исполнением и с изучением модели. Более того, позднее в данной главе будет приведено доказательство того, что планирование при помощи очень маленьких шагов может являться наиболее эффективным подходом даже при решении задач, связанных исключительно с планированием, если эти задачи слишком велики для точного решения.

9.2. Объединение планирования, исполнения и обучения

Когда планирование осуществляется оперативно, в процессе взаимодействия с окружающей средой, возникает ряд интересных обстоятельств. Полученная в результате взаимодействия новая информация может менять модель и, таким образом, взаимодействовать с планированием. Иногда возникает необходимость подстраивать некоторым образом процесс планирования под анализируемые в данный момент или ожидаемые в будущем состояния или решения. Если процесс принятия решений так же, как и процесс изучения модели, требует больших вычислительных мощностей, то может возникнуть необходимость разделения вычислительных ресурсов между ними. Перед тем как приступить к изучению данных вопросов, в данном разделе мы рассмотрим Dyna-Q, простую структуру, объединяющую основные функции, необходимые агенту оперативного планирования. Все функции в Dyna-Q имеют простую, почти тривиальную форму. В последующих разделах мы подробно рассмотрим некоторые альтернативные способы получения каждой функции и соотношений между ними. На данный момент основной задачей является лишь проиллюстрировать обсуждаемые идеи и стимулировать их интуитивное восприятие.

Для агента планирования реальный опыт выполняет, по крайней мере, две функции: он может использоваться для совершенствования модели (добиваясь более точного ее соответствия реальной окружающей среде), а также для непосредственного улучшения функции ценности и стратегии, применяя те виды методов обучения с подкреплением, которые были рассмотрены нами в предыдущих главах. Первая называется *изучением модели*, вторая называется *прямым обучением с подкреплением* (прямым ОП). Возможные взаимосвязи между опытом, моделью, ценностями и стратегией приведены на рис. 9.2. Каждая стрелка показывает взаимосвязь влияния и предполагаемого улучшения. Заметьте, что опыт может совершенствовать функцию ценности и стратегию как непосредственно, так и с помощью модели. Последнее, называемое иногда *косвенным обучением с подкреплением*, является составной частью планирования.

Как прямые, так и косвенные методы имеют свои преимущества и недостатки. Косвенные методы обычно в большем объеме используют ограниченный объем опыта и, таким образом, достигают лучшей стратегии при меньших взаимодействиях с окружающей средой. Прямые методы, в свою очередь, намного проще и не подвержены влиянию



Рис. 9.2. Взаимосвязи между изучением, планированием и исполнением

ошибок, присутствующих в модели окружающей среды. Одни исследователи утверждают, что косвенные методы всегда превосходят прямые, а другие — что на прямых методах основывается все обучение людей и животных. Аналогичные споры в области психологии и искусственного интеллекта касаются противопоставления процесса познания обучению методом проб и ошибок, а также противопоставления тщательного планирования и принятия решений, рассматриваемых как реакции на текущие изменения. На наш взгляд, в таких спорах различия между этими альтернативами всегда преувеличиваются и большего понимания можно добиться, если сосредоточиться на сходствах данных методов, нежели на их различиях. Например, в данной книге подчеркивается глубокое сходство между динамическим программированием и методами временных различий, даже несмотря на то, что первый был разработан для планирования, а второй для обучения без использования модели.

Структура Dyna-Q включает в себя все показанные на рис. 9.2 непрерывно возникающие процессы — планирование, исполнение, изучение модели и прямое обучение с подкреплением (ОП). Метод планирования представляет собой метод одношагового табличного *Q*-планирования со случайной выборкой, показанный на рис. 9.1. Метод прямого ОП является одношаговым табличным *Q*-обучением. Метод изучения модели также является по сути табличным и предполагает, что все изменения происходят детерминированно. После каждого перехода $s_t, a_t \rightarrow s_{t+1}, r_{t+1}$ модель регистрирует в элементе таблицы, соответствующем паре s_t, a_t , прогноз, что за ней детерминированно следует пара s_{t+1}, r_{t+1} . Таким образом, если модель получает запрос о паре состояние—действие, которая уже наблюдалась ранее, то в качестве прогноза она просто возвращает наблюдавшееся последующее состояние и последующее вознаграждение. Во время планирования алгоритм *Q*-обучения осуществляет случайные выборки только среди пар



Рис. 9.3. Общая структура Дуна-агента

состояние—действие, которые уже наблюдались ранее (на шаге 1), таким образом, модель никогда не получает запросов о парах состояние—действие, относительно которых у нее нет информации.

Общая структура Dyna-агента, одним из примеров которой является алгоритм Dyna-Q, показана на рис. 9.3. Центральная колонка отображает основное взаимодействие между агентом и окружающей средой, результатом которого является траектория реального опыта. Стрелкой слева на рисунке представлено прямое обучение с подкреплением, использующее реальный опыт для улучшения функции ценности и стратегии. Справа представлены моделируемые процессы. Модель строится на реальном опыте и порождает имитируемый опыт. Под термином *управление поиском* следует понимать процесс, который выбирает начальные состояния и действия для имитируемого опыта, сгенерированного моделью. В конечном счете, планирование осуществляется путем применения методов обучения с подкреплением к имитируемому опыту так, как будто этот опыт действительно имел место. Обычно, как и в Dyna-Q, один и тот же метод обучения с подкреплением используется как для обучения на основе реального опыта, так и для планирования на основе имитируемого опыта. Таким образом, метод обучения с подкреплением является «общим инструментом» обучения и планирования. Обучение и планирование глубоко интегрированы в том смысле, что они используют практически одни и те же механизмы, различаясь только источником получения опыта.

Инициализировать $Q(s, a)$ и $Model(s, a)$ для всех $s \in \mathcal{S}$ и $a \in \mathcal{A}(s)$

Выполнять циклически:

- (а) $s \leftarrow$ текущее (нетерминальное) состояние
- (б) $a \leftarrow \varepsilon$ -жадное (s, Q)
- (в) Выполнить действие a , найти результирующее состояние s' и вознаграждение r
- (г) $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
- (д) $Model(s, a) \leftarrow s', r$ (предполагая детерминированную среду)
- (е) Повторить N раз:
 - $s \leftarrow$ случайное предыдущее состояние
 - $a \leftarrow$ случайное предыдущее действие для s
 - $s', r \leftarrow Model(s, a)$
 - $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

Рис. 9.4. Алгоритм Dyna-Q. $Model(s, a)$ обозначает содержание модели (прогнозируемое состояние и вознаграждение) для пары состояния—действие s, a . Прямое обучение с подкреплением, изучение модели и планирования осуществляются на шагах (d), (e) и (f) соответственно. Если пропустить шаги (e) и (f), то получится алгоритм N -шагового табличного Q -обучения

По сути, планирование, исполнение, изучение модели и прямое ОП возникают в Dyna-агентах одновременно и параллельно. Тем не менее, чтобы иметь возможность реализовать данные процессы на последовательном компьютере, приходится полностью указывать порядок их возникновения в течение временного шага. В Dyna-Q процессы исполнения, изучения модели и прямого ОП требуют небольших вычислительных ресурсов, и предполагается, что на это затрачивается лишь незначительное время. Оставшееся время на каждом шаге может быть потрачено на процесс планирования, который требует значительных вычислительных ресурсов. Предположим, что на каждом шаге после исполнения, изучения модели и прямого ОП остается время для совершения N итераций (шаги 1–3) алгоритма Q -планирования. На рис. 9.4 полностью представлен алгоритм Dyna-Q.

Пример 9.1 (Лабиринт Dyna). Рассмотрим простой лабиринт, показанный на рис. 9.5 справа вверху. В каждом из 46 состояний возможны четыре действия: *up* (вверх), *down* (вниз), *right* (вправо) и *left* (влево), которые по отдельности выбираются агентом в направлении соответствующего соседнего состояния, за исключением ситуации, когда движение блокировано преградой или границей лабиринта, в этом случае агент остается на прежнем месте. Вознаграждение равно нулю за все перемещения, за исключением перемещения, в результате которого достигается

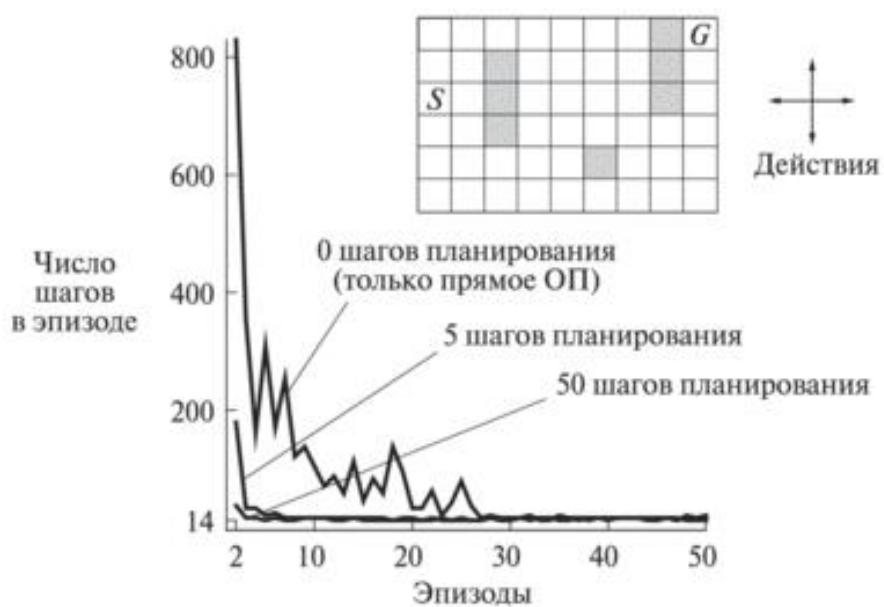


Рис. 9.5. Простой лабиринт (на вставке вверху) и усредненные кривые обучения агентов Dyna-Q с различным количеством шагов планирования за один реальный шаг. Задача — как можно быстрее пройти лабиринт из точки S в точку G

конечная цель, в этом случае вознаграждение равно +1. После достижения цели (G) агент возвращается в начальное состояние (S), чтобы начать новый эпизод. Это приведенная эпизодная задача с $\gamma = 0,95$.

В основной части рис. 9.5 изображены усредненные кривые обучения, полученные из эксперимента, в ходе которого в задаче с лабиринтом задействовался агент Dyna-Q. Начальные ценности действий были равны нулю, размер шага α равен 0,1, а параметр исследования ε равен 0,1. При жадном выборе среди одинаково предпочтительных действий одно выбиралось случайным образом. Агенты варьировали число шагов планирования N , которые они совершали за один реальный шаг. Для каждого N кривые показывают число шагов, предпринятых агентом в каждом эпизоде и усредненных по 30 повторениям эксперимента. При каждом повторении начальное число для генератора случайных чисел оставалось неизменным для всех алгоритмов. Вследствие этого первый эпизод был полностью одинаковым (около 1700 шагов) для всех значений N , поэтому его результаты не приводятся на рисунке. После первого эпизода результаты улучшаются при всех значениях N , однако особенно резко при больших значениях. Напомним, что при $N = 0$ агент не является планирующим агентом и использует только прямое обучение с подкреплением (одношаговое табличное Q -обучение). В данной

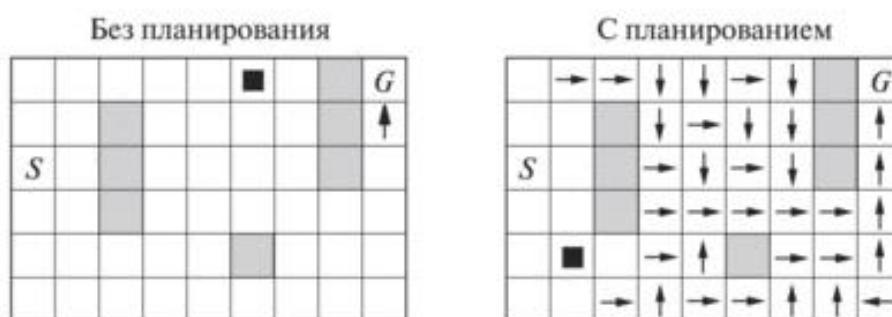


Рис. 9.6. Стратегии, найденные непланирующим и планирующими агентами Dyna-Q к середине второго эпизода. Стрелками обозначены жадные действия в каждом состоянии; в состоянии, в котором отсутствует стрелка, все действия эквивалентны. Чёрным квадратиком обозначено местонахождение агента

задаче такой агент был наиболее медленным, несмотря на то, что выбор значений параметров (α и ε) был оптимальным. Агенту, не использующему планирование, потребовалось около 25 эпизодов для достижения ε -оптимальных результатов, тогда как агенту с числом $N = 5$ потребовалось пять эпизодов, а агенту с числом $N = 50$ потребовалось только три эпизода.

Рисунок 9.6 демонстрирует, по какой причине агенты с планированием находили решение настолько быстрее, чем агент без планирования. На рисунке показаны стратегии, найденные агентами с числами $N = 0$ и $N = 50$ соответственно к середине второго эпизода. Без планирования ($N = 0$) каждый эпизод добавлял лишь один дополнительный шаг к стратегии, и, таким образом, изучался только один (последний) шаг. При планировании лишь один шаг изучался в течение первого эпизода, но уже в течение второго эпизода была выработана сложная стратегия, которая к концу эпизода почти достигнет начального состояния. Данная стратегия выработана процессом планирования, в то время как агент все еще находится около стартового состояния. К концу третьего эпизода будет найдена полностью оптимальная стратегия и будут получены наилучшие результаты.

В Dyna-Q обучение и планирование реализованы с помощью одного и того же алгоритма, который использует реальный опыт для обучения и имитируемый опыт для планирования. Так как планирование реализуется пошагово, планирование и исполнение соединить очень просто. Оба процесса происходят настолько быстро, насколько это позволяет их собственный алгоритм. Агент всегда готов скорректировать свое поведение и немедленно реагирует на поступающую с датчиков информацию, таким образом постоянно осуществляя фоновый процесс планирования.

Изучение модели также происходит в виде фонового процесса. При поступлении новой информации модель корректируется в сторону лучшего соответствия реальным условиям. С изменением модели происходящий процесс планирования постепенно вырабатывает новую линию поведения, соответствующую новой модели.

Упражнение 9.1. Непланирующий метод выглядит особенно плохо на рис. 9.6 вследствие того, что он является одношаговым; метод, использующий следы приемлемости, показал бы более удовлетворительные результаты. Как вы думаете, может ли метод со следами приемлемости показать такой же хороший результат, как и метод Dyna-Q? Аргументируйте свой ответ.

9.3. Когда модель неверна

В рассмотренном в предыдущем разделе примере с лабиринтом изменения модели были сравнительно небольшими. Вначале модель была пустая, а затем была заполнена достоверной информацией. В общем случае такого ожидать не приходится. Модели могут оказаться неверными вследствие того что либо окружающая среда имеет стохастический характер изменений и наблюдается лишь ограниченное число выборок, либо модель изучается с использованием аппроксимации функций, которая функционирует неидеально, или просто из-за того, что окружающая среда изменилась и ее новый характер еще не изучен. Если модель неверна, процесс планирования будет вырабатывать неоптимальную стратегию.

В некоторых ситуациях выработанная процессом планирования неоптимальная стратегия быстро приводит к обнаружению и исправлению ошибок моделирования. Это обычно происходит, когда модель слишком оптимистичная в том смысле, что прогнозирует завышенные вознаграждения и переходы состояний лучше, чем возможно при реальном эксперименте. Спланированная стратегия пытается воспользоваться данными возможностями и обнаруживает, что они не существуют.

Пример 9.2 (Заблокированный лабиринт). Пример лабиринта, иллюстрирующий данный сравнительно редкий вид ошибки моделирования и способ ее исправления, показан на рис. 9.7. Изначально существует короткий путь от старта к цели справа от барьера, как показано на верхнем левом рисунке. После 1000 временных шагов короткий путь «блокируется» и открывается более длинный путь слева от барьера, как по-

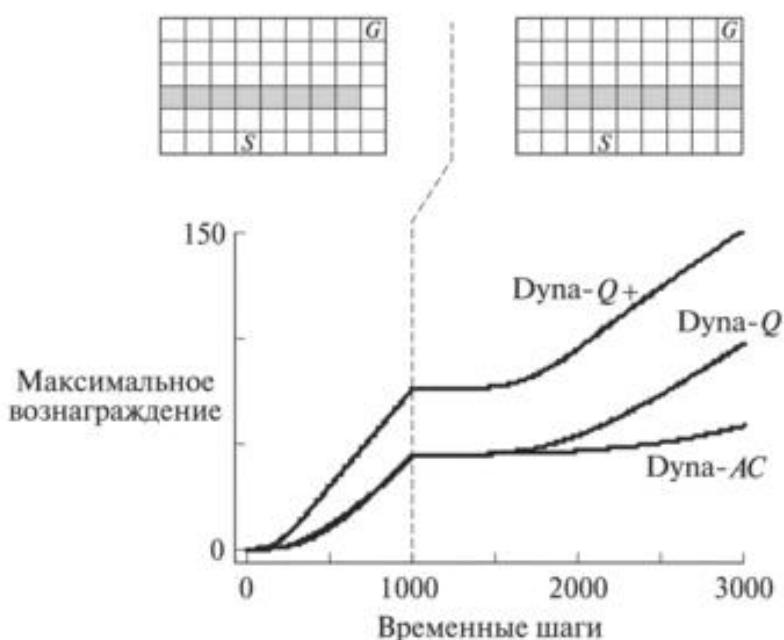


Рис. 9.7. Усредненные результаты Dyna-агентов в задаче с заблокированным лабиринтом. Левая окружающая среда использовалась в течение первых 1000 шагов, затем использовалась правая окружающая среда. Dyna-Q+ представляет собой агента Dyna-Q с исследовательским призом, который поощряет изучающие действия. Dyna-AC представляет собой Dyna-агента, который использует метод обучения исполнитель–критик вместо *Q*-обучения

казано на верхнем правом рисунке. На графике показано среднее накопленное вознаграждение для агента Dyna-Q и двух других Dyna-агентов. Из первой части графика видно, что все три Dyna-агента нашли короткий путь за 1000 шагов. Когда окружающая обстановка изменилась, на графике появилась горизонтальная часть, соответствующая периоду, в течение которого агенты не получали вознаграждения, так как они находились за барьером. Тем не менее после некоторого промежутка времени они смогли найти новый путь и новую оптимальную линию поведения.

Более серьезные трудности возникают, когда окружающая среда изменяется, улучшаясь, и используемая до этого верная стратегия не замечает данных улучшений. В такой ситуации, как видно из следующего примера, ошибка моделирования если и обнаруживается, то этот процесс может занять очень много времени.

Пример 9.3 (Короткий путь в лабиринте). Проблема, вызванная данным типом изменения окружающей среды, показана на примере с лабиринтом, изображенным на рис. 9.8. Изначально оптимальный путь проходит слева от барьера (верхний левый рисунок). Тем не менее по-

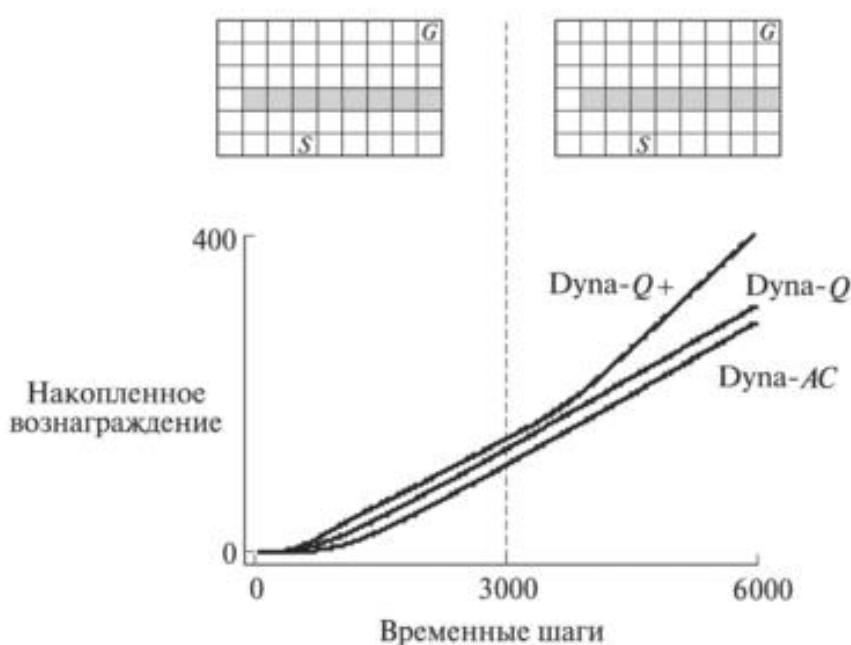


Рис. 9.8. Усредненные результаты для Дуна-агентов в задаче с коротким путем. Левая окружающая среда использовалась в течение первых 3000 шагов, затем использовалась правая окружающая среда

сле 300 шагов справа от барьера открывается более короткий путь, не закрывая при этом длинный (верхний правый рисунок). График показывает, что два из трех агентов так ни разу и не воспользовались коротким путем. По сути, они так и не поняли, что он существует. Соответствующие модели давали информацию, что короткого пути не существует; таким образом, чем дальше осуществлялся процесс планирования, тем меньше становилась вероятность, что кто-либо из данных агентов пойдет вправо и обнаружит короткий путь. Даже при ϵ -жадной стратегии существует очень небольшая вероятность того, что агент будет так много исследовать, что обнаружит короткий путь.

Общая проблема в данном случае заключается в очередной версии конфликта между изучением и применением. В контексте планирования изучение заключается в выборе действий, которые улучшают модель, тогда как применение означает реализацию оптимального поведения при заданной текущей модели. Необходимо, чтобы агент исследовал и находил изменения в окружающей среде, но не так интенсивно, чтобы результаты значительно ухудшались. Как и в предыдущем конфликте, исследование/применение здесь, вероятно, не существует одновременно верного и практического решения данной проблемы, однако простые эвристики обычно оказываются эффективными.

Агент Dyna-Q+, который нашел короткий путь в лабиринте, использует одну из таких эвристик. Для каждой пары состояния—действие этот агент отслеживает, какое количество временных шагов прошло с тех пор, как данная пара последний раз была использована во взаимодействии с окружающей средой. Чем больше прошло времени, тем больше (предположительно) шанс, что динамика данной пары изменилась и что ее модель уже некорректна. Чтобы стимулировать агента осуществлять долгое время не тестирующиеся действия, назначается специальное «бонусное вознаграждение» за имитируемый опыт, приводящий к данным действиям. В частности, если смоделированное вознаграждение за переход равно r , и переход не был осуществлен за n временных шагов, то дублирования планирования осуществляются, как если бы за данный переход полагалось вознаграждение $r + \kappa\sqrt{n}$, при некотором малом параметре κ . Это стимулирует агента испытывать все доступные переходы состояний и даже планировать длинные цепочки действий для того, чтобы осуществить данные испытания. Несомненно, все такие испытания имеют свою цену, но во многих случаях, например в задаче с лабиринтом, данный алгоритм полностью себя оправдывает.

Упражнение 9.2. По какой причине Dyna-агент с призом за исследования, Dyna-Q+, показывает лучшие результаты как в первой, так и во второй части экспериментов с блокированием и с коротким путем?

Упражнение 9.3. Если внимательно рассмотреть рис. 9.8, то можно заметить, что различия между агентами Dyna-Q+ и Dyna-Q слегка уменьшаются в первой части эксперимента. Чем это можно объяснить?

Упражнение 9.4 [Программирование]. Описанный выше приз за исследования, по сути, меняет расчетные значения состояний и действий. Необходимо ли это? Предположим, что приз $\kappa\sqrt{n}$ использовался не в дублировании, а исключительно при выборе действий, т. е. предположим, что всегда выбиралось то действие, для которого значение $Q(s, a) + \kappa\sqrt{n_{sa}}$ было максимальным. Проведите эксперимент с сеткой, который продемонстрирует слабые и сильные стороны такого подхода.

9.4. Приоритетная прогонка

В представленном в предыдущем разделе Dyna-агенте имитируемые переходы начинаются в парах состояния—действие, которые равномерно и случайно выбираются из всех уже испытанных пар. Однако равномерный выбор обычно не является лучшим; планирование может быть

значительно более эффективным, если имитируемые переходы и дублирования оперируют конкретными парами состояния—действие. Например, рассмотрим, что происходит в течение второго эпизода в задаче с лабиринтом (рис. 9.6). В начале этого эпизода только пара состояния—действие, которая непосредственно ведет к достижению цели, имеет положительную ценность; ценности всех остальных пар по-прежнему равны нулю. Это означает, что практически для всех переходов дублирование теряет смысл, так как все они просто переводят агента из одного состояния с нулевой ценностью в другое, таким образом, дублирование не оказывает никакого влияния. Лишь в случае с переходом в состояние, предшествующее достижению цели, или в случае с переходом в саму цель дублирование будет изменять какие-либо значения ценностей. Если имитируемые переходы генерируются равномерно, то, прежде чем будут осуществлены два данных дублирования, будет выполнено множество бесполезных операций дублирования. При дальнейшем планировании область полезных дублирований будет расти, однако результаты такого планирования будут значительно хуже, чем если бы оно было изначально сконцентрировано в тех областях, где оно наиболее эффективно. В очень больших задачах, которые в действительности являются основными объектами исследования, количество состояний настолько велико, что несфокусированное хаотичное изучение будет крайне неэффективно.

Данный пример показывает, что для получения хороших результатов изучение может быть сфокусировано на работе в *обратном направлении* по отношению к целевым состояниям. Конечно, нет необходимости придумывать метод специально под идею «целевого состояния». Необходимо иметь методы, работающие с общими функциями вознаграждения. Целевые состояния являются лишь частным случаем, помогающим лучше понять проблему. В общем случае необходим процесс, работающий в обратном направлении не только из целевого состояния, но из любого состояния, чья ценность была изменена. Предположим, что значения ценностей изначально скорректированы в соответствии с имеющейся моделью, как это было в примере с лабиринтом перед тем, как было найдено целевое состояние. Теперь предположим, что агент обнаружил изменение в окружающей среде и изменил расчетную ценность одного состояния. Обычно это означает, что ценности многих других состояний также должны быть изменены, однако какой бы то ни было эффект будут иметь дублирования лишь тех действий, которые непосредственно ведут в состояние с измененной ценностью. Если скорректированы ценности данных действий, то ценности предшествующих состояний, в свою

очередь, также могут измениться. В этом случае действия, приводящие к данным состояниям, должны подвергнуться дублированию, и, следовательно, предшествующие им состояния могут измениться. Таким образом, можно осуществлять обратный процесс из любого произвольного состояния, ценность которого была изменена, при этом либо осуществляя полезные дублирования, либо завершая данный процесс распространения.

Распространяясь в обратном направлении, процесс дублирования быстро расширяется, порождая множество пар состояния—действие, пригодных для дублирования. Однако дублирование не всех таких действий будет одинаково полезным. Ценности одних состояний могут изменяться значительно, в то время как ценности других состояний почти не меняются. Те пары, которые предшествуют сильно измененным парам действие—состояние, с высокой вероятностью сами будут значительно изменены. В стохастической окружающей среде колебания расчетных значений вероятностей переходов также вносят вклад в разброс величин изменений и в формирование порядка, согласно которому осуществляется дублирование пар. Обычно приоритеты при дублировании расставляются в соответствии с мерой их срочности и в дальнейшем осуществляют дублирование в порядке приоритета. Данная идея лежит в основе *приоритетной прогонки*. Формируется очередь, состоящая из всех пар состояния—действие, оценки ценностей которых существенно изменяются при дублировании. Приоритеты в данной очереди расставляются в соответствии с величиной изменения. После дублирования пары, находящейся во главе очереди, рассчитывается влияние данного действия на предшествующие пары. Если эффект данного влияния превышает некоторый порог, данная пара помещается в очередь с новым значением приоритета (если данная пара уже содержится в очереди, после осуществления дублирования значение ее приоритета повышается). Таким образом, влияние изменений эффективно распространяются по направлению от начала очереди до момента завершения процесса. Полный алгоритм для случая с детерминированной окружающей средой представлен на рис. 9.9.

Пример 9.4 (Приоритетная прогонка в задаче с лабиринтом). Было обнаружено, что приоритетная прогонка значительно увеличивает скорость процесса нахождения решений в задаче с лабиринтом, обычно от 5 до 10 раз. Типичный пример приведен на рис. 9.10. Указанные данные относятся к ряду задач с лабиринтом, имеющих структуру, аналогичную той, что показана на рис. 9.5, за исключением того,

Инициализировать $Q(s, a)$, $Model(s, a)$ для всех s, a , очистить $PQueue$. Выполнять циклически:

- (а) $s \leftarrow$ текущее (нетерминальное) состояние
- (б) $a \leftarrow$ стратегия (s, Q)
- (в) Выполнить действие a , найти результирующее состояние s' и вознаграждение r
- (г) $Model(s, a) \leftarrow s', r$
- (д) $p \leftarrow |r + \gamma \max_{a'} Q(s', a') - Q(s, a)|$
- (ж) Если $p > \theta$, то вставить s, a в $PQueue$ с приоритетом p
- (з) Повторить N раз, пока $PQueue$ не пуста:
 - $s, a \leftarrow$ первое($PQueue$)
 - $s', r \leftarrow Model(s, a)$
 - $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
 Повторить для всех \bar{s}, \bar{a} , предположительно ведущих к s :
 - $\bar{r} \leftarrow$ прогнозируемое вознаграждение
 - $p \leftarrow |\bar{r} + \gamma \max_a Q(s, a) - Q(\bar{s}, \bar{a})|$
 - Если $p > \theta$, то вставить \bar{s}, \bar{a} в $PQueue$ с приоритетом p

Рис. 9.9. Алгоритм приоритетной прогонки для случая с детерминированной окружающей средой

что задачи данного ряда имеют разный размер клеток сетки. Приоритетная прогонка имеет значительное преимущество над бесприоритетным методом Dyna-Q. Обеим системам требуется не больше чем

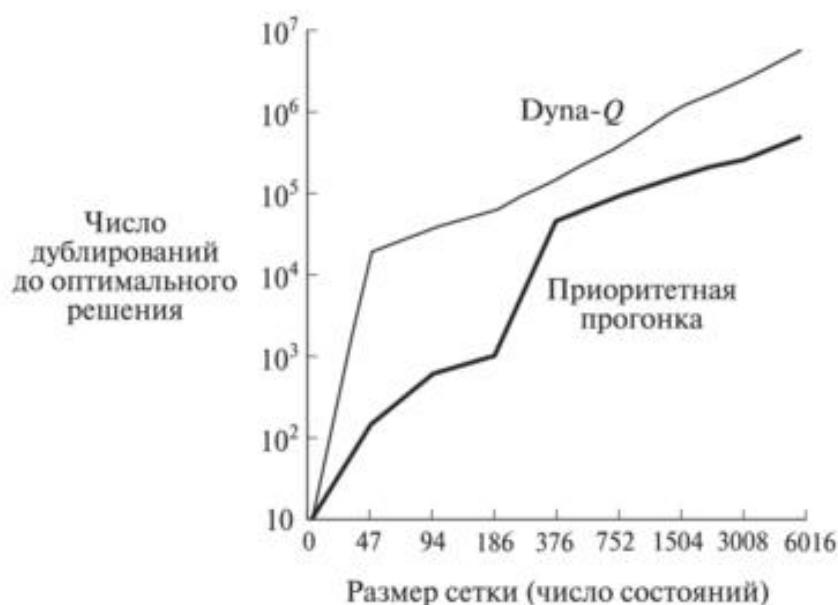


Рис. 9.10. Приоритетная прогонка значительно сокращает время обучения в задаче с Дуна-лабиринтом для сеток с различным размером клеток. Перепечатано с разрешения авторов из статьи [Peng and Williams (1993)]

$N = 5$ дублирований за один акт взаимодействия с окружающей средой.

Пример 9.5 (Управление перемещением стержня). Задача заключается в управлении перемещением стержня между произвольным образом расположенным барьерами по направлению к целевому положению за наименьшее число шагов (рис. 9.11). Стержень может перемещаться вдоль своей оси и перпендикулярно к ней; кроме этого, он может поворачиваться в любом направлении относительно своего центра. Длина каждого перемещения составляет примерно $1/20$ часть области рабочего пространства, а поворот осуществляется на 10 градусов. Перемещения стержня детерминированы и квантованы на 20×20 позиций. На рисунке изображены барьеры и кратчайший путь от старта до целевого положения, найденный приоритетной прогонкой. Данная задача является детерминированной, в ней 4 действия и 14 400 состояний (некоторые из них недостижимы из-за расположения барьеров). Данная зада-

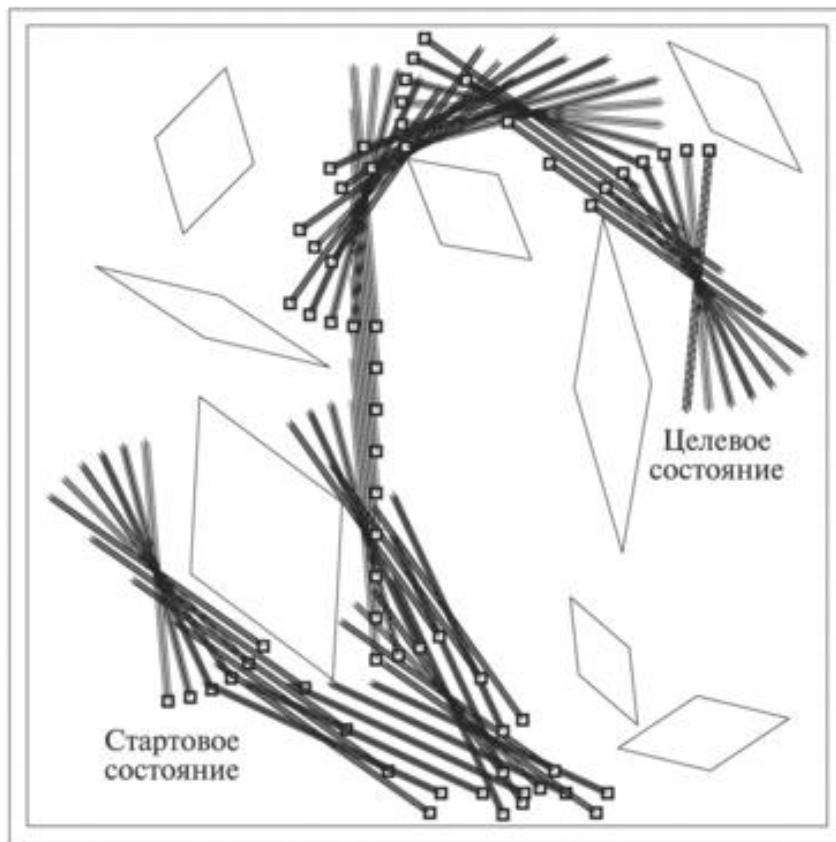


Рис. 9.11. Задача управления перемещением стержня и ее решение, найденное приоритетной прогонкой. Перепечатано с разрешения авторов из статьи [Moore and Atkeson (1993)]

ча, по-видимому, слишком велика, чтобы решать ее бесприоритетными методами.

Очевидно, что приоритетная прогонка — это очень эффективный метод, но оказывается, что при попытке применения имеющихся на данный момент алгоритмов в более интересных случаях возникают определенные проблемы. Главная проблема заключается в том, что в основе алгоритмов лежит предположение о дискретности состояний. При возникновении изменения в одном состоянии данные методы проводят расчеты всех предшествующих состояний, которые также могли подвергнуться изменению. Если для изучения модели или функции ценности используется аппроксимация функций, то единственное дублирование может затронуть значительное количество остальных состояний. При этом неясно, каким образом данные состояния будут распознаваться и обрабатываться. С другой стороны, общая идея сфокусировать исследования на состояниях, о которых можно предполагать, что их значения ценности были изменены, а затем переходить к предшествующим состояниям, в общем случае выглядит разумной. Дополнительное исследование может порождать более общие варианты метода приоритетной прогонки.

Почти очевидно, как можно распространить приоритетную прогонку на случай со стохастической окружающей средой. Функционирование модели поддерживается за счет фиксирования числа наблюдений каждой пары состояние—действие, а также информации о следующих состояниях. Наиболее разумным затем оказывается дублирование каждой пары при помощи не выборочного дублирования, как это делалось до сих пор, но при помощи полного дублирования, принимая в расчет все возможные следующие состояния и вероятности их возникновения.

9.5. Сравнение полного и выборочного вариантов дублирования

Примеры из предыдущих разделов приводят к идее упорядочить возможные сочетания методов планирования и обучения. В оставшейся части данной главы мы проанализируем некоторые детали этой идеи, начиная с относительных преимуществ полного и выборочного методов дублирования.

Большая часть данной книги посвящена различным видам дублирования, и к этому моменту уже рассмотрено достаточно много их раз-

новидностей. Рассмотрим одношаговые дублирования. Различия данных алгоритмов обусловлены в основном тремя бинарными факторами. Первые два фактора заключаются в том, выполняется ли дублирование ценностей состояний или действий и производится ли вычисление этих ценностей в соответствии с оптимальной стратегией или с произвольно заданной стратегией. Два данных фактора задают четыре класса дублирований для аппроксимации четырех функций ценности Q^* , V^* , Q^π и V^π . Оставшийся бинарный фактор заключается в том, является ли дублирование полным дублированием, которое учитывает все возможные события, или же *выборочным* дублированием, которое учитывает лишь некоторую выборку возможных событий. Три рассмотренных бинарных фактора задают восемь классов, семь из которых соответствуют специальным алгоритмам, как показано на рис. 9.12. (Восьмой случай не соответствует никакому полезному дублированию.)

Любое из этих одношаговых дублирований может использоваться в методах планирования. Рассмотренные ранее агенты Dyna-Q используют Q^* выборочные дублирования, однако они также могут использовать Q^* полные дублирования, а также выборочные или полные Q^π дублирования. Система Dyna-AC использует V^π выборочное дублирование совместно с обучающей стратегией. В случае со стохастическими задачами приоритетная прогонка всегда выполняется с использованием полного дублирования.

Когда в гл. 6 мы рассказывали об одношаговых выборочных дублированиях, были описаны возможные замены для всех полных дублирований. При отсутствии модели распределения полные дублирования невозможны, тем не менее выборочные дублирования могут быть выполнены с использованием выборочных переходов окружающей среды или выборочной модели. Подразумевалось, что если есть возможность, всегда предпочтительнее проводить полное дублирование, нежели выборочное. Так ли это на самом деле? Очевидно, что полное дублирование позволяет получать лучшие оценки, так как не подвержено влиянию ошибки выборки, однако оно требует большего объема расчетов, который всегда является ограниченным ресурсом в планировании. Чтобы адекватно оценить относительные преимущества полных и выборочных дублирований применительно к планированию, необходимо определить соответствующие требования к объему вычислений.

Для определенности, рассмотрим полные и выборочные дублирования для аппроксимации Q^* , а также частный случай дискретных состояний и действий, представленную в виде табличного поиска приближенную функцию ценности Q и модель в форме предполагаемых вероятно-

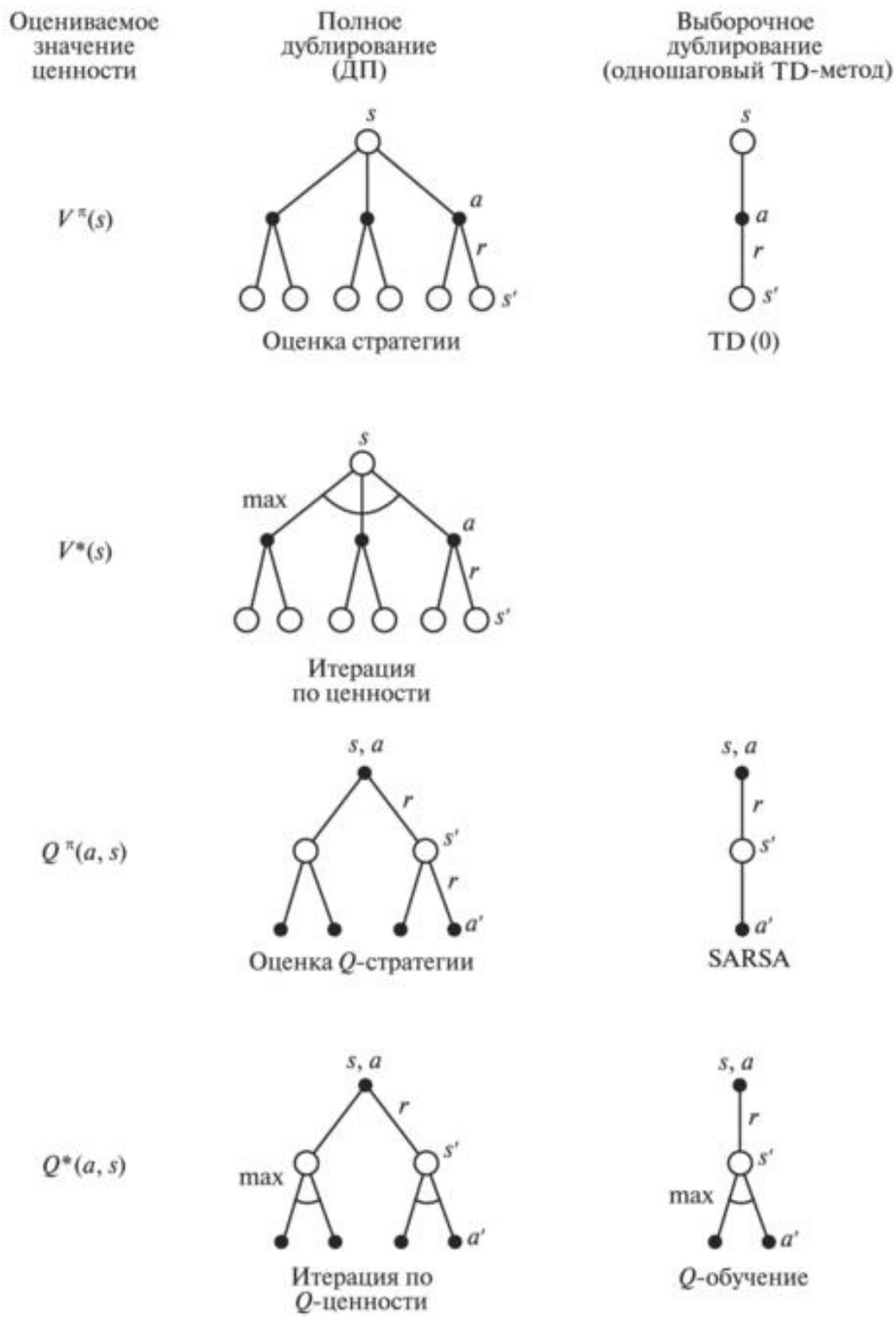


Рис. 9.12. Одношаговые варианты дублирования

стей смены состояний $\hat{P}_{ss'}^a$ и ожидаемых вознаграждений $\hat{R}_{ss'}^a$. Полное дублирование пары s, a состояние—действие имеет следующий вид:

$$Q(s, a) \leftarrow \sum_{s'} \hat{P}_{ss'}^a [\hat{R}_{ss'}^a + \gamma \max_{a'} Q(s', a')]. \quad (9.1)$$

Соответствующее выборочное дублирование s, a при заданном выборочном следующем состоянии s' представляет собой корректировку, подобную Q -обучению:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [\hat{R}_{ss'}^a + \gamma \max_{a'} Q(s', a') - Q(s, a)], \quad (9.2)$$

где α — чаще всего положительный размер шага, а ожидаемое значение вознаграждения $\hat{R}_{ss'}^a$, соответствующее данной модели, используется вместо выборочного вознаграждения, которое используется при применении Q -обучения при отсутствии модели.

Значительное различие между представленными таким образом полными и выборочными дублированиями обусловлено стохастичностью окружающей среды, а именно тем, что при заданных состояниях и действиях в дальнейшем может иметь место множество следующих состояний с различными вероятностями возникновения. Если возможно возникновение только одного следующего состояния, то рассмотренные выше полные и выборочные дублирования совпадают (при $\alpha = 1$). Если возможно возникновение нескольких следующих состояний, то между данными типами дублирований возникают значительные различия. В пользу полного дублирования говорит тот факт, что оно представляет собой точный расчет, результатом которого является новое значение $Q(s, a)$, точность которого ограничена лишь точностью значения $Q(s', a')$ в предыдущих состояниях. На выборочное дублирование, кроме этого, оказывает влияние ошибка выборки. С другой стороны, выборочное дублирование проще с точки зрения расчетов, так как оно учитывает только одно следующее состояние, а не все возможные. На практике, вычисления, требующиеся для дублирования, обычно зависят от числа пар состояния—действие, в которых оценивается функция Q . Пусть для стартовой пары (s, a) параметр b является коэффициентом ветвления, т. е. числом возможных следующих состояний s' , для которых $\hat{P}_{ss'}^a > 0$. Тогда полное дублирование данной пары потребует приблизительно в b раз больше вычислений, чем выборочное дублирование.

Если имеется достаточно времени для осуществления полного дублирования, то полученная оценка будет в общем случае лучше, чем оценка b выборочных дублирований, из-за отсутствия ошибки выборки. Однако если для завершения полного дублирования времени недостат-

точно, то всегда предпочтительнее использовать выборочные дублирования, так как они осуществляют, по крайней мере, некоторое улучшение расчетной оценки при общем их количестве, меньшем b . Такая ситуация часто встречается при решении больших задач с большим числом пар состояния—действие. При большом числе пар состояния—действие осуществление полных дублирований заняло бы слишком много времени. При этом может оказаться намного более эффективным применение нескольких дублирований ко многим парам состояния—действие, чем применение полных дублирований лишь к нескольким парам. Если затраты единицей затрат на вычисления, будут ли более эффективными несколько полных дублирований или в b раз большее количество выборочных дублирований?

На рис. 9.13 представлены результаты экспериментального анализа данного вопроса. На графике приведена зависимость ошибки оценки от времени проведения вычислений для полных и выборочных дублирований при различных значениях коэффициента ветвления b . В данном случае предполагалось, что все b последующих состояний равновероятны и что ошибка начальной оценки равна 1. Предполагается, что ценности следующих состояний определены правильно; таким образом, по завершении одного полного дублирования ошибка становится равной нулю. Уменьшение ошибки выборочного дублирования в этом случае происходит по закону $(1/\sqrt{t}) \cdot (b-1)/b$, где t — это число осуществленных выборочных дублирований (предполагается усреднение по выборке, т. е. $\alpha = 1/t$). Главное наблюдение заключается в том, что для умеренно

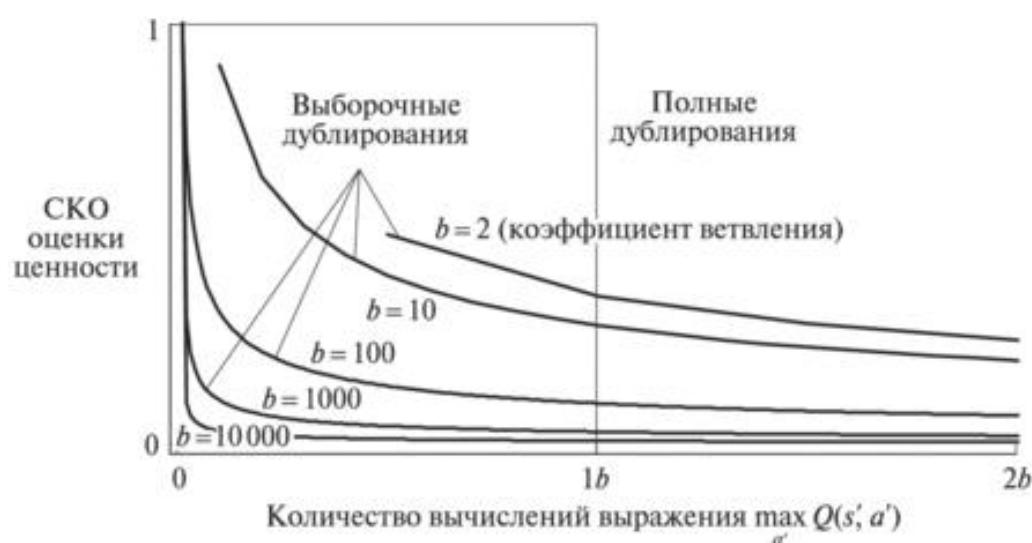


Рис. 9.13. Сравнение эффективности полных и выборочных дублирований

больших значений b ошибка спадает очень быстро уже при количестве выборочных дублирований, составляющем лишь малую часть b . В этих случаях многие пары состояния—действие могли бы значительно улучшить свои ценности, с различием в пределах нескольких процентов от результата, которое дает полное дублирование, за время, которое потребовалось бы на полное дублирование одной пары состояния—действие.

Показанное на рис. 9.13 преимущество выборочных дублирований в реальных условиях, возможно, будет еще более значительным. В реальных задачах ценности последующих состояний сами будут являться оценками, скорректированными при помощи дублирований. Делая оценки более точными, выборочные дублирования будут иметь второе преимущество, заключающееся в том, состоящее в том, что скопированные с последующих состояний ценности будут более точными. Данные результаты говорят о том, что выборочные дублирования предпочтительнее полных дублирований при решении задач с большим стохастическим фактором ветвления и содержащих большое число состояний, ценности которых должны быть определены с высокой степенью точности.

Упражнение 9.5. В проведенном выше анализе предполагалось, что все b возможных следующих состояний имели одинаковую вероятность возникновения. Предположим, что данное распределение было совсем не равномерным, и некоторые из b состояний имели значительно более высокую вероятность возникновения, чем большинство других состояний. Увеличит или уменьшит данная поправка преимущество выборочных дублирований над полными? Аргументируйте свой ответ.

9.6. Траекторная выборка

В данном разделе сравним два вида распределения дублирований. Согласно традиционному подходу, пришедшему из динамического программирования, прогонки осуществляются по всему пространству состояний (или состояний—действий), при этом дублирование каждого состояния (пары состояния—действие) выполняется один раз за прогонку. Это может вызвать проблемы при решении больших задач из-за нехватки времени для выполнения даже одной полной прогонки. Во многих задачах подавляющее большинство состояний не представляют собой никакого интереса, так как они либо возникают при очень плохой стратегии, либо имеют очень низкую вероятность возникновения. Исчерпывающая прогонка по определению уделяет одинаковое время всем частям пространства состояний, вместо того чтобы сфокусироваться только на областях пространства, в которых она действительно необходима. Как уже

обсуждалось в гл. 4, исчерпывающие прогонки и предлагаемые ими равновероятные рассмотрения всех состояний не являются обязательными свойствами динамического программирования. Теоретически прогонки могут иметь какое угодно распределение (для гарантии сходимости все состояния и пары состояния—действие в пределе должны посещаться бесконечное число раз), но на практике часто применяются именно исчерпывающие прогонки.

Согласно второму подходу, выборка из пространства состояний или пар состояния—действие производится в соответствии с некоторым распределением. Выборка может быть равномерной, как в случае с агентом Dyna-Q, однако это порождает те же проблемы, что и исчерпывающая прогонка. Практичесе выбирать дублирования в соответствии с распределением с интегрированной оценкой ценности стратегий, т. е. в соответствии с распределением, которое наблюдается при следовании текущей стратегии. Одним из преимуществ данного распределения является тот факт, что оно может быть быстро сгенерировано; необходимо просто взаимодействовать с моделью, придерживаясь текущей стратегии. В эпизодной задаче процесс начинается в стартовом состоянии (или в состоянии, соответствующем распределению стартовых состояний), и далее проводится имитация вплоть до завершающего состояния. В непрерывной задаче процесс начинается в любом месте, и далее происходит его имитация. В обоих случаях выборочные переходы состояний и вознаграждения задаются моделью, а выборочные действия определяются текущей стратегией. Другими словами, происходит имитация отдельных траекторий, и осуществляются дублирования всех встречающихся в процессе состояний или пар состояния—действие. Такой метод генерации опыта и дублирований называется *траекторной выборкой*.

Сложно представить себе более эффективный способ распределения дублирований, соответствующий распределению с интегрированной оценкой ценности стратегий, отличный от траекторной выборки. Если бы имелось точное представление распределения с интегрированной оценкой, то можно было бы осуществить прогонку по всем состояниям, взвесив дублирование каждого из них в соответствии с таким распределением. Однако при этом снова встает вопрос о затратах на расчеты, связанные с исчерпывающими прогонками. Вероятно, можно было бы выбирать и корректировать отдельные пары состояния—действие из распределения, но даже в том случае, если бы это удалось сделать достаточно эффективно, какие преимущества предоставил бы данный алгоритм относительно имитации траекторий? Кроме того, практически невозможно точно знать распределение с интегрированной оценкой ценности

стратегий. Распределение меняется, как только изменяется стратегия, и расчет такого распределения потребует вычислительных ресурсов, сравнимых с теми, которые требуются для полной оценки стратегии. Таким образом, рассмотрев альтернативные варианты, можно прийти к выводу, что траекторная выборка является наиболее эффективным и изящным методом.

Является ли распределение с интегрированной оценкой ценности стратегий хорошим? На первый взгляд кажется, что это хороший выбор, по крайней мере лучше, чем равномерное распределение. Например, если вы учитесь играть в шахматы, то изучаете позиции, которые могут возникнуть в реальной игре, а не случайные позиции шахматных фигур. Последние могут также являться интересными позициями, однако их оценка будет отличаться от оценки позиций в реальной игре. Кроме этого, из гл. 8 известно, что у распределения с интегрированной оценкой появляются значительные преимущества, когда используется аппроксимация функций. На данный момент только для этого распределения можно гарантировать сходимость при использовании общей линейной аппроксимации функций. Независимо от того, используется аппроксимация функций или нет, можно ожидать, что применение распределения с интегрированной оценкой значительно увеличит скорость планирования.

Использование распределения с интегрированной оценкой ценности стратегий может быть выгодным, так как при этом обширные, не представляющие интерес области пространства не рассматриваются. Однако использование данного распределения может оказаться и вредным, так как снова и снова приводит к дублированию одних и тех же частей пространства. Для оценки данного эффекта мы провели небольшой эксперимент. Для того чтобы выделить влияние дублирования распределения, были использованы одношаговые полные табличные дублирования, определенные согласно выражению (9.1). В *равномерном* случае дублированию подвергалась каждая пара состояния—действие, а в случае с интегрированной оценкой ценности стратегий имитировались эпизоды, а также выполнялось дублирование каждой пары состояния—действие, которая возникала при текущей ε -жадной стратегии ($\varepsilon = 0,1$). Рассматривались эпизодные и неприведенные задачи, произвольно сформированные следующим образом. В каждом из $|S|$ состояний были возможны два действия, каждое из которых приводило к одному из b (коэффициент ветвления) следующих равновероятных состояний с различным произвольным выбором b состояний для каждой пары состояния—действие. Кроме этого, каждый переход с вероятностью 0,1 вел в тер-

миимальное состояние, завершающее эпизод. Ожидаемое вознаграждение на каждом переходе выбиралось из гауссовского распределения со средним 0 и дисперсией 1. В любой момент процесс планирования можно остановить и полностью рассчитать $V^\pi(s_0)$, истинную ценность стартового состояния при жадной стратегии π , при заданной текущей функцией ценности действия Q как показатель того, насколько хорошо агент показал бы себя в новом эпизоде, в котором он действовал жадным образом (при этом подразумевается, что модель верна).

На рис. 9.14 вверху показаны результаты, усредненные по 200 выборочным задачам с 1000 состояниями и коэффициентом ветвления, равным 1, 3 и 10. Качество найденных стратегий представлено в виде функции от числа завершенных полных дублирований. Во всех случаях результатом выборки в соответствии с распределением с интегрированной оценкой ценности стратегий является изначально более быстрое планирование с последующим замедлением на длинной дистанции. При меньших коэффициентах ветвления эффект от данного распределения был сильнее, а период более быстрого планирования длился дольше. В ходе других экспериментов обнаружилось, что данный эффект усиливается с увеличением числа состояний. Например, в нижней части рисунка представлены результаты для коэффициента ветвления, равного 1, в задаче с 10 000 состояний. В данном случае преимущество использования интегрированной оценки ценности стратегий заметнее и длится дольше.

Появление таких результатов ожидаемо. В краткосрочном периоде выборка, соответствующая распределению с интегрированной оценкой ценности стратегий, помогает сфокусироваться на состояниях, которые наступают вскоре после стартового. Если имеется много состояний и коэффициент ветвления невелик, данный эффект будет значительным и долгосрочным. В длительной перспективе использование распределения с интегрированной оценкой может производить отрицательный эффект, так как часто возникающим состояниям уже присвоены их правильные ценности. Выбор таких состояний бесполезен, тогда как выбор других состояний может, вообще говоря, оказывать положительное действие. Вероятно, именно по этой причине исчерпывающий ни на чем не сфокусированный подход обеспечивает лучшие результаты в длительной перспективе, по крайней мере, при решении небольших задач. Данные результаты не являются окончательными, так как они соответствуют лишь задачам, сгенерированным конкретным случайным образом, однако указывают на то, что выборка, соответствующая распределению с интегрированной оценкой, может иметь большое преимущество при решении больших задач, в особенности задач, в которых при ис-

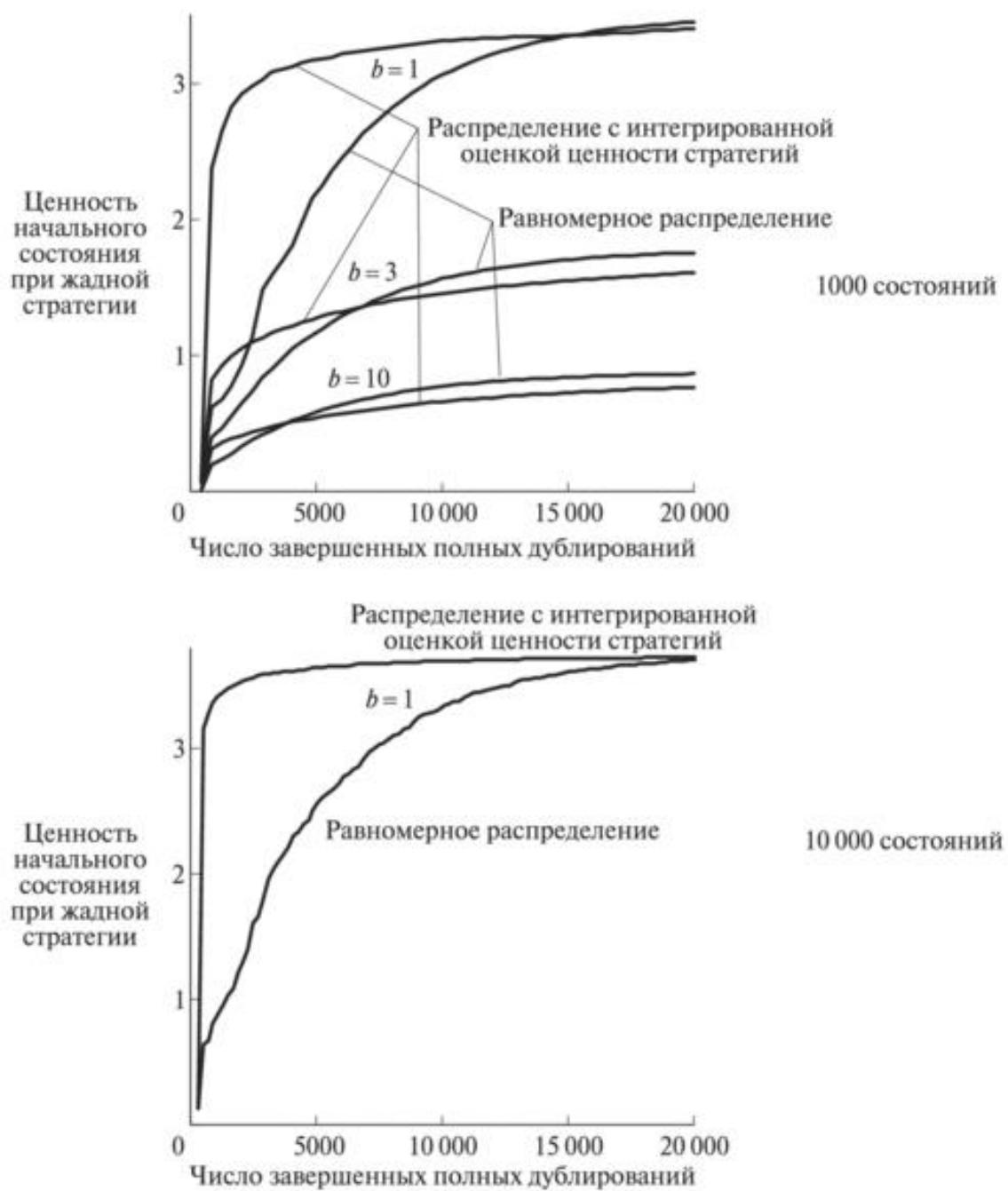


Рис. 9.14. Относительная эффективность дублирований, распределенных равномерно в пространстве состояний, в сравнении с результатами дублирований, использующих имитируемые траектории с интегрированной оценкой ценности стратегий. Результаты соответствуют случайно сгенерированным задачам двух различных размеров с различными коэффициентами ветвления b

пользовании такого распределения посещаются лишь небольшие подпространства в пространстве состояний—действий.

Упражнение 9.6. Некоторые графики на рис. 9.14 на ранней стадии имеют всплески и неровности, особенно верхний график, соответствующий случаю $b = 1$ и равномерному распределению. Как это можно объяснить? Какие из представленных результатов соответствуют вашей гипотезе?

Упражнение 9.7 [Программирование]. Если у вас имеется доступ к достаточно мощному компьютеру, попробуйте повторить эксперимент, результаты которого представлены в нижней части рис. 9.14. Затем повторите данный эксперимент с новым значением $b = 3$. Объясните полученные результаты.

9.7. Эвристический поиск

Методы планирования в пространстве состояний, преобладающие в области искусственного интеллекта, в совокупности известны под термином *эвристический поиск*. Несмотря на внешнее отличие от рассмотренных до сих пор в данной главе методов планирования, эвристический поиск и некоторые его идеи могут эффективно взаимодействовать с данными методами. В отличие от последних эвристический поиск направлен не на изменение приближенной или «эвристической» функции ценности, но лишь на усовершенствование процесса выбора действий при заданной текущей функции ценности. Другими словами, эвристический поиск представляет собой планирование в части выработки стратегии.

В эвристическом поиске для каждого встретившегося состояния рассматривается большое дерево возможных продолжений. Приближенная функция ценности применяется к листьям дерева и затем подвергается дублированию в направлении текущего состояния. Процесс дублирования в рамках дерева поиска аналогичен процессу тах-дублирований (для функций V^* и Q^*), рассмотренных в данной книге. Процесс дублирования останавливается в узлах состояния—действие, относящихся к текущему состоянию. Как только вычислены дублированные ценности для данных узлов, лучший из них выбирается в качестве текущего состояния, и затем все остальные дублированные ценности отбрасываются.

В традиционном эвристическом поиске не осуществляется никаких действий по сохранению дублированных ценностей при помощи измене-

ния приближенной функции ценности. Фактически функция ценности в общем случае задается человеком и больше не изменяется в процессе поиска. Тем не менее полезно будет рассмотреть случай, в котором происходит изменение функции ценности во времени, при использовании либо дублированных ценностей, рассчитанных в течение эвристического поиска, либо любого другого метода, представленного в данной книге. В некотором смысле данный подход мы применяли постоянно. Наши жадные и ε -жадные методы выбора действий схожи с эвристическим поиском, хотя и в меньшем масштабе. Например, для расчета жадного действия при заданной модели и функции ценности состояния необходимо из каждого возможного состояния посмотреть вперед в каждое следующее состояние, осуществить дублирование вознаграждений и расчетных ценностей и затем выбрать наилучшее действие. Так же как и в случае с традиционным эвристическим поиском, данный процесс рассчитывает дублированные ценности возможных действий, но не предпринимает попыток сохранить их. Таким образом, эвристический поиск можно рассматривать как продолжение идеи жадной стратегии после единственного шага.

Целью поиска, осуществляемого далее чем на один шаг, является получение более эффективного алгоритма выбора действий. Если имеется идеальная модель и неидеальная функция ценности действий, то фактически результатом более глубокого поиска обычно является более эффективная стратегия¹⁾. Очевидно, что если поиск продолжается до конца эпизода, то эффект неидеальной функции ценности устраняется, а определенное таким образом действие должно быть оптимальным. Если поиск обладает такой значительной глубиной k , что величина γ^k очень мала, то действия будут почти оптимальными. С другой стороны, чем глубже поиск, тем больше требуется вычислительных ресурсов, что обычно приводит к увеличению времени отклика. Хорошим примером является разработанная Тезауро программа TD-Gammon для игры в нарды на гроссмейстерском уровне (см. разд. 11.1). Данная система использовала алгоритм TD(λ) для изучения функции ценности последующих состояний, проводя множество игр сама с собой с использованием формы эвристического поиска для осуществления ходов. В качестве модели в TD-Gammon использовались имеющиеся знания о вероятностях выпадения цифр на игральных костях и предположение, что соперник всегда выбирает лучшие для него с точки зрения TD-Gammon действия.

¹⁾ Существуют интересные исключения из данного утверждения. См. книгу [Pearl (1984)].

Тезауро обнаружил, что чем более глубоким был эвристический поиск, тем более правильные ходы осуществлял TD-Gammon, при этом тем больше ему требовалось времени для выбора хода. Нарды имеют большой коэффициент ветвления, при этом ходы должны осуществляться в течение нескольких секунд. Реально осуществимым оказался выборочный поиск лишь на несколько шагов вперед, однако результатом даже такого поиска становилось значительное улучшение алгоритма выбора действий.

До сих пор эвристический поиск рассматривался нами в качестве метода выбора действия, однако данная функция может быть и не основной. Эвристический поиск также предлагает способы осуществления выборочно распределенных дублирований. Это может привести к более точной и быстрой аппроксимации оптимальной функции ценности. Большой объем исследований, связанных с эвристическим поиском, был посвящен тому, каким образом сделать поиск наиболее эффективным. Дерево поиска растет избирательно, интенсивнее вдоль одних направлений и медленнее вдоль других. Например, дерево поиска обычно более развито в направлении тех действий, который с наибольшей вероятностью считаются наилучшими, и менее развито в направлениях тех действий, который агент, вероятно, больше не станет выбирать. Можно ли использовать подобную идею для улучшения распределения дублирований? Нельзя ли это осуществить при помощи приоритетной корректировки тех пар состояния—действие, значения ценности которых наиболее близки к максимальным значениям, возможным в данном состоянии? По нашим сведениям, данная и другие возможности формирования распределения дублирований, основанные на идее, позаимствованной из эвристического поиска, до сих пор не исследованы.

Не следует упускать из виду наиболее очевидное направление, в котором сосредоточен эвристический поиск при проведении дублирований: текущее состояние. Одна из основных причин эффективности эвристического поиска заключается в том, что его дерево поиска является узконаправленным в сторону состояний и действий, которые могут последовать непосредственно за текущим состоянием. Вы, к примеру, можете чаще играть в шахматы, а не в шашки, однако когда вы играете в шашки, для эффективной игры необходимо думать о шашках, о текущей позиции на доске, о возможных следующих ходах и позициях именно в данной игре. Как бы вы ни выбирайте действия, это будут состояния и действия с наивысшим приоритетом для дублирований и с наиболее точной приближенной функцией ценности. При этом не только ваши вычисления должны быть сосредоточены на предстоящих событиях, но и на

это же должны быть направлены ваши ограниченные ресурсы памяти. В шахматах, например, слишком много возможных позиций, чтобы хранить различные оценки ценности для каждой из них, однако основанные на эвристическом поиске шахматные программы могут легко сохранять оценки миллионов позиций, с которыми они сталкиваются, осуществляя обзор будущих позиций из одного-единственного состояния. Такое сосредоточение ресурсов памяти и вычислительных ресурсов на принятии текущего решения и является, вероятно, основной причиной такой высокой эффективности эвристического поиска.

Распределение дублирований может быть скорректировано аналогичным образом для их концентрации на текущем состоянии и вероятных последующих состояниях. В качестве предельного случая можно применить как раз методы эвристического поиска для формирования дерева поиска, как это показано на рис. 9.15. Если дублирования распределены именно таким образом и при этом используется представление в виде таблицы, то будет осуществляться в точности такое же дублирование, что и в эвристическом поиске. Любой поиск в пространстве состояний может быть представлен этим способом при помощи набора большого количества отдельных одношаговых дублирований. Таким образом, наблюдаемое улучшение результатов с более глубоким поиском

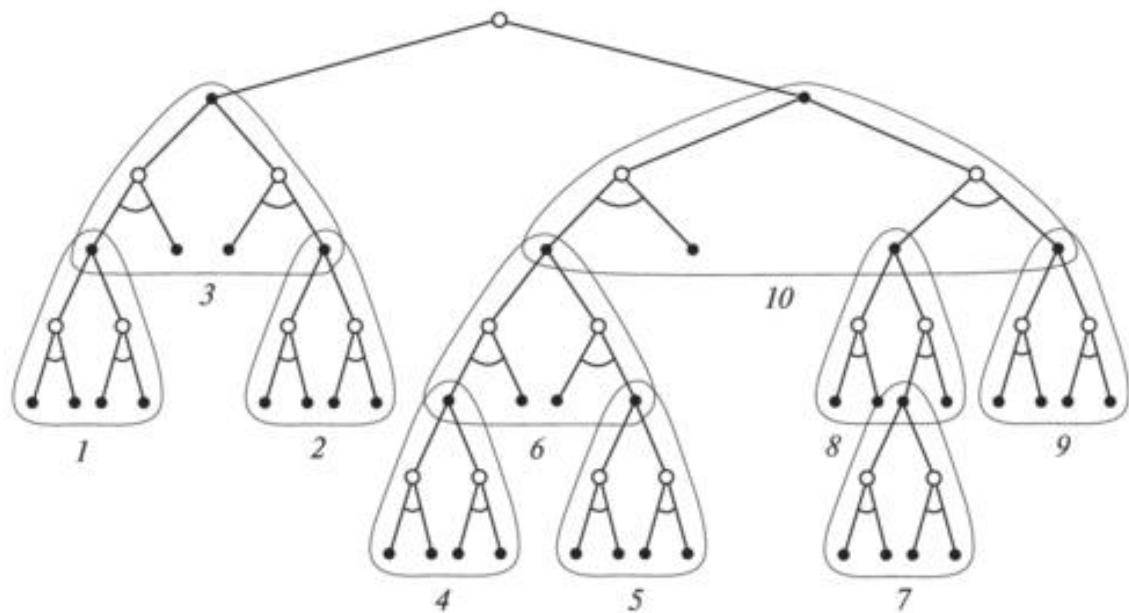


Рис. 9.15. Глубокие дублирования, используемые в эвристическом поиске, могут быть реализованы при помощи последовательности одношаговых дублирований (очерчены на рисунке). Представленный в данном случае порядок действий относится к выборочному поиску в глубину

происходит не за счет использования многошаговых дублирований, но за счет концентрации дублирований на состояниях и действиях, непосредственно следующих за данным состоянием. При концентрации большей части вычислительных ресурсов на расчетах, связанных с конкретными действиями, можно добиться намного лучших результатов, чем при использовании ни на чем не сконцентрированного дублирования.

9.8. Итоги

В данной главе была подчеркнута тесная взаимосвязь между планированием оптимального поведения и обучением оптимальному поведению. В обоих случаях присутствует оценка одной и той же функции ценности, и в обоих случаях чаще всего оценки корректируются пошагово, длинными сериями небольших операций дублирования. Поэтому сама собой на-прашивается идея об объединении процессов планирования и обучения. Добиться этого можно, просто разрешив обоим процессам корректировать одну и ту же оцениваемую функцию ценности. Кроме этого, любой из методов обучения может быть преобразован в метод планирования при помощи замены реального опыта на имитируемый опыт (сгенерированный моделью). В этом случае обучение и планирование становятся еще более схожими; они, возможно, будут выражаться идентичными алгоритмами, использующими два разных источника получения опыта.

Естественно объединить методы планирования с процессами исполнения и изучения модели. Планирование, исполнение и изучение модели взаимодействуют между собой (рис. 9.2), при этом результат деятельности каждого процесса используется последующим процессом для дальнейших улучшений; все остальные взаимодействия между ними либо не требуются, либо запрещаются. Согласно наиболее простому подходу, все процессы должны происходить асинхронно и параллельно. Если одновременно нескольким процессам требуется вычислительные ресурсы, то их распределение может происходить практически произвольным образом, в зависимости от того, какая структура распределения наилучшим образом подходит к конкретной задаче.

В данной главе мы рассмотрели несколько факторов, которые определяют различия методов планирования в пространстве состояний. Одним из наиболее важных факторов является распределение дублирований, т. е. определение направления поиска. Приоритетная прогонка фокусируется на состояниях, которые предшествуют тем, ценности которых были недавно изменены. Примененный к обучению с подкреплением эвристический поиск направлен, среди прочего, на состояния, сле-

дующие за данным. Траекторная выборка является эффективным способом использования распределения с интегрированной оценкой ценности стратегий. Все рассмотренные способы могут значительно ускорить процесс планирования, и сейчас их активно исследуют.

Еще одним фактором различия является размер дублирований. Чем меньше дублирование, тем более инкрементными могут быть методы планирования. Одним из наименьших дублирований являются одношаговые выборочные дублирования. Согласно проведенному анализу, одношаговые выборочные дублирования могут оказаться более предпочтительными при решении больших задач. С этим вопросом связана глубина дублирований. Во многих случаях глубокие дублирования могут быть реализованы при помощи последовательности неглубоких дублирований.

9.9. Библиографические и исторические справки

- 9.1. Представленный здесь общий подход к планированию и обучению постепенно развивался в течение нескольких лет, в частности, авторами данной книги [Sutton (1990, 1991a, 1991b); Barto, Bradtke and Singh, (1991, 1995); Sutton and Pinette (1995); Sutton and Barto (1991b)]; большой вклад также внесли работы [Agre and Chapman (1990); Agre (1988); Bertsekas and Tsitsiklis (1989); Singh (1993)] и другие. Также большое влияние на авторов книги оказало исследование в области психологии скрытого обучения [Tolman (1932)] и взгляд на природу мышления с точки зрения психологии, изложенный в работах [Galanter and Gerstenhaber, (1956); Craik (1943); Campbell (1960); Dennett (1978)].
- 9.2–3. Термины *прямое* и *косвенное*, которые использовались для описания различных видов обучения с подкреплением, пришли из литературы по адаптивному управлению [Goodwin and Sin (1984)], где они используются для описания того же типа различий. Термин *идентификация системы* относится в адаптивном управлении к тому, что мы называем *изучением модели*; см. работы [Goodwin and Sin (1984); Ljung and Söderstrom (1983); Young (1984)]. Dyna-структура предложена в статье [Sutton (1990)], и результаты разделов 9.2–9.3 основаны на представленных там результатах.
- 9.4. Приоритетная прогонка была разработана одновременно и независимо в работах [Moore and Atkeson (1993)] и [Peng and Williams (1993)]. Представленные на рис. 9.10 результаты получены в статье [Peng and Williams (1993)]. Представленные на рис. 9.11 результаты получены Муром и Аткесоном.
- 9.5. Материал данного раздела основан на экспериментах, описанных в работе [Singh (1993)].

- 9.6. Использование траекторной выборки в неявном виде являлось частью обучения с подкреплением с самых первых этапов его развития, однако наиболее подробно данный процесс был рассмотрен в работах [Barto, Bradtke, and Singh (1991, 1995)]. Авторы отметили преимущества узко-направленных вычислений при планировании, упорядочив дублирования в асинхронном ДП-алгоритме, следя реальным или имитируемым траекториям марковских процессов принятия решений. Алгоритм, который проделывал это, используя полное дублирование ценностей, был назван *ДП в реальном времени* по аналогии с *эвристическим поиском в реальном времени*, который был предложен в статье [Korf (1990)]. Данный вопрос был также проработан в статье [Jalali and Ferguson (1989)], где получившийся алгоритм назван *нестационарным ДП*.
- 9.7. Для дальнейшего изучения эвристического поиска читателю предлагается обратиться к таким работам, как [Russel and Norvig (1995)] и [Korf (1998)]. В статье [Peng and Williams (1993)] идея прямого фокусирования дублирований развита гораздо глубже, чем предлагает материал данного раздела.
-

10 Важнейшие аспекты обучения с подкреплением

На протяжении всей этой книги авторы старались представить обучение с подкреплением не как набор отдельных методов, а как совокупность логически согласованных идей, пронизывающих эти методы. Каждую из таких идей можно рассматривать как некое направление в пространстве нескольких измерений, вдоль которой меняются методы. Совокупность таких направлений порождает обширное пространство возможных методов. Исследуя данное пространство в терминах его измерений, можно добиться наиболее глубокого и ясного понимания рассматриваемой проблемы. В этой главе идея многомерного пространства методов используется для обобщения предлагаемого в данной книге представления об обучении с подкреплением, а также для того, чтобы выявить некоторые наиболее значимые пробелы в освещении данной области.

10.1. Единый подход

Все методы обучения с подкреплением, рассмотренные ранее в данной книге, содержат три общие ключевые идеи. Первая из них: целью каждого из методов является оценка функций ценности. Вторая: каждый из методов использует дублирование ценностей вдоль реальных или возможных траекторий состояний. Третья: каждый из методов следует общей стратегии обобщенной итерации по стратегиям (ОИС), т. е. они вычисляют приближенную функцию ценности и приближенную стратегию, и при этом стараются улучшить данную функцию, основываясь на стратегии, и стратегию, основываясь на этой функции. Три эти общие для всех методов идеи определяют собой объект исследования, описанный в данной книге. Предполагается, что функции ценности, дублирование и ОИС являются мощными организующими принципами, потенциально применимыми ко всем моделям интеллекта.

Два наиболее важных направления, вдоль которых изменяются методы, показаны на рис. 10.1. Эти направления связаны с типом дублирования, применяемого для улучшения функции ценности. Вертикальное направление соответствует различию между выборочными дублированиями (основанными на траектории выборки) и полными дубли-

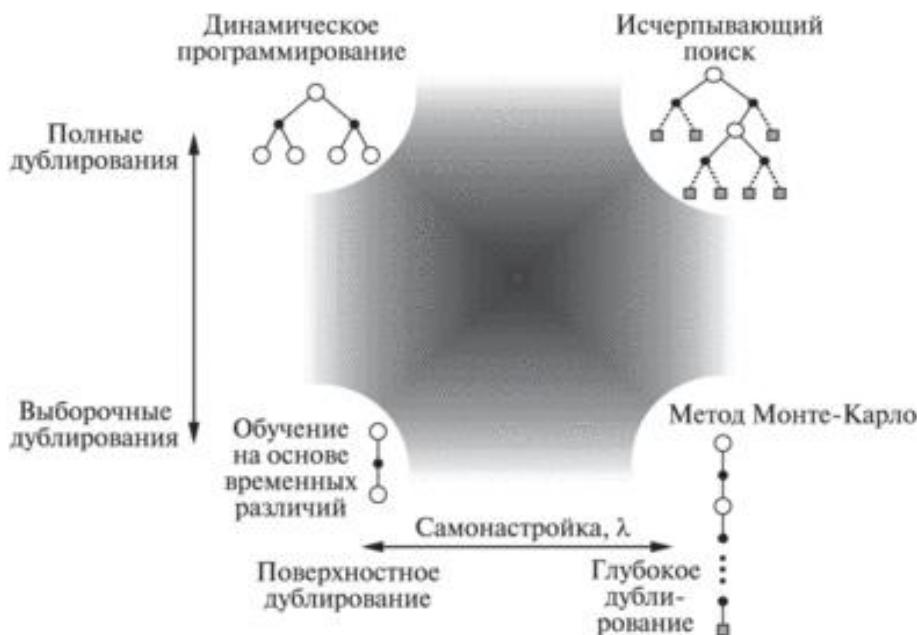


Рис. 10.1. Сечение пространства методов обучения с подкреплением

рованиями (основанными на распределении возможных траекторий). Полные дублирования, очевидно, требуют наличия модели, тогда как выборочные дублирования могут быть выполнены как при наличии модели, так и при ее отсутствии (еще один фактор различия). Горизонтальное направление соответствует глубине дублирований, т. е. степени самонастройки. В трех из четырех углов данного сечения пространства располагаются три основных метода расчета ценностей: динамическое программирование, TD-метод и метод Монте-Карло. Вдоль нижней границы пространства располагаются методы выборочного дублирования, начиная от одношаговых TD-дублирований, и заканчивая Монте-Карло дублированиями полной выгоды. Между данными алгоритмами находится спектр методов, включающий методы, в основе которых лежат n -шаговые дублирования, такие как λ -дублирование, реализованные при помощи следов приемлемости.

Методы ДП представлены в левом верхнем углу данного пространства, так как они включают одношаговые полные дублирования. Правый верхний угол пространства представляет собой предельный случай полного дублирования, настолько глубокого, что оно осуществляется вплоть до заключительного (терминального) состояния (или, в случае с непрерывной задачей, до тех пор, пока при помощи коэффициентов приведения влияние любого последующего вознаграждения не будет снижено до пренебрежимо малой величины). Это случай ис-

черпывающего поиска. Промежуточными методами вдоль этого направления являются такие методы, как эвристический поиск и связанные с ним методы, которые осуществляют поиск и дублирование до некоторой ограниченной глубины, возможно, избирательно. Вдоль вертикального направления также имеются промежуточные методы. Данное направление содержит методы, которые сочетают в себе полные и выборочные дублирования; сюда же относятся методы, которые сочетают в себе выборки и распределения в рамках одного дублирования. Внутренняя часть квадрата заполнена, чтобы представить символически пространство всех промежуточных методов.

Третье важное направление относится к аппроксимации функций. Аппроксимацию функций можно рассматривать как ортогональный спектр возможностей, начинающийся с табличных методов в качестве предельного случая объединения состояний, продолжающийся различными линейными методами и завершающийся набором разнообразных нелинейных методов. Данное направление можно представить символически как перпендикуляр к плоскости страницы на рис. 10.1.

Другим направлением, которое рассматривалось в данной книге, является бинарное различие между методами с интегрированной оценкой ценности стратегий и методами с разделенной оценкой ценности стратегий. В первом случае агент ищет функцию ценности для стратегии, которой он придерживается в данный момент, тогда как во втором случае он пытается найти функцию ценности для стратегии, которая в данный момент является наилучшей по его мнению. Эти две стратегии всегда отличны друг от друга вследствие необходимости исследования. Взаимодействие данного направления с направлениями самонастройки и аппроксимации функций, которое обсуждалось в гл. 8, наглядно демонстрирует преимущества рассмотрения пространства методов с точки зрения направлений. Несмотря на то что имеет место взаимодействие между указанными тремя направлениями, многие другие направления оказались незначимыми, тем самым сильно упрощая процесс анализа и повышая его значимость.

Кроме четырех только что рассмотренных направлений в данной книге были определены и некоторые другие.

Определение выгоды. Является ли задача эпизодной или непрерывной, приведенной или неприведенной?

Ценности состояний, ценности действий или ценности последующих состояний. Какие ценности нужно оценивать? Если вычисля-

ются только ценности состояний, то для выбора действий требуется либо модель, либо отдельная стратегия (как в методе исполнитель—критик).

Выбор/изучение действий. Каким образом необходимо выбирать действия, чтобы добиться необходимого соотношения между изучением и применением? Нами были рассмотрены лишь простейшие способы решения этого вопроса: ϵ -жадный и softmax-выбор действий, а также оптимистичный способ задания начальных ценностей.

Синхронность или асинхронность. Осуществляются ли дублирования одновременно для всех состояний или они следуют по очереди, подчиняясь некоторому порядку?

Замещающие или накапливающиеся следы. Если применяются следы приемлемости, какой из их типов предпочтителен?

Реальный или имитируемый опыт. Какой опыт предпочтительнее использовать при дублировании — реальный, имитируемый или оба одновременно? Если оба, то в каком соотношении?

Объект дублирований. Какие состояния или пары состояния—действие должны подвергаться дублированию? Не использующие модель методы могут выбирать только среди состояний и пар состояния—действие, которые имели место в реальности, но основанные на модели методы могут осуществлять произвольный выбор. Здесь открывается много потенциально очень привлекательных возможностей.

Когда осуществлять дублирования. Осуществлять ли дублирования в процессе выбора действия или лишь после осуществления выбора?

Ресурсы памяти. Как долго необходимо сохранять в памяти дублированные ценности? Должны ли они сохраняться все время или только на период проведения вычислений, связанных с выбором действия, как в эвристическом поиске?

Конечно, данный список не является исчерпывающим, а перечисленные направления не являются взаимоисключающими. Как правило, отдельные алгоритмы отличаются по многим другим направлениям, и многие алгоритмы находятся в нескольких местах на различных направлениях. Например, Dyna-методы используют как реальный, так и имитируемый опыт, формируя одну и ту же функцию ценности. Кроме этого, целесообразно иметь множество функций ценности, вычисленных различными способами или с использованием различных представлений состояний и действий. Тем не менее данные направления формируют согласованный набор идей, используемых для описания и исследования обширного пространства возможных методов.

10.2. Некоторые другие новые направления

В данном пространстве методов обучения с подкреплением остается еще очень много неисследованного. Например, даже для табличного случая сходимость к оптимальной функции не была доказана ни для одного из методов, использующих многошаговые дублирования. Что касается методов планирования, то основные идеи, такие как траекторная выборка и узконаправленные выборочные дублирования, практически совсем не исследованы. При ближайшем рассмотрении областей пространства они, несомненно, окажутся более сложными и обладающими более сложной внутренней структурой, чем это кажется на первый взгляд. Существуют также другие направления, вдоль которых можно рассматривать обучение с подкреплением, которые еще не были упомянуты и благодаря которым пространство методов значительно расширяется. Здесь мы опишем некоторые из этих направлений, а также рассмотрим некоторые вопросы и аспекты из предыдущих глав.

Одна из наиболее важных проблем в обучении с подкреплением, решение которой значительно расширяет его границы, связана с устранением условия, согласно которому состояния обладают марковским свойством. Существует несколько интересных подходов к рассмотрению немарковского случая. Большинство из этих методов направлено на создание из заданного сигнала состояния и его прошлых ценности нового марковского или почти марковского сигнала. Например, один из методов основывается на теории частично наблюдаемых марковских процессов принятия решений (ЧНМПР). ЧНМПР являются финитными МПР, в которых само состояние не наблюдается, однако наблюдается другой сигнал «восприятия», стохастически связанный с данным состоянием. Теория ЧНМПР активно исследуется для случая, когда имеется полное знание динамики ЧНМПР. В этом случае для расчета на каждом временном шаге вероятности пребывания окружающей среды в каждом состоянии, лежащего в основе МПР, можно использовать методы Байеса. Полученное таким образом распределение вероятностей может затем использоваться в качестве нового сигнала состояния в начальной задаче. Недостатком байесовского метода применительно к ЧНМПР является значительная потребность в вычислительных ресурсах, а также необходимость полной модели окружающей среды. К последним работам, посвященным этому методу, относятся [Littman, Cassandra and Kaelbling (1995)], [Parr and Russel (1995)] и [Chrisman (1992)]. Если условие наличия полной модели ЧНМПР-динамики не выполняется, то суще-

ствующая теория почти бесполезна. Тем не менее можно все же попытаться сформировать марковский сигнал состояния из последовательности восприятий. В данной области было исследовано множество статистических и различных специальных методов (см., например, работы [McCallum (1992, 1993, 1995); (Lin and Mitchell (1992); Chapman and Kaelbling (1991); Moore (1994); Rivest and Schapire (1987); Colombetti and Dorigo (1994); Whitehead and Ballard (1991); Hochreiter and Schmidhuber (1997)].

Все эти методы включают в себя создание улучшенного представления состояния из немарковского представления, полученного от окружающей среды. Согласно другому подходу, представление состояния остается в неизменном виде, при этом используются методы, на которые не оказывает такого отрицательного влияния тот факт, что представление немарковское (см., например, работы [Singh, Jaakkola and Jordan (1994, 1995); Jaakkola, Singh and Jordan (1995)]. На самом деле, таким способом могут рассматриваться многие методы аппроксимации функций. Например, методы объединения состояний для аппроксимации функций по своей сути эквивалентны немарковскому представлению, в котором все элементы пространства состояний создают общее восприятие. В обоих случаях теоретическая сторона проблемы достаточно очевидна, тогда как практическая сторона совсем не очевидна и находится на стадии исследования. В данной ситуации можно только догадываться, укажут ли две данные линии на общий метод решения двух данных проблем.

Другое очень важное направление в обучении с подкреплением, значительно расширяющее его границы, — это объединение идей блочного построения и иерархии. Предварительное обучение с подкреплением заключается в изучении функции ценности и одношаговых моделей динамики окружающей среды. Однако многое из того, что изучают люди, не подпадает в точности ни под одну из этих категорий. К примеру, рассмотрим, что нам известно о завязывании шнурков, о разговоре по телефону или о поездке в Лондон. Изучив все эти вещи, затем можно выбирать среди них и планировать, как будто они являются примитивными действиями. То, что мы изучаем, для того чтобы уметь совершать эти действия, не является традиционными функциями ценности или одношаговыми моделями. Мы способны планировать и изучать на различных уровнях, а также легко устанавливать связи между ними. Мы не изучаем ценности непосредственно, а привыкаем к быстрой оценке значений ценности при возникновении новой ситуации или новой информации. Серьезное исследование обучения с подкреплением направ-

лено на изучение именно таких способностей (см., например, работы [Watkins (1989); Dayan and Hinton (1993); Singh (1992a, 1992b); Ring (1994), Kaelbling (1993b); Sutton (1995)].

Кроме этого, разработаны методы, позволяющие с выгодой использовать структуры конкретных задач. К примеру, многие задачи имеют представления состояний, которые по своей сути являются списками переменных, как, например, показания множества сенсоров, а иногда действия представимы в виде списков своих компонент. Полная или почти полная независимость некоторых переменных от остальных иногда может быть использована для получения более эффективных специальных видов алгоритмов обучения с подкреплением. Иногда даже оказывается возможным разбиение задачи на несколько подзадач, которые могут быть решены отдельными агентами обучения. Задача обучения с подкреплением обычно может быть структурирована различными способами. Некоторые из них отражают реальные аспекты задачи, такие как существование физических сенсоров, а другие являются результатом попыток разбиения задачи на более простые подзадачи. Возможности использования преимущества конкретных структур в обучении с подкреплением и связанные с этим задачи планирования изучались многими исследователями (см., например, работы [Boutilier, Dearden and Goldszmidt (1995); Dean and Lin (1995)]. Существуют также связанные с этим исследования многоагентного или распределенного обучения с подкреплением (см., например, работы [Littman (1994); Markey (1994); Crites and Barto (1996); Tan (1993)].

В заключение необходимо подчеркнуть, что обучение с подкреплением представляет собой *общий* подход к обучению на основе взаимодействия. В большинстве случаев не требуется вмешательства специализированных учителей и наличия специальных знаний, но при этом, если данные дополнительные средства доступны, их обычно используют. Например, часто процесс обучения с подкреплением можно ускорить, давая агенту советы или намеки (см. [Clouse and Utgoff (1992); Maclin and Shavlik (1994)] или показывая агенту поучительные примеры поведенческих траекторий [Lin (1992)]. Согласно еще одному методу облегчения обучения, связанному с «формированием навыка» в психологии, необходимо дать агенту обучения серию сравнительно легких задач, из которых состоит интересующая нас более сложная задача (см. [Selfridge, Sutton and Barto (1985)]. Эти и другие еще не исследованные методы имеют возможность наполнить такие термины машинного обучения, как *учить* и *тренировать*, новым смыслом, более близким к миру животных и человека.

11 Конкретные примеры

В этой завершающей главе рассматривается ряд конкретных примеров применения обучения с подкреплением. Некоторые из них являются важными прикладными проблемами. Один из примеров, шашечная программа Сэмюеля, представляет в основном исторический интерес. Изложенный здесь материал призван показать некоторые компромиссы и вопросы, которые возникают в реальных задачах. Будет показано, например, каким образом имеющиеся специфические знания учитываются при формулировании и решении задачи. Будут изучены также некоторые моменты, связанные с представлением задач, что обычно очень важно для их успешного решения. Алгоритмы, используемые в некоторых из этих конкретных примеров, в основном являются более сложными, чем те, которые мы уже рассматривали. Решение задач обучения с подкреплением еще не стало рутинным процессом, так что искусства в данном вопросе требуется не меньше, чем науки. Упрощение задач и попытки сделать их более очевидными является одной из целей проводимых в настоящее время исследований в области обучения с подкреплением.

11.1. Программа TD-Gammon

Автором одной из наиболее впечатляющих реализаций обучения с подкреплением на сегодняшний день является Джерри Тезауро со своей программой для игры в нарды [Tesauro (1992, 1994, 1995)]. При реализации программы, названной Тезауро *TD-Gammon*, использовался достаточно ограниченный объем знаний об игре в нарды, однако при этом она научилась играть необычайно хорошо, почти на уровне сильнейших в мире гроссмейстеров. Алгоритмом обучения в TD-Gammon являлась простая комбинация TD(λ)-алгоритма и нелинейной аппроксимации функций, использовавшей многослойную нейронную сеть, которая обучалась при помощи метода обратного распространения TD-ошибок.

Нарды очень распространены во всем мире, по ним проводится большое количество турниров и матчи регулярного чемпионата мира. В определенной степени нарды являются азартной игрой, которая привлекает участников возможностью заработать значительную сумму де-

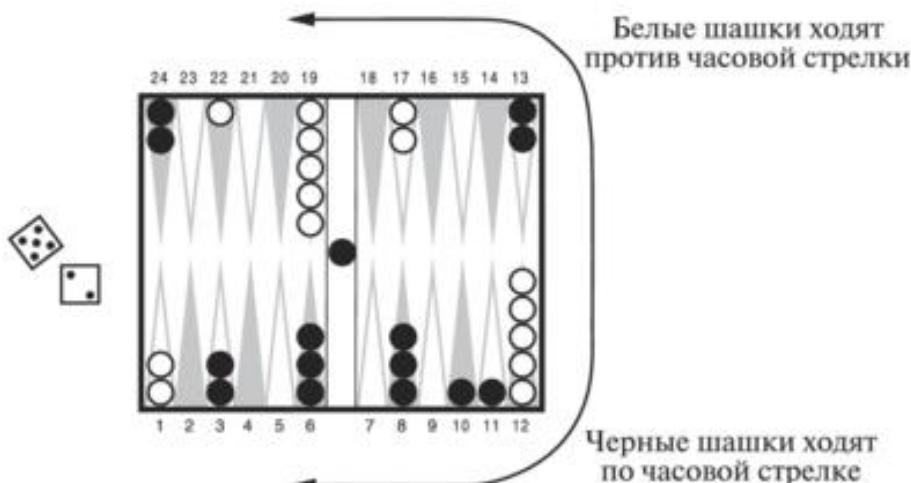


Рис. 11.1. Позиция в нардах

нег. Вероятно, существует больше профессиональных игроков в нарды, чем в шахматы. В нарды играют 15 черными и 15 белыми шашками на доске с 24 ячейками, называемыми *пунктами*.

На рис. 11.1 представлена типичная для начала игры позиция, показанная с точки зрения игрока белыми шашками.

На этом рисунке игрок белыми шашками только что бросил кости, на которых выпало 5 и 2. Это означает, что он может пойти одной из своих шашек на 5 шагов, а другой (или той же самой) — на 2 шага. Например, он может пойти двумя шашками из пункта 12, одной в пункт 17, а другой — в пункт 14. Цель игрока белыми шашками состоит в том, чтобы переместить все свои шашки в последний квадрат (пункты 19–24), а затем снять с доски. Первый, кто снимет с доски все свои шашки, побеждает. Одна из сложностей заключается в том, что, двигаясь в противоположных направлениях, шашки мешают друг другу. Например, если бы на рис. 11.1 был ход игрока черными шашками, он мог бы воспользоваться выпавшей на костях двойкой для того, чтобы передвинуть шашку из пункта 24 в пункт 22 и «выбить» оттуда белую шашку. Выбитые шашки перемещаются в центр доски (где уже находится выбитая до этого черная шашка), откуда они вновь вступают в игру с начальной позиции. Тем не менее если в одном пункте находятся две шашки, то соперник не может переместиться в этот пункт; в этом случае шашки защищены от выбивания. Таким образом, игрок белыми шашками не может использовать выпавшие ему очки 5–2 для того, чтобы переместить какую-либо из шашек из пункта 1, так как возможные конечные пункты заняты группами черных шашек. Формирование групп, занимающих

расположенные рядом пункты, является одной из основных стратегий игры в нарды.

В игре имеется ряд других сложных моментов, однако приведенного выше описания вполне достаточно для понимания основных принципов игры в нарды. При 30 шашках и 24 возможных позициях (26, если считать позицию в центре доски и за ее пределами) очевидно, что количество возможных комбинаций в нардах просто огромно, гораздо больше, чем число элементов памяти в любом из существующих компьютеров. Количество ходов, возможных в каждой позиции, также велико. При стандартных игральных костях, существует 20 различных вариантов ходов. Рассматривая последующие ходы, такие как ответы противника, необходимо также принимать во внимание возможные комбинации выпадения костей. В результате дерево игры имеет коэффициент ветвления, примерно равный 400. Это слишком много для эффективного использования традиционных методов эвристического поиска, которые доказали свою эффективность в таких играх, как шахматы и шашки.

С другой стороны, данная игра хорошо подходит под возможности методов TD-обучения. Несмотря на то что сама игра стохастическая, полное описание состояния можно дать в любой момент. Игра развивается посредством последовательности ходов и позиций до момента завершения, когда выигрывает один из игроков. Результат игры можно интерпретировать как окончательное вознаграждение, которое нужно предсказать. С другой стороны, описанные до этого теоретические результаты нельзя применить в данной задаче. Число состояний настолько велико, что невозможно использовать таблицу поиска, и соперник представляет собой меняющийся во времени источник неопределенности.

В TD-Gammon использована нелинейная форма метода TD(λ). Оценка ценности $V_t(s)$ любого состояния (позиции на доске) s представляла собой оценку вероятности выиграть при старте из состояния s . Для этого вознаграждения полагались равными нулю за все временные шаги, кроме тех, на которых достигался выигрыш. Для реализации функции ценности в TD-Gammon использовалась стандартная многослойная нейронная сеть, наподобие той, что представлена на рис. 11.2. (Реальная нейронная сеть имела два дополнительных элемента в последнем слое для оценки вероятности выигрыша каждого игрока особым методом, называемым «триктрак» или «нарды».) Сеть состояла из слоя входных элементов, слоя скрытых элементов и последнего слоя выходных элементов. Вход сети представлял собой образ позиции в игре, а выход являлся оценкой значения ценности данной позиции.



Рис. 11.2. Нейронная сеть, используемая в TD-Gammon

В первой версии TD-Gammon, TD-Gammon 0.0, игровые позиции были представлены в сети относительно простым способом, в котором содержалось небольшое количество знаний об игре. Тем не менее в нем содержались основные знания о том, каким образом работают нейронные сети и как наилучшим образом представить информацию. Полезно рассмотреть метод представления, выбранный Тезауро. Всего в сети имелось 198 входных элементов. Для каждого пункта на доске четыре элемента указывали число белых шашек в пункте. Если в пункте не было ни одной белой шашки, то все четыре элемента получали нулевую ценность. Если в пункте находилась одна белая шашка, то первый элемент получал ценность 1. Если в пункте было две белых шашки, то как первый, так и второй элемент получали ценность 1. Если в пункте находилось три или больше белых шашек, то все три первых элемента получали ценность 1. Если в пункте находилось более трех шашек, то также задействовался четвертый элемент, показывая дополнительное к первым трем число шашек.

Обозначим через n число шашек в пункте. Если $n > 3$, то четвертый элемент имел значение ценности $(n - 3)/2$. При четырех элементах для белых шашек, четырех элементах для черных шашек и при 24 пунктах общее число элементов было равно 192. Два дополнительных элемента содержали информацию о количестве черных и белых шашек в центре доски (каждому из которых присваивалось значение ценности $n/2$, где n — это число шашек в центре доски), а еще два элемента указывали количество черных и белых шашек, которые успешно покинули преде-

лы доски (им присваивается значение ценности, равное $n/15$, где n — это число шашек, уже покинувших доску). Наконец, еще два элемента в бинарной форме показывали, кто в данный момент делает ход. Логика построения данной схемы достаточно очевидна. По существу, Тезауро постарался представить позиции наиболее простым способом, не стараясь особенно минимизировать число элементов. Он поставил в соответствие один элемент каждой возможной позиции, принципиально отличающейся от остальных, которая с большой вероятностью имела бы существенное значение в игре. Кроме этого, он присвоил этим элементам близкие значения, в данном случае от 0 до 1.

Располагая представленной таким образом позицией в игре, сеть рассчитывала оценку ее ценности стандартным методом. Каждой связи входного элемента со скрытым элементом соответствовал действительный вес. Сигналы от каждого входного элемента умножались на соответствующие весовые коэффициенты и суммировались в скрытом элементе. Результатом $h(j)$ на выходе скрытого элемента j являлась нелинейная сигма-функция взвешенной суммы:

$$h(j) = \sigma\left(\sum_i w_{ij}\phi(i)\right) = \frac{1}{1 + \exp\{-\sum_i w_{ij}\phi(i)\}},$$

где $\phi(i)$ — ценность i -го входного элемента, а w_{ij} — вес его связи с j -м скрытым элементом. Результат сигмоидальной функции всегда находится в пределах от 0 до 1, он интерпретируется как вероятность, основанная на суммировании шансов. Вычисления при переходе от скрытых элементов к выходным аналогичны. Каждая связь скрытого элемента с выходным элементом имеет свой собственный вес. Выходной элемент формирует взвешенную сумму, используя которую, получают аналогичную сигма-нелинейность.

В TD-Gammon использовался алгоритм TD(λ) на основе наискорейшего спуска, описанный в разд. 8.2, с градиентами, рассчитанными при помощи алгоритма обратного распространения ошибки [Rumelhart, Hinton and Williams (1996)]. Напомним, что общее правило корректировки в этом случае следующее:

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)]\vec{e}_t, \quad (11.1)$$

где θ_t — это вектор, содержащий все изменяемые параметры (в данном случае весовые коэффициенты сети), а \vec{e}_t — это вектор следов приемлемости, для каждого компонента θ_t , скорректированного по правилу

$$\vec{e}_t = \gamma\lambda\vec{e}_{t-1} + \nabla_{\vec{\theta}_t}V_t(s_t),$$

где $\vec{e}_0 = \vec{0}$. Градиент в данном уравнении можно рассчитать при помощи процедуры обратного распространения. Для задачи об игре в нарды, в которой $\gamma = 1$ и вознаграждение за все, кроме победы, равно нулю, доля TD-ошибки в правиле обучения обычно составляет $V_t(s_{t+1}) - V_t(s_t)$, как показано на рис. 11.2.

Для того чтобы применить данное правило обучения, необходимо иметь результаты множества игр в нарды. Тезауро получил бесконечную последовательность игр, заставив своего игрока в нарды играть против самого себя. При выборе ходов программа TD-Gammon учитывала каждую из двух десятков комбинаций на игральных костях и соответствующие возможные позиции. Эти возможные позиции являются *следующими состояниями*, как это было описано в разд. 6.8. Сеть должна была оценить все их ценности. После этого выбирался ход, результатом которого была позиция с наибольшей ценностью. Придерживаясь данной схемы, в которой программа TD-Gammon делала ходы за обоих игроков, можно было легко сгенерировать большое количество игр. Каждая игра рассматривалась как эпизод с последовательностью позиций в качестве состояний s_0, s_1, s_2, \dots . Нелинейное TD-правило (11.1) Тезауро применял пошагово, т. е. после каждого отдельно хода.

Весовые коэффициенты в сети изначально были приравнены к малым случайным значениям. Таким образом, начальные оценки были полностью произвольными. Поскольку ходы выбирались на основании этих оценок, начальные ходы неизбежно получались беспорядочными, и первые игры длились сотни и тысячи ходов, пока какая-либо из сторон не выигрывала практически случайно. Тем не менее после проведения нескольких десятков игр результаты стали стремительно улучшаться.

После примерно 300 000 игр, проведенных против самой себя, программа TD-Gammon 0.0, как было сказано выше, научилась играть не хуже, чем лучшие программы для игры в нарды того времени. Это был поразительный результат, так как все предыдущие лучшие компьютерные программы использовали обширные знания об игре в нарды. Например, программой-чемпионом того времени была, вероятно, программа Нейронарды (Neurogammon), написанная тем же Тезауро, использовавшая нейронные сети, а не TD-обучение. Обучение сети в Нейронардах происходило на основе запрограммированного экспертами по игре в нарды большого набора типичных ходов. Кроме этого, изначально сеть наделялась множеством свойств, относящихся конкретно к игре в нарды. Нейронарды представляли собой тщательно отлаженную, высокоэффективную программу игры в нарды, которая уверенно выиграла международную олимпиаду по нардам в 1989 г. С другой сто-

роны, в TD-Gammon 0.0 не было заложено, по сути, никаких знаний об игре в нарды. То, что данная программа играла на том же уровне, что и Нейронарды, является ярким свидетельство высокого потенциала методов обучения, основанных на игре против самого себя.

Учитывая такой успех программы TD-Gammon 0.0 с нулевыми знаниями об игре в нарды, было принято решение о внесении некоторых изменений: были добавлены некоторые характерные именно для игры в нарды свойства, и при этом был оставлен основанный на игре против самого себя TD-метод обучения. Результатом внесения данных изменений стала программа TD-Gammon 1.0, которая была значительно лучше, чем все существовавшие на тот момент программы игры в нарды, а серьезное сопротивление ей могли оказать лишь люди, эксперты по игре в нарды. Более поздние версии программы, TD-Gammon 2.0 (40 скрытых элементов) и TD-Gammon 4.0 (80 скрытых элементов), были дополнены выборочной двойной процедурой поиска. При выборе ходов эти программы просчитывали не только позиции, которые могли быть непосредственным результатом хода, но и возможные комбинации игровых костей у противника, а также его вероятные ходы. В предположении, что соперник выбирает ходы, которые являются для него наилучшими в данной ситуации, рассчитывались ожидаемые ценности каждого из возможных ходов, после чего выбирался лучший из них. Для ускорения процесса второй этап поиска проводился только для тех возможных ходов, которые высоко оценивались после первого этапа поиска, в среднем количество таких ходов равнялось 4 или 5. Двойной поиск оказывал влияние только на выбранные ходы; процесс обучения оставался неизменным. Самая последняя версия программы, TD-Gammon 3.0, использовала 160 скрытых элементов и выборочный тройной поиск. В TD-Gammon используется комбинация найденных функций ценности и поиска верных решений, как в методах эвристического поиска. В более поздней работе [Tesauro and Galperin (1997)] начато исследование методов траекторной выборки в качестве альтернативы поиску.

Тезауро провел значительное количество игр со своей программой против игроков в нарды мирового класса. Результаты этих игр приведены в табл. 11.1. Основываясь на них, а также на анализе, проведенном экспертами игры в нарды (см. работы [Robertie (1992); Tesauro (1995)]), можно сказать, что программа TD-Gammon 3.0 играет так же или почти так же, как лучшие в мире игроки среди людей. Она уже может стать чемпионом мира. Эти программы оказали сильное влияние на стратегию игры лучших игроков в нарды. Например, программа TD-Gammon разыгрывала определенные начальные позиции способом,

Таблица 11.1
Результаты игр программы TD-Gammon

Программа	Скрытые элементы	Тренировочные игры	Соперники	Результаты
TD-Gammon 0.0	40	300 000	другие программы	близко к лучшим
TD-Gammon 1.0	80	300 000	Robertie, Magriel, ...	-13 очков / 51 игра
TD-Gammon 2.0	40	800 000	различные гроссмейстеры	-7 очков / 38 игр
TD-Gammon 2.1	80	1 500 000	Robertie	-1 очко / 40 игр
TD-Gammon 3.0	80	1 500 000	Kazaros	+6 очков / 20 игр

отличным от общепринятых схем, которые использовались лучшими игроками. Учитывая успех TD-Gammon и основываясь на дальнейшем анализе игры, ведущие игроки в нарды разыгрывают теперь эти позиции тем же способом, что и программа TD-Gammon [Tesauro (1995)].

11.2. Программа игры в шашки Сэмюеля

Известным предшественником программы TD-Gammon, созданной Тесауро, были работы Артура Сэмюеля [Samuel (1959, 1967)] по созданию программ, обучавшихся игре в шашки. Сэмюель одним из первых стал эффективно применять методы эвристического поиска и то, что сейчас называется обучением на основе временных различий. Его программы для игры в шашки помимо того, что представляют исторический интерес, являются наглядными обучающими примерами. В данной книге особое значение придается связи методов Сэмюеля с современными методами обучения с подкреплением и объясняется логика, которой следовал Сэмюель, выбирая данные методы.

Впервые Сэмюель написал программу игры в шашки для компьютера IBM 701 в 1952 г. Создание первой *обучающейся* программы было завершено в 1955 г., а в 1956 г. она была продемонстрирована по телевидению. Поздние версии данной программы играли на хорошем уровне, хотя выдающихся результатов не показывали. Сэмюеля привлекали игровые программы, как способ исследования машинного обучения, так как игры, будучи проще «взятых из жизни» задач, тем не менее позволяют проводить результативные исследования того, как эвристические методики можно использовать совместно с обучением. Он выбрал обь-

ектом своего изучения шашки вместо шахмат, так как относительная простота первых позволяет полностью сконцентрироваться на обучении.

Программы Сэмюеля играли, осуществляя поиск вперед из каждого текущего состояния. Они использовали то, что сейчас называется методами эвристического поиска, чтобы определить, каким образом развивать дерево поиска и когда останавливать поиск. В процессе очередного поиска заключительные позиции на доске оценивались функцией ценности или «оценочным полиномом», при помощи линейной аппроксимации функций. В этом и в других вопросах Сэмюель, вероятно, основывался на предположениях работы Шеннона [Shannon (1950)]. В частности, при поиске наилучших ходов из данной позиции программа Сэмюеля основывалась на шенноновском минимаксном алгоритме. При прохождении в обратном направлении дерева поиска из получивших оценку заключительных состояний, каждое состояние получало оценку позиции, которая являлась бы результатом наилучшего хода из данного состояния, предполагая, что машина всегда старается максимизировать оценку, тогда как соперник всегда старается ее минимизировать. Сэмюель назвал данный алгоритм *усиленной оценкой* позиции. Когда алгоритм минимакса достигал корней дерева поиска — текущей позиции, это приводило к наилучшему ходу в предположении, что соперник использует такой же критерий оценки с поправкой на его точку зрения. Некоторые версии программ Сэмюеля использовали сложные методы управления поиском, аналогичные тем, которые известны как «альфа-бета» отсечения (см. работу [Pearl (1984)]).

Сэмюель использовал два основных метода обучения, простейший из которых он назвал *обучением методом механического запоминания*. Он заключался в простом запоминании описания каждой встретившейся в течение игры позиции на доске вместе с ее усиленной оценкой ценности, определенной алгоритмом минимакса. В результате, если уже встречавшаяся ранее позиция возникала снова в качестве заключительной позиции дерева поиска, глубина поиска увеличивалась, при условии что сохраненная ценность данной позиции содержала в кэш-памяти результаты одного или более проведенных ранее поисков. Изначальная проблема данного алгоритма заключалась в том, что у программы не было стимула двигаться по кратчайшему пути к победе. Сэмюель задал ей «направление движения» путем небольшого уменьшения ценности позиции каждый раз, когда она получала очередное значение (называемое уровнем) в процессе минимаксного анализа. «Если перед программой не ставится выбор между позициями на доске, оценки которых различаются лишь номером уровня, она будет автоматически де-

лать наиболее выгодный выбор, выбирая низкоуровневый вариант, если она выигрывает, и высокоуровневый вариант, если она проигрывает» (см. [Samuel (1959)], с. 80). Сэмюель считал использование данной методики, аналогичной алгоритму приведения, необходимым для успешного обучения. Результатом обучения методом механического запоминания было медленное, но постоянное улучшение игры, которое было наиболее эффективно в эндшпиле. Программа Сэмюеля стала «начинающим игроком уровня выше среднего» после обучения в процессе многих игр против самой себя, против множества игроков среди людей, а также по книгам в режиме обучения с учителем.

Обучение методом механического запоминания и другие аспекты работы Сэмюеля заключают в себе основную идею обучения на основе временных различий — ценность состояния должна быть равна ценности вероятных следующих состояний. Ближе всего к данной идее Сэмюель подошел при реализации своего второго метода обучения, алгоритма обучения на основе обобщения, направленного на корректировку параметров функции ценности. Концептуально метод Сэмюеля был аналогичен методу, который впоследствии применял Тезауро в своих программах TD-Gammon. Программа играла множество игр против другой своей же версии и после каждого хода осуществляла дублирование. Предложенная Сэмюелем идея дублирования представлена в виде диаграммы на рис. 11.3. Каждый белый кружок представляет собой позицию, в которой программа делает ход, так называемую *позицию хода*, а каждый черный кружок представляет собой позицию, в которой ходит соперник. Дублирование применялось к ценности каждой позиции хода после хода с каждой стороны, результатом чего являлась вторая позиция хода. Дублирование осуществлялось по направлению к минимаксному значению поиска, начинавшегося из второй позиции хода. Таким образом, общий эффект был таким же, как от дублирования, состоящего из одного полного хода в реальных условиях и последующего поиска среди возможных событий, как это показано на рис. 11.3. В действительности алгоритм Сэмюеля был значительно сложнее, чем тот, что представлен здесь, однако основную идею мы описали.

Алгоритм Сэмюеля не содержал вознаграждений в явном виде. Вместо этого он фиксировал вес наиболее важного свойства, свойства *преимущества шашек*, которое заключалось в сравнении числа шашек, которые имелись у программы, с числом шашек у противника, учитывая больший вес дамок и тот факт, что превращение шашек в дамки выгоднее делать, когда выигрываешь, чем когда проигрываешь. Таким образом, целью программы Сэмюеля было улучшение преимущества

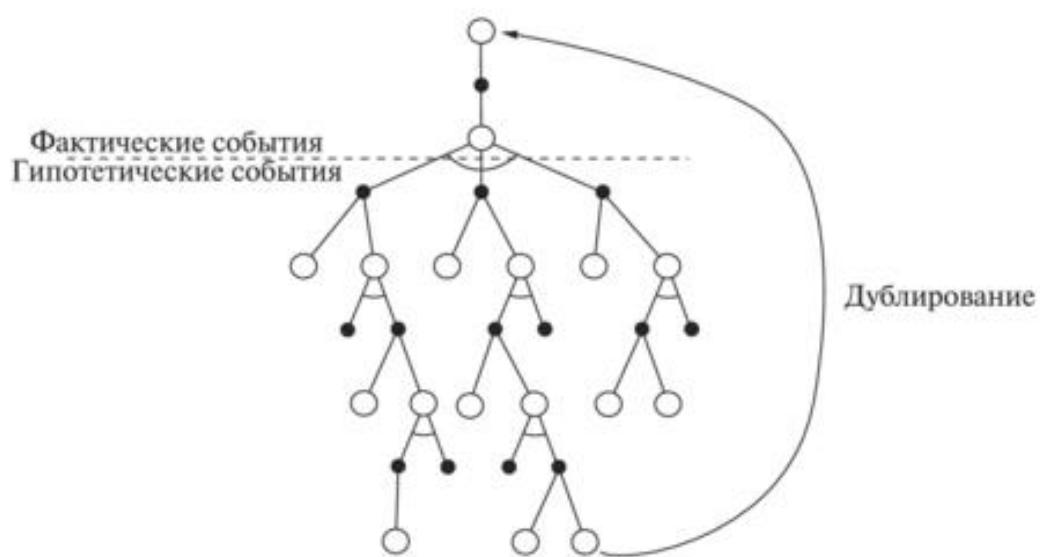


Рис. 11.3. Диаграмма предшествующих состояний программы Сэмюеля для игры в шашки

своих шашек, что в игре в шашки является главной составляющей победы.

Тем не менее методы обучения Сэмюеля, возможно, были лишены важной части полноценного алгоритма временных различий. Обучение на основе временных различий можно рассматривать как способ добиться согласованности функции ценности с самой собой, и это определенно присутствует в методе Сэмюеля. Однако, кроме этого, необходимо увязать функцию ценности с истинными ценностями состояний. Обычно этого достигают при помощи вознаграждений и введения коэффициентов приведения или при помощи введения фиксированной ценности заключительного состояния. Однако методы Сэмюеля не предусматривают ни вознаграждений, ни какого-либо особого подхода к заключительным позициям в игре. Как отмечал сам Сэмюель, для того чтобы его функция ценности была согласованной, необходимо было всего лишь присвоить постоянные ценности всем позициям. Он надеялся избежать подобных решений, присваивая своему элементу, отвечающему за преимущество шашек, большой, неизменяемый вес. Однако, несмотря на то что это уменьшало вероятность нахождения бесполезных оценочных функций, это не запрещало их появление. Например, можно было получить постоянную функцию, задав изменяемые веса таким образом, чтобы свести на нет влияние неизменяемого веса.

Так как нахождение полезной оценочной функции не являлось обязательным условием для методик обучения Сэмюеля, то существовала

возможность, что оценочная функция ухудшится по мере приобретения опыта. В действительности Сэмюэль указывал на это, наблюдая данную тенденцию в течение продолжительных серий тренировочных игр программы против самой себя. Для того чтобы программа снова начала улучшаться, требовалось вмешательство Сэмюеля, который вновь приравнивал к нулю вес с наибольшим абсолютным значением. По его мнению, такое радикальное вмешательство уводило программу от локального оптимума, но при этом оно уводило программу и от оценочной функции, которая, хотя и была согласованной, но никак не влияла на выигрыш или проигрыш игры.

Несмотря на обозначенные выше проблемы, программа игры в шашки Сэмюеля, используя обобщающий метод обучения, достигла уровня игры «выше среднего». Хорошие игроки-любители характеризовали эту программу как «хитрую, но такую, которую можно победить» [Samuel (1959)]. В отличие от программы, основанной на методе механического запоминания, данная программа хорошо играла в середине игры, показывая при этом достаточно слабую игру в начальной и в завершающей стадиях. Кроме того, в данной программе была заложена возможность поиска в пространстве свойств для отыскания тех из них, которые являлись наиболее важными для формирования функции ценности. Более поздняя версия программы [Samuel (1967)] получила улучшенную процедуру поиска, включавшего в себя альфа-бета отсечение решений, активное использование обучения с учителем, называемого «формальным обучением», а также иерархические таблицы поиска, называемые сигнатурными таблицами [Griffith (1966)], для представления функции ценности вместо линейной аппроксимации функций. Данная версия программы обучалась игре намного лучше версии 1959 года, хотя также достигала лишь среднего уровня игры. Программа игры в шашки Сэмюеля была признана значительным достижением в области искусственного интеллекта и машинного обучения.

11.3. Акробот

Обучение с подкреплением применяется во множестве физических задач управления (например, см. подборку задач робототехники [Connell and Mahadevan (1993)]. Одной из таких задач является задача *акробота* (Acrobot), двухстержневого робота, со свободно качающейся нижней частью, который является грубой аналогией раскачивающегося на перекладине гимнаста (рис. 11.4). В первом шарнире (соответствующем

Цель: поднять стержень над этой линией



Рис. 11.4. Акробот

рукам гимнаста на перекладине) не может создаваться крутящий момент, однако во втором шарнире (соответствующем изгибающейся талии гимнаста) момент создаваться может. Система имеет четыре непрерывных параметра состояния: положения двух шарниров и их скорости. Уравнения движения приведены на рис. 11.5. Данная система подробно изучалась специалистами по системам управления (см., например, [Spong (1994)]) и исследователями в области машинного обучения (см., например, [DeJong and Spong (1994); Boone (1997)]).

При управлении акроботом целью является раскачать нижний стержень так, чтобы его конец («ноги») оказался выше первого шарнира на величину, равную длине стержня, за минимальное время. В данной задаче прикладываемый ко второму шарниру закручивающий момент ограничен тремя позициями: положительный момент заданной величины, отрицательный момент той же величины или отсутствие момента. За каждый шаг, предшествующий достижению цели, когда заканчивается

$$\begin{aligned}\ddot{\theta}_1 &= -d^{-1}(d_2\ddot{\theta}_2 + \phi_1), \\ \ddot{\theta}_2 &= \left(m_2l_{c2}^2 + I_2 - \frac{d_2^2}{d_1}\right)^{-1} \left(\tau + \frac{d_2}{d_1}\phi_1 - m_2l_1l_{c2}\dot{\theta}_1^2 \sin \theta_2 - \phi_2\right), \\ d_1 &= m_1l_{c1}^2 + m_2(l_1^2 + l_{c2}^2 + 2l_1l_{c2} \cos \theta_2) + I_1 + I_2, \\ d_2 &= m_2(l_{c2}^2 + l_1l_{c2} \cos \theta_2) + I_2, \\ \phi_1 &= -m_2l_1l_{c2}\dot{\theta}_2^2 \sin \theta_2 - 2m_2l_1l_{c2}\dot{\theta}_2\dot{\theta}_1 \sin \theta_2 + (m_1l_{c1} + m_2l_1)g \cos\left(\theta_1 - \frac{\pi}{2}\right) + \phi_2, \\ \phi_2 &= m_2l_{c2}g \cos\left(\theta_1 + \theta_2 - \frac{\pi}{2}\right)\end{aligned}$$

Рис. 11.5. Уравнения движения моделируемого акробата. В процессе моделирования использовались временные шаги размером в 0,05 секунд и действия, выбирающиеся через каждые четыре временных шага. Приложенный ко второму шарниру крутящий момент обозначен $\tau \in \{+1, -1, 0\}$. Ограничения на положения шарниров отсутствовали, однако угловые скорости были ограничены следующими интервалами: $\theta_1 \in [-4\pi, 4\pi]$ и $\dot{\theta}_2 \in [-9\pi, 9\pi]$. Константы имели значения соответственно $m_1 = m_2 = 1$ (массы стержней), $l_{c1} = l_{c2} = 0,5$ (длина до центра масс стержней), $I_1 = I_2 = 1$ (моменты инерции стержней) и $g = 9,8$ (ускорение свободного падения)

эпизод, дается вознаграждение -1 . Коэффициент приведения γ равен 1 . Таким образом, оптимальной ценностью $V^*(s)$ каждого состояния s является минимальное время достижения цели (целое число шагов), начиная из данного состояния.

В работе [Sutton (1996)] задача раскачивания акробота рассматривалась как оперативная и безмодельная. Несмотря на то что осуществлялось моделирование акробота, его модель никаким образом не могла использоваться агентом/оператором. Обучение и взаимодействие происходили, как будто использовался физически реальный акробот. В начале каждого эпизода оба стержня робота были направлены строго вертикально вниз, находясь в покое. Агент обучения с подкреплением воздействовал моментами на шарнир до тех пор, пока цель не достигалась, что, в конце концов, всегда происходило. После этого акробот приводился в свое первоначальное состояние, и начинался новый эпизод.

Для обучения применялся алгоритм SARSA(λ) с линейной аппроксимацией функций, мозаичным кодированием и замещающими следами, вид которого приведен на рис. 8.8. Имея небольшой дискретный набор действий, удобно использовать отдельные наборы фрагментов для каждого действия. Следующим шагом необходимо было выбрать непрерывные переменные, с помощью которых задавалось состояние. Опытный разработчик, вероятно, задал бы состояние при помощи угловой позиции и скорости центра масс второго стержня, что сделало бы решение более простым и позволило бы в дальнейшем проводить обобщения. Но поскольку это было лишь тестовой задачей, было использовано более прямое представление состояния с помощью позиций и скоростей стержней: θ_1 , $\dot{\theta}_1$, θ_2 и $\dot{\theta}_2$. Обе угловые скорости ограничены физикой акробота (см. рис. 11.5), а два угла ограничены в силу геометрии системы интервалом $[0, 2\pi]$. Таким образом, пространство состояний данной задачи представляет собой ограниченную прямоугольную четырехмерную область.

Возникает вопрос, какое разбиение на фрагменты использовать. Существует много возможных вариантов, рассмотренных в гл. 8. Одним из таких вариантов является использование полной сетки, проводя разбиение четырехмерного пространства по всем пространствам на много маленьких четырехмерных фрагментов. В качестве альтернативы можно было бы провести разбиение только по одному измерению, получив тем самым гиперплоские полосы. В данном случае необходимо выбрать пространство, подвергающееся разбиению. Кроме этого, во всех случаях необходимо выбрать ширину разбиения, количество фрагментов каждого типа и, если используется несколько разных разбиений, определить, каким образом они будут соединяться между собой. Также можно бы-

ло бы провести разбиение по двум или трем измерениям, получив другие фрагменты. Например, если бы ожидалось, что скорости двух стержней будут сильно зависеть друг от друга, тем самым, меняя при этом свои значения, можно было бы использовать множество фрагментов, которые содержали бы оба данных измерения. Если бы особое значение придавалось области около нулевой скорости, то данную область можно было бы разбить на более мелкие фрагменты.

Саттон применял различные несложные разбиения. Каждое из четырех измерений было разделено на шесть равных интервалов. Седьмой интервал был определен для угловых скоростей таким образом, чтобы разбиения можно было смещать на произвольную часть интервала во всех измерениях (см. гл. 8, подраздел «Мозаичное кодирование»). Из 48 использовавшихся мозаичных разбиений 12 осуществлялись во всех четырех измерениях, как рассмотрено выше, разбивая каждое пространство на $6 \times 7 \times 6 \times 7 = 1764$ фрагмента. Другие 12 осуществлялись в трех измерениях (по 3 произвольно сдвинутых мозаичных разбиения для каждого из 4 наборов из трех измерений), и еще 12 осуществлялись в двух измерениях (по 2 мозаичных разбиения для каждого из 6 наборов из двух измерений). Наконец, оставшиеся 12 мозаичных разбиений относились только к одному измерению (по 3 разбиения для каждого из 4 измерений). В результате каждому действию соответствовало примерно 25 000 фрагментов. Это достаточно небольшое число позволяло избежать хэширования. Все мозаичные разбиения смещались на произвольную часть интервала в любом измерении, где это было необходимо.

Остальным параметрам алгоритма обучения были заданы значения соответственно

$$\alpha = \frac{0,2}{48}, \quad \lambda = 0,9, \quad \varepsilon = 0 \quad \text{и} \quad Q_0 = 0.$$

В данной задаче выгоднее всего было использование жадной стратегии ($\varepsilon = 0$), так как для правильной работы алгоритма необходимо иметь длинные последовательности правильных действий. Одно действие, направленное на изучение, могло испортить всю последовательность правильных действий. Изучение, в свою очередь, стимулировалось заданием изначально оптимистичных ценностей действий с нулем в качестве минимального значения. Как было показано в разд. 2.7 и примере 8.2, при этом агента не устраивают любые вознаграждения, полученные им вначале, и он пытается пробовать новые действия.

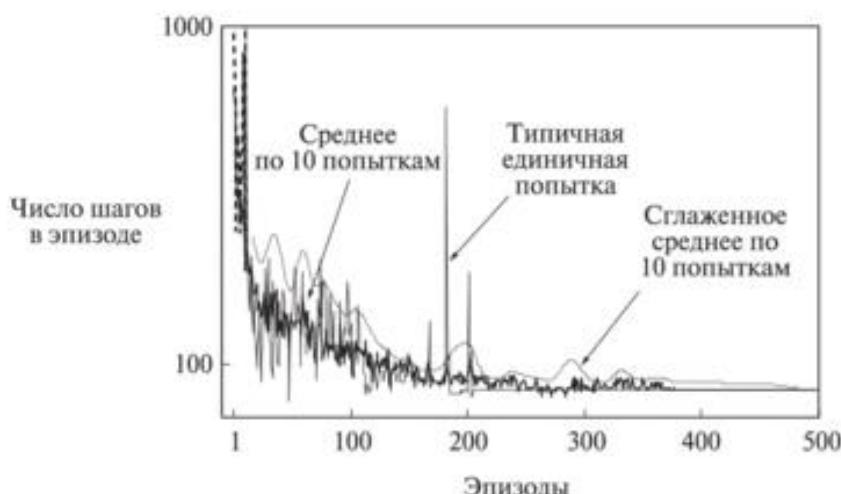


Рис. 11.6. Кривые обучения для метода SARSA(λ) в задаче с акроботом

На рис. 11.6 изображены кривые обучения, соответствующие описанному выше алгоритму обучения в задаче акробата. На примере выделенной кривой, соответствующей одной попытке, видно, что отдельные эпизоды иногда были чрезвычайно продолжительными. В таких эпизодах акробат обычно многократно поворачивался вокруг своего второго шарнира, тогда как первый шарнир лишь немного поворачивался относительно вертикального положения.

Несмотря на то что этот процесс часто длился в течение большого числа временных шагов, он всегда, в конце концов, завершался с уменьшением ценностей действий. Все попытки заканчивались нахождением стратегии, эффективно решающей данную задачу, что обычно занимало около 75 шагов. Типичное решение показано на рис. 11.7. Вначале акробат симметрично раскачивался туда и обратно, при этом второй стержень был всегда направлен вниз. Затем, как только система получала достаточно энергии, второй стержень качался вверх и достигал необходимой высоты.

11.4. Управление лифтом

Каждому из нас, вероятно, приходилось ожидать лифт. Мы нажимаем кнопку и ожидаем, когда приедет лифт в нужном нам направлении. Иногда приходится ждать некоторое время, особенно если слишком много пассажиров и мало лифтов. Время нашего ожидания зависит от стратегии управления, которой руководствуется лифт при выборе направления движения. Например, если пассажиры на нескольких этажах ожидают прибытия лифта, куда приедет лифт в первую очередь?

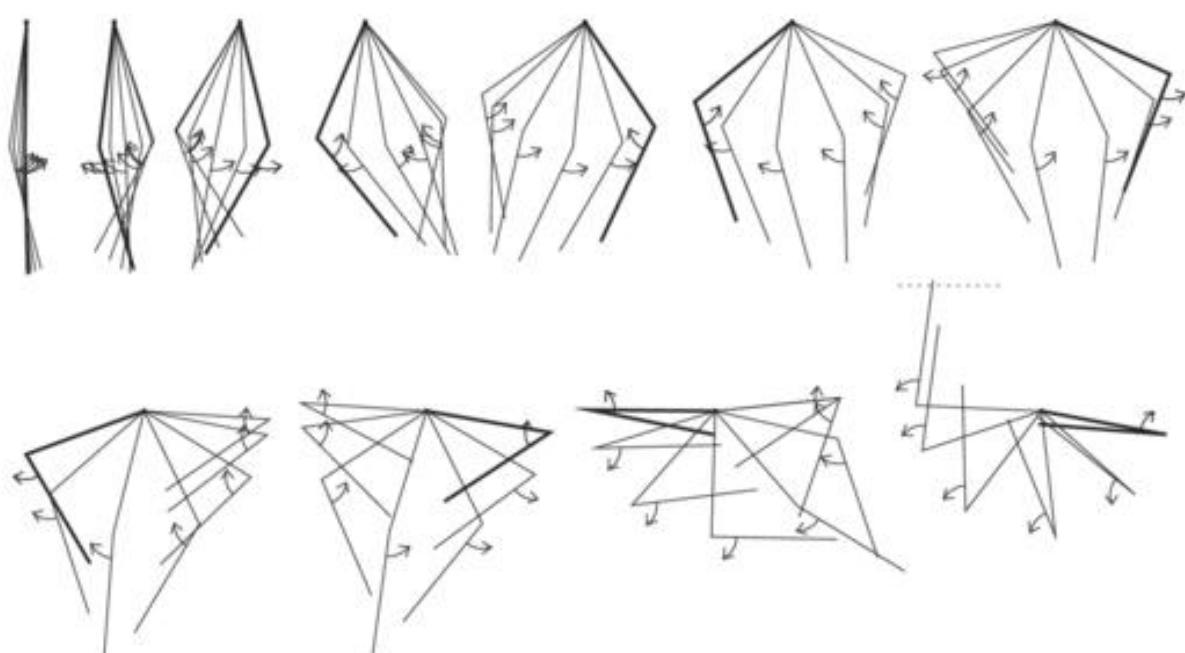


Рис. 11.7. Типичный процесс обучения акробата. Каждая группа представляет собой серию последовательных позиций, начинающуюся с выделенной линии. Стрелками обозначены моменты, прикладываемые ко второму шарниру

Если никто не ожидает лифта, каким образом должны располагаться лифты, ожидая следующего вызова? Задача управления лифтом является хорошим примером представляющей собой практическую ценность стохастической задачи оптимального управления, которая слишком велика для решения классическими методами, такими как динамическое программирование.

В работах [Crites and Barto (1996); [Crites (1996)] изучалось применение методов обучения с подкреплением к системе с четырьмя лифтами и десятью этажами, показанной на рис. 11.8. Справа на рисунке изображены места вызова лифта и время его ожидания. Каждому лифту соответствует местоположение, направление и скорость движения, кроме этого, у каждого лифта есть набор кнопок, при помощи которых пассажир задает нужный ему этаж. Используя довольно грубую дискретизацию непрерывных переменных данной задачи, Крайтс и Барто получили, что система имеет примерно 10^{22} состояний. Такое большое пространство состояний исключает возможность применения классических методов динамического программирования, таких как итерация по ценностям. Даже если дублирование одного состояния занимало бы одну микросекунду, на одну полную прогонку в пространстве состояний потребовалось бы более 1000 лет.

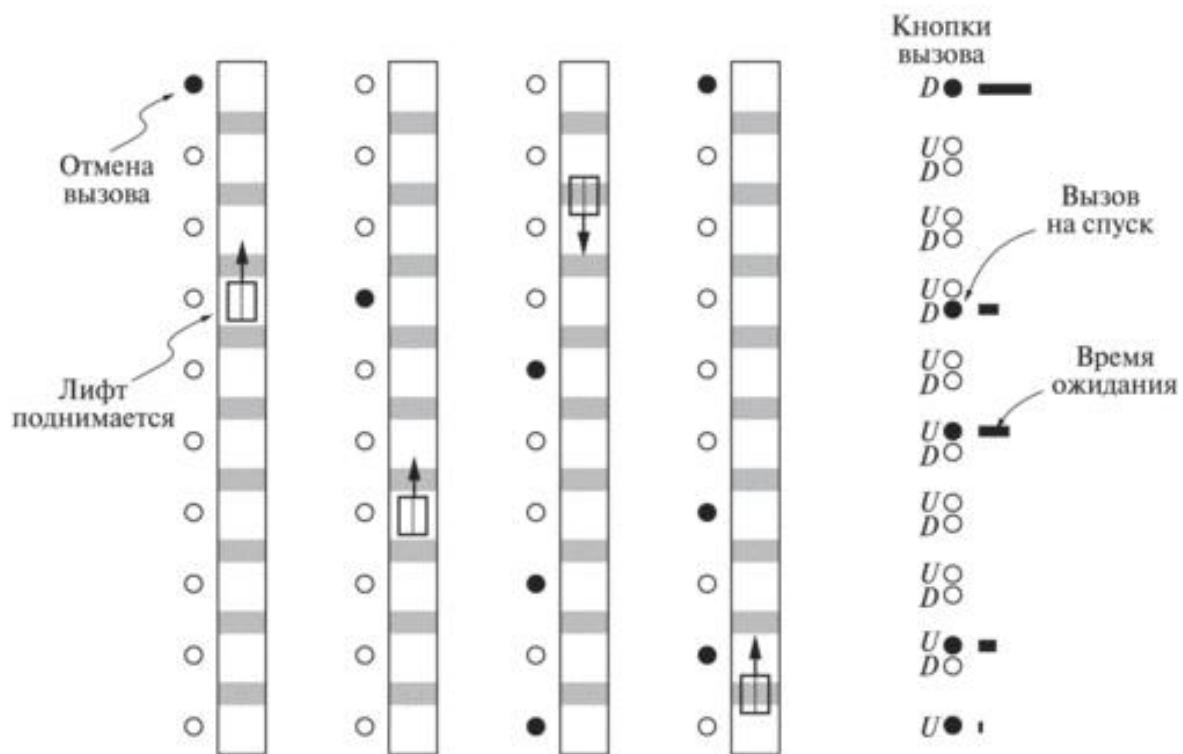


Рис. 11.8. Четыре лифта в десятиэтажном здании

На практике современные программы управления лифтами разрабатываются эвристически и испытываются с помощью имитаторов зданий. Имитаторы представляют собой достаточно сложные и подробные программы. Физика каждой кабины лифта моделируется в непрерывном времени с непрерывными переменными состояний. Прибытие пассажиров моделируется с помощью дискретных стохастических событий, происходящих с частотой, постоянно меняющейся в зависимости от пассажиропотока в данный моделируемый день. Очевидно, что временем наиболее интенсивного пассажиропотока и, следовательно, максимальной нагрузки на программу управления лифтами являются утренние и вечерние часы пик. По сути, программа управления разрабатывается именно для этих напряженных периодов времени.

Эффективность программы управления измеряется несколькими различными способами, в каждом из которых оценивается средний пассажиропоток, проходящий через систему. Средним временем ожидания является время, в течение которого пассажир ожидает прибытия лифта, средним временем системы является время, в течение которого пассажир добирается до нужного ему этажа. Еще одна часто встречающаяся статистика — процент пассажиров, чье время ожидания лифта превышает 60 секунд. Целью Крайтса и Барто было уменьшение среднего зна-

чения *квадрата времени ожидания*. Данный метод используется чаще всего, так как наряду со снижением времени ожидания стимулируется равномерное обслуживание пассажиров, без каких-либо предпочтений со стороны программы.

Крайтс и Барто применили одношаговую версию *Q*-обучения, модернизированную для наиболее выгодного использования особенностей данной задачи. Самая важная из них относилась к определению действия. Вначале каждый лифт принимал свои решения независимо от других лифтов. Затем на эти решения накладывался ряд ограничений. Во-первых, лифт, перевозящий пассажиров, не мог пропустить этаж, на котором кому-либо из них нужно выйти. Во-вторых, лифт не мог изменить направление движения, пока не выйдет последний пассажир, которому нужно это направление. Кроме этого, кабина лифта могла останавливаться на этаже, только если кому-либо из пассажиров необходимо войти или выйти из лифта, и она не могла остановиться, чтобы подобрать пассажиров, если на данном этаже уже остановился другой лифт. Наконец, если возникал выбор, двигаться вверх или вниз, лифт обязательно должен был ехать вверх (иначе в вечерние часы пик все лифты постоянно находились бы внизу). Последние три ограничения были введены в программу на основе предыдущих наблюдений для того, чтобы упростить задачу. В результате введения данных ограничений каждому лифту нужно было принимать небольшое количество простых решений. Единственным решением, которое необходимо было принимать, было решение о том, останавливаться или нет на проезжающем кабиной этаже, на котором находились ожидающие лифта пассажиры. Во всех остальных случаях необходимость принятия решения отсутствовала.

Тот факт, что каждый лифт принимает решения достаточно редко, позволяет использовать второе упрощение в данной задаче. Между теми моментами, когда необходимо вмешательство агента обучения, система совершает дискретные переходы из одного момента времени, в котором необходимо принять решение, в другой. Задача принятия решения с непрерывным временем, рассматриваемая как система с дискретным временем, известна как *квазимарковский* процесс принятия решений. С достаточной степенью точности данный процесс можно рассматривать так же, как и другие марковские процессы принятия решений, вводя вознаграждения при каждом дискретном переходе как интеграл по вознаграждениям за соответствующий непрерывный интервал времени. Представление о выгоде, в свою очередь, превращается из приведенной суммы будущих вознаграждений в приведенный *интеграл*

будущих вознаграждений:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad \text{превращается в} \quad R_t = \int_0^{\infty} e^{-\beta t} r_{t+\tau} d\tau,$$

где r_t в левой части является обычным непосредственным вознаграждением при дискретном времени, а $r_{t+\tau}$ в правой части является мгновенным вознаграждением в момент времени $t + \tau$. В задаче с лифтами вознаграждение при непрерывном времени представляет собой отрицательное значение суммы квадратов периодов ожидания всех пассажиров. Параметр $\beta > 0$ играет ту же роль, что и параметр приведения $\gamma \in [0, 1)$.

Теперь можно объяснить основную идею расширения Q -обучения до квазимарковской задачи принятия решений. Предположим, что система находится в состоянии s и осуществляет действие a в момент времени t_1 , а затем в момент времени t_2 в состоянии s' необходимо принять следующее решение. После такого дискретного перехода квазимарковское резервное копирование Q -обучения для табличной функции ценности действия Q будет иметь следующий вид:

$$\begin{aligned} Q(s, a) \leftarrow Q(s, a) + \alpha \left[\int_{t_1}^{t_2} e^{-\beta(\tau-t_1)} r_{\tau} d\tau + \right. \\ \left. + e^{-\beta(t_2-t_1)} \max_{a'} Q(s', a') - Q(s, a) \right]. \end{aligned}$$

Отметим, что величина $e^{-\beta(t_2-t_1)}$ выступает в роли переменного коэффициента приведения, который зависит от времени, прошедшего между событиями. Данный метод был предложен в работе [Bradtko and Duff (1995)].

Одна из сложностей заключалась в том, что определенное таким образом вознаграждение — отрицательное значение суммы квадратов времен ожидания — обычно не будет известна лифту во время его движения. Это происходит потому, что в реальной системе лифтов никогда неизвестно, сколько людей ожидает лифта на каком-либо из этажей, но известно лишь, сколько времени прошло с того момента, когда была нажата кнопка вызова лифта. Такая информация доступна при моделировании, и Крайтс и Барто использовали ее для достижения наилучших результатов. Они также экспериментировали с другим методом, в котором использовалась только информация, которая была бы известна в ситуации оперативного обучения с реальным набором лифтов. В этом

случае можно использовать информацию о том, как давно была нажата кнопка вызова лифта, совместно с оценкой интенсивности пассажиропотока для расчета *прогнозируемой* суммы квадратов времен ожидания для каждого этажа. Использование данного алгоритма измерения вознаграждений оказалось почти столь же эффективным, как и использование фактической суммы квадратов времен ожидания.

При аппроксимации функций для представления функции ценности действия использовалась нелинейная нейронная сеть, обучаемая методом обратного распространения ошибки обучения. Крайтс и Барто экспериментировали с множеством различных способов представления состояний для сети. После продолжительных исследований наилучшие результаты были получены при использовании 47 входных элементов, 20 скрытых элементов и двух выходных элементов, по одному для каждого действия. Оказалось, что способ представления состояния посредством входных элементов имел определяющее значение для эффективности обучения. Среди 47 входных элементов были следующие:

- 18 элементов. Два элемента содержали информацию о каждой из девяти наружных кнопок вызова лифта вниз. Значения одного элемента были действительными и выражали время, прошедшее с момента нажатия кнопки, а второй, бинарный элемент был активным до момента нажатия кнопки.
- 16 элементов. По одному элементу для каждого возможного местоположения и направления движения кабины, от которой требовалось принятие решения. В частности один из этих элементов был активен в любой момент времени.
- 10 элементов. Местоположение остальных лифтов по 10 этажам. Каждый лифт имел свою «информацию-отпечаток», которая зависела от скорости и направления его движения. Например, остановившийся лифт вызвал активацию только того элемента, который соответствовал текущему этажу, но движущийся лифт вызывал активацию нескольких элементов, соответствующих этажам, по направлению к которым он двигался, с наиболее высоким уровнем активности элементов, соответствующих ближайшим этажам. Наличие информации о том, какая из остальных кабин занимала определенное местоположение, не предусматривалось.
- 1 элемент. Этот элемент был активен, если лифт, от которого требовалось принятие решения, находился на верхнем этаже с ожидающим пассажиром.

- 1 элемент. Этот элемент был активен, если лифт, от которого требовалось принятие решения, находился на этаже с пассажиром, который ожидал в течение наиболее длительного времени.
- 1 элемент. Элемент нарушения равновесия был всегда активен.

Использовалось две структуры. В структуре RL1 каждому лифту соответствовала своя собственная функция ценности действия и своя собственная нейронная сеть. В структуре RL2 была только одна сеть и одна функция ценности действия, при этом при обучении сети учитывался опыт всех четырех лифтов. В обоих случаях каждый лифт принимал свои решения независимо от других лифтов, но все они получали один и тот же сигнал вознаграждения. Это вносило дополнительную стохастичность, так как вознаграждение каждого лифта отчасти зависело от действий других лифтов, которые он не мог контролировать. В структуре, в которой каждый лифт имел свою собственную функцию ценности действий, разные лифты могли придерживаться различной стратегии (хотя, по сути, они должны стремиться к одной стратегии). С другой стороны, структура с общей функцией ценности действий могла обучаться быстрее, так как она одновременно использовала опыт всех лифтов. Несмотря на то что обучение системы моделировалось, в данном случае снижение времени обучения играло главную роль. Обучение методами обучения с подкреплением проводилось на протяжении четырех дней компьютерного времени на 100 мегагерцевых процессорах (что соответствовало примерно 60 000 часам моделируемого времени). Несмотря на то что это значительное количество вычислений, оно чрезвычайно мало по сравнению тем, которое потребовалось бы любому из стандартных алгоритмов динамического программирования.

Обучение сетей происходило при помощи моделирования множества вечерних часов пик, во время которых принимались решения оперативного управления с использованием формируемых функций ценности действий. Крайтс и Барто использовали для выбора действий softmax-метод Гиббса, описанный в разд. 2.3, постепенно понижая «температуру» по ходу обучения. Нулевая температура использовалась во время тестов, в ходе которых оценивалась эффективность выработанной управляющей программы.

На рис. 11.9 представлены результаты работы нескольких управляющих программ во время смоделированного вечернего часа пик, который исследователи называли *нижним пиком* интенсивности пассажиропотока. Программы управления включали в себя методы, аналогичные тем, что на практике применяются в данной отрасли, множество эври-

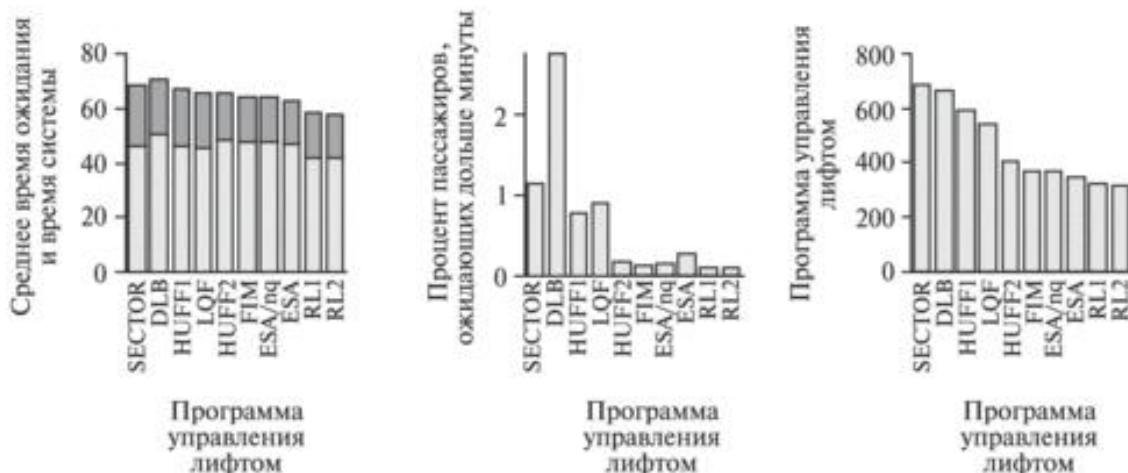


Рис. 11.9. Сравнение программ управления лифтами. Программа управления SECTOR аналогична программам, применяемым в реальных системах. Программы управления RL1 и RL2 были получены при помощи обучения с подкреплением

стических методов, изощренные алгоритмы исследования, многократно повторявшие сложные оперативные алгоритмы оптимизации (см. работу [Bao et al. (1994)]), кроме того, две программы управления обучались, используя две описанные структуры обучения с подкреплением. Согласно всем оценкам эффективности, управляющие программы обучения с подкреплением показывали лучшие результаты. Несмотря на то что оптимальная стратегия для данной задачи неизвестна, а современный технический уровень сложно точно определить вследствие того, что детали таких программ управления обычно не раскрываются, полученные программы управления показывают очень хороший результат.

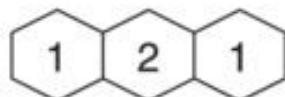
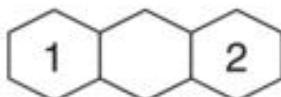
11.5. Динамическое распределение каналов

Важной задачей при управлении системой сотовой связи является проблема эффективного использования доступного диапазона частот для предоставления услуг наибольшему числу пользователей. Данная проблема обостряется со стремительным ростом числа используемых мобильных телефонов. Здесь рассматривается исследование [Singh and Bertsekas (1997)], в котором применяется обучение с подкреплением к данной задаче.

Преимуществом мобильной связи является то, что канал связи — набор частот — может использоваться одновременно многими абонентами, если их телефонные аппараты физически находятся достаточно далеко друг от друга, чтобы они не интерферировали между собой. Минималь-

ное расстояние, на котором еще не возникает интерференции, называется *ограничением по использованию канала*. В сотовой телефонной системе зона предоставления услуг всегда разделена на некоторое количество областей, называемых сотами. В пределах каждой соты находится станция, которая управляет всеми вызовами внутри соты. Весь доступный частотный диапазон всегда разделен на некоторое количество каналов. Каналы затем должны быть распределены по сотам и по вызовам, совершающимся внутри сот, не нарушая при этом ограничения по использованию канала. Существует множество способов реализации данной схемы, некоторые из которых лучше остальных с точки зрения надежности и своевременности предоставления каналов для новых вызовов или для вызовов, идущих из данной соты в другую, когда звонящий выходит за границы одной соты. Если для нового или переместившегося из соседней соты вызова нет доступных каналов, то вызов будет потерян или заблокирован. Синг и Бертсекас рассматривали задачу распределения каналов таким образом, чтобы число заблокированных вызовов было минимальным.

Разберем суть задачи на простом примере. Рассмотрим ситуацию, когда на три соты приходится два канала. Соты расположены последовательно одна за другой, и при этом в соседних сотах не может использоваться один канал без нарушения ограничения по использованию канала. Если в левую соту поступает вызов по каналу 1, в то время как в правой соте обслуживается вызов по каналу 2, как показано на левом рисунке, то любой входящий вызов в среднюю соту должен быть блокирован.



Очевидно, что в этой ситуации правая и левая соты должны использовать для вызовов канал 1. Тогда для нового входящего вызова в среднюю соту может быть использован канал 2 без нарушения ограничения на использование канала. Такие коллизии и возможные способы оптимизации типичны для задач распределения каналов. В близких к реальным ситуациях с множеством сот, каналов и вызовов и с высокой степенью неопределенности времени и места возможных входящих и исходящих вызовов задача минимизации блокируемых вызовов при распределении каналов может оказаться чрезвычайно сложной.

Согласно наиболее простому подходу к решению данной задачи, каждой соте ставится в соответствие свой канал таким образом, чтобы огра-

ничение на использование канала не нарушалось, даже если все каналы во всех сотах задействованы одновременно. Это называется методом *фиксированного распределения*. В методе *динамического распределения*, с другой стороны, все каналы потенциально доступны для всех сот и распределяются по сотам динамически по мере поступления вызовов. Если данный алгоритм реализован правильно, то он способен обеспечить преимущество временных изменений в пространственном и временном распределениях вызовов с целью обслужить большее количество абонентов. Например, если много вызовов сконцентрировано в небольшом количестве сот, в данных сотах может быть задействовано большее количество каналов без увеличения числа блокируемых вызовов в менее загруженных сотах.

Задача распределения каналов может быть сформулирована как полумарковский процесс принятия решений, по аналогии с задачей управления лифтами в предыдущем разделе. Состояние в полу-МППР формулировке имеет две компоненты. Первая компонента содержит в себе информацию о полной конфигурации всей сотовой системы, храня для каждой соты текущее состояние (занят или свободен) каждого канала данной соты. Типичная сотовая система с 49 сотами и 70 каналами имеет 70^{49} конфигураций. Такое огромное число конфигураций исключает возможность использования традиционных методов динамического программирования. Вторая компонента состояния является показателем того, какого типа событие вызвало переход состояния: новый вызов, завершение вызова или переадресация вызова. Данная компонента определяет возможные действия. Когда поступает новый вызов, возможными действиями являются либо выделение свободного канала, либо блокировка вызова при отсутствии свободных каналов. При завершении вызова, т. е. когда абонент завершает соединение, системе разрешено перераспределять используемые в данной соте каналы таким образом, чтобы создать наилучшую конфигурацию. В момент времени t непосредственным вознаграждением r_t является количество осуществляемых в данный момент вызовов, выгода при этом равна

$$R_t = \int_0^{\infty} e^{-\beta t} r_{t+\tau} d\tau,$$

где параметр $\beta > 0$ играет ту же роль, что и параметр приведения γ . Максимизация математического ожидания данной выгода, по сути, является минимизацией ожидаемого (приведенного) количества заблокированных вызовов в интервале времени, стремящемся к бесконечности.

Это еще одна задача, которую можно значительно упростить, если рассматривать ее с точки зрения последующих состояний (см. разд. 6.8). Для каждого состояния и действия мгновенным результатом является новая конфигурация, т. е. последующее состояние. Функция ценности формируется на основе только этих конфигураций. Для осуществления выбора среди возможных действий конфигурация должна была быть определена и оценена. Затем выбиралось действие, результатом которого была конфигурация с наибольшим значением ценности. Например, когда в соту поступал входящий вызов, ему мог быть выделен любой свободный канал, при отсутствии свободных каналов вызов блокировался. Новую конфигурацию, являющуюся результатом каждого распределения, было легко рассчитать, так как она всегда представляла собой простую детерминированную последовательность распределения. При завершении вызова освободившийся канал снова был готов к использованию при поступающим вызовам. В этом случае действия по перераспределению освободившихся каналов по поступающим вызовам в данной соте также принимались во внимание. После этого вновь выбиралось действие, результатом которого была конфигурация с наибольшей оценкой ценности.

Для функции ценности применялась линейная аппроксимация: оценка ценности конфигурации представляла собой взвешенную сумму характеристик. Конфигурации были представлены двумя наборами характеристик: характеристика доступности каждой соты и характеристика загруженности каждой пары сота—канал. Для любой конфигурации характеристика доступности соты заключала в себе информацию о количестве дополнительных вызовов, которые эта сота могла принять без возникновения конфликтов, если остальная часть сот была бы заморожена в текущей конфигурации. Для любой заданной конфигурации характеристика загруженности пары сота-канал заключала в себе информацию о числе, которое данный канал использовал в данной конфигурации в радиусе четырех сот от данной. Все характеристики были отнормированы и принимали значения от -1 до 1 . Для корректировки весов использовался полумарковский вариант линейного TD(0) метода.

Синг и Бертsekas сравнивали три метода распределения каналов на модели, представляющей собой 7×7 сотовую матрицу с 70 каналами. Ограничение на использование канала состояло в том, что вызовы должны были находиться на расстоянии не менее трех сот друг от друга. Вызовы поступали в случайном порядке в соответствии с распределением Пуассона с различным средним для различных сот. Продолжительность вызовов определялась случайным образом в соответствии с экспоненци-

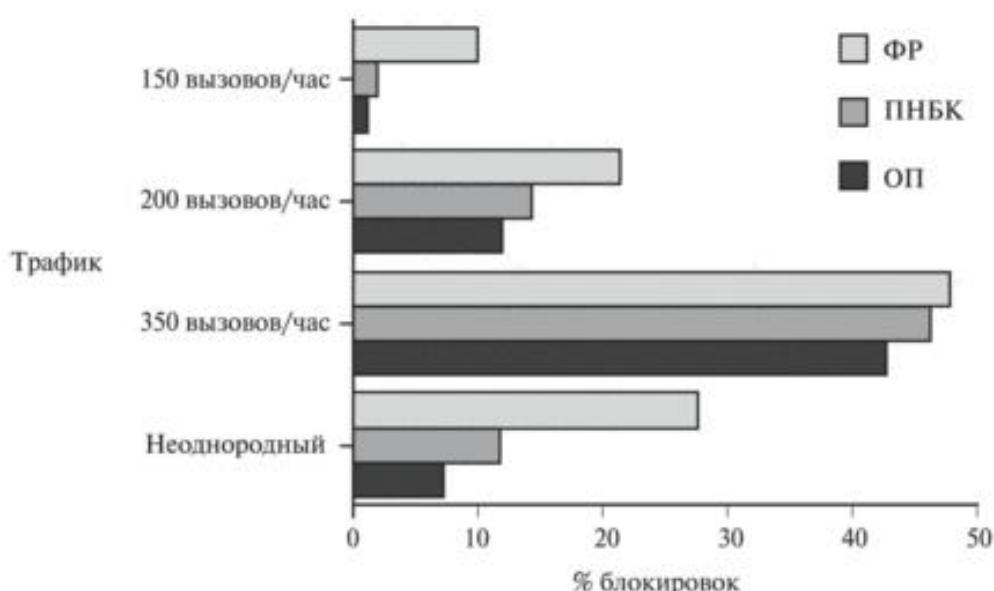


Рис. 11.10. Результаты ФР, ПНБК и ОП методов распределения каналов для различной средней интенсивности входящего трафика

альным распределением со средним, равным трем минутам. Сравнивались следующие методы: метод фиксированного распределения (ФР), метод динамического распределения, названный «перераспределением с направленной блокировкой каналов» (ПНБК), и метод обучения с подкреплением (ОП). ПНБК (см. работу [Zhang and Yum (1989)]) оказался наилучшим методом динамического распределения каналов из тех, что были описаны в литературе. Этот эвристический метод распределял каналы по сотам аналогично методу ФР, однако при необходимости канал мог быть позаимствован у соседних сот. Метод упорядочивал каналы в каждой соте и использовал этот порядок для определения канала, который необходимо было позаимствовать у соседних сот, и для динамического перераспределения вызовов по каналам внутри соты.

На рис. 11.10 представлены вероятности блокировки вызова, соответствующие данным методам, для средней интенсивности входящего трафика 150, 200 и 300 вызовов в час, и, кроме того, для случая, когда разным сотам соответствовали трафики разных интенсивностей. Метод обучения с подкреплением обучался оперативно. Приведенные результаты соответствуют его асимптотическим показателям, однако обучение при этом происходило стремительно. Метод ОП блокировал вызовы реже всех остальных методов при любых интенсивностях входящего трафика сразу после начала обучения. Отметим, что различия методов уменьшались с увеличением интенсивности трафика. Это ожидаемый результат, так как при насыщении системы трафиком для метода дина-

мического распределения вызовов оставалось все меньше возможностей использования наиболее эффективных схем. Тем не менее на практике наиболее важными являются результаты методов, которые они показывают, когда система не насыщена трафиком. В коммерческих целях сотовые телефонные системы строятся с достаточной пропускной способностью, при которой количество заблокированных вызовов редко достигает 10%.

В докладе [Nie and Haykin (1996)] также изучалось применение обучения с подкреплением к задаче динамического распределения каналов. Авторы сформулировали задачу несколько иначе, чем Синг и Бертсекас. Вместо того чтобы напрямую пытаться минимизировать вероятность блокировки вызова, их система пыталась минимизировать более косвенную величину, характеризующую эффективность системы. Каждой используемой конфигурации каналов присваивалась некоторая цена, которая зависела от расстояний между вызовами, использующими один канал. Конфигурации, в которых каналы использовались множеством вызовов, находящихся близко друг от друга, были предпочтительнее конфигураций, в которых использующие один канал вызовы находились друг от друга на большом расстоянии. Най и Хайкин сравнивали свою систему с методом MAXAVAIL (см. работу [Sivarajan, McEliece and Ketchum (1990)], считавшимся одним из лучших методов динамического распределения каналов. Для каждого нового вызова данный метод выбирал канал, который максимизировал общее число доступных каналов в системе. Най и Хайкин показали, что достигнутая при помощи такой системы обучения с подкреплением вероятность блокирования вызова была близка к вероятности, полученной методом MAXAVAIL, на модели с 49 сотами и 70 каналами в различных условиях. Ключевым здесь является тот факт, что получаемая при помощи обучения с подкреплением стратегия распределения каналов может быть реализована оперативно и намного более эффективно, чем в методе MAXAVAIL, который требует так много оперативных вычислений, что просто не может применяться к большим системам.

Рассмотренные в данном разделе примеры сейчас являются предметом исследования, а многие связанные с ними вопросы до сих пор не имеют ответов. Тем не менее на этих примерах можно убедиться, что существует много различных способов применения обучения с подкреплением к одним и тем же практическим задачам из реальной жизни. В ближайшем будущем мы ожидаем усовершенствования существующих и появления новых приложений обучения с подкреплением к задачам, относящимся к системам связи.

11.6. Задача планирования

В области промышленного производства, как и во многих других областях, требуется выполнение набора задач при одновременном соблюдении ограничений по времени и по ресурсам. Согласно временным ограничениям, одни задачи должны быть закончены перед тем, как начнется выполнение других; ограничения по ресурсам делают невозможным одновременное выполнение задач, требующих одних и тех же ресурсов (например, одна машина не может одновременно выполнять две задачи). Целью является создание расписания, определяющего, когда начинается выполнение каждой задачи и какие при этом будут задействованы ресурсы, для того чтобы не нарушить никакие из ограничений и затратить как можно меньше времени. Это является задачей планирования. Задача планирования обычно решается при помощи эвристических алгоритмов, которые учитывают преимущества конкретных свойств в каждом отдельном случае.

В работах [Zhang and Dietterich (1995, 1996); Zhang (1996)] было применено обучение с подкреплением к задаче планирования, так как отражающие специфику конкретной области эвристические алгоритмы требовали большого количества вычислительных и временных ресурсов. Была поставлена цель показать, как можно использовать обучение с подкреплением для быстрого формирования краткосрочных планов, удовлетворяющих всем ограничениям в конкретной области, и таким образом сократить число требуемых рабочих рук. Они обратились к решавшейся в NASA проблеме загрузки космического корабля (SSPP – Space Shuttle Payload Processing problem), в которой требуется составить план задач, необходимых для монтажа и тестирования единиц полезной нагрузки в грузом отсеке корабля. В случае с SSPP необходимо было спланировать от двух до шести полетов корабля, для подготовки которых требовалось решить от 34 до 164 задач. Примером задачи является ПРОВЕРКА-ПОСЛЕДОВАТЕЛЬНОСТИ-ЗАДАЧ-ПОЛЕТА, которая длится 7200 единиц времени и требует следующих ресурсов: два специалиста по контролю качества, два техника, один комплект автоматического испытательного оборудования ATE, одна система обработки, управления и отображения сигнала SPCDS и одна скоростная комплексная испытательная система HITS. Некоторые ресурсы разделены на группы, и если для решения задачи требуется больше одного ресурса конкретного типа, то эти ресурсы должны принадлежать одной и той же группе. Например, если для решения задачи требуется два специалиста по контролю качества, они оба должны входить в группу

специалистов по контролю качества, работающих в одной смене на требующемся участке работ. Формирование бесконфликтного плана работ, который удовлетворял бы всем временным и ресурсным ограничениям, не составляет большой сложности. Однако целью является формирование бесконфликтного плана работ наименьшей возможной продолжительности, что является намного более сложной задачей.

Как можно решить данную задачу, используя обучение с подкреплением? Задача планирования обычно формулируется как поиск в пространстве планов, который называется дискретной, или комбинаторной, задачей оптимизации. Обычный метод решения данной задачи последовательно генерирует планы, пытаясь улучшить каждый следующий план по сравнению с предыдущим, избавляясь от нарушений заданных ограничений и сокращая продолжительность задачи (метод поиска экстремума или локального поиска). Можно рассматривать это как неассоциативную задачу обучения с подкреплением, относящуюся к рассмотренному в гл. 2 типу задач с очень большим количеством возможных действий — всех возможных планов! Однако помимо проблемы большого количества действий любое решение, полученное данным способом, будет обладать еще одним недостатком: оно будет просто *отдельно взятым планом одного отдельно взятого* примера работ. В противоположность этому, в работах [Zhang and Dietterich (1995, 1996) требовалось, чтобы результатом их системы обучения была *стратегия*, которая могла бы быстро формировать эффективные планы для любой задачи SSPP. Они добивались того, чтобы данная стратегия накапливала опыт формирования планов в данной конкретной области.

Для формирования представления того, каким образом решить данную задачу, они обратились к уже существующему подходу к решению задачи SSPP — единственному подходу, используемому в NASA на тот момент времени, итеративному методу устранения недостатков, описанному в работе [Zwebel, Daun and Deale (1994)]. Отправной точкой является *план критического пути*, — план, который соответствует временным ограничениям, но игнорирует ресурсные ограничения. Данный план может быть сформирован следующим образом: каждая предшествующая запуску задача планируется настолько поздно, насколько это позволяют временные ограничения, а каждая задача после приземления планируется как можно раньше, в соответствии с данными ограничениями. Группы ресурсов определяются случайным образом. Для корректировки планов используются два типа операторов. Они могут быть применены к любой задаче, в которой нарушаются ресурсные ограничения. Оператор **Reassign-Pool** (Переопределить-Группу) изменяет группу, на-

значенную одному из ресурсов данной задачи. Данный тип оператора применяется, только если он может переопределить группу таким образом, чтобы удовлетворялись ресурсным требованиям. Оператор **Move** (Сдвинуть) сдвигает задачу на более раннее или более позднее время, в котором будут удовлетворяться ее ресурсные требования, и использует метод критического пути для перепланирования всех временных зависимостей в данной задаче.

На каждом шаге в итеративном методе поиска и устранения недостатков к текущему плану применяется один оператор, выбираемый в соответствии со следующими правилами. Находится наиболее ранняя задача, в которой нарушаются ресурсные ограничения, и к ней, если это возможно, применяется оператор **Reassign-Pool**. Если применяется больше одного оператора, т. е. если возможно несколько переназначений группы, то случайным образом выбирается одно из них. Если не применяется оператор **Reassign-Pool**, то применяется оператор **Move**, который выбирается случайным образом на основе эвристики, согласно которой предпочтительны небольшие сдвиги задач, имеющих небольшое число временных зависимостей и ресурсные требования, близкие к тем, которые будут иметь место при сдвиге данной задачи в новую позицию в плане. После того как применен данный оператор, определяется число нарушений ограничений в получившемся плане. Решение о принятии или непринятии данного нового плана принимает смоделированная процедура нормализации. Если обозначить через ΔV число нарушений ограничений, устраниемых посредством исправления ошибок, то новый план принимается с вероятностью $e^{-\Delta V/T}$, где T представляет собой текущую температуру вычислений, которая постепенно убывает в процессе поиска. Если план принимается, то он становится текущим планом для следующей итерации; в противном случае алгоритм предпринимает очередную попытку исправить ошибки в старом плане, что обычно приводит к другим результатам, благодаря наличию элемента случайности при выборе решений. Поиск останавливается, когда удовлетворяются все ограничения. Наименьший период выполнения задачи получают, повторяя данный алгоритм несколько раз и выбирая при этом наименее протяженный по времени бесконфликтный план.

В работах [Zhang and Dietterich (1995, 1996)] рассматривались планы в целом как состояния обучения с подкреплением. Действиями были подходящие операторы **Reassign-Pool** и **Move**, количество которых обычно было порядка 20. Задача рассматривалась как эпизодная. Каждый эпизод начинался с того же плана критического пути, с которого начинался итеративный алгоритм исправления ошибок, и за-

канчивался, когда находился план, не нарушающий никаких ограничений. Начальное состояние — план критического пути — обозначен s_0 . Вознаграждения определялись таким образом, чтобы стимулировать быстрое формирование бесконфликтных планов наименьшей продолжительности. Система получала небольшое отрицательное вознаграждение ($-0,001$) на каждом временном шаге, результатом которого был план, по-прежнему нарушащий ограничение. Это стимулировало агента быстро находить бесконфликтные планы, т. е. при небольшом количестве исправлений ошибок в состоянии s_0 . Стимулирование системы к формированию краткосрочных планов представляет собой более сложную задачу, так как понятие краткосрочности плана зависит от конкретной SSPP-задачи. Самый краткосрочный план в сложном случае со множеством задач и ограничений будет длиннее аналогичного плана в более простом случае. В [Zhang and Dietterich (1995)] выведена формула для коэффициента потребления ресурсов (КПР), который представлял собой независимую от конкретной задачи меру продолжительности плана. Для учета сложностей, связанных с конкретными задачами, в формулу входит величина, характеризующая распределение ресурсов в состоянии s_0 . Поскольку с увеличением продолжительности плана возрастает соответствующий ему КПР, взятая с отрицательным знаком величина КПР окончательного бесконфликтного плана использовалась в качестве вознаграждения в конце каждого эпизода. Если на пути от начального плана s до конечного бесконфликтного плана s_f было произведено N исправлений ошибок, то при заданной таким образом функции вознаграждение было равно

$$-\text{КПР}(s_f) - 0,001(N - 1).$$

Построение данной функции вознаграждений преследовало две цели: стимулировать систему находить бесконфликтные планы небольшой продолжительности и выполнять данную задачу как можно быстрее. Однако в действительности система обучения с подкреплением имеет только одну цель — максимизировать ожидаемую выгоду; таким образом, конкретные значения вознаграждений определяют, каким образом система будет выполнять обе поставленные задачи. При мгновенном вознаграждении, равном $-0,001$, система будет считать, что одно исправление ошибки, т. е. один шаг в процессе планирования, стоит $0,001$ часть КПР. Таким образом, если, например, перед оптимальной стратегией стоит выбор, получить бесконфликтный план при помощи одного исправления ошибки или двух, дополнительное исправление будет вы-

полнено, только если это повлечет за собой уменьшение окончательного КПР больше, чем на тысячную долю.

Для нахождения функции ценности в [Zhang and Dietterich (1995)] использовался алгоритм TD(λ). Аппроксимация функций формировалась при помощи многослойной нейронной сети методом обратного распространения ошибки TD-обучения. Действия выбирались согласно ε -жадной стратегии, при убывающем по мере обучения параметре ε . Для нахождения жадного действия использовался одношаговый предварительный просмотр. Знания авторов позволили им легко прогнозировать планы, получавшиеся в результате каждого операции по исправлению ошибки. Они экспериментировали с несколькими различными вариантами данного алгоритма, пытаясь улучшить его результаты. Как один из вариантов, после каждого эпизода TD(λ) алгоритм применялся в обратном направлении, распространяя при этом следы приемлемости не на прошлые состояния, но на будущие. Результаты этих экспериментов показали, что такой алгоритм оказался более точным и эффективным, нежели прямое обучение. В том случае, если TD-ошибка была значительной, иногда использовались многократные корректировки весовых коэффициентов в сети. Это эквивалентно динамическому изменению параметра размера шага в зависимости от ошибки в процессе обучения.

Авторы также экспериментировали с методикой *повторного воспроизведения опыта*, предложенной в работе [Lin (1992)]. В любой точке обучения агент запоминал наилучший эпизод в данной точке. После каждого четырех эпизодов он повторно воспроизводил запомненный эпизод, изучая его, как если бы это был новый эпизод. В начале обучения в работе [Zhang and Dietterich (1995)] аналогичным образом позволялось системе обучаться на основе эпизодов, сгенерированных хорошей программой-планировщиком, а затем повторно воспроизводить их в процессе обучения. Для того чтобы ускорить процесс предварительного просмотра в больших задачах, которые обычно имеют фактор ветвления, примерно равный 20, использовался вариант алгоритма, названный *жадным поиском по случайной выборке*, который оценивал жадное действие, рассматривая только случайные выборки действий, увеличивая объем выборки до тех пор, пока жадное действие данной выборки не оказывалось истинным жадным действием с заранее заданным уровнем доверия. В заключение, обнаружив, что обучение может значительно замедлиться при наличии чрезмерного количества циклов в процессе планирования, они внедрили в систему процесс проверки на наличие циклов, и, таким образом, при обнаружении цикла система должна была корректировать свои действия. Несмотря на то что все эти методы,

вероятно, повышали эффективность обучения, остается неясным, насколько значительным был их вклад в успех данной системы.

В [Zhang and Dietterich (1995, 1996)] проводились эксперименты с двумя разными структурами сети. В первом варианте их сети каждый план был представлен в виде 20 заданных вручную свойств. Для определения данных свойств они изучали небольшие задачи планирования с целью найти свойства, при помощи которых можно было спрогнозировать КПР. Например, исследование небольших задач показало, что только четыре группы ресурсов могли стать причиной проблем с распределением ресурсов. Были рассчитаны среднее и стандартное отклонение каждой из этих неиспользуемых частей ресурсных групп, результатом чего стали 8 действительнозначных свойств. Два других свойства представляли собой КПР текущего плана и процент от времени, в течение которого нарушались ресурсные ограничения. В сети имелось 20 входных элементов, по одному на каждое свойство, скрытый слой из 40 сигмоидальных элементов и выходной слой из 8 сигмоидальных элементов. Выходные элементы содержали информацию о ценности плана, используя код, в котором, по сути, ценность была представлена позицией пика активности восьми элементов. Используя подходящую TD-ошибку, весовые коэффициенты сети корректировались при помощи обратного распространения ошибки обучения методом многократной корректировки весовых коэффициентов, который был упомянут выше.

Во втором варианте системы ([Zhang and Dietterich (1996)] использовалась более сложная нейронная сеть с временной задержкой (НСВЗ), заимствованная из области исследований по распознаванию речи [Lang, Waibel and Hinton (1990)]. В данной версии системы каждый план разбивался на последовательность блоков (максимальные интервалы времени, в течение которых не менялись задачи и распределения ресурсов), а каждый блок представлялся набором свойств, аналогичных тем, что использовались в первой программе. Затем система прогоняла набор «ядерных» сетей сквозь совокупность блоков для создания более абстрактных свойств. Поскольку разные планы имели разное количество блоков, другой слой усреднял полученные абстрактные свойства по каждым трем блокам. Затем последний слой из 8 сигмоидальных выходных элементов задавал ценность плана, используя тот же код, что и первая версия системы. Всего данная сеть имела 1123 регулируемых весовых коэффициентов.

Был сконструирован набор из 100 искусственных задач, разделенный на подмножества, которые использовались для обучения, определения момента остановки обучения (группа достоверности) и завершающего

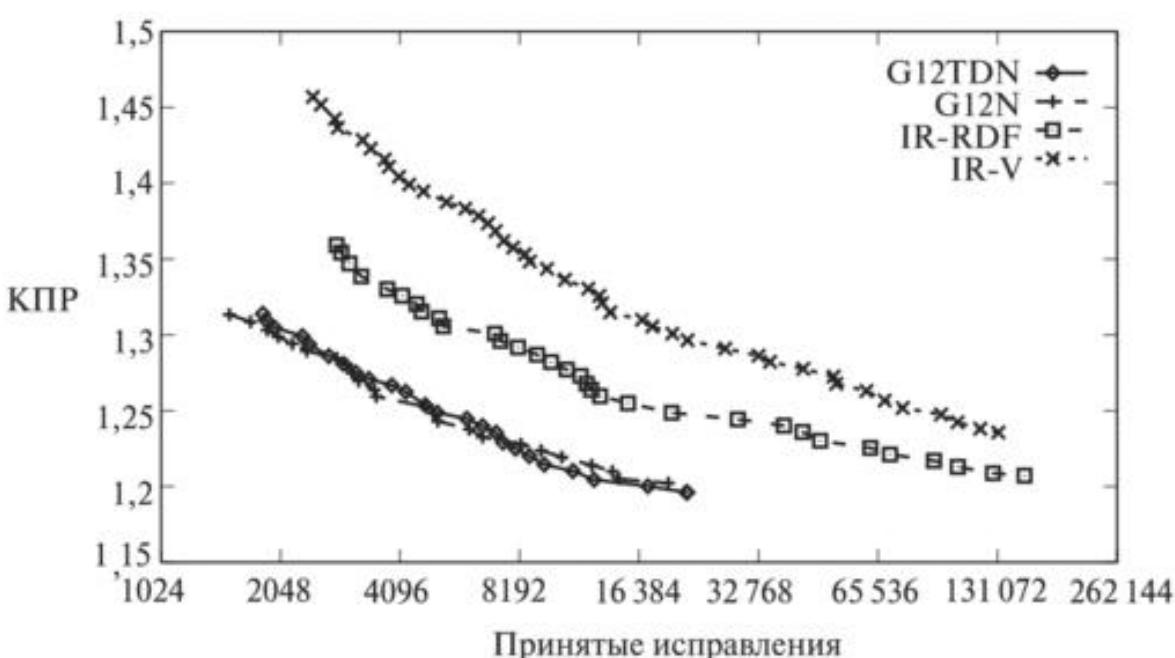


Рис. 11.11. Сравнение числа принятых исправлений ошибок в плане. Дан- ный график перепечатан из статьи [Zhang and Dietterich (1996)] с разрешения авторов

тестирования. В процессе обучения система тестировалась при помощи группы достоверности после каждого 100 эпизодов, и обучение останавливалось, когда результаты в группе достоверности переставали изменяться, что обычно занимало примерно 10 000 эпизодов. Сети проходили обучение с различными значениями параметра λ (0,2 и 0,7) и с тремя разными наборами данных обучения. В конце обучения сохранялось как окончательный набор весовых коэффициентов, так и набор весовых коэффициентов, при котором наблюдались наилучшие результаты в группе достоверности. Каждый набор весовых коэффициентов рассматривался как отдельная сеть, результатом чего оказались 12 сетей, каждая из которых соответствовала отдельному алгоритму планирования.

На рис. 11.11 показано сравнение усредненных результатов 12 сетей с временными задержками (TDNN — Time Delay Neural Network), обозначенных как G12TDN, с результатами двух вариантов итеративного алгоритма исправления ошибок [Zwebel, Dawn and Deale (1994)]. В первом варианте использовалось число нарушаемых ограничений как функция, минимизируемая смоделированной процедурой нормализации (IR-V). Во втором варианте использовалось значение КПР (IR-RDF). На рисунке также приведены результаты применения первой версии системы, в которой не использовалась сеть TDNN (G12N). Соответству-

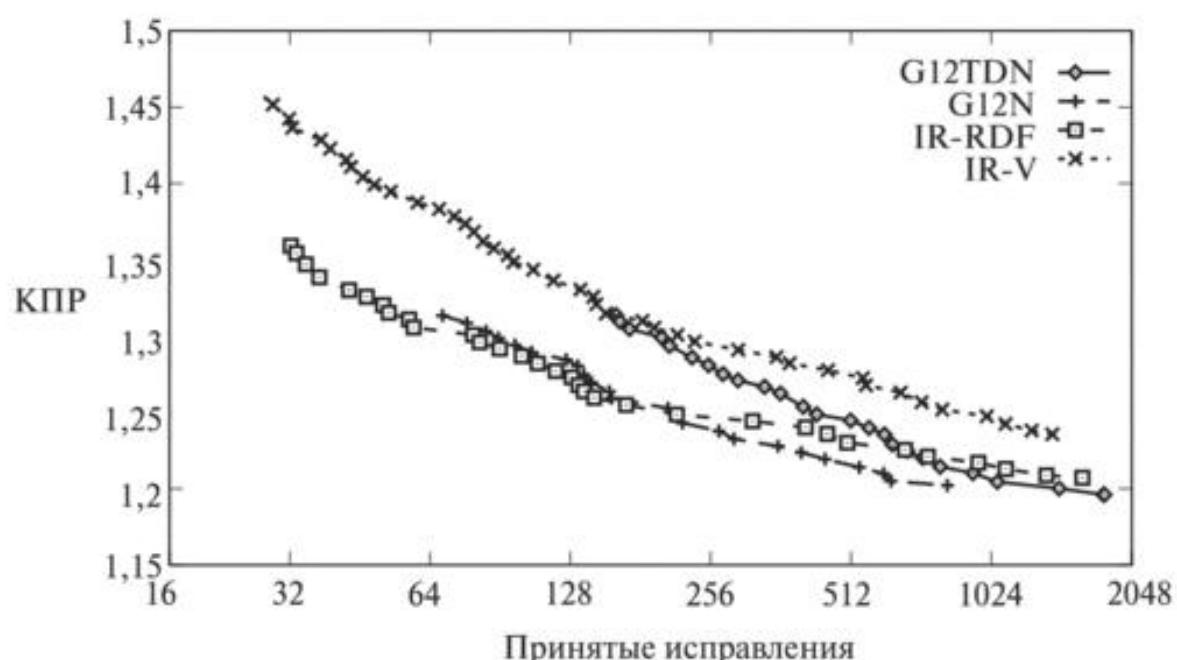


Рис. 11.12. Сравнение времени работы центрального процессора. Данный график перепечатан из статьи [Zhang and Dietterich (1996)] с разрешения авторов

ющее наилучшему плану среднее значение КПР, найденное при помощи многократного запуска алгоритма, представлено на графике в виде зависимости от общего числа исправления ошибок в плане (в логарифмическом масштабе). Эти результаты показывают, что данная система обучения позволяла получать алгоритмы планирования, которые требовали намного меньшего количества исправлений ошибок для формирования бесконфликтных планов такого же уровня качества, что и планы, найденные при помощи итеративного алгоритма исправления ошибок. На рис. 11.12 сравнивается машинное время, которое требуется каждому алгоритму планирования для формирования планов с различными КПР. В соответствии с данной мерой эффективности алгоритма наилучшее сочетание машинного времени и качества плана предлагает алгоритм, не использующий TDNN (G12N). Алгоритм TDNN (G12TDN) тратил много времени на процесс ядерного сканирования, однако в [Zhang and Dietterich (1995, 1996)] отмечается, что существует множество способов заставить данный алгоритм действовать быстрее.

Исходя из представленных результатов, нельзя сделать вывод о явном преимуществе применения обучения с подкреплением в задачах планирования и в других сложных задачах поиска. Однако ясно, что возможно использование методов обучения с подкреплением для улуч-

шения эффективности поиска. Система планирования из [Zhang and Dietterich (1995, 1996)] представляет собой первый успешный пример, в котором обучение с подкреплением использовалось в *пространстве планирования*, в котором действия являются сформированными планами (в данном случае, планами проведения работ), а действия представляют собой корректировки плана. Данное применение обучения с подкреплением является более абстрактным по сравнению с тем, что рассматривалось до этого. Обратите внимание на то, что в данном случае система не просто обучалась созданию *одного* хорошего плана (по сути, бесполезному умению); она обучалась быстрому формированию хороших планов для целого класса аналогичных задачах планирования. Очевидно, что, разрабатывая данный пример, авторы работ [Zhang and Dietterich (1995, 1996)] сами многому научились методом проб и ошибок. Но не следует забывать, что это было революционным исследованием новых возможностей обучения с подкреплением. Можно ожидать, что с накоплением опыта задачи аналогичного типа и сложности будут решаться быстрее и легче.

Список обозначений

- t — шаг дискретного времени
 T — момент времени завершения эпизода
 s_t — состояние в момент t
 a_t — действие в момент t
 r_t — вознаграждение в момент t , зависящее, как и s_t , от a_{t-1} и s_{t-1}
 R_t — выгода (совокупное приведенное вознаграждение) после момента t
 $R_t^{(n)}$ — n -шаговая выгода (разд. 7.1)
 R_t^λ — λ -выгода (разд. 7.2)
 π — стратегия, правило принятия решения
 $\pi(a)$ — вероятность исполнения действия a в задаче n -ру�ого бандита
 $\pi(s)$ — действие, выполненное в состоянии s при *детерминированной* стратегии π
 $\pi(s, a)$ — вероятность исполнения действия a в состоянии s при *стохастической* стратегии π
 S — множество всех нетерминальных состояний
 S^+ — множество всех состояний, включая завершающее состояние
 $A(s)$ — множество действий, возможных в состоянии s
 \mathbb{R} — множество вещественных чисел
 $P_{ss'}^a$ — вероятность перехода из состояния s в состояние s' при выполнении действия a
 $R_{ss'}^a$ — ожидаемое немедленное вознаграждение при переходе из s в s' при выполнении действия a
 $V^\pi(s)$ — ценность состояния s при стратегии π (ожидаемая выгода)
 $V^*(s)$ — ценность состояния s при оптимальной стратегии
 V, V_t — оценки для V^π или V^*
 $Q^\pi(s, a)$ — ценность выполняемого действия a в состоянии s при стратегии π

$Q^*(s, a)$ — ценность выполняемого действия a в состоянии s при оптимальной стратегии

Q, Q_t — оценки для Q^π или Q^*

$\vec{\theta}_t$ — вектор параметров, от которых зависит V_t или Q_t

$\vec{\phi}_s$ — вектор свойств, представляющих состояние s

δ_t — ошибка временного различия в момент t

$e_t(s)$ — след приемлемости для состояния s в момент t

$e_t(s, a)$ — след приемлемости для пары состояния–действие

γ — коэффициент приведения

ε — вероятность случайного действия в ε -жадной стратегии

α, β — параметры, задающие размер шага

λ — коэффициент затухания для следов приемлемости

Список литературы

- Agre, P. E. (1988). *The Dynamic Structure of Everyday Life*. Ph.D. thesis, Massachusetts Institute of Technology. AI-TR 1085, MIT Artificial Intelligence Laboratory.
- Agre, P. E. and Chapman, D. (1990). What are plans for? *Robotics and Autonomous Systems*, 6:17–34.
- Alhus, J. S. (1971). A theory of cerebellar function. *Mathematical Biosciences*, 10:25–61.
- Albus, J. S. (1981). *Brain, Behavior, and Robotics*. Byte Books, Peterborough, NH.
- Anderson, C. W. (1986). *Learning and Problem Solving with Multilayer Connectionist Systems*. Ph.D. thesis. University of Massachusetts, Amherst.
- Anderson, C. W. (1987). Strategy learning with multilayer connectionist representations. *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 103–114. Morgan Kaufmann, San Mateo, CA.
- Anderson, J. A., Silverstein, J. W. Ritz, S. A., and Jones, R. S. (1977). Distinctive features, categorical perception, and probability learning: Some applications of a neural model. *Psychological Review*, 84:413–451.
- Andreae, J. H. (1963). STELLA: A scheme for a learning machine. In *Proceedings of the 2nd IFAC Congress, Basle*, pp. 497–502. BuKerworths, London,
- Andreae, J. H. (1969a). A learning machine with monologue. *International Journal of Man-Machine Studies*. 1:1–20.
- Andreae, J. H. (1969b). Learning machines—a unified view. In A. R. Meetham and R. A. Hudson (eds.). *Encyclopedia of Information, Linguistics, and Control*, pp. 261–270. Pergamon, Oxford,
- Andreae, J. H. (1977). *Thinking with the Teachable Machine*. Academic Press, London.
- Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 30–37. Morgan Kaufmann, San Francisco.
- Rao, G. Cassandras, C. G., Djaferis, T. E., Gandhi, A. D. and Looze, D. P. (1994). Elevator dispatchers for down peak traffic. Technical report. ECE Department, University of Massachusetts, Amherst.
- Barnard, E. (1993). Temporal-difference methods and Markov models. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:357–365.

- Barto, A. G. (1985). Learning by statistical cooperation of self-interested neuron-like computing elements. *Human Neurobiology*, 4:229–256.
- Barto, A. G. (1986). Game-theoretic cooperativity in networks of self-interested units. In J. S. Denker (ed.), *Neural Networks for Computing*, pp. 41–46. American Institute of Physics, New York,
- Barto, A. G. (1990). ConnectioHist learning for control: An overview. In T. Miller, R. S. Sutton, and P. J. Werbos (eds.). *Neural Networks for Control*, pp. 5–58. MIT Press, Cambridge, MA.
- Barto, A. G. (1991). Some learning tasks from a control perspective. In L. Nadel and D. L. Stein (eds.), *1990 Lectures in Complex Systems*, pp. 195–223. Addison-Wesley, Redwood City, CA.
- Barto, A. G. (1992). Reinforcement learning and adaptive critic methods, in D. A. WhiLe and D. A. Sofge (eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pp. 469–191. Van Nostrand Reinhold, New York.
- Barto, A. G. (1995a). Adaptive critics and the basal ganglia. In J. C. Houk, J. L. Davis, and D. G. Beiser (eds.), *Models of Information Processing in the Basal Ganglia*, pp. 215–232. MIT Press, Cambridge, MA.
- Barto, A. G. (1995b), Reinforcement learning. In M. A. Arbib (ed.). *Handbook of Brain Theory and Neural Networks*, pp. 804–809. MIT Press, Cambridge, MA.
- Barto, A. G. and Anandan, P. (1985). Pattern recognizing stochastic learning automata, *IEEE Transactions on Systems, Man, and Cybernetics*, 15:360–375,
- Barto, A. G., and Anderson, C. W. (1985). Structural learning in connectionist systems. In *Program of the Seventh Annual Conference of the Cognitive Science Society*, pp. 43–54.
- Barto, A. G., Anderson, C. W., and Sutton, R. S. (1982). Synthesis of nonlinear control surfaces by a layered associative search network. *Biological Cybernetics*, 43:175–185.
- Barto, A. G., Bradtke, S. J., and Singh, S. P. (1991). Real-time learning and control using asynchronous dynamic programming. Technical Report 91–57. Department of Computer and Information Science, University of Massachusetts, Amherst.
- Barto, A. G. Bradtke, S. J., and Singh, S. P (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138.
- Barto, A. G., and Duff, M. (1994). Monte Carlo matrix inversion and reinforcement learning. In J. D. Cohen, G. Tesauro, and J. Alspector (eds.). *Advances in Neural Information Processing Systems: Proceedings of the 1993 Conference*, pp. 687–694. Morgan Kaufmann, San Francisco.
- Barto, A. G., and Jordan, M. I. (1987). Gradient following without back-propagation in layered networks. In M. Caudill and C. Butler (eds.), *Proceedings of the IEEE First Annual Conference on Neural Networks*, pp. II629–II636. SOS Printing, San Diego, CA.
- Barto, A. G., and Sutton, R. S. (1981a). Goal seeking components for adaptive intelligence: An initial assessment Technical Report AFWAL-TR-81-1070. Air

- Force Wright Aeronautical Laboratories/Avionics Laboratory, Wright-Patterson AFB, OH.
- Barto, A. G., and Sutton, R. S. (1981b). Landmark learning: An illustration of associative search. *Biological Cybernetics*, 42:1–8.
- Barto, A. G., and Sutton, R. S. (1982). Simulation of anticipatory responses in classical conditioning by a neuron-like adaptive element. *Behavioural Brain Research*, 4:221–235.
- Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835–846. Reprinted in J. A. Anderson and E. Rosenfeld (eds.), *Neurocomputing: Foundations of Research*, pp. 535–549. MIT Press, Cambridge, MA, 1988.
- Barto, A. G., Sutton, R. S., and Brouwer, P. S. (1981). Associative search network: A reinforcement learning associative memory. *Biological Cybernetics*, 40:201–211.
- Bellman, R. E. (1956). A problem in the sequential design of experiments. *Sankhya*, 16:221229.
- Bellman, R. E. (1957a). *Dynamic Programming*. Princeton University Press, Princeton.
- Bellman, R. E. (1957b). A Markov decision process. *Journal of Mathematical Mechanics*, 6:679–684.
- Bellman, R. E., and Dreyfus, S. E. (1959). Functional approximations and dynamic programming. *Mathematical Tables and Other Aids to Computation*, 13:247–251.
- Bellman, R. E., Kalaba, R., and Kotkin, B. (1973). Polynomial approximation—A new computational technique in dynamic programming: Allocation processes. *Mathematical Computation*, 17:155–161.
- Berry, D. A., and Fristedl, B. (1985). *Bandit Problems*. Chapman and Hall, London.
- Bertsekas, D. P. (1982). Distributed dynamic programming. *IEEE Transactions on Automatic Control*, 27:610–616.
- Bertsekas, D. P. (1983). Distributed asynchronous computation of fixed points. *Mathematical Programming*, 27:107–120.
- Bertsekas, D. P. (1987). *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Englewood Cliffs, NJ.
- Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA.
- Bertsekas, D. P., and Tsitsiklis, J. N. (1989). *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Englewood Cliffs, NJ.
- Bertsekas, D. P., and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.
- Biermann, A. W. Fairfield, J. R. C., and Beres, T. R. (1982). Signature table systems and learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 12:635–648.

- Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Clarendon, Oxford.
- Booker, L. B. (1982). *Intelligent Behavior as an Adaptation to the Task Environment*. Ph.D. thesis. University of Michigan, Ann Arbor.
- Boone, G. (1997). Minimum-time control of the acrobot. In *1997 International Conference on Robotics and Automation*, pp. 3281–3287. IEEE Robotics and Automation Society.
- Boutilier, C., Dearden, R., and Goldszmidt, M. (1995). Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1104–. Morgan Kaufmann.
- Boyan, J. A., and Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. In G. Tesauro, D. S. Touretzky, and T. Leen (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1994 Conference*, pp. 369–376. MIT Press, Cambridge, MA.
- Boyan, J. A., Moore, A. W., and Sutton, R. S. (eds.). (1995). *Proceedings of the Workshop on Value Function Approximation. Machine Learning Conference J995*. Technical Report CMU-CS-95-206. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.
- Bradtko, S. J. (1993). Reinforcement learning applied to linear quadratic regulation. In S. J. Hanson, J. D. Cowan, and C. L. Giles (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1992 Conference*, pp. 295–302. Morgan Kaufmann, San Mateo, CA.
- Bradtko, S. J. (1994). *Incremental Dynamic Programming for On-Line Adaptive Optimal Control*. Ph.D. thesis, University of Massachusetts, Amherst, Appeared as CMPSCI Technical Report 94-62.
- Bradtko, S. J., and Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57,
- S. J. Bradtko, B. E. Ydstie, and A. G. Barto (1994). Adaptive linear quadratic control using policy iteration. In *Proceedings of the American Control Conference*, pp. 3475–3479. American Automatic Control Council, Evanston, IL.
- Bradtko, S. J., and Duff, M. O. (1995). Reinforcement learning methods for continuous-time Markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1994 Conference*, pp. 393–400. MIT Press, Cambridge, MA.
- Bridle, J. S. (1990). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimates of parameters. In D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems: Proceedings of the 1989 Conference*, pp. 211–217. Morgan Kaufmann, San Mateo, CA.
- Broomhead, D. S., and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.
- Bryson, A. E., Jr. (1996). Optimal control—1950 to 1985. *IEEE Control Systems*, 13(3):26–33.

- Bush, R. R., and MosteNer, F. (1955). *Stochastic Models for Learning*. Wifey, New York.
- Byrne, J. H., Gingrich, K., J., and Baxter, D. A. (1990). Computational capabilities of single neurons: Relationship to simple forms of associative and nonassociative learning in *aplysia*. In R. D. Hawkins and G. H. Bower (eds.), *Computational Models of Learning*, pp. 31–63, Academic Press, New York,
- Campbell, D. T. (1960). Blind variation and selective survival as a general strategy in knowledge-processes. In M. C. Yovits and S. Cameron (eds.), *Self-Organizing Systems*, pp. 205–231. Pergamon, New York.
- Carlsrom, J., and Nordstrom, E. (1997). Control of self-similar ATM call traffic by reinforcement learning. In *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications 3*, pp. 54–62. Erlbaum, Hillsdale, NJ.
- Chapman, D., and Kaelbling, L. P. (1991). Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings of the Twelfth International Conference on Artificial Intelligence*, pp. 726–731. Morgan Kaufmann, San Mateo, CA,
- Chow, C.-S., and Tsitsiklis, J. N. (1991). An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Transactions on Automatic Control*, 36:898–914.
- Chrisman, L. (1992). Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 183–188. AAAI/MIT Press, Menlo Park, CA.
- Christensen, J., and Korf, R. E. (1986). A unified theory of heuristic evaluation functions and its application to learning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 148–152, Morgan Kaufmann, San Mateo, CA.
- Cichosz, P. (1995). Truncating temporal differences; On the efficient implementation of TD(A) for reinforcement learning. *Journal of Artificial Intelligence Research*, 2:287–318.
- Clark, W. A., and Farley, R. G. (1955). Generalization of pattern recognition in a self-organizing system. In *Proceedings of the 1955 Western Joint Computer Conference*, pp. 86–91.
- Clouse, J. (1996). *On Integrating Apprentice Learning and Reinforcement Learning TITLE2*, Ph.D. thesis, University of Massachusetts, Amherst. Appeared as CMPSCI Technical Report 96-026.
- Clouse, J., and Utgoff, P. (1992). A teaching method for reinforcement learning systems. In *Proceedings of the Ninth International Machine Learning Conference*, pp. 92–101. Morgan Kaufmann, San Mateo, CA.
- Colombetti, M., and Dorigo, M. (!994). Training agent to perform sequential behavior. *Adaptive Behavior*, 2(3):247–275.
- Connell, J. (1989). A colony architecture for an artificial creature. Technical Report AI-TR- ! 151. MIT Artificial Intelligence Laboratory, Cambridge, MA.

- Connell, J., and Mahadevan, S. (1993). *Robot Learning*. Kluwer Academic, Boston.
- Craik, K. J. W. (1943). *The Nature of Explanation*. Cambridge University Press, Cambridge.
- Crites, R. H. (1996). *Large-Scale Dynamic Optimization Using Teams of Reinforcement Learning Agents*. Ph.D. thesis. University of Massachusetts. Amherst.
- Crites, R. H., and Barto, A. G. (1996). Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (eds.). *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pp. 1017–1023. MIT Press, Cambridge, MA.
- Curtiss, J. H. (1954). A theoretical comparison of the efficiencies of two classical methods and a Monte Carlo method for computing one component of the solution of a set of linear algebraic equations. In H. A. Meyer (ed.), *Symposium on Monte Carlo Methods*, pp. 191–233. Wiley, New York.
- Cziko, G. (1995). *Without Miracles: Universal Selection Theory and the Second Darwinian Revolution*. MIT Press, Cambridge, MA.
- Daniel, J. W. (1976). Splines and efficiency in dynamic programming. *Journal of Mathematical Analysis and Applications*, 54:402–407.
- Dayan, P. (1991). Reinforcement comparison. In D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton (eds.), *Connectionist Models: Proceedings of the 1990 Summer School*, pp. 45–51. Morgan Kaufmann, San Mateo, CA.
- Dayan, P. (1992). The convergence of TOIII for general λ . *Machine Learning*, 8:341–362.
- Dayan, P., and Hinton, G. E. (1993). Feudal reinforcement learning. In S. J. Hanson, J. D. Cohen, and C. L. Giles (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1992 Conference*, pp. 271–278. Morgan Kaufmann, San Mateo, CA.
- Dayan, P. and Sejnowski, T. (1994). TD(A) converges with probability 1. *Machine Learning*, 14:295–301.
- Dean, T., and Lin, S.-H. (1995). Decomposition techniques for planning in stochastic domains. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1121–1127, Morgan Kaufmann, See also Technical Report CS-95-10, Brown University, Department of Computer Science, 1995.
- DeJong, G., and Spong, M. W. (1994). Swinging up the acrobot: An example of intelligent control. In *Proceedings of the American Control Conference*, pp. 2158–2162. American Automatic Control Council, Evanston, IL.
- Denardo, E. V. (1967). Contraction mappings in the theory underlying dynamic programming. *SIAM Review*, 9:165–177.
- Dennett, D. C. (1978). *Brainstorms*, pp. 71–89. Bradford/MIT Press, Cambridge, MA.

- Dietterich, T. G., and Flann, N. S. (1995). Explanation-based learning and reinforcement learning: A unified view. In A. Prieditis and S. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 176–184. Morgan Kaufmann, San Francisco.
- Doya, K. (1996). Temporal difference learning in continuous time and space. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pp. 1073–1079, MIT Press, Cambridge, MA.
- Doyle, P. G., and Snell, J. L. (1984). *Random Walks and Electric Networks*. The Mathematical Association of America. Carus Mathematical Monograph 22.
- Dreyfus, S. E. and Law, A. M. (1977). *The Art and Theory of Dynamic Programming*. Academic Press, New York.
- Duda, R. O., and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Wiley, New York.
- Duff, M. O. (1995). Q-learning for bandit problems. In A. Prieditis and S. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 209–217. Morgan Kaufmann, San Francisco.
- Estes, W. K. (1950). Toward a statistical theory of learning. *Psychological Review*, 57:94–107.
- Farley, B. G., and Clark, W. A. (1954). Simulation of self-organizing systems by digital computer, *IRE Transactions on Information Theory*, 4:76–84.
- Feldbaum, A. A. (1965). *Optimal Control Systems*. Academic Press, New York.
- Friston, K. L. Tononi, G., Rekke, G. N. Sporns, O., and Edelman, G. M. (1994). Value-dependent selection in the brain: Simulation in a synthetic neural model. *Neuroscience*, 59:229–243.
- Fu, K. S. (1970). Learning control systems—Review and outlook. *IEEE Transactions on Automatic Control*, 15:2)0–221.
- Galanter, E., and Gerstenhaber, M. (1956). On thought: The extrinsic theory. *Psychological Review*, 63:218–227.
- Gallant, S. I. (1993). *Neural Network Learning and Expert Systems*. MIT Press, Cambridge, MA,
- Gällmo, O. and Asplund, L. (1995). Reinforcement learning by construction of hypothetical targets. In J. Alspector, R. Goodman, and T. X. Brown (eds.), *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications 2*, pp. 300–307. Erlbaum, Hillsdale, NJ:
- Gardner, M. (1973). Mathematical games. *Scientific American*, 228(1); 108–115.
- Gelperin, A. Hopfield, J. J., and Tank, D. W. (1985). The logic of *Umax* learning. In A. Selverston (ed.), *Model Neural Networks and Behavior*, pp. 247–261. Plenum Press, New York.
- Giltins, J. C. and Jones, D. M. (1974). A dynamic allocation index for the sequential design of experiments. In J. Gani, K. Sarkadi, and I. Vineze (eds.), *Progress in Statistics*, pp. 241–266, North-Holland, Amsterdam-London.

- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.
- Goldstein, H. (1957). *Classical Mechanics*. Addison-Wesley, Reading, MA.
- Goodwin, G. C., and Sin, K. S. (1984). *Adaptive Filtering Prediction and Control*. Prentice-Hall, Englewood Cliffs, NJ.
- Gordon, G. J. (1995). Stable function approximation in dynamic programming. In A. Prieditis and S. Russell (eds.). *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 261–268, Morgan Kaufmann, San Francisco. An expanded version was published as Technical Report CMU-CS-95-103, Carnegie Mellon University, Pittsburgh, PA, 1995.
- Gordon, G. J. (1996). Stable lilted reinforcement learning. In D. S. Touretzky, M. C. Mozer, M. F. Hasselmo (eds.). *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pp. 1052–1058. MIT Press, Cambridge, MA.
- Griffith, A. K. (1966). A new machine learning technique applied to the game of checkers. Technical Report Project MAC, Artificial intelligence Memo 94, Massachusetts institute of Technology, Cambridge, MA.
- Griffith, A. K. (1974). A comparison and evaluation of three machine learning procedures as applied to the game of checkers. *Artificial Intelligence*, 5:137–148.
- Gullapalli. V. (1990). A stochastic reinforcement algorithm for learning real-valued functions. *Neural Networks*, 3:671–692.
- Gurvits, L., Lin, L. J., and Hanson, S. J. (1994). Incremental learning of evaluation functions for absorbing Markov chains: New methods and theorems. Preprint,
- Hampson, S. E. (1983). *A Neural Model of Adaptive Behavior*. Ph.D. (thesis. University of California, Irvine.
- Hampson, S. E. (1989). *Connectionist Problem Solving: Computational Aspects of Biological Learning*. Birkhauser, Boston.
- Hawkins, R. D., and Kandel, E. R. (1984). Is there a cell-biological alphabet for simple forms of learning? *Psychological Review*, 91:375–391,
- Hersh, R., and Griego, R. J. (1969). B row man motion and potential theory. *Scientific American*, 220:66–74.
- Hilgard, E. R., and Bower, G. H. (1975). *Theories of Learning*, Prentice-Hall, Englewood Cliffs, NJ.
- Hinton, G. E. (1984). Distributed representations. Technical Report CMU-CS-84-157. Department of Computer Science, Carnegie-Mellon University, Pittsburgh, РЛ.
- Hochreiter, S., and Schmidhuber, J. (1997). LSTM can solve hard time lag problems. In *Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference*, pp. 473–479. MIT Press, Cambridge, MA.
- Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

- Holland, J. H. (1976). Adaptation, In R. Rosen and F. M. Snell (eds.), *Progress in Theoretical Biology*, vol. 4, pp. 263–293. Academic Press, New York.
- Holland, J. H. (1986). Escaping brittleness: The possibility of general-purpose learning algorithms applied to rule-based systems, in R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.). *Machine Learning: An Artificial Intelligence Approach*, vol. 2, pp. 593–623, Morgan Kaufmann, San Mateo, CA.
- Houk, J. C., Adams, J. L., and Barto, A. G. (1995). A model of how the basal ganglia generates and uses neural signals that predict reinforcement. In J. C. Houk, I. L. Davis, and D. G. Beiser (eds.). *Models of Information Processing in the Basal Ganglia*, pp. 249–270. MIT Press, Cambridge, MA.
- Howard, R. (1960). *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA.
- Hull, C. L. (1943). *Principles of Behavior*. Appleton-Century, New York. Hull, C. L. (1952). *A Behavior System*. Wiley, New York.
- Jaakkola, T., Jordan, M. I., and Singh, S. P. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1 185–1201.
- Jaakkola, T., Singh, S. P., and Jordan, M. I. (1995). Reinforcement learning algorithm for partially observable Markov decision problems. In G. Tesauro, D. S. Touretzky, T. Leen (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1994 Conference*, pp. 345–352, MIT Press, Cambridge, MA.
- Jalali, A., and Ferguson, M. (1989). Computationally efficient adaptive control algorithms for Markov chains. In *Proceedings of the 28th Conference on Decision and Control*, pp. 1283–1288.
- Kaelbling, L. P. (1993a). Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 167–173. Morgan Kaufmann, San Mateo, CA.
- Kaelbling, L. P. (1993b). *Learning in Embedded Systems*. MIT Press, Cambridge, MA.
- Kaelbling, L. P. (ed.). (1996). Special issue of *Machine Learning* on reinforcement learning. 22.
- Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.
- Kakutani, S. (1945). Markov processes and the Dirichlet problem. *Proceedings of the Japan Academy*, 21:227–233.
- Kalos, M. H., and Whitloek, P. A. (1986). *Monte Carlo Methods*. Wiley, New York.
- Kanerva, P. (1988). *Sparse Distributed Memory*. MIT Press, Cambridge, MA.
- Kanerva, P. (1993). Sparse distributed memory and related models. In M. H. Hassoun (ed.), *Associative Neural Memories: Theory and Implementation*, pp. 50–76, Oxford University Press, New York,

- Kashyap, R. L., Blaydon, C. C., and Fu, K. S. (1970). Stochastic approximation. In J. M. Mendel and K. S. Fu (eds.). *Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications*, pp. 329–355. Academic Press, New York.
- Keerthi, S. S., and Ravindran. B. (1997). Reinforcement learning. In E. Ficsler and R. Beale (eds.), *Handbook of Neural Computation*, C3. Oxford University Press, New York.
- Kimble, G. A. (1961). *Hilgard and Marquis' Conditioning and Learning*. Applet et on-Century- Crofts, New York.
- Kimble, G. A. (1967). *Foundations of Conditioning and Learning*, Appleton-Centuiy-Crofts, New York,
- Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.
- Klopff, A. H. (1972). Brain function and adaptive systems—A heterostatic theory. Technical Report AFCRL-72-0164, Air Force Cambridge Research Laboratories, Bedford, MA. A summary appears in *Proceedings of the International Conference on Systems, Man, and Cybernetics*. IEEE Systems, Man, and Cybernetics Society, Dallas, TX, 1974.
- Klopff, A. H. (1975). A comparison of natural and artificial intelligence. *SICARTNewsletter*, 53:11–13.
- Klopff, A. H. (1982). *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence*. Hemisphere, Washington, DC.
- Klopff, A. H. (1988}. A neuronal model of classical conditioning. *Psychobiology*, 16:85–125.
- Kohonen, T. (1977). *Associative Memory: A System Theoretic Approach*. Springer-Verlag, Berlin.
- Korf, R. E. (1988). Optimal path finding algorithms. In L. N. Kanal and V. Kumar (eds.). *Search in Artificial Intelligence*, pp. 223–267. Springer Verlag, Berlin.
- Korf, R. E. (1990). Real-time heuristic search. *Artificial Intelligence*, 42:189–211.
- Kraft, L. G., and Campagna, D. R (1990). A summary comparison of CMAC neural network and traditional adaptive control systems. In T. Miller, R. S. Sutton, and P. J. Werbos (eds.), *Neural Networks for Control*, pp. 143–169. MIT Press, Cambridge, MA.
- Kraft, L. G., Miller, W. T., and Dietz, D. (1992). Development and application of CMAC neural network-based control. In D. A. White and D. A. Sofge (eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pp. 215–232. Van Nostrand Reinhold, New York.
- Kumar, P. R., and Varaiya, P. (1986). *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice-Hall, Englewood Cliffs, NJ.
- Kumar, P. R. (1985). A survey of some results in stochastic adaptive control. *SIAM Journal of Control and Optimization*, 23:329–380,
- Kumar, V. and Kanal L. N. (1988). The CDP: A-unifying formulation for heuristic search, dynamic programming, and branch-and-bound. In L. N. Kanal

- and V. Kumar (eds.), *Search in Artificial Intelligence*, pp. 1–37. Springer-Verlag, Berlin.
- Kushner, H. J., and Dupuis, P. (1992). *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer-Verlag, New York.
- Lai, T. L. (1987). Adaptive treatment allocation and the multi-armed bandit problem. *The Annals of Statistics*, 15(3):109S- .
- Lang, K. J., Waibel, A. H., and Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:33–43,
- Lin, C.-S., and Kim, H. (1991). CMAC-based adaptive critic self-learning control. *IEEE Transactions on Neural Networks*, 2:530–533.
- Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321.
- Lin, L.-J., and Mitchell, T. (1992). Reinforcement learning with hidden states. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pp. 271–280. MIT Press, Cambridge, MA.
- Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 157–163. Morgan Kaufmann, San Francisco.
- Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In A. Prieditis and S. Russell (eds.). *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 362–370. Morgan Kaufmann, San Francisco.
- Littman, M. L., Dean, T. L., and Kaelbling, L. P. (1995). On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, pp. 394–402.
- Ljung, L., and Soderstrom, T. (1983). *Theory and Practice of Recursive Identification*. MIT Press, Cambridge, MA.
- Lovejoy, W. 5. (1991). A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28:47–66.
- Luce, D. (1959). *Individual Choice Behavior*. Wiley, New York.
- Maclin, R., and Shavlik, J. W. (1994). Incorporating advice into agents that learn from reinforcements, In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 694–699. AAAI Press, Menlo Park, CA.
- Mahadevan, S. (1996). Average reward reinforcement learning: Foundations, algorithms, and empirical results, *Machine Learning*, 22:159–196.
- Markey, K. L. (1994). Efficient learning of multiple degree-of-freedom control problems with quasi-independent Q-agents, In M. C. Mozer, P. Smolensky, D. S. Touretzky, J. L. Elman, and A. S. Weigend (eds.), *Proceedings of the 1990 Connectionist Models Summer School*. Erlbaum, Hillsdale, NJ.
- Mazur, J. E. (1994). *jteaming and Behavior*, 3rd ed. Prentice-Hall, Englewood Cliffs, NJ.

- McCallum, A. K. (1992). Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 183–188. AAAI/MIT Press, Menlo Park, CA.
- McCallum, A. K. (1993). Overcoming incomplete perception with utile distinction memory. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 190–196, Morgan Kaufmann, San Mateo, CA.
- McCallum, A. K. (1995). *Reinforcement Learning with Selective Perception and Hidden State*. Ph.D. thesis, University of Rochester, Rochester, NY.
- Mendel, J. M. (1966). A survey of learning control systems. *ISA Transactions*, 5:297–303.
- Mendel, J. M., and McLaren, R. W. (1970). Reinforcement learning control and pattern recognition systems. In J. M. Mendel and K. S. Fu (eds.). *Adaptive, Learning and Pattern Recognition Systems: Theory and Applications*, pp. 287–318. Academic Press, New York.
- Michie, D. (1961). Trial and error. In S. A. Barnett and A. McLaren (eds.), *Science Survey, Part 2*, pp. 129–145, Penguin, Harmondsworth.
- Michie, D. (1963). Experiments on the mechanisation of game learning. 1. characterization of the model and its parameters. *Computer Journal*, 1:232–263.
- Michie, D. (1974). *On Machine Intelligence*. Edinburgh University Press, Edinburgh.
- Michie, D., and Chambers, R. A. (1968). BOXES: An experiment in adaptive control. In E. Dale and D. Michie (eds.), *Machine Intelligence 2*, pp. 137–152. Oliver and Boyd, Edinburgh.
- Miller, S., and Williams, R. J. (1992). Learning to control a bioreactor using a neural net Dyna-Q system. In *Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems*, pp. 167–172. Center for Systems Science, Dunham Laboratory, Yale University, New Haven.
- Miller, W. T., Scalera, S. M., and Kim, A. (1994). Neural network control of dynamic balance for a biped walking robot. In *Proceedings of the Eighth Yale Workshop on Adaptive and learning Systems*, pp. 156–161. Center for Systems Science, Dunham Laboratory, Yale University, New Haven.
- Minsky, M. L. (1954). *Theory of Neural-Analog Reinforcement Systems and Its Application to the Brain-Model Problem*. Ph.D. thesis, Princeton University.
- Minsky, M. L. (1961). Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49:8–30. Reprinted in E. A. Feigenbaum and J. Feidman (eds.). *Computers and Thought*, pp. 406–450. McGraw-Hill, New York, 1963.
- Minsky, M. L. (1967). *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, NJ.
- Montague, P. R. Dayan, P., and Sejnowski, T. J. (1996). A framework for mesencephalic dopamine systems based on predictive Hebbian learning. *Journal of Neuroscience*, 16:1936–1947.

- Moore, A. W. (1990). *Efficient Memory-Based Learning for Robot Control*. Ph.D. thesis. University of Cambridge,
- Moore, A. W. (1994). The parti-game algorithm for variable resolution reinforcement learning in multidimensional spaces. In J. D. Cohen, G. Tesauro and i. Alspector (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1993 Conference*, pp. 711–718, Morgan Kaufmann, San Francisco.
- Moore, A. W., and Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130.
- Moore, J. W., Desmond, J. E., Berthier, N. E., Blazis, E. J., Sutton, R. S., and Barto, A. G. (1986). Simulation of the classically conditioned nictitating membrane response by a neuronlike adaptive element: I. Response topography, neuronal firing, and interstimulus intervals. *Behavioural Brain Research*, 21:143–154.
- Narendra, K. S., and Thathachar, M. A. L. (1974). Learning automata—A survey. *IEEE Transactions on Systems, Man, and Cybernetics*, 4:323–334.
- Narendra, K. S., and Thathachar, M. A. L. (1989). *Learning Automata: An Introduction*. Prentice-Hall, Englewood Cliffs, NJ.
- Narendra, K. S., and Wheeler, R. M. (1986). Decentralized learning in finite Markov chains. *IEEE Transactions on Automatic Control*, AC31(6):519–526.
- Nie, J., and Haykin, S. (1996). A dynamic channel assignment policy through Q-learning. CRL Report 334. Communications Research Laboratory, McMaster University, Hamilton, Ontario,
- Page, C. V. (1977). Heuristics for signature table analysis as a pattern recognition technique. *IEEE Transactions on Systems, Man, and Cybernetics*, 7:77–86.
- Parr, R., and Russell, S. (1995). Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1088–1094, Morgan Kaufmann.
- Pavlov, P. I. (1927). *Conditioned Reflexes*. Oxford University Press, London.
- Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, MA.
- Peng, J. (1993). *Efficient Dynamic Programming-Based Learning for Control*. Ph.D. thesis, Northeastern University, Boston.
- Peng, J. and Williams, R. J. (1993). Efficient learning and planning within the Dyna framework. *Adaptive Behavior*, 1(4):437–454,
- Peng, J., and Williams, R. J. (1994). Incremental multi-step Q-learning. In W. W. Cohen and H. Hirsh (eds.). *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 226–232. Morgan Kaufmann, San Francisco.
- Peng, J., and Williams, R. J. (1996). Incremental multi-step Q-learning. *Machine Learning*, 22:283–290.
- Phansalkar, V. V., and Thathachar, M. A. L. (1995). Local and global optimization algorithms for generalized learning automata. *Neural Computation*, 7:950–973.

- Poggio, T., and Girosi, F. (1989). A theory of networks for approximation and learning. A.I. Memo 1140. Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.
- Poggio, T., and Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982.
- Powell, M. J. D. (1987). Radial basis functions for multivariate interpolation: A review. In J. C. Mason and M. G. Cox (eds.), *Algorithms for Approximation*, pp. 143–167, Clarendon Press, Oxford.
- Puterman, M. L. (1994). *Markov Decision Problems*. Wiley, New York.
- Puterman, M. L., and Shin, M. C. (1978). Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24:1127–1137.
- Reetz, D. (1977). Approximate solutions of a discounted Markovian decision process. *Bonner Mathematische Schriften*, 98:77–92.
- Ring, M. B. (1994). *Continual Learning in Reinforcement Environments*. Ph.D. thesis, University of Texas, Austin.
- Rivest, R. L., and Schapire, R. E. (1987). Diversity-based inference of finite automata, in *Proceedings of the Twenty-Eighth Annual Symposium on Foundations of Computer Science*, pp. 78–87, Computer Society Press of the IEEE, Washington, DC.
- Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58:527–535.
- Robertie, B. (1992). Carbon versus silicon: Matching wits with TD-Gammon. *Inside Backgammon*, 2:14–22.
- Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington, DC.
- Ross, S. (1983). *Introduction to Stochastic Dynamic Programming*. Academic Press, New York.
- Rubinstein, R. Y. (1981). *Simulation and the Monte Carlo Method*. Wiley, New York.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. I, *Foundations*. Bradford/MIT Press, Cambridge, MA.
- Rummery, G. A. (1995). *Problem Solving with Reinforcement Learning*. Ph.D. thesis, Cambridge University,
- Rummery, G. A., and Niranjan, M. (1994). On-line Q-learning using connections! systems. Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University.
- Russell, S., and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ.

- Rust, J. (1996). Numerical dynamic programming in economics. In H. Amman, D. Kendrick, and J. Rust (eds.), *Handbook of Computational Economics*, pp. 614–722. Elsevier, Amsterdam.
- Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development*, 3:211–229. Reprinted in E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, pp. 71–105. McGraw-Hill, New York, 1963.
- Samuel, A. L. (1967). Some studies in machine learning using the game of checkers. II—Recent progress. *IBM Journal on Research and Development*, 11:601–617.
- Schultz, D. G., and Melsa, J. L. (1967). *State Functions and Linear Control Systems*. McGraw-Hill, New York.
- Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275:1593–1598.
- Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 298–305. Morgan Kaufmann, San Mateo, CA.
- Schweitzer, P. J., and Seidmann, A. (1985). Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110:568–582.
- Selfridge, O. J., Sutton, R. S., and Barto, A. G. (1985). Training and tracking in robotics. In A. Joshi (ed.), *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 670–672. Morgan Kaufmann. San Mateo, CA,
- Shannon, C. E. (1950). Programming a computer for playing chess. *Philosophical Magazine*, 41:256–275.
- Shewchuk, J., and Dean, T. (1990). Towards learning time-varying functions with high input dimensionality, in *Proceedings of the Fifth IEEE International Symposium on Intelligent Control*, pp. 383–388. IEEE Computer Society Press, Los Alamos, CA.
- Singh, S. P. (1992a). Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 202–207. AAAI/MIT Press, Menlo Park, CA,
- Singh, S. P. (1992b). Scaling reinforcement learning algorithms by learning variable temporal resolution models. In *Proceedings of the Ninth International Machine Learning Conference*, pp. 406–415. Morgan Kaufmann, San Mateo, CA.
- Singh, S. P. (1993). *Learning to Solve Markovian Decision Processes*. Ph.D. thesis. University of Massachusetts, Amherst. Appeared as CMPSCI Technical Report 93–77.
- Singh, S. P., and Bertsekas, D. (1997). Reinforcement learning for dynamic channel allocation in cellular telephone systems. In *Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference*, pp. 974–980. MIT Press, Cambridge, MA.

- Singh, S. P., Jaakkola, T., and Jordan, M. I. (1994). Learning without state-estimation in partially observable Markovian decision problems. In W. W. Cohen and H. Hirsch (eds.), *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 284–292. Morgan Kaufmann, San Francisco.
- Singh, S. P., Jaakkola, T., and Jordan, M. I. (1995). Reinforcement learning with soft slate aggregation. In G. Tesauro, D. S. Touretzky, T. Leen (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1994 Conference*, pp. 359–368. MIT Press, Cambridge, MA.
- Singh, S. P., Jaakkola, T., Littman, M. L., and Szepesvari, C. (in preparation). Convergence results for single-step on-policy reinforcement-learning algorithms.
- Singh, S. P., and Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158.
- Sivarajan, K. N., McElicee, R. J., and Ketchum, J. W. (1990). Dynamic channel assignment in cellular radio. In *Proceedings of the 40th Vehicular Technology Conference*, pp. 631–637.
- Skinner, B. P. (1938). *The Behavior of Organisms*. Appleton-Century, New York.
- Sofge, D. A., and White, D. A. (1992). Applied learning: Optimal control for manufacturing. In D. A. White and D. A. Sofge (eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pp. 259–281. Van Nostrand Reinhold, New York.
- Spong, M. W. (1994). Swing up control of the acrobot. In *Proceedings of the 1994 IEEE Conference on Robotics and Automation*, pp. 2356–2361. IEEE Computer Society Press, Los Alamitos, CA.
- Staddon, J. E. R. (1983). *Adaptive Behavior and Learning*. Cambridge University Press, Cambridge.
- Sutton, R. S. (1978a). Learning theory support for a single channel theory of the brain. Unpublished report.
- Sutton, R. S. (1978b). Single channel theory: A neuronal theory of learning. *Brain Theory Newsletter*, 4:72–75. Center for Systems Neuroscience, University of Massachusetts, Amherst, MA.
- Sutton, R. S. (1978c). A unified theory of expectation in classical and instrumental conditioning. Bachelors thesis, Stanford University.
- Sutton, R. S. (1984). *Temporal Credit Assignment in Reinforcement Learning*. Ph.D. thesis, University of Massachusetts, Amherst.
- Sutton, R. S. (1988). Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44.
- Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pp. 216–224. Morgan Kaufmann, San Mateo, CA.
- Sutton, R. S. (1991a). Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bulletin*, 2:160–163. ACM Press.

- Sutton, R. S. (1991b). Planning by incremental dynamic programming. In L. A. Birnbaum and G. C. Collins (eds.), *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 353–357. Morgan Kaufmann, San Mateo, CA.
- Sutton, R. S. (1995). TD models: Modeling the world at a mixture of time scales. In A. Priedis and S. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 531–539. Morgan Kaufmann, San Francisco.
- Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky, M. C. Mozer and M. E. Hasselmo (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pp. 1038–1044. MIT Press, Cambridge, MA.
- Sutton, R. S. (ed.). (1992). Special issue of *Machine Learning* on reinforcement learning, 8, Also published as *Reinforcement Learning*. Kluwer Academic. Boston, 1992.
- Sutton, R. S., and Barto, A. G. (1981a). Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review*, 88:135–170.
- Sutton, R. S., and Barto, A. G. (1981b). An adaptive network that constructs and uses an internal model of its world. *Cognition and Brain Theory*, 3:217–246.
- Sutton, R. S. and Barto, A. G. (1987). A temporal-difference model of classical conditioning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pp. 355–378. Erlbaum, Hillsdale, NJ.
- Sutton, R. S., and Barto, A. G. (1990). Time-derivative models of Pavlovian reinforcement. In M. Gabriel and J. Moore (eds.), *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, pp. 497–537. MIT Press, Cambridge, MA.
- Sutton, R. S., and Pinette, B. (1985). The learning of world models by connectionist networks. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pp. 54–64.
- Sutton, R. S., and Singh, S. (1994). On bias and step size in temporal-difference learning. In *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems*, pp. 91–96. Center for Systems Science, Dunham Laboratory, Yale University, New Haven.
- Sutton, R. S., and Whitehead, D. S. (1993). Online learning with random representations. In *Proceedings of the Tenth International Machine Learning Conference*, pp. 314–321. Morgan Kaufmann, San Mateo, CA.
- Tadepalli, P., and Ok, D. (1994). H-learning: A reinforcement learning method to optimize undiscounted average reward. Technical Report 94-30-01, Oregon State University, Computer Science Department, Corvallis.
- Tan, M. (1991). Learning a cost-sensitive internal representation for reinforcement learning, in L. A. Birnbaum and G. C. Collins (eds.), *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 358–362. Morgan Kaufmann, San Mateo, CA,

- Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 330–337, Morgan Kaufmann, San Mateo, CA.
- Tesauro, G. J. (1986). Simple neural models of classical conditioning. *Biological Cybernetics*, 55:187–200.
- Tesauro, G. J. (1992). Practical issues in temporal difference learning. *Machine Learning*, 8:257–277.
- Tesauro, G. J. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219.
- Tesauro, G. J. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38:58–68.
- Tesauro, G. J., and Galperin, G. R. (1997). On-line policy improvement using Monte-Carlo search. In *Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference*, pp. 1068–1074. MIT Press, Cambridge, MA.
- Tham, C. K. (1994). *Modular On-Line Function Approximation for Scaling up Reinforcement Learning*. PhD thesis, Cambridge University.
- Thathachar, M. A. L. and Sastry, P. S. (1985). A new approach to the design of reinforcement schemes for learning automata, *IEEE Transactions on Systems, Man, and Cybernetics*, 15:168–175.
- Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294.
- Thompson, W. R. (1934). On the theory of apportionment. *American Journal of Mathematics*, 57:450–457.
- Thorndike, E. L. (1911). *Animal Intelligence*. Hafner, Darien, CT,
- Thorp, E. O. (1966). *Beat the Dealer: A Winning Strategy for the Game of Twenty-One*. Random House, New York.
- Tolman, E. C. (1932). *Purposive Behavior in Animals and Men*. Century, New York.
- Tsetlin, M. L. (1973). *Automaton Theory and Modeling of Biological Systems*, Academic
- Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16:185–202.
- Tsitsiklis, J. N. and Van Roy, B. (1996). Feature-based methods for large scale dynamic programming. *Machine Learning*, 22:59–94.
- Tsitsiklis, J. N., and Van Roy, B. (1997a). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42:674–690,
- Tsitsiklis, J. N., and Van Roy, B. (1997b). Average cost temporal-difference learning. *IEEE Transactions on Automatic Control*, 42:674–690,
- Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59:433–460. Reprinted in E. A. Feigenbaum and J. Feldman (eds), *Computers and Thought*, pp. 11–35. McGraw-Hill, New York, 1963.

- Ungar, L. H. (1990). A bioreactor benchmark for adaptive network-based process control. In W. T. Miltz, R. S. Sutton, and P. J. Werbos (eds.), *Neural Networks for Control*, pp. 387–402. MIT Press, Cambridge, MA.
- Waltz, M. D., and Fu, K. S. (1965). A heuristic approach to reinforcement learning control systems. *IEEE Transactions on Automatic Control*, 10:390–398.
- Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. Ph.D. thesis, Cambridge University.
- Watkins, C. J. C. H., and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.
- Werbos, P. J. (1977). Advanced forecasting methods for global crisis warning and models of intelligence. *General Systems Yearbook*, 22:25–38.
- Werbos, P. J. (1982). Applications of advances in nonlinear sensitivity analysis. In R. F. Drenick and F. Kozin (eds.), *System Modeling and Optimization*, pp. 762–770. Springer-Verlag, Berlin.
- Werbos, P. J. (1987). Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man, and Cybernetics*, 17:7–20.
- Werbos, P. J. (1988). Generalization of back propagation with applications to a recurrent gas market model. *Neural Networks*, 1:339–356.
- Werbos, P. J. (1989). Neural networks for control and system identification. In *Proceedings of the 28th Conference on Decision and Control*, pp. 260–265. IEEE Control Systems Society.
- Werbos, P. J. (1990). Consistency of HDP applied to a simple reinforcement learning problem. *Neural Networks*, 3:179–189.
- Werbos, P. J. (1992). Approximate dynamic programming for real-time control and neural modeling. In D. A. White and D. A. Sofge (eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pp. 493–525. Van Nostrand Reinhold, New York.
- White, D. J. (1969). *Dynamic Programming*. Holden-Day, San Francisco.
- White, D. J. (1985). Real applications of Markov decision processes. *Interfaces*, 15:73–83.
- White, D. J. (1988). Further real applications of Markov decision processes. *Interfaces*, 18:5561.
- White, D. J. (1993). A survey of applications of Markov decision processes. *Journal of the Operational Research Society*, 44:1073–1096.
- Whitehead, S. D., and Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning*, 7:45–83.
- Whitt, W. (1978). Approximations of dynamic programs I, *Mathematics of Operations Research*, 3:231–243.
- Whittle, P. (1982). *Optimization over Time*, vol. 1. Wiley, New York.
- Whittle, P. (1983). *Optimization over Time*, vol. 2. Wiley, New York.

- Widrow, B., Gupta, N. K., and Maitra, S. (1973). Punish/reward: Learning with a critic in adaptive threshold systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 3:45 5–465.
- Widrow, B. and Hoff, M. E. 119601. Adaptive switching; circuits. In *1960 WESCON Convention Record Part IV*, pp. 96–104. Institute of Radio Engineers, New York. Reprinted in J. A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research*, pp. 126–134. MIT Press, Cambridge, MA, 1988.
- Widrow, B., and Smith, F. W. (1964). Pattern-recognizing control systems, in J. T. Ton and R. H. Wilcox (eds.), *Computer and Information Sciences*, pp. 288–317. Spartan, Washington, DC.
- Widrow, B., and Stearns, S. D. (1985). *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.
- Williams, R. J. (1986). Reinforcement learning in connectionist networks: A mathematical analysis. Technical Report ICS 8605, Institute for Cognitive Science, University of California at San Diego, La Jolla.
- Williams, R. J. (1987). Reinforcement-learning connectionist systems. Technical Report NU- CCS-87-3. College of Computer Science, Northeastern University, Boston,
- Williams, R. J. (1988). On the use of back propagation in associative reinforcement learning. In *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1263–1270. IEEE San Diego section and IEEE TAB Neural Network Committee.
- Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.
- Williams, R. J., and Baird, L. C. (1990). A mathematical analysis of actor-critic architectures for learning optimal controls through incremental dynamic programming. In *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*, pp. 96–101. Center for Systems Science, Dunham Laboratory, Yale University, New Haven.
- Wilson, S. W. (1994). ZCS: A zeroth order classifier system. *Evolutionary Computation*, 2:118.
- Witten, I. H. (1976). The apparent conflict between estimation and control—A survey of the two-armed problem. *Journal of the Franklin Institute*, 301:161–189.
- Witten, I. H. (1977). An adaptive optimal controller for discrete-time Markov environments. *Information and Control*, 34:286–295.
- Witten, I. H., and Corbin, M. J. (1973). Human operators and automatic adaptive controllers: A comparative study on a particular control task. *International Journal of Man-Machine Studies*, 5:75–104.
- Yee, R. C., Saxena, S., Utgoff, P. E., and Barto, A. O. (1990). Explaining temporal differences to create useful concepts for evaluating states. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 882–888. AAAI Press, Menlo Park, CA.

- Young, P. (1984). *Recursive Estimation and Time-Series Analysis*. Springer-Verlag, Berlin.
- Zhang, M., and Yum, T. P. (1989). Comparisons of channel-assignment strategies in cellular mobile telephone systems, *IEEE Transactions on Vehicular Technology*, 38:211–215.
- Zhang, W. (1996). *Reinforcement Learning for Job-shop Scheduling*. Ph.D. thesis, Oregon State University. Technical Report CS-96-30-1.
- Zhang, W., and Dietterich, T. G. (1995). A reinforcement learning approach to job-shop scheduling. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1 114–1120, Morgan Kaufmann.
- Zhang, W., and Dietterich, T. G. (1996). High-performance job-shop scheduling with a time-delay TD(i) network. In D. S. Touretzky, M. C. Mozer, M. E. Hasselmo (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pp. 1024–1030, MIT Press, Cambridge, MA.
- Zweben, M., Daun, B., and Deale, M. (1994). Scheduling and rescheduling with iterative repair. In M. Zweben and M. S. Fox (eds.). *Intelligent Scheduling*, pp. 241–255. Morgan Kaufmann, San Francisco.

Предметный указатель

- N*-шаговые TD-методы (*N*-step TD methods)
и TD (λ (and TD(λ))) 209–212¹⁾, 234
и методы Монте-Карло (and Monte Carlo methods) 203–206, 210–212
проблемы с ними (problems with) 207
- N*-шаговые TD-методы (*N*-step TD methods) 203–208
- N*-шаговые выгоды (*N*-step returns) 205–206, 234
- N*-шаговые дублирования (*N*-step backups) 203–206, 234
- Q*-обучение (*Q*-learning)
его сходимость (convergence of) 186, 199, 235, 264, 267
и планирование (and planning) 281–282
и следы приемлемости (and eligibility traces) 223–227, 235
- Q*-обучение (*Q*-learning) 40, 186–189, 199, 275, 340
- Q*-планирование (*Q*-planning) 281–282, 283–285
- Q*-функции, см. Функции ценности действий (*Q*-functions) 115, см. также Action-value functions
- R*-обучение (*R*-learning) 192–195, 200
- λ -выгода (λ -return) 210–213, 231–233, 234, 244
- «На пути к искусственному интеллекту», статья (Мински) (T2A «Steps Toward Artificial Intelligence» paper (Minsky)) 33, 37, 38
- BOXES, регулятор (BOXES) 34, 155, 273
- MENACE, система, которая методом проб и ошибок учились игре в крестики-нолики (MENACE (Matchbox Educable Naughts and Crosses Engine)) 34, 115
- $Q(\lambda)$ ($Q(\lambda)$) 223–227, 235
- SARSA (Sarsa) 182–186, 199, 258
- SARSA(λ) (Sarsa(λ))
и TD(λ) (and TD(λ)) 220–221
линейный, наискорейший спуск (linear, gradient-descent) 259–262, 275
- SARSA(λ) (Sarsa(λ)) 220–222, 235

¹⁾ Страницы с номерами, набранными курсивом, рекомендуются для ознакомления в первую очередь.

SNARC, стохастические нейро-аналоговые калькуляторы с подкреплением
(SNARCs (Stochastic Neural-Analog Reinforcement Calculators)) 33
softmax-метод (Softmax method) 49–50, 72

Автономная корректировка (Off-line updating) 208

Агент (Agent) 14, 74–78

Агенты Dyna (Dyna agents)

vs. приоритетная прогонка (vs. prioritized sweeping) 294–296

агент Dyna-AC (Dyna-AC) 290, 299

агент Dyna-Q (Dyna-Q) 282–290

агент Dyna-Q+ (Dyna-Q+) 290, 291

структура агента (architecture) 284

Агенты Dyna (Dyna agents) 282–291, 311, 317

Акробот: пример прикладной задачи (Acrobot): case study, 331–335

Алгоритмы REINFORCE (REINFORCE algorithms) 276

Алгоритмы обновления ценностей «на месте» (In-place algorithms) 119, 141

Алгоритмы оценивания (Estimator algorithms) 71

Андрэ, Джон (Andreae, John) 34, 114, 140

Аппроксимация (Approximation) 109–110

Аппроксимация функций (Function approximation)

контрипример аппроксимации функций, с самонастройкой на основе

разделенной оценки ценности стратегий (counterexamples to, with off-policy bootstrapping) 265–270

линейные методы аппроксимации (linear methods for) 246–258

методы наискорейшего спуска для аппроксимации (gradient-descent methods for) 242–246

управление с аппроксимацией функций (control with) 258–263

Аппроксимация функций (Function approximation) 237–276, 318

Априорное знание, его использование (Prior knowledge, using) 28, 80, 319

Асинхронное динамическое программирование (Asynchronous dynamic programming) 133–134, 137–138, 141, 311

Ассоциативное обучение (Associative learning) 32

Ассоциативное обучение с подкреплением (Associative reinforcement learning) 72

Ассоциативный алгоритм поощрения-наказания (Associative reward-penalty algorithm) 277

Ассоциативный поиск (Associative search) 67–68, 72

Балансировка стержня, см. BOXES (Pole-balancing) 84, 90, 113, 248, 271
(рис.). см. также BOXES

Безмодельные методы, см. Прямое обучение с подкреплением (Model-free methods). см. Direct RL

Беллман, Ричард (Bellman, Richard) 30, 37, 71, 72, 115, 140

Беллмановская ошибка (Bellman error) 268–270, 274

Бинарная задача о бандите (Binary bandit tasks) 52–55, 68, 72

- Бинарные свойства (Binary features)
против радиально-базисных функций (vs. radial basis functions) 255
- Бинарные свойства (Binary features) 248, 252
- Биореактор: пример (Bioreactor): example, 78, 113
- Блокированный лабиринт: пример (Blocking maze): example, 288–290
- Блэкджек: пример (Blackjack): examples, 143–146, 154, 166
- Вершины-действия (Action nodes) 96
- Вершины-состояния (State nodes) 96
- Взаимодействие (Interaction)
агент–среда (agent–environment) 75
обучение на основе взаимодействия (learning from) 12
- Взаимодействие (Interaction) 17
- Вознаграждения (Rewards)
в задаче об n руках бандите (in the n -armed bandit problem) 43
и цели (and goals) 80–82
эталонные (reference) 62
- Вознаграждения (Rewards) 18–19, 74–75, 76, 77, 110
- Вторичные подкрепляющие стимулы (Secondary reinforcers) 37
- Выборочные дублирования (Sample backups) 171, 296, 301, 313
- Выгода (Return)
 n -шаговая выгода, или уточненная n -шаговая усеченная выгода (n -step, or corrected n -step truncated) 205
единные обозначения для нее (unified notation for) 85–86
и функции ценности (and value functions) 96–97, 103
- Выгода (Return) 82–85, 110, 315
- Гамильтон, Уильям (Hamilton, William) 30, 114
- Генетические алгоритмы (Genetic algorithms) 20, 23, 36, 71, 276
- Глобальный оптимум (Global optimum) 240
- Гольф: пример (Golf): example, 99–101, 103–104, 109
- Гоночный трек: пример (Racetrack): example, 162–163, 166
- Грубое кодирование (Coarse coding) 248–252, 255
- Групповая корректировка или Групповое обучение (Batch updating or Batch training) 178–191, 199
- ДП в реальном времени (Real-time DP) 311
- ДП, см. Динамическое программирование (DP). см. Dynamic programming
- Действия (Actions)
в задаче об n руках бандите (in the n -armed bandit problem) 43
- Действия (Actions) 74–75
- Детеныш газели: пример (Gazelle calf): example, 16–17
- Диаграммы предшествующих состояний (Backup diagrams)
для n -шаговых TD-методов (for n -step TD methods) 205
для $Q(\lambda)$ (for $Q(\lambda)$) 225, 227

- для SARSA (for Sarsa(λ)) 221
- для TD(λ) (for TD(λ)) 210
- для TD (0) (for TD (0)) 171
- для ДП (for DP) 97–98, 104–105
- для методов Монте-Карло (for Monte Carlo methods) 147, 149
- для одношагового Q -обучения (for one-step Q-learning) 188
- для одношаговых методов (for one-step methods) 297
- для полуудублирований (for half-backups) 102
- для программы Сэмюэля, играющей в шашки (for Samuel's checker player) 330
- сложные (complex) 209
- Диаграммы предшествующих состояний (Backup diagrams) 98, 234
- Дilemma изучения-использования (Exploration-exploitation dilemma)
 - в задаче об n руках бандите (in the n -armed bandit problem) 43–44, 49, 68–72
- Дilemma изучения-использования (Exploration-exploitation dilemma) 14, 165, 182, 289–291
- Динамическое программирование (ДП) (Dynamic programming (DP))
 - vs. методы Монте-Карло (vs. Monte Carlo methods) 142–151, 164–166, 168, 314
 - ДП в приращениях (incremental) 139
 - аппроксимация функций и ДП (function approximation and) 238, 265–268, 272, 274, 276
 - диаграммы предшествующих состояний для ДП (backup diagrams for) 97–98, 104–105
 - и искусственный интеллект (and artificial intelligence) 139
 - обучение на основе временных различий и ДП (temporal-difference learning and) 168–171, 174, 199, 314
 - обучение с подкреплением и ДП (reinforcement learning and) 20, 30–31, 40, 117, 139
 - эффективность ДП (efficiency of) 137–138
- Динамическое программирование (ДП) (Dynamic programming (DP)) 30–31, 72, 117–141
- Динамическое распределение каналов: в системах сотовой телефонной связи (Dynamic channel allocation): in cellular telephone systems, 343–349
- Дуальное управление (Dual control) 71
- Дублирования (Backups)
 - n -шаговые (n -step) 203–205, 206, 209–213
 - в динамическом программировании (in dynamic programming) 119
 - в шашечной программе Сэмюэля (in Samuel's checker player) 328–330
 - в эвристическом поиске (in heuristic search) 308–310
 - и аппроксимация функций (function approximation and) 238–239, 241, 247
 - и приоритетная прогонка (prioritized sweeping and) 293–294
 - и траекторная выборка (trajectory sampling and) 303–306

- их распределение (distribution of) 239–241, 247, 265–267, 301–304, 308, 310, 311
полное vs. выборочное (full vs. sample) 296–301
Дублирования (Backups) 25, 98
- Ж**адные действия (Greedy actions) 44
Жадные стратегии (Greedy policies)
 ε -жадные стратегии (ε -greedy policies) 155
Жадные стратегии (Greedy policies) 105, 124, 151
Жадных или ε -жадных действий методы выбора (Greedy or ε -greedy action selection methods) 45–47, 71
- Завтрак: пример (Breakfast): example, 16–17, 40
Задача Дирихле (Dirichlet problem) 166
Задача игрока (Gambler's problem) 131–133
Задача об N -руком бандите (N -armed bandit problem)
 и ассоциативный поиск (associative search and) 67–68
 и методы Монте-Карло (Monte Carlo methods and) 163
 и методы, основанные на ценности действия (action-value methods and) 45–50
 и начальные оценки ценности действий (initial action-value estimates and) 60–62
 и нестационарные среды (nonstationary environments and) 58–59
 методы преследования для них (pursuit methods for) 64–66
 методы сравнения с подкреплением для них (reinforcement comparison methods for) 62–64
 оценивание vs. инструктирования в этой задаче (evaluation vs. instruction in) 50–56
Задача об N руком бандите (N -armed bandit problem) 35, 43–44, 71
Задача об автомобиле на горе (Mountain-car task) 262–263
Задача планирования: пример прикладной проблемы (Job-shop scheduling):
 case study, 349–355
Задача прогнозирования (Prediction problem) 118
Задача среднего вознаграждения, см. Неприведенные продолжающиеся задачи (Average-reward case). см. Undiscounted continuing tasks
Задачи о бандите, см. Задача об N -руком бандите (Bandit problems) 71, 163.
 см. также N -armed bandit problem
Задачи с сеткой (Grid tasks or Gridworld examples)
 и ДП (and DP) 121–123
 и следы приемлемости (and eligibility traces) 221–222
 и функции ценности (and value functions) 98–100, 106–108
 прогулка у пропасти (cliff walking) 187
 с учетом ветра (windy) 184–186
Задачи с сеткой (Grid tasks or Gridworld examples)
Закон влияния (Law of Effect) 32–33, 72

Замещающие следы или Методы замещения следов, см. Следы приемлемости (Replacing traces or Replace-trace methods) 228–231, 271 (рис.).
см. также Eligibility traces

ИИ, см. Искусственный интеллект (AI). см. Artificial intelligence
Игра «сам-с-собой» (Self-play) 28, 326, 328
Иерархия и модульность (Hierarchy and modularity) 318
Избирательная самонастраивающаяся адаптация (Selective bootstrap adaptation) 34, 276–277
Избирательное обучение (Selectional learning) 32, 33, 34
Изучающие старты (Exploring starts) 149, 152, 155, 165
Изучающие ходы или действия (Exploratory moves or actions) 22, 25 (рис.), 28, 223–226
Изучение модели, настройка модели (Model-learning) 282–291, 311
Имитируемый опыт (Simulated experience) 142
Инструктирование vs. оценивание (Instruction vs. evaluation) 42, 50–56
Информационное состояние (Information state) 69, 72
Искусственный интеллект (ИИ) (Artificial intelligence (AI))
и ДП (and DP) 139
Искусственный интеллект (ИИ) (Artificial intelligence (AI)) 12, 14
Использование приведенных величин (Discounting) 83–84, 86, 315
История обучения с подкреплением (History of reinforcement learning) 29–40
Итерация по стратегиям, см. Обобщенная итерация по стратегиям (Policy iteration)
с помощью методов Монте-Карло (by Monte Carlo methods) 150–151
Итерация по стратегиям, см. Обобщенная итерация по стратегиям (Policy iteration) 126–129. см. также Generalized policy iteration

Квазиньютоновские методы (Quasi-Newton methods) 273
Классифицирующие системы (Classifier systems) 36, 39, 276
Клопф, Харри (Klopf, Harry) 36–39, 198, 235
Кодирование по методу Канервы (Kanerva coding) 256–258, 274
Компания «Автомобили напрокат»: пример (Jack's car rental): example, 127–129, 197
Контирипример Бэрда (Baird's counterexample) 265–267, 270, 274
Короткий путь в лабиринте: пример (Shortcut maze): example, 289–291
Коэффициент ветвления (Branching factor) 300
Коэффициент приведения (Discount-rate parameter) 83, 339
Критик (Critic) 36, 189–190, 199
Кубик Рубика: пример (Rubik's cube): example, 76

Лабиринт Dyna (Dyna maze) 285–288, 295
Линейные методы, аппроксимация функций с их использованием (Linear methods, function approximation using)
границы прогнозируемой ошибки (bound on prediction error) 247
сходимость (convergence of) 247, 273–274

Линейные методы, аппроксимация функций с их использованием (Linear methods, function approximation using) 246–247

Локальный оптимум (Local optimum) 241, 243–245, 247

МК-методы, см. Методы Монте-Карло (MC methods)

всех посещений (every-visit) 143, 149, 166, 231

первого посещения (first-visit) 143–144, 149, 166, 231

с параметром α (constant- α) 169, 172, 175–179, 181, 211

МК-методы, см. Методы Монте-Карло (MC methods). см. также Monte Carlo methods

МНК, см. Метод наименьших квадратов (LMS). см. Least-mean-square algorithm

МППР, см. Марковские процессы принятия решений (MDPs). см. Markov decision processes

Марковские процессы принятия решений (МППР), см. Частично наблюдаемые МППР, Полумарковские процессы принятия решений (Markov decision processes (MDPs)) 29, 30, 40, 92–94, 111–114.

см. также Partially observable MDPs, Semi-Markov decision process

Метод Монте-Карло с изучающими стартами (Monte Carlo with Exploring Starts (Monte Carlo ES)) 152–154

Метод наименьших квадратов (МНК) (Least-mean-square (LMS) algorithm) 36, 273

Методы Монте-Карло, см. МК-методы (Monte Carlo methods)

n -шаговое дублирование и МК-методы (n -step backups and) 203–206, 210–212

диаграммы предшествующих состояний для них (backup diagrams for) 147, 149

и TD-обучение (and TD learning) 168–182, 209, 215

и методы ДП (and DP methods) 142–143, 164–166

и управление (and control) 150–154

их пошаговая реализация (incremental implementation of) 163–164

их преимущества (advantages of) 164–166

их сходимость (convergence of) 143, 152–154, 158

следы приемлемости и МК-методы (eligibility traces and) 203, 212, 231, 233–235

управление МК-методом с интегрированной оценкой ценности стратегий (on-policy control by) 155–158

управление МК-методом с разделенной оценкой ценности стратегий (off-policy control by) 160–163

Методы Монте-Карло, см. МК-методы (Monte Carlo methods) 142–166. см. также MC methods

Методы дерева решений (Decision-tree methods) 242, 277

Методы интервального оценивания (Interval estimation methods) 69, 72

- Методы исполнителя—критика (Actor—critic methods)
и агенты Dyna (and Dyna) 289 (рис.)
следы приемлемости для... (eligibility traces for) 227–228, 235
- Методы исполнителя—критика (Actor—critic methods) 39, 62, 189–192, 199–200
- Методы наискорейшего спуска (Gradient-descent methods) 242–247, 258–261,
272–276
- Методы остаточного градиента (Residual-gradient methods) 274
- Методы преследования (Pursuit methods) 64–67
- Методы сравнения с подкреплением (Reinforcement comparison methods)
62–64, 72, 190
- Методы среднего выборочного (Sample-average methods) 46
- Методы, основанные на ценности действия (Action-value methods) 45–49
- Минимакс (Minimax) 22, 328–330
- Мински, Марвин (Minsky, Marvin) 32, 38, 39, 139
- Мичи, Дональд (Michie, Donald) 34–35, 113, 114, 166, 273
- Модели (окружающей среды) (Models (of the environment))
и планирование (and planning) 21, 278–288
неверные (incorrect) 288–291
типы (types of) 278–279
- Модели (окружающей среды) (Models (of the environment)) 21, 111, 278–279
- Модели классического (павловского) условного рефлекса (Classical
conditioning models) 39
- Модели распределения (Distribution models) 278–279, 299
- Модифицированная итерация по стратегиям (Modified policy iteration) 141
- Модифицированное Q-обучение (Modified Q-learning) 199
- Мозаично распределенная память (Sparse distributed memory) 274
- Мыльный пузырь: пример (Soap bubble): example, 149, 166
- Накапливающиеся следы или Методы накапливания следов (Accumulating
traces or Accumulate-trace methods)**
замещающие следы (vs. replacing traces) 228–231, 271 (рис.)
- Накапливающиеся следы или Методы накапливания следов (Accumulating
traces or Accumulate-trace methods) 213, 217. см. также Eligibility
traces
- Нарды: программа для них, см. TD-Gammon (Backgammon): program for. см.
TD-Gammon
- Неассоциативные задачи и методы (Nonassociative problems and methods) 36,
42, 57, 350
- Нейробиология (Neuroscience) 39, 235
- Нейронные сети (Neural networks) 38, 39, 40, 276, 322–325
- Немарковские задачи (Non-Markov problems) 89, 90, 192, 233–234, 317–318
- Неполное знание (Incomplete knowledge) 111
- Непрерывное время (Continuous time) 75, 115, 337, 338–340
- Непрерывное действие (Continuous action) 117, 192, 237, 259, 277
- Непрерывное состояние (Continuous state) 89, 115, 117, 139, 237, 248, 252, 263

- Непрямое обучение с подкреплением (Indirect reinforcement learning) 282–283, 311
- Несамонастраивающиеся методы (Nonbootstrapping methods) 270–272
- Нестационарность (Nonstationarity) 49, 58–60, 163, 239
- ОИС, см. Обобщенная итерация по стратегиям (GPI). см. Generalized policy iteration
- Обобщение (Generalization) 237–276
- Обобщенная итерация по стратегиям (ОИС) (Generalized policy iteration (GPI))
- и TD-обучение (and TD learning) 182, 193, 197–198
 - и методы Монте-Карло (and Monte Carlo methods) 150, 155, 160, 165
- Обобщенная итерация по стратегиям (ОИС) (Generalized policy iteration (GPI)) 135–137, 138, 258, 259, 313
- Обобщенное подкрепление (Generalized reinforcement) 39
- Обратный подход к следам приемлемости (Backward view of eligibility traces)
- в $Q(\lambda)$ (in $Q(\lambda)$) 223
 - в $TD(\lambda)$ (in $TD(\lambda)$) 213–219
 - с аппроксимацией функций (with function approximation) 245, 259
 - эквивалентность с прямым подходом (equivalence with forward view) 217–220
- Обратный подход к следам приемлемости (Backward view of eligibility traces) 203, 215 (рис.), 234, 235
- Обучающийся автомат (Learning automata) 35, 54–55, 71, 72
- Обучение, см. по типу (Learning)
- и планирование (and planning) 21, 278–311, 318–319
 - на примерах, см. Обучение с учителем (from examples). см. Supervised learning
 - через взаимодействие (from interaction) 12, 21, 27
- Обучение, см. по типу (Learning). см. также by type
- Объединение состояний (State aggregation) 245–246, 248, 273–276, 318
- Одношаговое табличное Q-планирование со случайной выборкой (Random-sample one-step tabular Q-planning) 280–282
- Одношаговые методы (One-step methods)
- диаграммы предшествующих состояний для них (backups diagrams for) 297
- Одношаговые методы (One-step methods) 203
- Окружающая среда (Environment)
- и агент (and agent) 74–77
 - марковское свойство и среда (Markov property and) 89
 - модели среды (models of) 278
- Окружающая среда (Environment) 12, 74
- Оперативная корректировка (On-line updating) 206, 215, 235
- Оптимальное управление (Optimal control) 29–31, 112–114

- Оптимальность (Optimality,)
для TD(0) (of TD(0)) 178–182
и аппроксимация (and approximation) 109–110
и функции ценности (and value functions) 103–109
- Оптимальность (Optimality,)
- Оптимистичные начальные ценности (Optimistic initial values) 60–62, 263
- Опыт (Experience) 142, 279
- Отсроченное вознаграждение (Delayed reward) 13, 116, 234
- Оценивание vs. инструктирования (Evaluation vs. instruction) 42, 50–56
- Оценка на основе принципа максимума правдоподобия (Maximum-likelihood estimate) 181
- Оценка стратегии (Policy evaluation)
для ценности действий (for action values) 148–149, 182–184
ее завершение (termination of) 121
итеративная (iterative) 119–121
оценка одной стратегии при использовании другой (of one policy while following another) 158–160
с помощью TD-методов (by TD methods) 168–184
с помощью ДП (by DP) 118–123
с помощью методов Монте-Карло (by Monte Carlo methods) 143–149,
- Оценка стратегии (Policy evaluation) 118–123, 122 (рис.)
- Оценочная обратная связь (Evaluative feedback) 49–72
- Перегонка нефти: пример (Petroleum refinery): example, 16–18
- Планирование (Planning)
в агенте Dyna (in Dyna) 282–291
и ДП (and DP) 280
и обучение с подкреплением (and reinforcement learning) 14, 21
и обучение (and learning) 21, 278–311
и траекторная выборка (and trajectory sampling) 301–306
и эвристический поиск (and heuristic search) 306–310
интегрированное с действиями и обучением (integrated with action and learning) 282–291
планирование в пространстве состояний vs. планирование в пространстве планов (state-space vs. plan-space) 279
пошаговое (incremental) 282, 310
с помощью *Q*-обучения (by *Q*-learning) 280–282
совещательность против реагирования в нем (deliberation vs. reaction in) 21, 283
частичного порядка (partial-order) 279
- Планирование (Planning) 21, 278–311, 318–319
- Повторное воспроизведение опыта (Experience replay) 352
- Поглощающее состояние (Absorbing state) 85
- Поездка домой: пример (Driving home): example, 171–174
- Показатели Гиттисса (Gittins indices) 71, 72

- Полное знание (Complete knowledge) 31, 111
Полные дублирования (Full backups) 119, 138, 296–301, 313–314
Полумарковский процесс принятия решений (Semi-Markov decision process) 337–339, 344–345
Последовательный план экспериментов (Sequential design of experiments) 71, 113
Послесостояния (Afterstates) 195–197
Предпочтения действий (Action preferences)
 в задаче n -ру�ого бандита (in the n -armed bandit problem) 62–63, 64, 65
Предпочтения действий (Action preferences) 192, 227
Представления состояний (State representations)
 использование конкретной структуры (exploiting structure in) 319
 улучшенные (augmenting) 317–318
Представления состояний (State representations) 86–87
Пример с очередью, управление доступом (Queuing example, access-control) 193–194, 197
Примеры с лабиринтами (Maze examples) 285–288, 289, 291, 294
Примеры с роботом (Robot examples)
 перемещение предметов (pick and place) 77
 сбор банок (recycling) 78, 80, 92–96, 106, 112
 сбор мусора (trash collecting) 16–18
Примеры с роботом (Robot examples) 74, 76, 80, 82, 84, 248, 331
Приоритетная прогонка (Prioritized sweeping) 293–294, 295 (рис.), 310, 311
Приятие решений как реакция на текущие изменения (Reactive decision-making) 283
Проблема загрузки космического корабля: Прикладная задача (Space shuttle payload processing (SSPP)): case study, 349–351
Проблема обучения с подкреплением (Reinforcement learning problem) 13, 74–115
Программа для игры в шашки: программа Сэмюэля (Checkers player): Samuel's, 139, 276, 320, 327–331
Прогулка у пропасти: пример (Cliff walking): example, 187, 188 (рис.)
Продолжающиеся задачи (Continuing tasks) 83, 85–87, 113
Прокат автомобилей: пример (Car rental): example, 127–129, 197
Проклятие размерности (Curse of dimensionality) 29, 137, 254
Просмотровые таблицы, см. табличные методы (Lookup tables). см. tabular methods
Простой метод $Q(\lambda)$ (Naive $Q(\lambda)$) 226
Пространство сопряженности (Contingency space) 53 (рис.), 72
Прямое обучение с подкреплением (Прямое ОП) (Direct reinforcement learning (Direct RL)) 282–286, 311
Прямой подход к следам приемлемости (Forward view of eligibility traces)
 в $Q(\lambda)$ (in $Q(\lambda)$) 223–227
 в $TD(\lambda)$ (in $TD(\lambda)$) 209–213
 с аппроксимацией функций (with function approximation) 243–245

- с переменной λ (with variable λ) 233
эквивалентность с обратным подходом (equivalence with backward view) 217–220, 235
Прямой подход к следам приемлемости (Forward view of eligibility traces) 203, 211 (рис.), 234
Психология и обучение с подкреплением, см. Модели классического условного рефлекса, Закон влияния, Вторичные подкрепляющие стимулы (Psychology and reinforcement learning)
и ассоциации стимул–реакция (and stimulus–response associations) 18
и следы стимулов (and stimulus traces) 235
и формирование навыка (and shaping) 319
Психология и обучение с подкреплением, см. Модели классического условного рефлекса, Закон влияния, Вторичные подкрепляющие стимулы (Psychology and reinforcement learning) 29, 71, 311.
см. также Classical conditioning models, Law of Effect, Secondary reinforcers
Психология обучения животных, см. Психология и обучение с подкреплением (Animal learning psychology). см. Psychology and reinforcement learning
Пятикарточный покер: пример (Draw poker): example, 90–91

РБФ, см. Радиально-базисные функции (RBFs). см. Radial basis functions
Радиально-базисные функции (РБФ) (Radial basis functions (RBFs)) 255, 274
Распознавание образов (Pattern recognition) 13, 32
Распределение Больцмана (Boltzmann distribution) 49
Распределение Гиббса (Gibbs distribution) 49–50
Распределение коэффициентов уверенности (Credit assignment) 32, 203, 235
Распределение с интегрированной оценкой ценности стратегий (On-policy distribution)
vs. равномерного распределение (vs. uniform distribution) 302–304
Распределение с интегрированной оценкой ценности стратегий (On-policy distribution) 241, 247, 265–266
Расстояние Хемминга (Hamming distance) 258
Рекурсивные методы наименьших квадратов (Recursive-least-square methods) 273
Робот, собирающий банки: пример (Recycling robot): examples, 16–18, 78, 80, 92–96, 106, 112
Розенблattt, Фрэнк (Rosenblatt, Frank) 33

СКО, см. Среднеквадратическая ошибка (RMS error). см. Root mean-squared error
СКО, Среднеквадратическая ошибка (MSE). см. Mean-squared error
Самонастройка (Bootstrapping)
ее оценка (assessment of) 175–179, 270–273, 274
и TD-обучение (and TD learning) 169, 174

- и ДП (and DP) 139
- и аппроксимация функций (and function approximation) 239, 245, 247, 265–272, 272–273
- и методы Монте-Карло (and Monte Carlo methods) 147
- и следы приемлемости (and eligibility traces) 270–271
- Самонастройка (Bootstrapping) 139
- Свойства (Features) 246–261, 276
- Свойство уменьшения ошибки (Error reduction property) 206
- Сеть CMAC, см. Мозаичное кодирование (CMAC). *см.* Tile coding
- Сеть ассоциативной памяти (Associative memory network) 276
- Сигнатурные таблицы (Signature tables) 276–331
- Следы приемлемости для пар состояния–действие (State-action eligibility traces) 220–221, 231
- Следы приемлемости, см. Накапливающиеся следы, Замещающие следы (Eligibility traces)
 - для методов исполнитель-критик (for actor-critic methods) 227–228, 235
 - и Q-обучение (and Q-learning) 223–227, 235
 - и SARSA (and Sarsa) 220–222, 235
 - с переменной λ (with variable λ) 231–233, 235
 - характеристики с учетом следов приемлемости (performance with) 271 (рис.)
- Следы приемлемости, см. Накапливающиеся следы, Замещающие следы (Eligibility traces) 203–235. *см. также* Accumulating traces, Replacing traces
- Сложные дублирования (Complex backups) 209
- Случайные блуждания: примеры (Random walk)
 - n -шаговые методы для них (with n -step methods on) 206–208
 - TD(λ) для них (TD(λ) on) 219–220
 - TD-методы vs. методы Монте-Карло для них (TD vs. MC methods on) 175–179
 - алгоритм λ -выгоды для них (λ -return algorithm on) 212
 - групповая корректировка для них (batch updating on) 179
 - для 19 состояний (19-state) 207–208, 212, 219–220
 - для пяти состояний (five state) 175–179, 206–207
 - получение значений ценности для них с помощью TD(0) (values learned by TD(0) on) 177 (рис.)
- Случайные блуждания: примеры (Random walk): examples,
- Смоделированные совокупности эпизодов (Sample models) 164, 278–279
- Состояние (состояния), см. Представления состояний (State (s)) 75–78, 87–91, 110, 112. *см. также* State representations
- Состояния-прототипы (Prototype states) 258
- Сотовые телефоны: динамическое распределение каналов (Cellular telephones): dynamic channel allocation in, 343–349
- Среднеквадратическая ошибка (Root mean-squared error) 177 (рис.)

- Среднеквадратическая ошибка (СКО) (Mean-squared error (MSE)) 239–241, 247
- Стохастически эквивалентная оценка (Certainty-equivalence estimate) 181–182, 199
- Стратегия (Policy)
- ε -жадная (ε -greedy) 155
 - гибкая, ε -гибкая (soft, ε -soft) 129, 155
 - детерминированная (deterministic) 121, 355
 - жадная (greedy) 105, 124, 151
 - оптимальная (optimal) 103
 - оценивания (estimation) 160
 - поведения (behavior) 160
 - случайная равновероятная (equiprobable random) 123
 - стохастическая (stochastic) 75
- Стратегия (Policy) 75
- Схемы усреднения (Averagers) 268
- Сэмюэль, Артур, см. Программа для игры в шашки (Samuel, Arthur) 38, 39, 139, 276. см. также Checkers player:
- Удовольствие и боль (Pleasure and pain) 18–20, 82
- Улучшение стратегии (Policy improvement)
- и методы Монте-Карло (and Monte Carlo methods) 150–151
 - теорема об улучшении стратегии (theorem of) 123–126
- Улучшение стратегии (Policy improvement) 123–127
- Управление лифтами: прикладная задача (Elevator dispatching): case study, 332–342
- Управление очередью запросов к серверам, пример (Access-control queuing example) 193–194, 197, 200
- Управление перемещением стержня: пример (Rod maneuvering): example, 294–296
- Управление поиском (Search control) 284
- Уравнение Гамильтона—Якоби—Беллмана (Hamilton-Jacobi-Bellman equation) 115
- Уравнения Беллмана (Bellman equations)
- для Q^* (for Q^*) 104
 - для Q^π (for Q^π) 100
 - для V^* (for V^*) 104
 - для V^π (for V^π) 97
 - и ДП (and DP) 130, 138
 - и дублирования (and backups) 138
 - их решение (solving the) 105–109
- Уравнения Беллмана (Bellman equations) 29, 115, 118
- Условия сходимости стохастической аппроксимации (Stochastic approximation convergence conditions) 60
- Фарли и Кларк (Farley and Clark) 32, 274

- Функции ценности действия (Action-value functions)
и TD-обучение (and TD learning) 182–184, 186, 190, 193
и аппроксимация функций (and function approximation) 258–259
и метод Монте-Карло (and Monte Carlo methods) 149–150
оптимальные (optimal) 103, 106, 114
Функции ценности действия (Action-value functions) 96, 114, 315
- Хеширование (Hashing) 254, 274
Холланд, Джон (Holland, John) 36, 39, 71, 198–199, 276
- Целевое состояние (Goal state) 293
Цели (Goals) 13, 16, 80–82
Ценности действий (Action values)
в задаче об n -руком бандите (in the n -armed bandit problem) 43, 45–46, 71
Ценности действий (Action values) 96, 100–102, 149
Ценности состояний или Функции ценности состояний (State values or State value functions) 96–100, 103–105, 114
Цепочка приборов с зарядовой связью (Bucket brigade) 199
- ЧН-МППР, см. Частично наблюдаемые МППР (POMDPs). см. Partially observable MDPs
Частично наблюдаемые МППР (Partially observable MDPs (POMDPs)) 31, 72, 317–318
- Шахматы (Chess) 16–18, 80, 109, 114, 195, 277
Шашечная программа Сэмюэля (Samuel's)
Шашки (Checkers) 80, 87, 308–310, 327–331
Шеннон, Клод (Shannon, Claude) 38, 114, 327–328
- Эволюционные методы (Evolutionary methods) 21, 23, 27, 36, 276, 279
Эволюция (Evolution) 21, 32
Эвристический поиск (Heuristic search)
в программе TD-Gammon (in TD-Gammon) 325
как последовательность дублирований (as sequence of backups) 310–311
(рис.)
как разложение уравнений Беллмана (as expanding Bellman equations) 108
Эвристический поиск (Heuristic search) 306–310, 314
Эвристический поиск в реальном времени (Real-time heuristic search) 311
Эвристическое динамическое программирование (Heuristic dynamic programming) 139
Эпизоды, эпизодные задачи (Episodes, episodic tasks) 83–86, 112
Эталонное вознаграждение (Reference reward) 62–63
- Алгоритм λ -выгоды (λ -return algorithm) 211–212, 217–221
алгоритм типа Гаусса–Зейделя (Gauss–Seidel-style algorithm) 141
- И обучение с подкреплением (and reinforcement learning) 28, 32–40, 112

- Марковское свойство (Markov property) 87–91, 111, 165, 317–318
машинное обучение (Machine learning)
специальные выпуски журнала по обучению с подкреплением (special journal issues on reinforcement learning) 40
машинное обучение (Machine learning) 13, 14, 40
метод $Q(\lambda)$ Пенга (Peng's $Q(\lambda)$) 223, 225–227 (рис.), 235
методы с интегрированной оценкой ценности стратегий (On-policy methods)
SARSA с интегрированной оценкой ценности стратегий (Sarsa as) 182–186, 220–222
методы Монте-Карло с интегрированной оценкой ценности стратегий (Monte Carlo) 155–158
методы с интегрированной оценкой ценности стратегий vs. методы с разделенной оценкой ценности стратегий (vs. off-policy methods) 165, 186–188
методы с интегрированной оценкой ценности стратегий (On-policy methods) 155, 165
методы с разделенной оценкой ценности стратегий (Off-policy methods)
 Q -обучение как эти методы (Q -learning as) 223
ДП как эти методы (DP as) 265
методы Монте-Карло с разделенной оценкой ценности стратегий (Monte Carlo) 158–163
проблемы с самонастройкой в методах с разделенной оценкой ценности стратегий (problems with bootstrapping in) 265–270, 272
методы с разделенной оценкой ценности стратегий (Off-policy methods) 160–161, 259
- Параметры, определяющие размер шага (Step-size parameters) 26–27, 57, 58–60
пример (Driving): example, 78
- Система STeLLA (STeLLA system) 33

Оглавление

Предисловие редактора серии «Adaptive computation and machine learning»	5
Предисловие	6
Часть I. Постановка задачи и подходы к ее решению...	11
Глава 1. Введение.....	12
1.1. Обучение с подкреплением	12
1.2. Примеры.....	16
1.3. Элементы обучения с подкреплением.....	18
1.4. Подробный пример: крестики-нолики	21
1.5. Итоги	29
1.6. История обучения с подкреплением	29
1.7. Библиографические и исторические справки	40
Глава 2. Оценочная обратная связь.....	42
2.1. Задача об n -руком бандите.....	43
2.2. Методы вычисления значений ценности действий	45
2.3. Выбор действия с помощью операции softmax	49
*2.4. Оценивание в сравнении с инструктированием	50
2.5. Пошаговая реализация обучения	56
2.6. Нестационарные задачи	58
2.7. Оптимистичные начальные оценки.....	60
*2.8. Сравнение с подкреплением	62
*2.9. Методы преследования	65
*2.10. Ассоциативный поиск	67
2.11. Итоги	69
2.12. Библиографические и исторические справки	71
Глава 3. Задача обучения с подкреплением.....	74
3.1. Взаимосвязь агент — окружающая среда	74
3.2. Цели и вознаграждения	80
3.3. Выгода	82
3.4. Единые обозначения для непрерывных задачий и заданий, состоящих из эпизодов	85

*3.5. Марковское свойство	86
3.6. Марковские процессы принятия решений	92
3.7. Функции ценности	96
3.8. Оптимальные функции ценности	103
3.9. Оптимальность и аппроксимация	109
3.10. Итоги	110
3.11. Библиографические и исторические справки	112
Часть II. Фундаментальные методы решения	116
Глава 4. Динамическое программирование	117
4.1. Оценка стратегии	118
4.2. Улучшение стратегии	123
4.3. Итерация по стратегиям	126
4.4. Итерация по ценностям	129
4.5. Асинхронное динамическое программирование	133
4.6. Обобщенная итерация по стратегиям	135
4.7. Эффективность динамического программирования	137
4.8. Итоги	138
4.9. Библиографические и исторические справки	140
Глава 5. Методы Монте-Карло	142
5.1. Оценка стратегии методами Монте-Карло	143
5.2. Оценка ценности действия методом Монте-Карло	149
5.3. Формирование управления методом Монте-Карло	150
5.4. Управление по методу Монте-Карло с интегрированной оценкой ценности стратегий	155
5.5. Оценивание одной стратегии при использовании другой ..	158
5.6. Управление по методу Монте-Карло с разделенной оцен- кой ценности стратегий	160
5.7. Пошаговая реализация	163
5.8. Итоги	164
5.9. Библиографические и исторические справки	166
Глава 6. Обучение на основе временных различий	168
6.1. Предсказание на основе временных различий	168
6.2. Преимущества TD-методов предсказания	174
6.3. Оптимальность метода TD(0)	178
6.4. SARSA: управление по TD-методу с интегрированной оценкой ценности стратегий	182
6.5. Q-обучение: управление по TD-методу с разделенной оценкой ценности стратегий	186

*6.6. Методы исполнитель—критик	189
*6.7. <i>R</i> -обучение для неприведенных продолжающихся задач ..	192
6.8. Игры, послесостояния и другие особые случаи	195
6.9. Итоги	197
6.10. Библиографические и исторические справки	198
Часть III. Единый подход	201
Глава 7. Следы приемлемости	202
7.1. <i>n</i> -шаговое TD-прогнозирование	203
7.2. Прямой подход к методам TD(λ)	209
7.3. Обратный подход к методам TD(λ)	213
7.4. Эквивалентность прямого и обратного представлений	217
7.5. SARSA(λ)	220
7.6. Метод Q(λ)	223
*7.7. Следы приемлемости для методов типа исполнитель— критик	227
7.8. Замещающие следы	228
7.9. Проблемы реализации	231
*7.10. Переменный параметр λ	232
7.11. Итоги	233
7.12. Библиографические и исторические справки	234
Глава 8. Обобщение и аппроксимация функций	237
8.1. Прогнозирование ценности при помощи аппроксимации функции	238
8.2. Методы наискорейшего спуска	242
8.3. Линейные методы	246
8.4. Управление с аппроксимацией функции	258
8.5. Самонастройка с разделенной оценкой ценности страте- гий	264
8.6. Нужна ли самонастройка?	270
8.7. Итоги	272
8.8. Библиографические и исторические справки	273
Глава 9. Планирование и обучение	278
9.1. Модели и планирование	278
9.2. Объединение планирования, исполнения и обучения	282
9.3. Когда модель неверна	288
9.4. Приоритетная прогонка	291
9.5. Сравнение полного и выборочного вариантов дублирова- ния	296

9.6. Траекторная выборка	301
9.7. Эвристический поиск	306
9.8. Итоги	310
9.9. Библиографические и исторические справки	311
Глава 10. Важнейшие аспекты обучения с подкреплением	313
10.1. Единый подход	313
10.2. Некоторые другие новые направления	317
Глава 11. Конкретные примеры	320
11.1. Программа TD-Gammon	320
11.2. Программа игры в шашки Сэмюеля	327
11.3. Акробот	331
11.4. Управление лифтом	335
11.5. Динамическое распределение каналов	342
11.6. Задача планирования	348
Список обозначений	357
Список литературы	359
Предметный указатель	380

Минимальные системные требования определяются соответствующими требованиями программы Adobe Reader версии не ниже 11-й для операционных систем Windows, Mac OS, Android, iOS, Windows Phone и BlackBerry; экран 10"

Научное электронное издание

Серия: «Адаптивные и интеллектуальные системы»

**Саттон Ричард С.
Барто Эндрю Г.**

ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ

Ведущие редакторы *M. C. Стригунова, И. А. Маховая*

Редактор *Н. А. Шихова*

Художник *Н. А. Лозинская*

Художественный редактор *Н. А. Новак*

Оригинал-макет подготовлен *M. Ю. Копаницкой* в пакете L^AT_EX 2_•

Подписано к использованию 18.09.14.

Издательство «БИНОМ. Лаборатория знаний»

125167, Москва, проезд Аэропорта, д. 3

Телефон: (499) 157-5272

e-mail: binom@Lbz.ru, <http://www.Lbz.ru>

Данная интеграционная система, которая является для организаций возможностью выйти в широкий временной срок, а также отсутствием для системы единого временного фактора, должна приводить к снижению производительности труда и развитию общества. Следует отметить, что в сфере инфраструктурных и социальных услуг система – обменение с внешней средой. Активные элементы представляют собой интегрированные технологии для функционирования общества. Они являются структурными компонентами инфраструктуры и являются основой для дальнейшего развития инфраструктуры концепции интегрированной инфраструктуры, а также взаимодействия с внешней средой в сфере инфраструктурных и социальных услуг. Важное значение имеет синхронизация инфраструктурных и социальных услуг, что способствует более эффективному функционированию инфраструктурного комплекса. Обменение с внешней средой является важным фактором для успешного функционирования инфраструктурных и социальных услуг, что способствует более эффективному функционированию инфраструктурных и социальных услуг.

Решение «Система интеграции национальных инфраструктурных и социальных услуг» включает в себя ряд мероприятий. Оно может представлять собой комплексное управление национальной инфраструктурой и социальными услугами в сфере инфраструктурных и социальных услуг, а также интеграцию инфраструктурных и социальных услуг в сфере инфраструктурных и социальных услуг.