Стюарт
Рассел

Питер
Норвиг

Искусственный интеллект
Современный подход
Четвертое издание

Том 2
Знания и рассуждения в условиях неопределенности

Искусственный интеллект
Современный подход

Четвертое издание

Том 2.
Знания и рассуждения в условиях неопределенности

Artificial Intelligence
A Modern Approach
Fourth Edition

Stuart J. Russel and Peter Norvig

Pearson

Искусственный интеллект
Современный подход

Четвертое издание

Том 2.
Знания и рассуждения в условиях неопределенности

С. Рассел, П. Норвиг

Диалектика

Москва · Санкт-Петербург
2021

ББК 32.813
Р24
УДК 004.8

ООО "Диалектика"

Перевод с английского и редакция *А.В. Слепцова*

По общим вопросам обращайтесь в издательство "Диалектика"
по адресу: info.dialektika@gmail.com, http://www.dialektika.com

Рассел, Стюарт, **Норвиг**, Питер.

Р24 Искусственный интеллект: современный подход, 4-е изд., том 2. Знания и рассуждения в условиях неопределенности. : Пер. с англ. — СПб. : ООО "Диалектика", 2021. — 480 с. — Парал. тит. англ.

ISBN 978-5-907365-26-1 (рус., том 2)

ISBN 978-5-907365-24-7 (рус., многотом.)

ББК 32.813

Научно-популярное издание

Стюарт Рассел, Питер Норвиг

Искусственный интеллект: современный подход
4-е издание, том 2
Знания и рассуждения в условиях неопределенности

Подписано в печать 02.08.2021. Формат 70x100/16
Усл. печ. л. 38,7. Уч.-изд. л. 29,6.
Тираж 400 экз. Заказ № 5941.

Отпечатано в АО "Первая Образцовая типография"
Филиал "Чеховский Печатный Двор"
142300, Московская область, г. Чехов, ул. Полиграфистов, д. 1
Сайт: www.chpd.ru, E-mail: sales@chpd.ru, тел. 8(499) 270-73-59

ООО "Диалектика", 195027, Санкт-Петербург, Магнитогорская ул., д. 30, лит. А, пом. 848

ISBN 978-5-907365-26-1 (рус., том 2)
ISBN 978-5-907365-24-7 (рус., многотом.)
ISBN 978-0-13-461099-3 (англ.)

Оглавление

Содержание

Часть IV

Неопределенные знания и рассуждения в условиях неопределенности

ГЛАВА 12

Количественная оценка неопределенности

В данной главе показано, как должен действовать агент в условиях неопределенности со степенью доверия, представленной в числовом виде.

12.1. Действия в условиях неопределенности

Агенты в реальном мире должны справляться с ▶**неопределенностью**, будь то по причине частичной наблюдаемости, недетерминизма или действий противников. Агент может никогда не знать наверняка, в каком состоянии он сейчас находится или где он окажется после выполнения некоторой последовательности действий.

Мы уже знакомы с агентами, решающими задачи, и логическими агентами, справляющимися с неопределенностью посредством отслеживания цепочки **доверительных состояний** — представления множества всех возможных состояний мира, которые могут в нем проявиться — и формирования условного плана, позволяющего справиться с любой возможной случайной ситуацией, о которой его датчики могут сообщить во время его выполнения. Подобный подход работает для простых задач, но имеет определенные недостатки.

- Агент должен рассмотреть *все возможные* объяснения для результатов наблюдений своих датчиков, и при этом не важно, насколько эти объяснения будут маловероятны. Такой подход приводит к формированию огромного доверительного состояния, полного почти невероятных возможностей.
- Правильный условный план, учитывающий любую возможность, может вырастать до сколь угодно больших размеров и должен учитывать сколь угодно маловероятные непредвиденные обстоятельства.
- Иногда не существует плана, который гарантированно приводит к цели, но агент все равно должен действовать. Он должен иметь какой-то способ сравнения достоинств различных планов, которые не являются гарантированно достигающими цели.

Например, предположим, что автоматизированному такси поставлена цель — доставить пассажира в аэропорт к заданному времени. Агент такси формирует план, A_{90}, предусматривающий выезд из дома за 90 минут до установленного времени отправления рейса и движение такси с разумной скоростью. Даже если аэропорт находится всего в 5 милях от дома, логический агент не сможет с абсолютной уверенностью прийти к заключению, что "План A_{90} позволяет добраться до аэропорта к назначенному времени". Вместо этого он придет к более слабому заключению: "План A_{90} позволяет прибыть в аэропорт вовремя, если машина не сломается и не попадет в аварию, и дорога не будет закрыта, и в машину не попадает метеорит, и..." Ни по одному из этих условий нельзя вынести гарантированно верное суждение, поэтому невозможно сделать вывод, что план обязательно будет успешным. Это логическая **проблема квалификации** (см. раздел 7.7.1), для которой до сих пор не найдено реального решения.

Тем не менее в некотором смысле план A_{90} действительно представляет собой правильное руководство к действию. Что имеется в виду под этим утверждением? Как уже говорилось в главе 2, под этим подразумевается, что из всех планов, которые могут быть выполнены, именно план A_{90}, как ожидается, позволит максимизировать показатели производительности агента (здесь это ожидание строится на основании знаний агента об окружающей среде). Показатели производительности включают своевременную доставку пассажира в аэропорт к указанному рейсу, предотвращение продолжительного, непродуктивного ожидания в аэропорту и исключение штрафов за превышение скорости по пути в аэропорт. Знания агента не позволяют гарантировать достижения любого из этих трех результатов при выполнении плана A_{90}, но могут обеспечить некоторую степень уверенности, что они будут достигнуты. Другие планы, например A_{180}, могут повысить степень уверенности агента в том, что он доставит пассажира до аэропорта вовремя, но одновременно повысят для него и вероятность продолжительного, скучного ожидания. ➔ *Следовательно, выбор правильного способа действия — **рационального решения** — зависит как от относительной важности различных целей, так и от степени уверенности в том, что они могут быть достигнуты.* В оставшейся части данного раздела эти идеи будут уточнены с целью подготовки к разработке общих теорий проведения рассуждений в условиях неопределенности и принятия рациональных решений, которые будут представлены в этой и последующих главах.

12.1.1. Учет наличия неопределенности

Рассмотрим простой пример рассуждений при наличии неопределенности: диагностика причин зубной боли у пациента. Диагностика — при медицинском обследовании, при ремонте автомобиля или в любых других случаях — почти всегда связана с неопределенностью. Попробуем записать правила для диагностики заболеваний зубов с использованием логики высказываний, что явным образом укажет на трудности, возникающие при простом логическом

подходе. Рассмотрим следующее простое правило (здесь *Toothache* — зубная боль, а *Cavity* — полость):

$$Toothache \Rightarrow Cavity.$$

Проблема состоит в том, что это правило неверно. Не у всех пациентов с зубной болью обязательно имеется полость, — у некоторых причиной боли может быть заболевание десен (*GumProblem*), нарыв (*Abscess*) или одна из нескольких иных сложных ситуаций.

$$Toothache \Rightarrow Cavity \vee GumProblem \vee Abscess\ldots$$

К сожалению, чтобы сделать это правило истинным, потребуется ввести в него почти бесконечный список возможных причин. Это правило можно попытаться преобразовать в причинное правило:

$$Cavity \Rightarrow Toothache.$$

Но и это правило нельзя назвать верным; не все зубы, имеющие полость, обязательно вызывают болевые ощущения. Единственный способ исправить данное правило состоит в том, чтобы сделать его логически исчерпывающим: дополнить левую сторону описаниями всех обстоятельств, которые должны иметь место для того, чтобы полость действительно вызывала зубную боль. Следовательно, попытка использовать логику для решения задач в проблемной области, подобной медицинской диагностике, оканчивается неудачей по следующим трем основным причинам.

- ▶**Экономия усилий**. Для формирования полного множества антецедентов или консеквентов, необходимого для составления правила, не имеющего исключений, потребуется слишком много работы, а применение таких правил будет слишком сложным.
- ▶**Неполнота теоретических знаний**. Медицинская наука не имеет полной теории для данной проблемной области.
- ▶**Неполнота практических знаний**. Даже если известны все теоретические правила, может иметь место неопределенность в отношении диагноза для конкретного пациента, поскольку не все необходимые обследования были или вообще могут быть выполнены.

Связь между зубной болью и наличием полости не является простым логическим следствием, действующим в обоих направлениях. Такая ситуация типична не только для медицинской диагностики, но и для большинства других проблемных областей, связанных с вынесением суждений: юриспруденции, бизнеса и экономики, проектирования, ремонта автомобилей, садоводства, датирования объектов или событий и т.д. Знания агента в лучшем случае позволяют сформулировать релевантные суждения только с определенной ▶**степенью уверенности** (*degree of belief*). Нашим основным инструментальным средством для работы со степенями уверенности будет ▶**теория вероятностей**. В соответствии с терминологией,

представленной в разделе 8.1, **онтологический вклад** логики и теории вероятностей одинаков — мир состоит из фактов, которые имеют либо не имеют места в каждом конкретном случае, но их **эпистемологический вклад** будет разным: логический агент уверен, что каждое высказывание должно быть истинным или ложным, либо у него нет никакого мнения, в то время как вероятностный агент может иметь числовую оценку степени уверенности в диапазоне от 0 (для высказываний, которые точно ложны) до 1 (для высказываний, которые безусловно верны).

➨ *Теория вероятности предоставляет способ суммарного учета неопределенности, возникающей по причинам экономии усилий и неполноты знаний*, тем самым решая проблему квалификации. Можно не знать со всей уверенностью, что именно вызывает зубную боль у определенного пациента, но можно с уверенностью полагать, что, скажем, в 80% случаев — т.е. с вероятностью 0,8, — если пациент испытывает зубную боль, то ее источником является полость в зубе. Это означает, что из всех ситуаций, неотличимых от текущей в пределах тех знаний, которыми обладает агент, в 80% этих случаев у пациента должна быть полость в зубе. Подобная уверенность может быть основана на статистических данных — у 80% пациентов с зубной болью, наблюдавшихся до сих пор, была обнаружена зубная полость, — на некоторых общих знаниях из области стоматологии или на комбинации различных источников.

Один вносящий путаницу момент состоит в том, что при постановке диагноза в реальном мире нет никакой неопределенности: в зубе пациента либо есть полость, либо нет. Так что же означает наше утверждение, что вероятность наличия полости равна 0,8? Разве она не должна быть равна 0 или 1? Ответ состоит в том, что вероятностные высказывания делаются в отношении *состояния знаний* агента, а не в отношении *реального мира*. Мы говорим: "Вероятность того, что пациент имеет зубную полость, *принимая во внимание то, что он испытывает зубную боль*, равна 0,8". Если позднее выяснится, что пациент уже некоторое время страдает заболеванием десен, можно будет прийти к другому заключению: "Вероятность того, что пациент имеет зубную полость, принимая во внимание, что он испытывает зубную боль и страдает заболеванием десен, составляет 0,4". Если будут собраны дополнительные убедительные доказательства против наличия зубной полости, появится возможность утверждать: "Вероятность того, что пациент имеет зубную полость, с учетом всего того, что нам теперь известно, почти нулевая". Обратите внимание, что все приведенные выше заключения не противоречат друг другу, — в каждом есть собственное утверждение о различном состоянии знаний агента.

12.1.2. Неопределенность и рациональные решения

Еще раз вернемся к плану поездки в аэропорт A_{90}. Предположим, что он обеспечивает 97%-ный шанс успешного вылета назначенным рейсом. Означает ли это, что выбор данного плана будет рациональным решением? Вовсе необязательно: могут существовать другие планы — например, A_{180} — с большей вероятностью

успешного вылета. Если *жизненно* важно не пропустить именно этот рейс, то стоит рискнуть подождать в аэропорту подольше. А что можно сказать о плане A_{1440}, предусматривающем заблаговременный выезд из дома за 24 часа до отправления самолета? В большинстве ситуаций это будет не лучший выбор, поскольку, хотя он практически гарантирует прибытие в аэропорт вовремя, он предполагает и невыносимо долгое ожидание, не говоря уже о возможности малоприятной диеты из предлагаемого в аэропорту меню.

Чтобы сделать подобный выбор, агент прежде всего должен иметь сведения о ►**предпочтениях** среди различных возможных ►**результатов** различных планов. Любой результат — это полностью определенное состояние, включая такие факторы, как своевременность прибытия и длительность ожидания в аэропорту. Для представления предпочтений и количественных рассуждений о них мы будем использовать ►**теорию полезности** (*utility theory*). (Термин "полезность" в данном контексте обозначает "свойство быть полезным".) Теория полезности утверждает, что для агента каждое состояние (или последовательность состояний) имеет определенную степень полезности (или просто *полезность*) и что агент всегда должен отдавать предпочтение состояниям с более высокой полезностью.

Для агента полезность состояния является величиной относительной. Например, полезность состояния, в котором белые могут поставить мат черным при игре в шахматы, очевидно высока для агента, играющего белыми, и очень низка для агента, играющего черными. Но мы не можем строго следовать оценкам в 1, 1/2 и 0 баллов, которые диктуются правилами проведения шахматных турниров, — одни игроки (включая авторов книги) могут быть в восторге от ничьей с чемпионом мира, тогда как другие игроки (включая прежнего чемпиона мира), едва ли будут ей особенно рады. В любом случае личные вкусы или предпочтения не должны учитываться: можно полагать, что агент, отдающий предпочтение мороженому с вкраплениями жевательной резинки "Халапеньо" вместо изюма или шоколадных чипсов, — очень странный, но нельзя утверждать, что он очевидно нерационален. Функция полезности может учитывать любое множество предпочтений — необычных или типичных, благородных или порочных. Можно даже учитывать полезность альтруистического поведения, просто включив оценку благополучия других как один из факторов.

Предпочтения, выраженные в виде полезности, комбинируются с вероятностями в общей теории рациональных решений, называемой ►**теорией принятия решений** (*decision theory*):

Теория принятия решений = теория вероятностей + теория полезности.

Фундаментальная идея теории принятия решений состоит в том, что ►*любой агент является рациональным тогда и только тогда, когда он выбирает действие, позволяющее достичь наибольшей ожидаемой полезности, усредненной по всем возможным результатам данного действия*. Это — принцип ►**максимальной ожидаемой полезности** (*Maximum Expected Utility* — **MEU**). Здесь "ожидаемой" означает

"средней"; точнее, это "статистическое среднее" значений полезностей, взвешенных по вероятности их получения. Мы наблюдали этот принцип в действии в главе 5, когда обсуждали выбор оптимальных решений при игре в нарды. В действительности это совершенно общий принцип принятия решений для агентов, действующих в одиночку.

На рис. 12.1 приведен набросок структуры агента, использующего теорию принятия решений для выбора действия. На некотором абстрактном уровне этот агент идентичен агентам, описанным в главах 4 и 7, которые поддерживают доверительное состояние, отражающее историю восприятий на текущий момент. Основное различие заключается в том, что доверительное состояние агента, действующего в соответствии с теорией принятия решений, представляет не только *возможности* для состояний мира, но и их *вероятности*. Основываясь на доверительном состоянии и некоторых знаниях о результатах действий, агент может сделать вероятностные предсказания о результатах выполнения действия и, следовательно, выбрать действие с наибольшей ожидаемой полезностью.

---

function DT-AGENT(*percept*) **returns** действие *action*
  persistent: *belief_state*, доверительное состояние — вероятностные убеждения в отношении текущего состояния мира
    action, действие агента

  обновить *belief_state* с учетом действия *action* и восприятия *percept*
  вычислить результирующие вероятности для действий actions
    на основании описаний действий *action* и текущего доверительного состояния *belief_state*
  выбрать действие *action* с наивысшей ожидаемой полезностью,
    исходя из вероятностей результатов и информации о полезности
  return *action*

Рис. 12.1. Агент, действующий на основании теории принятия решений и выбирающий рациональные действия

В этой и следующей главах изложение в основном сосредоточено на задаче представления данных и вычислений с учетом вероятностной информации в целом. Глава 14 посвящена методам решения конкретных задач представления и обновления доверительного состояния во времени и прогнозированию результатов. В главе 15 рассматриваются способы комбинирования теории вероятностей с выразительными формальными языками, такими как логика первого порядка и языки программирования общего назначения. В главе 16 теория полезности рассматривается более подробно, а в главе 17 разрабатываются алгоритмы планирования последовательностей действий в стохастических средах. В главе 18 все эти идеи распространяются на многоагентные проблемные среды.

12.2. Вероятность — основная система обозначений

Чтобы агент мог представлять и использовать вероятностную информацию, необходим формальный язык представления неопределенных знаний. Язык теории вероятностей традиционно является неформальным, разработанным человеком-математиком для других математиков. Стандартное введение в элементарную теорию вероятностей вы найдете в приложении А. В этом разделе выбран иной подход, более удобный для потребностей ИИ, в котором теория вероятностей соединяется с понятиями формальной логики.

12.2.1. О каких вероятностях идет речь

Подобно логическим утверждениям, вероятностные утверждения относятся к возможным мирам. В то время как логические утверждения говорят, какие из возможных миров являются строго недопустимыми (все те, в которых утверждение является ложным), вероятностные утверждения говорят о том, насколько вероятными являются различные миры. В теории вероятностей множество всех возможных миров называют ▶**пространством элементарных событий**. Возможные миры являются *взаимоисключающими* и *исчерпывающими* — два возможных мира не могут иметь место одновременно, иметь место в реальности допустимо лишь для одного возможного мира. Например, если мы собираемся бросить две (отличимые друг от друга) кости, существует 36 возможных миров, которые следует рассмотреть: (1,1), (1,2), …, (6,6). Для обозначения пространства элементарных событий используется греческая буква Ω (прописная буква "омега"), а буква ω (строчная буква "омега") используется для ссылок на элементы этого пространства, т.е. на конкретные возможные миры, которые в данном контексте также называют ▶**элементарными событиями**.

Полностью определенная ▶**вероятностная модель** связывает числовую вероятность $P(\omega)$ с каждым возможным миром.[1] Основные аксиомы теории вероятностей говорят о том, что каждый возможный мир характеризуется вероятностью в пределах от 0 до 1 и что суммарная вероятность всего множества возможных миров равна 1.

$$0 \le P(\omega) \le 1 \quad \text{для каждого } \omega \text{ и } \sum_{\omega \in \Omega} P(\omega) = 1 \tag{12.1}$$

Например, если предположить, что каждая кость выполнена без изъянов и при броске они не мешают друг другу, то каждый из возможных миров (1,1), (1,2), …, (6,6) характеризуется вероятностью 1/36. Если некоторые грани костей будут

[1] На данный момент мы предполагаем множество возможных миров дискретным и счетным. Корректная обработка непрерывных множеств требует учета определенных сложных моментов, которые менее актуальны для большинства целей в ИИ.

дополнительно утяжелены, то одни миры будут иметь более высокую вероятность, а другие — более низкую, но общая их сумма все равно составит 1.

Вероятностные утверждения и запросы обычно касаются не конкретных возможных миров, а некоторых их множеств. Например, нас может интересовать вероятность того, что при броске двух костей сумма будет равна 11, или вероятность того, что выпадут два одинаковых значения, и т.д. В теории вероятностей эти множества называются ▶**событиями** или **исходами** — этот термин уже широко использовался в главе 10 для иной концепции. В логике множество миров соответствует **высказыванию** на формальном языке, в частности для каждого высказывания соответствующее множество содержит только те возможные миры, в которых это высказывание истинно. (Таким образом, в этом контексте "событие" и "высказывание" означают примерно одно и то же, за исключением того, что высказывание выражается средствами формального языка.) Вероятность, связанная с высказыванием, определяется как сумма вероятностей тех возможных миров, в которых оно истинно.

$$\text{Для любого высказывания } \varphi,\ P(\varphi) = \sum_{\omega \in \Omega} P(\omega)\,. \qquad (12.2)$$

Например, при броске симметричных костей вероятность выпадения суммы 11 можно определить как $P(Total = 11) = P((5,6)) + P((6,5)) = 1/36 + 1/36 = 1/18$. Следует отметить, что теория вероятности не требует полного знания вероятностей для каждого возможного мира. Например, если мы полагаем, что кости были переделаны так, чтобы при броске на них чаще выпадали одинаковые значения, то можем *утверждать*, что $P(doubles) = 1/4$, даже не принимая во внимание то, что при броске чаще будут выпадать, скажем, две шестерки, а не две двойки. Так же, как и в случае логических утверждений, это утверждение *ограничивает* лежащую в основе вероятностную модель без полного ее определения.

Вероятности, такие как $P(Total = 11)$ и $P(doubles)$, принято называть ▶**безусловными** или ▶**априорными вероятностями**, — они касаются степени доверия к высказываниям *при отсутствии какой-либо другой информации*. Чаще всего, однако, у нас есть *некоторая* информация, обычно называемая ▶**свидетельством**, которая уже была получена ранее. Например, при броске двух костей первая из них уже может остановиться со значением 5 и мы, затаив дыхание, ждем, когда остановится вторая. В этом случае нас интересует не априорная вероятность результата броска костей, а ▶**условная** или ▶**апостериорная вероятность** выпадения дубля, *когда на первой кости* (Die_1) *уже выпала пятерка*. Эта вероятность записывается как $P(doubles|\mathrm{Die}_1 = 5)$, где символ "|" читается как "при условии".[2]

Аналогичным образом, если отправиться к стоматологу на регулярное плановое обследование, то априорная вероятность $P(cavity) = 0{,}2$ может представлять

[2] Оператор | имеет наименьший возможный приоритет, поэтому выражение $P(...|\ldots)$ всегда означает $P((...)|(\ldots))$.

интерес, но если отправиться стоматологу потому, что болит зуб, то важнее будет апостериорная вероятность $P(cavity \mid toothache) = 0{,}6$.

Важно понимать, что вероятность $P(cavity) = 0{,}2$ по-прежнему остается *имеющей силу* и после появления *зубной боли*, просто она уже не является особенно полезной. Принимая решения, агенту нужно обусловить *все* те свидетельства, которые он наблюдал. Также важно понимать различие, существующее между обусловливанием и логическим следствием. Утверждение, что $P(cavity \mid toothache) = 0{,}6$, не означает "Всякий раз, когда имеет место *зубная боль*, можно сделать заключение, что наличие в зубе *полости* имеет место с вероятностью 0,6". В действительности оно означает "Всякий раз, когда имеет место *зубная боль* и *у нас нет никакой дополнительной информации*, можно сделать заключение, что наличие в зубе *полости* имеет место с вероятностью 0,6". Это дополнительное обусловливание имеет большое значение. Например, получив дополнительную информацию о том, что стоматолог так и не обнаружил в зубах пациента никаких полостей, мы, определенно, не захотим прийти к заключению, что наличие в зубе полости имеет вероятность 0,6; вместо этого нам нужно будет использовать заключение $P(cavity \mid toothache \wedge \neg cavity) = 0$.

Говоря языком математики, апостериорные вероятности определяются в терминах априорных вероятностей следующим образом. Для любых высказываний a и b мы имеем

$$P(a \mid b) = \frac{P(a \wedge b)}{P(b)}, \tag{12.3}$$

что выполняется всякий раз, когда $P(b) > 0$. Например,

$$P(doubles \mid Die_1 = 5) = \frac{P(doubles \wedge Die_1{=}5)}{P(Die_1{=}5)}.$$

Это определение обретает смысл, если вспомнить, что наблюдение правил b исключает все те возможные миры, в которых b является ложным, оставляя множество, суммарная вероятность которого — просто $P(b)$. В пределах этого множества миры, в которых a является истинным, должны удовлетворять условию $a \wedge b$ и представляют собой дробь $P(a \wedge b)/P(b)$.

Определение апостериорной (или условной) вероятности — уравнение (12.3) — может быть записано в другой форме, называемой ►**правилом умножения вероятностей**:

$$P(a \wedge b) = P(a \mid b)P(b). \tag{12.4}$$

Правило умножения вероятностей, возможно, легче запомнить: оно построено на основании того факта, что для того, чтобы $a \wedge b$ было истинно, необходимо, чтобы b было истинно, а также чтобы истинно было a при данном b.

12.2.2. Язык высказываний в вероятностных утверждениях

В этой и последующих главах высказывания, описывающие множества возможных миров, как правило, будут записываться с использованием нотации, в которой сочетаются элементы логики высказываний, и нотации удовлетворения ограничений. Согласно терминологии, предложенной в разделе 2.4.7, это **развернутое представление**, в котором возможный мир представлен в виде множества пар *переменная/значение*. Также возможно использование еще более выразительного **структурного представления**, как показано в главе 15.

В теории вероятностей переменные называются ▶**случайными переменными** и их имена всегда начинаются с прописной буквы. Таким образом, в примере с бросанием костей переменные *Total* и Die_1 являются случайными. Каждая случайная переменная является функцией, отображающей проблемную область возможных миров Ω в некоторую ▶**область определения значений** (*range*) — множество возможных значений, которые она может принимать. Областью определения переменной *Total* в случае двух костей является множество {2,..., 12}, а областью определения переменной Die_1 — {1,..., 6}. Имена значений всегда записываются строчными буквами, поэтому сумму всех значений переменной X можно записать как $\Sigma_x P(X = x)$. Булева случайная переменная имеет область определения {*true*, *false*}. Например, высказывание о том, что были выброшены дубли, можно записать как $Doubles = true$. (Альтернативным вариантом области определения для булевых переменных является множество {0, 1}, и в этом случае говорят, что переменная имеет распределение ▶**Бернулли**.) По соглашению высказывания вида $A = true$ сокращаются до просто a, тогда как высказывания вида $A = false$ сокращаются до $\neg a$. (Использованные в предыдущем разделе имена *doubles*, *cavity* и *toothache* являются сокращением именно этого типа.)

Области определения значений переменных могут представлять собой множество произвольных признаков. Например, для переменной *Age* (*возраст*) можно выбрать область определения в виде множества {*juvenile*, *teen*, *adult*} (т.е. *ребенок*, *подросток*, *взрослый*), а для переменной *Weather* (*погода*) областью определения могут быть значения {*sun*, *rain*, *cloud*, *snow*} (т.е. *солнечно*, *дождь*, *облачно*, *снег*). Если неоднозначное понимание исключено, то обычно принято использовать само значение в тех высказываниях, где определенная переменная имеет это значение; так, значение *sun* можно непосредственно использовать в высказывании $Weather = sun$.[3]

Все предыдущие примеры имеют конечные области определения значений. Переменные также могут иметь бесконечные области определения — либо

[3] Эти соглашения, взятые вместе, приводят к потенциальной неоднозначности в обозначениях при суммировании значений булевых переменных. Например, $P(a)$ — вероятность того, что переменная A имеет значение *true*, тогда как в выражении $\Sigma_a P(a)$ это просто ссылка на вероятность одного из значений a переменной A.

дискретные (например, целые числа), либо непрерывные (например, действительные числа). Для любой переменной с упорядоченной областью определения также допускаются неравенства, такие как $NumberOfAtomsInUniverse \geq 10^{70}$.

Наконец, можно объединить все эти виды элементарных высказываний (включая сокращенные формы для булевых переменных), используя стандартные логические связки логики высказываний. Например, высказывание "Вероятность того, что в зубах пациентки есть полость, с учетом того, что она является подростком и не испытывает зубной боли, составляет 0,1" можно записать следующим образом.

$$P(cavity \mid \neg toothache \wedge teen) = 0{,}1$$

Также в вероятностной нотации для обозначения операции конъюнкции часто используют запятую, поэтому в приведенном выше высказывании левую часть можно было бы записать просто как $P(cavity \mid \neg toothache, teen)$.
Иногда в обсуждение требуется включить вероятности *всех* возможных значений случайной величины. Понятно, что в этом случае можно было бы использовать такую запись:

$$\begin{aligned} P(Weather = sun) &= 0{,}6, \\ P(Weather = rain) &= 0{,}1, \\ P(Weather = cloud) &= 0{,}29, \\ P(Weather = snow) &= 0{,}01, \end{aligned}$$

но для сокращения можно применить следующий вариант записи:

$$\mathbf{P}(Weather) = (0{,}6;\ 0{,}1;\ 0{,}29;\ 0{,}01).$$

Здесь выделение **P** полужирным шрифтом указывает, что результатом является вектор чисел, расположенных в некотором предопределенном порядке $\langle sun, rain, cloud, snow \rangle$ в соответствии с областью определения переменной *Weather*. Говорят, что высказывание **P** задает ▶**распределение вероятностей** для случайной переменной *Weather*, т.е. присвоение вероятности для каждого возможного значения этой случайной переменной. (В подобном случае при конечной дискретной области определения значений такое распределение называется ▶**категориальным распределением**.) Нотация **P** также используется для условных распределений: $\mathbf{P}(X \mid Y)$, присваивая значения $P(X = x_i \mid Y = y_j)$ для каждой возможной пары i, j.

Для непрерывных переменных просто невозможно записать все распределение в виде вектора, поскольку в нем существует бесконечно много значений. Вместо этого можно определить вероятность того, что случайная величина принимает некоторое значение x как параметризованную функцию от x, обычно называемую ▶**функцией плотности распределения вероятностей**. Например, высказывание

$$P(NoonTemp = x) = Uniform(x;\ 18C,\ 26C)$$

выражает уверенность в том, что значение температуры в полдень (переменная *NoonTemp*) будет равномерно распределено между значениями 18 и 26 градусов по Цельсию.

Функция плотности распределения вероятностей (также часто называемая просто **функцией распределения вероятностей**) по смыслу отличается от дискретных распределений. Утверждение, что плотность вероятности равномерно распределена от 18°C до 26°C, означает, что существует 100%-ная вероятность того, что значение температуры в полдень попадет в этот диапазон шириной в 8°C, и 50%-ная вероятность того, что оно попадет в любой поддиапазон шириной 4°C этого диапазона, и т.д. Принято записывать плотность вероятности для непрерывной случайной величины X в области значения x как $P(X=x)$ или просто как $P(x)$. Интуитивно понятное определение $P(x)$ — это вероятность того, что значение X попадает в произвольно малую область, начинающуюся от x, деленную на ширину этой области:

$$P(x) = \lim_{dx \to 0} P(x \le X \le x + dx)/dx.$$

Для переменной *NoonTemp* имеем

$$P(NoonTemp = x) = Uniform(x; 18C, 26C) = \begin{cases} \frac{1}{8C} & \text{если } 18C \le x \le 26C \\ 0 & \text{в противном случае.} \end{cases}$$

Здесь C обозначает шкалу температуры в градусах Цельсия (а не является константой). В выражении $P(NoonTemp = 20{,}18C) = \frac{1}{8C}$ обратите внимание, что $\frac{1}{8C}$ — это не вероятность, а *плотность вероятности*. Вероятность того, что переменная *NoonTemp* имеет значение *точно* $20{,}18°C$, равна нулю, потому что диапазон 20,18°C имеет нулевую ширину. Некоторые авторы используют разные символы для дискретных вероятностей и плотностей вероятностей; но мы в этой книге будем использовать обозначение P для конкретных значений вероятности и **P** — для векторов значений в обоих случаях, поскольку в действительности путаница возникает очень редко и уравнения чаще всего идентичны. Обратите внимание, что вероятности — это безразмерные числа, тогда как значения функций распределения вероятностей выражаются в некоторых единицах измерения. В нашем примере это единица, обратная градусу Цельсия. Если тот же самый интервал температур потребуется выразить в градусах по Фаренгейту, он будет иметь ширину 14,4 градуса, а плотность вероятности будет иметь значение $1/14{,}4F$.

В дополнение к распределениям по отдельным переменным нам потребуются обозначения и для распределений по нескольким переменным. Для этой цели будем использовать запятые. Например, выражение **P**(*Weather*, *Cavity*) определяет вероятности всех комбинаций значений переменных *Weather* и *Cavity* и представляет собой таблицу вероятностей размером 4×2, называемую ▶**совместным рас-**

пределением вероятностей для переменных *Weather* и *Cavity*. Также можно смешивать в выражениях переменные и конкретные значения, например **P**(*sun*, *Cavity*) является двухэлементным вектором, включающим вероятности наличия в зубе полости в солнечный день и отсутствия полости в зубе в солнечный день.

Использование обозначения **P** делает определение выражений гораздо более кратким, чем они могли бы быть в противном случае. Например, правило умножения вероятностей (см. уравнение (12.4)) для всех возможных значений переменных *Weather* и *Cavity* можно записать в виде единственного уравнения:

$$\mathbf{P}(Weather, Cavity) = \mathbf{P}(Weather \mid Cavity)\mathbf{P}(Cavity)$$

вместо следующих $4 \times 2 = 8$ уравнений (с использованием сокращений W и C):

$$P(W = sun \wedge C = true) = P(W = sun \mid C = true)\ P(C = true)$$
$$P(W = rain \wedge C = true) = P(W = rain \mid C = true)\ P(C = true)$$
$$P(W = cloud \wedge C = true) = P(W = cloud \mid C = true)\ P(C = true)$$
$$P(W = snow \wedge C = true) = P(W = snow \mid C = true)\ P(C = true)$$
$$P(W = sun \wedge C = false) = P(W = sun \mid C = false)\ P(C = false)$$
$$P(W = rain \wedge C = false) = P(W = rain \mid C = false)\ P(C = false)$$
$$P(W = cloud \wedge C = false) = P(W = cloud \mid C = false)\ P(C = false)$$
$$P(W = snow \wedge C = false) = P(W = snow \mid C = false)\ P(C = false).$$

Как вырожденный случай выражение **P**(*sun*, *cavity*) не содержит переменных и, следовательно, является вектором нулевой размерности, который можно рассматривать как скалярное значение.

На данный момент мы уже определили синтаксис высказываний и вероятностных утверждений, а также дали часть семантики: уравнение (12.2) определяет вероятность высказывания как сумму вероятностей миров, в которых оно выполняется. Для завершения семантики необходимо сказать, чем эти миры являются и как определить, выполняется ли высказывание в некотором мире. Мы заимствуем эту часть непосредственно из семантики логики высказываний следующим образом. ➔ *Возможный мир определяется как присваивание значений всем рассматриваемым случайным переменным.*

Легко показать, что это определение удовлетворяет основному требованию, согласно которому возможные миры должны быть взаимно исключающими и исчерпывающими (см. упражнение 12.5). Например, если случайными переменными являются *Cavity*, *Toothache* и *Weather*, то существует $2 \times 2 \times 4 = 16$ возможных миров. Более того, истинность любого заданного высказывания легко может быть определена в подобных мирах посредством тех же самых рекурсивных вычислений истинности, которые использовались нами в логике высказываний (см. раздел 7.4.2).

Обратите внимание, что некоторые случайные переменные могут быть избыточными в том смысле, что их значения во всех случаях могут быть получены из значений других переменных. Например, в мире двух игральных

костей переменная *Doubles* будет иметь значение *true* только в тех случаях, когда $Die_1 = Die_2$. Включение переменной *Doubles* в качестве одной из случайных переменных в дополнение к переменным Die_1 и Die_2, как кажется, увеличивает количество возможных миров с 36 до 72, но, конечно же, ровно половина из этих 72 миров будет логически невозможной и, следовательно, иметь вероятность 0.

Из приведенного выше определения возможных миров следует, что вероятностная модель полностью определяется совместным распределением вероятностей для всех случайных переменных — так называемым ▶**полным совместным распределением вероятностей**. Например, при наличии случайных переменных *Cavity*, *Toothache* и *Weather* полным совместным распределением вероятностей будет **P**(*Cavity*, *Toothache*, *Weather*). Это совместное распределение может быть представлено в виде таблицы размерностью $2 \times 2 \times 4$, содержащей 16 значений. Поскольку вероятность каждого высказывания является суммой по всем возможным мирам, полного совместного распределения в принципе достаточно для вычисления вероятности любого высказывания. Примеры того, как это можно сделать, будут приведены в разделе 12.3.

12.2.3. Аксиомы вероятности и их обоснованность

Основные аксиомы вероятности (уравнения (12,1) и (12,2)) подразумевают определенные отношения между степенями доверия, которые могут быть отнесены к логически связанным высказываниям. Так, можно вывести знакомые отношения между вероятностью высказывания и вероятностью его отрицания:

$$
\begin{aligned}
P(\neg a) &= \textstyle\sum_{\omega \in \neg a} P(\omega) = && \text{по уравнению (12.2)} \\
&= \textstyle\sum_{\omega \in \neg a} P(\omega) + \sum_{\omega \in a} P(\omega) - \sum_{\omega \in a} P(\omega) = && \\
&= \textstyle\sum_{\omega \in \Omega} P(\omega) - \sum_{\omega \in a} P(\omega) = && \text{группируя первые 2 члена} \\
&= 1 - P(a) && \text{по (12.1) и (12.2).}
\end{aligned}
$$

Также можно вывести известную формулу для вероятности дизъюнкции, которую иногда называют ▶**формулой** (или **принципом**) **включений-исключений:**

$$P(a \vee b) = P(a) + P(b) - P(a \wedge b). \qquad (12.5)$$

Это правило можно легко запомнить, отметив, что те случаи, когда высказывание a является истинным, вместе с теми случаями, когда высказывание b является истинным, безусловно, охватывают все те случаи, когда истинно высказывание $a \vee b$; но в сумме двух множеств случаи их пересечения будут учтены дважды, поэтому необходимо вычесть $P(a \wedge b)$.

Уравнения (12.1) и (12.5) часто называют ▶**аксиомами Колмогорова** в честь математика Андрея Колмогорова, показавшего, как построить остальную часть теории вероятностей на этом простом фундаменте и как справиться с трудностями,

вызванными непрерывными переменными.[4] Хотя уравнение (12.2) имеет определенную особенность, уравнение (12.5) показывает, что аксиомы действительно ограничивают степень уверенности, которую агент может иметь в отношении логически связанных высказываний. Это аналогично тому факту, что логический агент не может одновременно быть уверен в высказываниях A, B и $\neg(A \wedge B)$, так как не существует возможного мира, в котором они все три одновременно являются истинными. Однако при использовании вероятностей высказывания относятся не к миру непосредственно, а к собственному состоянию знания агента. Почему же тогда агент не может придерживаться следующего множества убеждений (даже если они нарушают аксиомы Колмогорова)?

$$P(a) = 0{,}4 \quad P(b) = 0{,}3 \quad P(a \wedge b) = 0{,}0 \quad P(a \vee b) = 0{,}8 \tag{12.6}$$

Такого рода вопрос был предметом яростных дебатов, продолжавшихся в течение десятилетий между теми, кто отстаивал допустимость использования вероятностей как единственной обоснованной формы оценки степеней уверенности, и теми, кто отстаивал альтернативные подходы.

Один из аргументов в пользу аксиом вероятностей, впервые сформулированный в 1931 году Бруно де Финетти ([554], 1983), заключается в следующем. Если агент имеет некоторую степень уверенности в истинности высказывания a, то он должен быть способен сформулировать оценку того, в какой степени он безразличен к ставке за или против высказывания a.[5] Можно рассматривать подобную ситуацию как игру между двумя агентами: агент 1 утверждает: "Моя степень уверенности в истинности события a равна 0,4". Затем агент 2 вправе выбрать, будет ли он делать ставку за или против высказывания a, выбирая ставки, совместимые с заявленной степенью уверенности. То есть агент 2 может решить сделать ставку на то, что событие a произойдет, поставив 6 долл. против 4 долл. агента 1. Или же агент 2 может сделать ставку на то, что будет иметь место событие $\neg a$, поставив 4 долл. против 6 агента 1. Когда исход события a станет известен, тот, кто оказался прав, заберет деньги. Если степень уверенности агента недостаточно точно отражает состояние мира, можно рассчитывать на то, что в долговременной перспективе он будет проигрывать деньги агенту-противнику, убеждения которого более точно отражают его состояние.

Теорема де Финетти относится не к выбору правильных значений для отдельных вероятностей, а к выбору значений вероятностей логически связанных

---

[4] Эти трудности включают множество Витали, четко определенного измеримого подмножества в интервале [0, 1] с неопределенной неизмеримой длиной.

[5] Можно возразить, что предпочтения агента применительно к балансам разных ставок являются таковыми, что возможность потерять 1 долл. не уравновешивается равной возможностью выиграть 1 долл. Один из возможных ответов на подобное возражение состоит в том, что суммы ставок должны быть достаточно малыми для того, чтобы можно было избежать данной проблемы. Анализ, проведенный Сэведжем ([1984], 1954), позволяет полностью исключить из рассмотрения эту проблему.

высказываний. ➡ *Если агент 1 руководствуется множеством степеней уверенности, нарушающим аксиомы теории вероятностей, то всегда существует комбинация ставок агента 2, гарантирующая, что агент 1 будет терять деньги при каждой ставке.* Например, предположим, что агент 1 руководствуется множеством степеней уверенности, приведенным в уравнении (12.6). На рис. 12.2 показано, что если агент 2 решит ставить 4 долл. на a, 3 долл. — на b и 2 долл. — на $\neg(a \vee b)$, то агент 1 всегда будет терять деньги, независимо от исходов для a и b. Из теоремы де Финетти следует, что ни один рациональный агент не может иметь убеждений, нарушающих аксиомы вероятности.

Высказывание	Степень уверенности агента 1	Ставка агента 2	Ставка агента 1	Результаты для агента 1			
				a, b	$a, \neg b$	$\neg a, b$	$\neg a, \neg b$
a	0,4	4 на a	6 на $\neg a$	-6	-6	4	4
b	0,3	3 на b	7 на $\neg b$	-7	3	-7	3
$a \vee b$	0,8	2 на $\neg(a \vee b)$	8 на $a \vee b$	2	2	2	-8
				-11	-1	-1	-1

Рис. 12.2. Поскольку агент 1 придерживается несогласованных убеждений, агент 2 может подобрать множество из трех ставок, гарантирующих постоянный проигрыш для агента 1, независимо от исходов для a и b

Одно общее возражение в отношении теоремы де Финетти состоит в том, что эта игра со ставками является довольно надуманной. Например, что будет, если один из игроков откажется делать ставку? Закончится ли на этом спор? Ответ на данный вопрос состоит в том, что эта игра со ставками представляет собой абстрактную модель для ситуации принятия решений, в которую любой агент *неизбежно* вовлечен в любой момент. Каждое действие (включая бездействие) — это своего рода ставка, а каждый исход может рассматриваться как положительное или отрицательное вознаграждение за эту ставку. Отказ делать ставку подобен отказу позволить времени двигаться.

В пользу применения вероятностей были выдвинуты и другие весомые философские аргументы, из которых наиболее заметными можно считать работы Кокса ([489], 1946), Карнапа ([374], 1950) и Джейнса (2003). В каждой из них предлагается множество аксиом для рассуждений со степенями доверия: отсутствие противоречий, соответствие положениям обычной логики (например, если степень доверия к A возрастает, то степень доверия к $\neg A$ должна уменьшаться) и т.д. Единственная спорная аксиома состоит в том, что степени доверия должны быть представлены числами или по крайней мере вести себя, как числа, например обладать свойством транзитивности (если степень доверия к A больше, чем степень доверия к B, которая больше, чем степень доверия к C, то степень доверия к A должна

быть больше, чем к C) и свойством сравнимости (степень доверия к A должна быть либо равна, либо больше, либо меньше, чем степень доверия к B). Можно доказать, что применение вероятностей является единственным подходом, удовлетворяющим все эти аксиомы.

Но мир таков, каков он есть, и практические свидетельства иногда оказываются более весомыми, чем доказательства. Успех систем формирования рассуждений, основанных на теории вероятностей, оказался гораздо более эффективным аргументом в пользу пересмотра многих взглядов, чем любая философская аргументация. В следующем разделе показано, как приведенные выше аксиомы можно применить к логическому выводу.

12.3. Логический вывод с использованием полных совместных распределений

В этом разделе описывается простой метод ▶**вероятностного вывода**, т.е. вычисления апостериорных вероятностей для высказываний, сформулированных как ▶**запросы** на основании наблюдаемых свидетельств. В качестве "базы знаний", из которой можно будет вывести ответы на все запросы, мы будем использовать полное совместное распределение. По ходу дела также будет представлено несколько полезных методов манипулирования уравнениями, включающих вероятности.

Начнем с очень простого примера — проблемной области, состоящей только из трех булевых переменных, *Toothache*, *Cavity* и *Catch* (*щипцы*) (неприятные ощущения от захвата зуба стальными стоматологическими щипцами все еще свежи в памяти автора). Полное совместное распределение этих переменных представляет собой таблицу размером $2 \times 2 \times 2$, представленную на рис. 12.3.

	toothache		¬*toothache*	
	catch	¬*catch*	*catch*	¬*catch*
cavity	0,108	0,012	0,072	0,008
¬*cavity*	0,016	0,064	0,144	0,576

Рис. 12.3. Полное совместное распределение для мира *Toothache*, *Cavity*, *Catch*

Обратите внимание, что вероятности в этом совместном распределении в сумме составляют 1, что и требуется согласно аксиомам вероятности. Также обратите внимание, что уравнение (12.2) предоставляет прямой способ вычисления вероятности любого высказывания, простого или сложного: достаточно определить те возможные миры, в которых данное высказывание является истинным, а затем сложить их вероятности. Например, имеется шесть возможных миров, в которых высказывание $cavity \vee toothache$ является истинным:

$$P(cavity \vee toothache) = 0{,}108 + 0{,}012 + 0{,}072 + 0{,}008 + 0{,}016 + 0{,}064 = 0{,}28.$$

Одна из задач, которые встречаются особенно часто, состоит в том, чтобы извлечь из подобной таблицы распределение вероятностей по некоторому подмножеству переменных или по одной переменной. Например, складывая элементы первой строки на рис. 12.3, получим безусловную или ►**маргинальную вероятность**[6] события *cavity*:

$$P(cavity) = 0{,}108 + 0{,}012 + 0{,}072 + 0{,}008 = 0{,}2.$$

Этот процесс называется ►**маргинализацией** или **исключением из суммы**, поскольку суммирование вероятностей для *каждого* возможного значения других переменных исключает их из уравнения. Можно записать следующее общее правило маргинализации для любых множеств переменных **Y** и **Z**:

$$\mathbf{P}(\mathbf{Y}) = \sum_{\mathbf{z}} \mathbf{P}(\mathbf{Y}, \mathbf{Z} = \mathbf{z}), \tag{12.7}$$

где $\Sigma_{\mathbf{z}}$ — сумма по всем возможным комбинациям значений множества переменных **Z**. Как обычно, в этом уравнении мы можем сократить $\mathbf{P}(\mathbf{Y}, \mathbf{Z} = \mathbf{z})$ до $\mathbf{P}(\mathbf{Y}, \mathbf{z})$. Например, для переменной *Cavity* уравнение (12.7) соответствует следующему уравнению:

$$\begin{aligned}\mathbf{P}(Cavity) &= \mathbf{P}(Cavity, toothache, catch) + \mathbf{P}(Cavity, toothache, \neg catch) + \\ &+ \mathbf{P}(Cavity, \neg toothache, catch) + \mathbf{P}(Cavity, \neg toothache, \neg catch) = \\ &= \langle 0{,}108;\ 0{,}016\rangle + \langle 0{,}012;\ 0{,}064\rangle + \langle 0{,}072;\ 0{,}144\rangle + \langle 0{,}008;\ 0{,}576\rangle = \\ &= \langle 0{,}2;\ 0{,}8\rangle.\end{aligned}$$

Используя правило умножения вероятностей (уравнение (12.4)), можно заменить $\mathbf{P}(\mathbf{Y}, \mathbf{z})$ в уравнении (12.7) на $\mathbf{P}(\mathbf{Y}|\mathbf{z})P(\mathbf{z})$, получив правило, называемое ►**правилом обусловливания**:

$$\mathbf{P}(\mathbf{Y}) = \sum_{\mathbf{z}} \mathbf{P}(\mathbf{Y} \mid \mathbf{z})P(\mathbf{z}). \tag{12.8}$$

Как оказалось, правила маргинализации и обусловливания являются очень полезными правилами для всех видов логических выводов, включающих вероятностные выражения.

В большинстве случаев нас будет интересовать задача вычисления *условных* (апостериорных) вероятностей некоторых переменных при наличии свидетельств, касающихся других переменных. Условные вероятности можно найти, вначале воспользовавшись уравнением (12.3) для получения выражения в терминах безусловных вероятностей, а затем рассчитав это выражение на основании полного

[6] Эта вероятность получила такое название, поскольку страховщики имеют общую привычку записывать суммы наблюдаемых частот событий на полях (*margin*) таблиц страхования.

совместного распределения. Например, ниже показано, как можно вычислить вероятность наличия полости в зубе, получив свидетельство о наличии зубной боли:

$$P(cavity \mid toothache) = \frac{P(cavity \wedge toothache)}{P(toothache)} = \\ = \frac{0{,}108 + 0{,}012}{0{,}108 + 0{,}012 + 0{,}016 + 0{,}064} = 0{,}6.$$

Просто для проверки можно также рассчитать вероятность того, что у пациента нет полости в зубе, если у него наблюдается зубная боль:

$$P(\neg cavity \mid toothache) = \frac{P(\neg cavity \wedge toothache)}{P(toothache)} = \\ = \frac{0{,}016 + 0{,}064}{0{,}108 + 0{,}012 + 0{,}016 + 0{,}064} = 0{,}4.$$

Эти два значения в сумме дают 1,0, как и должно быть. Обратите внимание на присутствие терма $P(toothache)$ в знаменателе обоих этих вычислений. Если бы переменная *Cavity* имела более двух значений, этот терм присутствовал бы в знаменателе для всех них. Фактически для распределения $P(Cavity \mid toothache)$ его можно рассматривать как константу **нормализации**, гарантирующую, что полученные вероятности в сумме составят 1. Во всех главах, в которых речь будет идти о вероятностях, для обозначения таких констант мы будем использовать символ α. Применив это обозначение, можно записать два предыдущих уравнения как одно:

$$\mathbf{P}(Cavity \mid toothache) = \alpha\, \mathbf{P}(Cavity, toothache) = \\ = \alpha\, [\mathbf{P}(Cavity, toothache, catch) + \mathbf{P}(Cavity, toothache, \neg catch)] = \\ = \alpha\, [(0{,}108; 0{,}016) + (0{,}012; 0{,}064)] = \alpha\, (0{,}12; 0{,}08) = (0{,}6; 0{,}4).$$

Другими словами, мы можем вычислить $\mathbf{P}(Cavity \mid toothache)$, даже не зная значения $P(toothache)$! Забыв на время о множителе $1/P(toothache)$, мы суммируем значения для *cavity* и $\neg cavity$, получив значения 0,12 и 0,08. Это правильная относительная пропорция, но в сумме они не дают 1, поэтому мы нормализуем эти значения делением каждого из них на 0,12 + 0,08, получив в результате истинные вероятности 0,6 и 0,4. Нормализация оказывается полезным упрощением во многих вероятностных расчетах, позволяя как сделать вычисления проще, так и выполнить расчеты, когда некоторые оценки вероятности (например, $P(toothache)$) недоступны.

На основании приведенного выше примера можно описать общую процедуру вероятностного вывода. Начнем со случая, когда запрос касается только одной переменной, X (например, *Cavity*). Пусть **E** будет списком переменных свидетельства (в нашем примере это только *Toothache*), **e** будет списком наблюдаемых значений

этих переменных, а **Y** будет представлять оставшиеся ненаблюдаемые переменные (в нашем примере это только *Catch*). Запрос будет иметь вид $\mathbf{P}(X \mid \mathbf{e})$ и может быть вычислен следующим образом:

$$\mathbf{P}(X \mid \mathbf{e}) = \alpha \mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y}), \tag{12.9}$$

где суммирование осуществляется по всем возможным **y** (т.е. по всем возможным комбинациям значений ненаблюдаемых переменных **Y**). Обратите внимание, что взятые вместе переменные X, **E** и **Y** образуют полное множество переменных для данной проблемной области, поэтому $\mathbf{P}(X, \mathbf{e}, \mathbf{y})$ представляет собой просто подмножество вероятностей из полного совместного распределения.

При наличии полного совместного распределения, с которым можно работать, уравнение (12.9) позволяет получить ответы на вероятностные запросы в отношении дискретных переменных. Однако оно недостаточно хорошо масштабируется, поскольку в проблемной области с n булевыми переменными потребуется входная таблица размером $O(2^n)$, обработка которой потребует времени $O(2^n)$. В реальных задачах вполне могут присутствовать сотни случайных переменных, и тогда оценка $O(2^n)$ для потребности в памяти и времени расчетов далеко выходит за пределы возможного: $2^{100} \approx 10^{30}$! И проблема здесь не только в объемах памяти и времени расчетов — еще более серьезная проблема состоит в том, что потребуется отдельно оценить на реальных примерах каждую из 10^{30} вероятностей, что делает объем необходимых экспериментальных данных просто астрономическим.

По этим причинам полное совместное распределение в табличной форме редко воспринимается как практический инструмент для построения систем рассуждений. На самом деле его, скорее, следует рассматривать как теоретическую основу, на которой можно строить более эффективные подходы, — как таблицы истинности являются теоретической основой для более практичных алгоритмов, таких как алгоритм DPLL, рассматривавшийся в главе 7. В оставшейся части этой главы будут представлены некоторые основные идеи, необходимые для подготовки к разработке реально осуществимых систем, описанных в главе 13.

12.4. Независимость

Давайте расширим полное совместное распределение, приведенное на рис. 12.2, добавив в него четвертую переменную, *Weather*. В результате полное совместное распределение будет иметь вид $\mathbf{P}(Toothache, Catch, Cavity, Weather)$ и представлять собой таблицу из $2 \times 2 \times 2 \times 4 = 32$ элементов (переменная *Weather* имеет четыре значения). Это распределение содержит четыре "варианта" таблицы, представленной на рис. 12.3, по одному на каждый вид погоды. Возникает вопрос: какую связь эти варианты имеют друг с другом и с первоначальной таблицей, построенной при наличии трех переменных? Как связаны друг с другом значения

$P(toothache, catch, cavity, cloud)$ и значения $P(toothache, catch, cavity)$? Для получения ответа можно воспользоваться правилом умножения вероятностей (уравнение (12.4)):

$$P(toothache, catch, cavity, cloud) = \\ = P(cloud \mid toothache, catch, cavity)P(toothache, catch, cavity).$$

Но человек, не верящий в возможность вмешательства свыше, едва ли сможет представить, что чьи-то проблемы с зубами способны повлиять на погоду. Поэтому следующее утверждение кажется вполне разумным:

$$P(cloud \mid toothache, catch, cavity) = P(cloud). \qquad (12.10)$$

Из этого можно вывести следующее:

$$P(toothache, catch, cavity, cloud) = P(cloud)P(toothache, catch, cavity).$$

Аналогичное уравнение существует для *каждого элемента* в распределении $\mathbf{P}(Toothache, Catch, Cavity, Weather)$. В действительности можно даже записать такое общее уравнение:

$$\mathbf{P}(Toothache, Catch, Cavity, Weather) = \mathbf{P}(Toothache, Catch, Cavity)\mathbf{P}(Weather).$$

Следовательно, 32-элементная таблица для четырех переменных может быть образована из одной 8-элементной таблицы и одной 4-элементной. Подобная декомпозиция схематически показана на рис. 12.4, *а*.

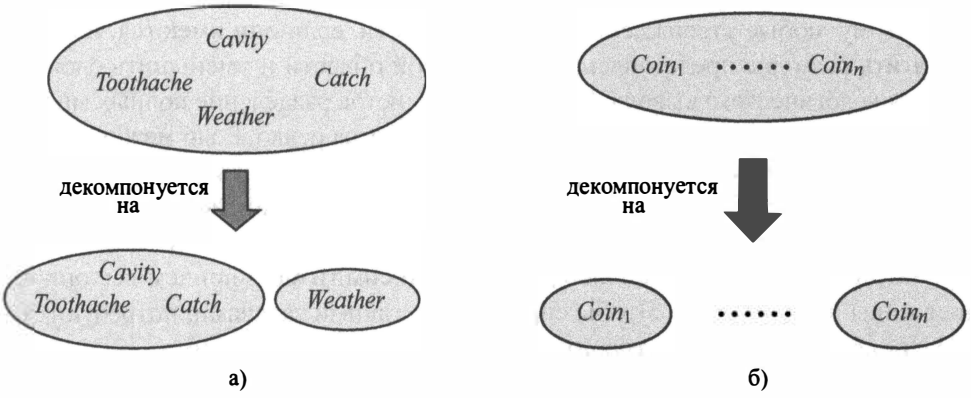


Рис. 12.4. Два примера разбиения большого совместного распределения на меньшие распределения с использованием свойства абсолютной независимости. **а)** Погода и проблемы с зубами независимы друг от друга. **б)** Броски монеты независимы друг от друга

Свойство, которое использовалось в уравнении (12.10), называется ▶**независимостью** (а также **маргинальной независимостью** или **абсолютной независимостью**). В частности, погода независима от чьих-то проблем с зубами. Независимость между высказываниями a и b может быть представлена следующим образом:

$$P(a \mid b) = P(a) \quad \text{или} \quad P(b \mid a) = P(b) \quad \text{или} \quad P(a \wedge b) = P(a)P(b). \qquad (12.11)$$

Все эти варианты записи эквивалентны (упражнение 12.15). Свойство независимости между переменными X и Y можно сформулировать следующим образом (все эти варианты записи также эквивалентны):

$$\mathbf{P}(X \mid Y) = \mathbf{P}(X) \quad \text{или} \quad \mathbf{P}(Y \mid X) = \mathbf{P}(Y) \quad \text{или} \quad \mathbf{P}(X, Y) = \mathbf{P}(X)\mathbf{P}(Y).$$

Утверждения о независимости обычно основаны на знаниях о проблемной области. Как показывает пример с зубной болью и погодой, они позволяют существенно сократить объем информации, необходимой для построения полного совместного распределения. Если все множество переменных может быть разделено на независимые подмножества, то полное совместное распределение может быть *факторизовано* на отдельные совместные распределения, заданные на этих подмножествах. Например, совместное распределение результатов n независимых бросков монеты $\mathbf{P}(C_1,..., C_n)$ включает 2^n элементов, но может быть представлено как произведение n распределений $\mathbf{P}(C_i)$ с одной переменной. С практической точки зрения независимость стоматологии и метеорологии является благоприятным фактором, поскольку в противном случае стоматологам могли бы потребоваться глубокие знания в области метеорологии, и наоборот.

Поэтому любые утверждения о независимости, если они имеются, позволяют сократить размеры представления проблемной области и уменьшить сложность проблемы логического вывода. К сожалению, чистое разделение полных множеств переменных по признаку независимости встречается редко. Если между двумя переменными существует хоть какая-то связь, пусть даже косвенная, свойство независимости уже не соблюдается. Более того, даже независимые подмножества могут оказаться достаточно большими; например, в области стоматологии могут рассматриваться десятки заболеваний и сотни симптомов, причем все они взаимосвязаны друг с другом. Чтобы справиться с такими задачами, потребуются методы, более тонкие, чем прямолинейная концепция независимости.

12.5. Правило Байеса и его использование

В разделе 12.2.1 было определено правило умножения вероятностей (уравнение (12.4)). На самом деле это правило можно записать в двух формах:

$$P(a \wedge b) = P(a \mid b)P(b) \quad \text{и} \quad P(a \wedge b) = P(b \mid a)P(a).$$

Приравняв правые части этих двух уравнений и разделив их на $P(a)$, получим

$$P(a \wedge b) = \frac{P(a \mid b)P(b)}{P(a)}. \tag{12.12}$$

Это уравнение известно как ▶**правило Байеса** (закон Байеса, теорема Байеса). Это простое уравнение лежит в основе всех современных систем искусственного интеллекта для вероятностного вывода.

Более общий случай правила Байеса для многозначных переменных можно записать в нотации **P** следующим образом:

$$\mathbf{P}(Y \mid X) = \frac{\mathbf{P}(X \mid Y)\mathbf{P}(Y)}{\mathbf{P}(X)}.$$

Как и прежде, это уравнение также следует рассматривать как представляющее множество уравнений, в каждом из которых рассматриваются конкретные значения переменных. Время от времени нам также придется использовать более общую версию, которая обусловлена некоторым фоновым свидетельством **e**:

$$\mathbf{P}(Y \mid X, \mathbf{e}) = \frac{\mathbf{P}(X \mid Y, \mathbf{e})\mathbf{P}(Y \mid \mathbf{e})}{\mathbf{P}(X \mid \mathbf{e})}. \tag{12.13}$$

12.5.1. Применение правила Байеса: простой случай

На первый взгляд, правило Байеса не кажется очень полезным. Оно позволяет вычислить единственный терм $P(b \mid a)$ на основании трех термов, $P(a \mid b)$, $P(b)$ и $P(a)$. Хотя это и похоже на шаг вперед и два шага назад, тем не менее правило Байеса находит очень широкое практическое применение, поскольку во многих случаях имеются хорошие оценки вероятностей для этих трех элементов и нужно вычислить четвертый. Часто воспринимаемые данные указывают на *результат* (*effect*), вызываемый какой-то неизвестной *причиной* (*cause*), и нам необходимо определить эту причину. В таких случаях правило Байеса принимает следующий вид:

$$P(cause \mid effect) = \frac{P(effect \mid cause)P(cause)}{P(effect)}.$$

Условная вероятность $P(effect \mid cause)$ предоставляет количественную оценку взаимосвязи в ▶**причинном** направлении, тогда как вероятность $P(cause \mid effect)$ описывает ▶**диагностическое** направление. В такой задаче, как определение медицинского диагноза, мы часто имеем условные вероятности причинно-следственных связей. Врач знает диагностические вероятности $P(symptoms \mid disease)$ и хочет поставить диагноз $P(disease \mid symptoms)$.

Например, врач знает, что менингит часто вызывает у пациента снижение подвижности шеи, — предположим, это наблюдается в 70% случаев. Врач также знает некоторые безусловные факты: априорная вероятность того, что у пациента менингит, равна 1/50 000, а априорная вероятность того, что у пациента будет сни-

жена подвижность шеи, равна 1%. Пусть s — высказывание, утверждающее, что у пациента снижена подвижность шеи, а m — высказывание, утверждающее, что у пациента менингит. Тогда можно записать следующее:

$$P(s \mid m) = 0{,}7$$
$$P(m) = 1/50\,000$$
$$P(s) = 0{,}01$$
$$P(m \mid s) = \frac{P(s \mid m)P(m)}{P(s)} = \frac{0{,}7 \times 1/50\,000}{0{,}01} = 0{,}0014. \qquad (12.14)$$

Таким образом, можно ожидать, что только у 0,14% пациентов со сниженной подвижностью шеи будет менингит. Обратите внимание, что даже если снижение подвижности шеи является весьма надежным свидетельством наличия менингита (с вероятностью 0,7), сама вероятность наличия менингита у пациента остается очень низкой. Это связано с тем, что априорная вероятность симптома снижения подвижности шеи (по любой причине) намного выше в сравнении с вероятностью менингита.

В разделе 12.3 был описан процесс, посредством которого можно избежать необходимости оценки априорной вероятности свидетельства (в данном случае — $P(s)$), вычислив вместо этого апостериорную вероятность для каждого значения переменной запроса (в данном случае — m и $\neg m$), а затем нормализовав результаты. Аналогичный процесс можно применять и при использовании правила Байеса. Итак, мы имеем

$$\mathbf{P}(M \mid s) = \alpha \langle P(s \mid m)P(m),\ P(s \mid \neg m)P(\neg m) \rangle.$$

Следовательно, чтобы воспользоваться этим подходом, необходимо вместо $P(s)$ вычислить значение $P(s \mid \neg m)$. К сожалению, бесплатных пирожных не бывает, — иногда это упрощает задачу, а иногда усложняет. Общая форма правила Байеса с нормализацией будет следующей:

$$\mathbf{P}(Y \mid X) = \alpha\, \mathbf{P}(X \mid Y)\mathbf{P}(Y), \qquad (12.15)$$

где α — константа нормализации, необходимая для того, чтобы сумма элементов в распределении $\mathbf{P}(Y \mid X)$ была равна 1.

Один из очевидных вопросов, касающихся правила Байеса, состоит в том, почему доступной может оказаться условная вероятность, реализуемая только в одном направлении, но не в другом. В проблемной области лечения менингита врач, возможно, знает, что при наличии симптома ограничения подвижности шеи менингит будет причиной в 1 из 5000 случаев. А это означает, что у врача имеется количественная информация в **диагностическом** направлении, от симптома к причине. Такому врачу использовать правило Байеса не требуется.

К сожалению, ➔ *знания в диагностическом направлении на практике встречаются намного реже, чем знания в причинном направлении.* В случае внезапной эпидемии менингита априорная вероятность этого заболевания, $P(m)$, повышается. Врач,

который вывел диагностическую вероятность $P(m \mid s)$ непосредственно из статистических наблюдений за пациентами перед эпидемией, не будет иметь представления о том, как обновить это значение после ее начала, тогда как врач, вычисляющий значение $P(m \mid s)$ из других трех значений, быстро обнаружит, что значение $P(m \mid s)$ должно увеличиваться пропорционально $P(m)$. Еще более важно то, что причинная информация $P(m \mid s)$ остается незатронутой эпидемией, поскольку она просто отражает, в чем выражается воздействие менингита на пациента. Использование прямых причинных знаний такого рода или знаний, основанных на модели, позволяет достичь надежности, которая крайне важна при создании вероятностных систем, применимых в реальном мире.

12.5.2. Использование правила Байеса: комбинирование свидетельств

Выше было показано, что правило Байеса может применяться для получения ответов на вероятностные запросы, в которых учтено условие, составляющее одно из свидетельств, например ограниченная подвижность шеи. В частности, было показано, что вероятностная информация часто доступна в форме $P(\mathit{effect} \mid \mathit{cause})$, где *effect* — результат, а *cause* — причина. А что произойдет, если свидетельств два или больше? Например, какой вывод может сделать стоматолог, если его пугающие стальные щипцы сомкнулись на больном зубе пациента? Если известно полное совместное распределение (см. рис. 12.2), можно сразу же отыскать ответ:

$$\mathbf{P}(\mathit{Cavity} \mid \mathit{toothache} \wedge \mathit{catch}) = \alpha \langle 0{,}108;\ 0{,}016 \rangle \approx \langle 0{,}871;\ 0{,}129 \rangle.$$

Но нам уже известно, что такой подход не масштабируется на большее количество переменных. Можно попробовать воспользоваться правилом Байеса для переформулировки этой задачи:

$$\begin{aligned} &\mathbf{P}(\mathit{Cavity} \mid \mathit{toothache} \wedge \mathit{catch}) = \\ &\quad = \alpha\, \mathbf{P}(\mathit{toothache} \wedge \mathit{catch} \mid \mathit{Cavity})\, \mathbf{P}(\mathit{Cavity}). \end{aligned} \tag{12.16}$$

Чтобы получить ответ на запрос в такой формулировке, необходимо знать условные вероятности конъюнкции $\mathit{toothache} \wedge \mathit{catch}$ для каждого значения *Cavity*. Это может быть легко осуществимо, если речь идет только о двух переменных свидетельства, но такой подход вновь становится источником затруднений при его масштабировании. Если имеется n возможных переменных свидетельства (рентгеновский снимок, гигиена полости рта и т.д.), то количество возможных комбинаций наблюдаемых значений, для которых необходимо будет знать условные вероятности, составит $O(2^n)$. Это не лучше, чем использование полного совместного распределения.

Чтобы улучшить ситуацию, необходимо найти некоторые дополнительные утверждения о рассматриваемой проблемной области, позволяющие упростить применяемые выражения. Понятие **независимости**, введенное в разделе 12.4, дает

ключ к этому решению, но требует уточнения. Было бы прекрасно, если бы переменные *Toothache* и *Catch* были независимыми, но они таковыми не являются: если зубной врач захватывает зуб щипцами, то он делает это, вероятно, потому, что в этом зубе есть полость, и именно наличие этой полости вызывает боль. Однако эти переменные *действительно* являются независимыми, *когда речь идет о наличии или отсутствии полости.* Причиной в каждом случае действительно является наличие полости в зубе, но ни одна из этих переменных не оказывает непосредственного влияния на другую: зубную боль определяет состояние нервов в зубе, тогда как точность наложения инструмента зависит прежде всего от навыков стоматолога, к которым зубная боль не имеет никакого отношения.[7] Математически это свойство записывается следующим образом:

$$\mathbf{P}(toothache \wedge catch \mid Cavity) = \mathbf{P}(toothache \mid Cavity)\mathbf{P}(catch \mid Cavity). \quad (12.17)$$

В этом уравнении выражена ▶ **условная независимость** переменных *toothache* и *catch*, если дана вероятность *Cavity*. Соответствующее выражение можно вставить в уравнение (12.16) с целью определения вероятности наличия полости:

$$\begin{aligned}&\mathbf{P}(Cavity \mid toothache \wedge catch) = \\ &\quad = \alpha\, \mathbf{P}(toothache \mid Cavity)\, \mathbf{P}(catch \mid Cavity)\, \mathbf{P}(Cavity).\end{aligned} \quad (12.18)$$

Теперь требования к наличию информации становятся такими же, как и при вероятностном выводе с использованием каждого свидетельства в отдельности: необходимо знать априорную вероятность $\mathbf{P}(Cavity)$ для переменной запроса и условную (апостериорную) вероятность каждого результата, если дана его причина.

Общее определение **условной независимости** двух переменных, X и Y, если дана третья переменная, Z, выражается следующей формулой:

$$\mathbf{P}(X, Y \mid Z) = \mathbf{P}(X \mid Z)\, \mathbf{P}(Y \mid Z).$$

Например, в проблемной области стоматологии кажется вполне резонным применить утверждение об условной независимости переменных *Toothache* и *Catch*, если дана вероятность *Cavity*:

$$\begin{aligned}&\mathbf{P}(Toothache, Catch \mid Cavity) = \\ &\quad = \mathbf{P}(Toothache \mid Cavity)\, \mathbf{P}(Catch \mid Cavity).\end{aligned} \quad (12.19)$$

Обратите внимание, что это утверждение несколько строже по сравнению с уравнением (12.17), в котором утверждается независимость только для конкретных значений *Toothache* и *Catch*. Если же воспользоваться свойством абсолютной независимости (уравнение (12.11)), то получим следующие эквивалентные формы, которыми также можно пользоваться (упражнение 12.21):

$$\mathbf{P}(X \mid Y, Z) = \mathbf{P}(X \mid Z) \quad \text{и} \quad \mathbf{P}(Y \mid X, Z) = \mathbf{P}(Y \mid Z).$$

[7] Предполагается, что пациент и стоматолог — разные люди.

В разделе 12.4 было показано, что утверждения при наличии абсолютной независимости позволяют выполнять декомпозицию полного совместного распределения на гораздо более мелкие распределения. Как оказалось, аналогичную декомпозицию допускают и утверждения при наличии условной независимости. Например, для утверждения в уравнении (12.19) декомпозицию можно вывести следующим образом:

$$\begin{aligned}
&\mathbf{P}(Toothache, Catch, Cavity) = \\
&= \mathbf{P}(Toothache, Catch \mid Cavity)\,\mathbf{P}(Cavity) = && \text{(по правилу умножения)} \\
&= \mathbf{P}(Toothache \mid Cavity)\,\mathbf{P}(Catch \mid Cavity)\,\mathbf{P}(Cavity) && \text{(по уравнению (12.19)).}
\end{aligned}$$

(То, что это уравнение действительно выполняется, читатель легко может проверить, обратившись к рис. 12.3.) Таким образом, большая исходная таблица теперь декомпонована на три меньшие таблицы. В исходной таблице было семь независимых значений. (Эта таблица включает $2^3 = 8$ значений, но их сумма должна быть равна 1, поэтому только 7 из них являются независимыми). Меньшие таблицы содержат в общей сложности $2 + 2 + 1 = 5$ независимых значений. (Распределение условных вероятностей, такое как $\mathbf{P}(Toothache \mid Cavity)$, включает две строки из двух значений, и в каждой строке их сумма равна 1, поэтому в данном случае есть только два независимых значения, а для априорного распределения, такого как $\mathbf{P}(Cavity)$, существует только одно независимое значение.) Переход от 7 к 5 независимым значениям может показаться не таким уж большим достижением, но выигрыш может оказаться намного больше при увеличении количества симптомов.

В общем случае для n симптомов, все из которых являются условно независимыми при заданной вероятности *Cavity*, размер представления растет как $O(n)$, а не $O(2^n)$. Это означает, что ➡ *утверждения об условной независимости могут обеспечивать масштабирование вероятностных систем; более того, такие утверждения могут быть подкреплены данными намного проще по сравнению с утверждениями об абсолютной независимости.* С концептуальной точки зрения переменная *Cavity* ▶ **разделяет** переменные *Toothache* и *Catch*, поскольку наличие полости в зубе является прямой причиной и зубной боли, и наложения щипцов на больной зуб. Разработка методов декомпозиции крупных вероятностных областей определения на слабо связанные подмножества с помощью свойства условной независимости стало одним из наиболее важных достижений в новейшей истории искусственного интеллекта.

12.6. Наивные байесовские модели

Приведенный выше пример из области стоматологии иллюстрирует часто встречающуюся ситуацию, в которой одна причина непосредственно влияет на целый ряд результатов, причем все эти результаты являются условно независимыми, когда дана эта причина. Полное совместное распределение может быть записано следующим образом:

$$\mathbf{P}(Cause, Effect_1, \ldots, Effect_n) = \mathbf{P}(Cause) \prod_i \mathbf{P}(Effect_i \mid Cause). \qquad (12.20)$$

Такое распределение вероятностей называется ▶**наивной байесовской** моделью, "наивной" — потому что часто используется (как упрощающее допущение) в тех случаях, когда переменные "результата" *не являются* строго независимыми при данной переменной причины. (Наивную байесовскую модель иногда называют ▶**байесовским классификатором**, и это не совсем корректное употребление термина побудило настоящих специалистов в области байесовских моделей называть ее не наивной, а **идиотской байесовской** моделью.) На практике наивные байесовские системы часто могут работать очень успешно, даже если предположение о независимости не является строго истинным.

Для использования наивной байесовской модели можно применить уравнение (12.20), чтобы получить вероятность причины с учетом некоторых наблюдаемых результатов. Обозначим наблюдаемые результаты как $\mathbf{E} = \mathbf{e}$, тогда как остальные переменные результата $\mathbf{Y}$ являются ненаблюдаемыми. Далее можно применить стандартный метод логического вывода из совместного распределения (уравнение (12.9)):

$$\mathbf{P}(Cause \mid \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(Cause, \mathbf{e}, \mathbf{y}).$$

Из уравнения (12.20) получаем

$$\begin{aligned} \mathbf{P}(Cause \mid \mathbf{e}) &= \alpha \sum_{\mathbf{y}} \mathbf{P}(Cause)\mathbf{P}(\mathbf{y} \mid Cause)\left(\prod_j \mathbf{P}(e_j \mid Cause)\right) = \\ &= \alpha\, \mathbf{P}(Cause)\left(\prod_j \mathbf{P}(e_j \mid Cause)\right) \sum_{\mathbf{y}} \mathbf{P}(\mathbf{y} \mid Cause) = \\ &= \alpha\, \mathbf{P}(Cause) \prod_j \mathbf{P}(e_j \mid Cause) \end{aligned} \qquad (12.21)$$

Здесь последняя строка может быть выведена потому, что сумма по $\mathbf{y}$ равна 1. Словами это уравнение можно интерпретировать так: для каждой возможной причины умножьте априорную вероятность причины на произведение условных вероятностей наблюдаемых результатов на данную причину, а затем нормализуйте результат. Время выполнения этих расчетов изменяется линейно по отношению к количеству наблюдаемых результатов и не зависит от количества ненаблюдаемых результатов (которое может быть очень большим в таких предметных областях, как медицина). В следующей главе будет показано, что это обычное явление в вероятностном выводе: переменные свидетельства, значения которых ненаблюдаемы, обычно "исчезают" из вычислений в полном составе.

12.6.1. Классификация текста с помощью наивной байесовской модели

Давайте посмотрим, как наивную байесовскую модель можно применить в задаче ►**классификации текстов**: дан некоторый текст, и необходимо установить, к какому из заранее определенного набора классов или категорий он относится. Здесь в качестве "причины" выступает переменная *Category*, а наличие или отсутствие в тексте определенных ключевых слов представлено переменными "результата" $HasWord_i$. Рассмотрим следующие два примера предложений, взятых из газетных статей.

1. В понедельник стоимость акций возросла — основные индексы прибавили 1%, поскольку сохраняется оптимизм в отношении сезона отчетности за первый квартал.
2. В понедельник проливные дожди по-прежнему охватывают большую часть восточного побережья, из-за чего в городе Нью-Йорк и других местах были сделаны предупреждения о возможности наводнения.

Наша задача заключается в том, чтобы отнести каждое предложение к некоторой *Category* — основному разделу газет: новости (*news*), спорт (*sports*), бизнес (*business*), погода (*weather*) или развлечения (*entertainment*). Наивная байесовская модель включает априорные вероятности $\mathbf{P}(Category)$ и условные вероятности $\mathbf{P}(HasWord_i \mid Category)$. Для каждой категории c вероятность $P(Category = c)$ оценивается как доля документов, относящихся к этой категории, из числа всех ранее просмотренных документов. Например, если в 9% статей идет речь о погоде, то $P(Category = weather) = 0{,}09$. Аналогичным образом, вероятности $\mathbf{P}(HasWord_i \mid Category)$ оцениваются как доля документов каждой категории, в которых присутствует слово i. Так, если примерно 37% статей о бизнесе содержат слово № 6, "акции", то вероятности $P(HasWord_6 = true \mid Category = business)$ можно присвоить значение 0,37.[8]

Чтобы классифицировать новый документ, необходимо проверить, какие ключевые слова в нем присутствуют, а затем применить уравнение (12.21), чтобы получить распределение апостериорных вероятностей по категориям. Если необходимо указать только одну категорию, выбирается та, у которой будет наибольшая апостериорная вероятность. Обратите внимание, что в этой задаче каждая

[8] Нужно проявлять осторожность, чтобы не присвоить нулевую вероятность словам, которые ранее не встречались в данной категории документов, поскольку нулевое значение уничтожит все остальные свидетельства в уравнении (12.21). То, что слово пока не встречалось, еще не означает, что оно никогда не встретится. Вместо этого следует зарезервировать небольшую часть распределения вероятностей для представления слов, "ранее не наблюдавшихся". Читайте главу 20 для получения более подробной информации по этому вопросу в целом и раздел 23.1.4, в котором представлены конкретные примеры словесных моделей.

переменная результата является наблюдаемой, поскольку всегда можно с уверенностью сказать, присутствует данное слово в документе или нет.

В наивной байесовской модели предполагается, что слова в документах появляются независимо друг от друга с частотой, определяемой категорией документа. Это предположение о независимости, очевидно, нарушается на практике. Например, фраза "первый квартал" встречается в статьях о бизнесе (или спорте) чаще, чем можно было бы предположить путем умножения вероятностей для отдельных слов "первый" и "квартал". Нарушение независимости обычно означает, что конечные апостериорные вероятности будут намного ближе к 1 или 0, чем они должны быть, — другими словами, такая модель будет проявлять излишнюю самоуверенность в своих предсказаниях. С другой стороны, даже с такими ошибками *рейтинг* возможных категорий часто оказывается весьма точным.

Наивные байесовские модели широко используются для определения языка и поиска документов, фильтрации спама и других задач классификации. Для решения таких задач, как медицинская диагностика, в которой фактические значения апостериорных вероятностей действительно имеют значение — например, при принятии решения, следует ли удалить аппендикс, — предпочтение отдают более сложным моделям, описываемым в следующей главе.

12.7. Очередное возвращение в мир вампуса

Комбинацию идей, изложенных в этой главе, можно применить для решения задачи выполнения вероятностных рассуждений в мире вампуса (полное описание мира вампуса приведено в главе 7). Неопределенность в мире вампуса возникает из-за того, что датчики агента предоставляют ему только частичную информацию о состоянии этого мира. Например, на рис. 12.5 показана ситуация, в которой каждый из трех не посещенных, но достижимых квадратов, [1,3], [2,2] и [3,1], может содержать яму. Чисто логический вывод не позволяет прийти к каким-либо заключениям о том, какой квадрат с наибольшей вероятностью окажется безопасным, поэтому логический агент может быть вынужден сделать случайный выбор. В этом разделе будет показано, что в подобной ситуации вероятностный агент может действовать гораздо успешнее, чем логический агент.

Наша цель — вычислить вероятность наличия ямы в каждом из этих трех квадратов. (В данном конкретном примере присутствие в них вампуса и золота игнорируется.) Относящиеся к этой задаче свойства мира вампуса включают, во-первых, то, что наличие ямы вызывает ощущение ветерка во всех соседних квадратах, и во-вторых то, что в каждом квадрате, отличном от [1,1], вероятность наличия в нем ямы равна 0,2. На первом этапе определяем множество необходимых случайных переменных, как показано ниже.

- Как и в случае логики высказываний, нам потребуется по одной булевой переменной P_{ij} для каждого квадрата; она будет принимать значение *true* тогда и только тогда, когда квадрат $[i,j]$ действительно содержит яму.
- Также нам потребуются булевы переменные B_{ij}, принимающие значение *true* тогда и только тогда, когда в квадрате $[i,j]$ ощущается ветерок. Из общего числа этих переменных нам достаточно будет рассмотреть только те, которые относятся к наблюдаемым квадратам, в данном случае — [1,1], [1,2] и [2,1].

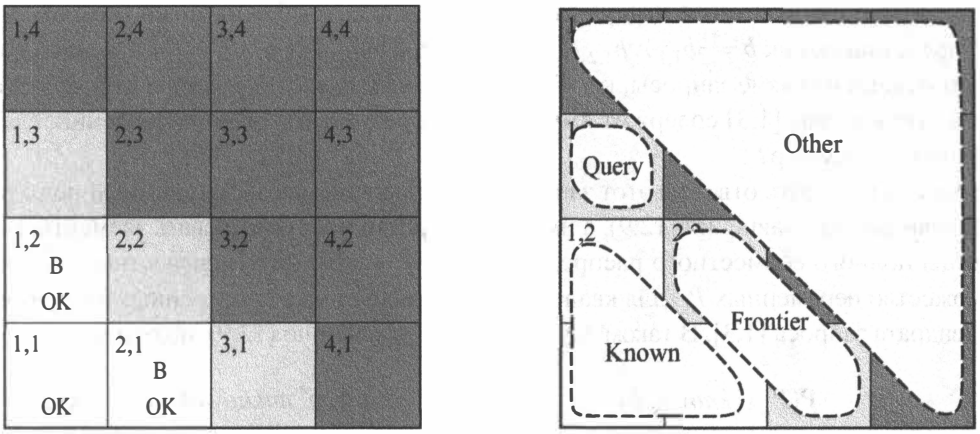


Рис. 12.5. а) После обнаружения ветерка как в квадрате [1,2], так и в квадрате [2,1] агент заходит в тупик — нет такого квадрата, который он мог бы обследовать без опасений. **б)** Распределение квадратов по категориям *Known* (известные), *Frontier* (периферийные) и *Other* (прочие) для формирования запроса (*Query*) в отношении квадрата [1,3]

На следующем этапе определяем полное совместное распределение $\mathbf{P}(P_{1,1},\ldots, P_{4,4}, B_{1,1}, B_{1,2}, B_{2,1})$. Применив правило умножения вероятностей, получаем следующее:

$$\mathbf{P}(P_{1,1},\ldots, P_{4,4}, B_{1,1}, B_{1,2}, B_{2,1}) = \\ = \mathbf{P}(B_{1,1}, B_{1,2}, B_{2,1} \mid P_{1,1},\ldots, P_{4,4})\mathbf{P}(P_{1,1},\ldots, P_{4,4}).$$

Эта декомпозиция позволяет очень легко определить, какими должны быть значения совместной вероятности. Первый терм представляет собой условную вероятность некоторой конфигурации данных о наличии ветерка, если дана конфигурация расположения ям; он принимает значение 1, если в квадратах, соседних с ямами, чувствуется ветерок, в противном случае принимает значение 0. Вторым термом является априорная вероятность конфигурации расположения ям. Каждый

квадрат может содержать яму с вероятностью 0,2, независимо от других квадратов, поэтому имеет место следующее:

$$\mathbf{P}(P_{1,1},\ldots, P_{4,4}) = \prod_{i,j=1,1}^{4,4} \mathbf{P}(P_{ij}). \qquad (12.22)$$

Для конкретной конфигурации с точно n ямами эта вероятность равна $0{,}2^n \times 0{,}8^{16-n}$.

В ситуации, показанной на рис. 12.2, *а*, свидетельство состоит из наблюдаемого ветерка (или его отсутствия) в каждом посещенном квадрате в сочетании с тем фактом, что каждый такой квадрат не содержит ямы. Эти факты можно сокращенно представить как $b = \neg b_{1,1} \wedge b_{1,2} \wedge b_{2,1}$ и $known = \neg p_{1,1} \wedge \neg p_{1,2} \wedge \neg p_{2,1}$. Нас интересуют ответы на такие запросы, как $\mathbf{P}(P_{1,3} \mid known, b)$: насколько велика вероятность того, что квадрат [1,3] содержит яму, учитывая результаты всех наблюдений, сделанных до сих пор?

Чтобы получить ответ на этот запрос, можно использовать стандартный подход, основанный на уравнении (12.9), а именно — просто просуммировать элементы таблицы полного совместного распределения. Пусть *Unknown* (неизвестное) — это множество переменных P_{ij} для квадратов, отличных от уже проверенных квадратов и квадрата запроса [1,3]. В таком случае, следуя уравнению (12.9), получим:

$$\mathbf{P}(P_{1,3} \mid known, b) = \alpha \sum_{unknown} \mathbf{P}(P_{1,3}, known, b, unknown). \qquad (12.23)$$

Полное совместное распределение вероятностей уже было определено, поэтому можно считать, что задача решена; точнее, осталось только выполнить вычисления. Количество неизвестных квадратов равно 12, следовательно, требуемая сумма состоит из $2^{12} = 4096$ термов. В общем случае количество термов в этой сумме растет экспоненциально в зависимости от количества квадратов.

Безусловно, напрашивается вопрос, а не являются ли другие квадраты не относящимися к делу? Как содержимое квадрата [4,4] может повлиять на наличие ямы в квадрате [1,3]? И действительно, эта догадка является приблизительно правильной, но ее необходимо уточнить. На самом деле здесь мы имеем в виду, что если бы мы знали значения переменных P для всех смежных квадратов, которые нас интересуют, то наличие (или отсутствие) ямы в других, более отдаленных, квадратах, уже не могло бы оказать влияния на нашу уверенность.

Пусть *Frontier* будет множеством переменных P (отличных от переменной запроса) всех тех квадратов, которые являются смежными с посещенными квадратами, — в нашем случае это квадраты [2,2] и [3,1]. Кроме того, пусть *Other* будет множеством переменных P для всех остальных неизвестных квадратов, — в нашем случае их имеется 10, как показано на рис. 12.5, *б*. С учетом этих определений $Unknown = Frontier \cup Other$. Ключевая догадка, приведенная выше, теперь может быть сформулирована следующим образом: наблюдения ветерка *условно*

независимы от других переменных, если даны известные переменные, переменные множества *Frontier* и переменная запроса. Чтобы воспользоваться этой идеей, необходимо преобразовать формулу запроса в такую форму, в которой данные о наличии ветерка становятся условно зависимыми от всех других переменных, а затем упростить полученное выражение с использованием утверждения об условной независимости:

$$\begin{aligned}
&\mathbf{P}(P_{1,3} \mid known, b) = \\
&= \alpha \sum_{unknown} \mathbf{P}(P_{1,3}, known, b, unknown) = && \text{(из уравнения (12.23))} \\
&= \alpha \sum_{unknown} \mathbf{P}(b \mid P_{1,3}, known, unknown)\mathbf{P}(P_{1,3}, known, unknown) = \\
& && \text{(правило умножения вероятностей)} \\
&= \alpha \sum_{frontier} \sum_{other} \mathbf{P}(b \mid known, P_{1,3}, frontier, other)\mathbf{P}(P_{1,3}, known, frontier, other) = \\
&= \alpha \sum_{frontier} \sum_{other} \mathbf{P}(b \mid known, P_{1,3}, frontier)\mathbf{P}(P_{1,3}, known, frontier, other),
\end{aligned}$$

где на конечном этапе используется утверждение об условной независимости: переменная b не зависит от *других переменных*, если даны известные переменные, переменные множества *Frontier* и переменная запроса $P_{1,3}$. Теперь первый терм в выражении не зависит от переменных множества *Other*, поэтому операцию суммирования можно переместить внутрь выражения:

$$\begin{aligned}
&\mathbf{P}(P_{1,3} \mid known, b) = \\
&= \alpha \sum_{frontier} \mathbf{P}(b \mid known, P_{1,3}, frontier) \sum_{other} \mathbf{P}(P_{1,3}, known, frontier, other).
\end{aligned}$$

Согласно утверждению о независимости, соответствующему приведенному в уравнении (12.22), терм априорной вероятности может быть факторизован, после чего все эти термы могут быть переупорядочены следующим образом:

$$\begin{aligned}
&\mathbf{P}(P_{1,3} \mid known, b) = \\
&= \alpha \sum_{frontier} \mathbf{P}(b \mid known, P_{1,3}, frontier) \sum_{other} \mathbf{P}(P_{1,3})P(known)P(frontier)P(other) = \\
&= \alpha P(known)\mathbf{P}(P_{1,3}) \sum_{frontier} \mathbf{P}(b \mid known, P_{1,3}, frontier)P(frontier) \sum_{other} P(other) = \\
&= \alpha' \mathbf{P}(P_{1,3}) \sum_{frontier} \mathbf{P}(b \mid known, P_{1,3}, frontier)P(frontier),
\end{aligned}$$

где на последнем этапе постоянный терм $P(known)$ вводится в нормализующую константу на основании того факта, что выражение $\sum_{other} P(other)$ равно 1.

Теперь в сумме по переменным множества *Frontier* $P_{2,2}$ и $P_{3,1}$ осталось только четыре терма. Использование свойств независимости и условной независимости позволило полностью исключить из рассмотрения все остальные квадраты.

Обратите внимание на то, что сумма вероятностей в выражении $\mathbf{P}(b \mid known, P_{1,3}, frontier)$ равна 1, если данные наблюдений о наличии ветерка совместимы с другими переменными, — в противном случае она равна 0. Следовательно, для

каждого значения $P_{1,3}$ выполняется суммирование по *логическим моделям* для переменных в множестве *Frontier*, согласующимся с известными фактами (это можно сравнить с тем, как осуществлялся перебор моделей на рис. 7.5 в разделе 7.3). Эти модели и связанные с ними априорные вероятности, *P*(*frontier*), показаны на рис. 12.6. Итак, мы получаем следующие значения:

$$\mathbf{P}(P_{1,3} \mid known, b) = \alpha' \langle 0{,}2(0{,}04 + 0{,}16 + 0{,}16),\ 0{,}8(0{,}04 + 0{,}16) \rangle \approx \langle 0{,}31,\ 0{,}69 \rangle.$$

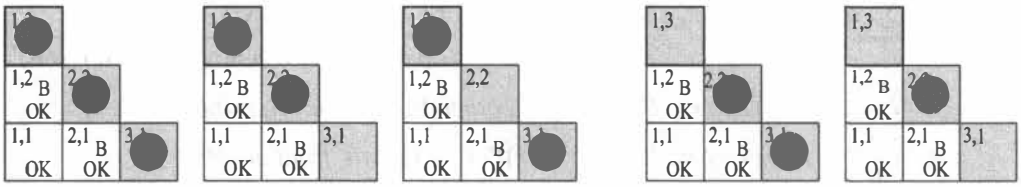


Рис. 12.6. Согласованные модели для переменных множества *Frontier* $P_{2,2}$ и $P_{3,1}$, показывающие значение *P*(*frontier*) для каждой модели. **а)** Три модели с $P_{1,3} = true$, где показаны две или три ямы. **б)** Две модели с $P_{1,3} = false$, где показаны одна или две ямы

Таким образом, квадрат [1,3] (и квадрат [3,1] по симметрии) содержит яму с вероятностью приблизительно 31%. Аналогичные вычисления, которые читатель вполне может выполнить самостоятельно, показывают, что квадрат [2,2] содержит яму с вероятностью приблизительно 86%. Агент в мире вампуса, определенно, должен избегать квадрата [2,2]! Обратите внимание, что логический агент из главы 7 ничего не знает о том, что выбор квадрата [2,2] может иметь намного худшие последствия, чем выбор любого другого из числа доступных. Логика может лишь сказать нам, что остается неизвестным, есть ли яма в квадрате [2,2], поэтому, чтобы получить больше информации по этому вопросу, необходимо обратиться к вероятностям.

В данном разделе было показано, что даже такие задачи, которые кажутся очень сложными, могут быть точно сформулированы в терминах теории вероятностей и решены с использованием простых алгоритмов. Для получения эффективных решений могут применяться соотношения, определяющие свойства независимости и условной независимости, что позволит упростить необходимые в расчетах операции суммирования. Эти соотношения часто соответствуют нашему интуитивному пониманию того, как следует выполнять декомпозицию задачи. В следующей главе будут разработаны формальные представления для таких соотношений, а также алгоритмы, оперирующие соответствующими представлениями и позволяющие эффективно осуществлять вероятностный логический вывод.

Резюме

В данной главе было предложено использовать теорию вероятности в качестве подходящей основы для рассуждений о неопределенности и дано самое общее представление о возможных способах ее использования.

- Неопределенность возникает как по причине экономии усилий, так и из-за отсутствия знаний. Ее невозможно избежать в сложной, недетерминированной или частично наблюдаемой проблемной среде.
- В оценках **вероятности** выражается неспособность агента прийти к определенному решению в отношении истинности высказывания. Вероятности обобщают степень уверенности агента в отношении свидетельств.
- **Теория принятия решений** объединяет убеждения и намерения агента за счет определения наилучшего действия как максимизирующего ожидаемую **полезность**.
- К основным типам вероятностных высказываний относятся **априорные** или **безусловные вероятности** и **апостериорные** или **условные вероятности** в отношении простых и сложных высказываний.
- Аксиомы вероятностей ограничивают допустимые значения вероятностей логически связанных высказываний. Агент, игнорирующий в своих действиях эти аксиомы, в некоторых обстоятельствах неизбежно будет вести себя нерационально.
- **Полное совместное распределение вероятностей** определяет вероятность каждого полного присваивания значений случайным переменным. Это распределение обычно слишком велико для того, чтобы его можно было создавать или использовать в явной форме, но если оно доступно, то может использоваться для получения ответа на любые запросы простым суммированием значений вероятностей для возможных миров, соответствующих высказываниям запроса.
- **Абсолютная независимость** между подмножествами случайных переменных позволяет выполнить декомпозицию полного совместного распределения на меньшие совместные распределения, что значительно уменьшает его сложность.
- **Правило Байеса** позволяет вычислять неизвестные вероятности из известных условных вероятностей, обычно в причинном направлении. При наличии многочисленных свидетельств применение правила Байеса приводит к возникновению таких же проблем масштабирования, которые возникают при использовании полного совместного распределения.
- Свойство **условной независимости**, вызванное наличием прямых причинных связей в проблемной области, позволяет провести декомпозицию полного совместного распределения на меньшие условные распределения.

В **наивной байесовской** модели предполагается наличие условной независимости всех переменных действия, если задана одна переменная причины; размеры этой модели увеличиваются линейно в зависимости от количества результатов.

- Агент в мире вампуса может вычислять вероятности ненаблюдаемых объектов мира и использовать их для принятия лучших решений в сравнении с простым логическим агентом. Условная независимость делает эти вычисления легко реализуемыми.

Библиографические и исторические заметки

Теория вероятностей появилась как средство анализа азартных игр. Примерно в 850 году н.э. индийский математик Махавирачарья описал, как подобрать набор ставок, исключающий возможность проигрыша (то, что мы сейчас называем *голландской книгой*). Первый значимый систематический анализ был проведен Джироламо Кардано примерно в 1565 году, но его работы были опубликованы только после его смерти (1663). К этому времени вероятность уже сформировалась как математическая дисциплина благодаря серии достижений, о которых Блез Паскаль сообщал в переписке с Пьером Ферма в 1654 году. Первым опубликованным учебником по теории вероятностей была книга Гюйгенса *De Ratiociniis in Ludo Aleae* ([1106], 1657). Взгляд на "лень и невежество" как источник неопределенности был предложен Джоном Арбетнотом в предисловии к его переводу этой книги Гюйгенса ([65], 1692).

Связь между вероятностью и рассуждением восходит по крайней мере к XIX веку: в 1819 году Пьер Лаплас сказал: "Теория вероятностей — это ни что иное, как здравый смысл, сведенный к расчетам". В 1850 году Джеймс Максвелл сказал: "Истинная логика для этого мира — исчисление вероятностей, которое учитывает величину вероятности, которая есть или должна быть в сознании разумного человека".

Долгие годы шли бесконечные дебаты по поводу источника и статуса значений вероятности. Сторонники ►**эмпирического** (частотного) подхода к вероятности настаивали на том, что эти значения могут быть получены только из *экспериментов*: если после тестирования 100 человек будет установлено, что десять из них имеют зубную полость, то можно будет утверждать, что вероятность образования такой полости равна приблизительно 0,1. Для сторонников этой точки зрения утверждение "вероятность образования зубной полости равна 0,1" означает, что значение 0,1 представляет собой долю случаев наличия полости, наблюдаемую в пределе бесконечного числа испытаний. Исходя из любой конечной выборки, можно оценить истинную долю, а также вычислить, насколько точной, скорее всего, является эта оценка.

Сторонники ►**объективистского** подхода полагают, что вероятности являются реальными аспектами Вселенной — склонностью самих объектов вести себя

определенным образом, а не просто описанием степени уверенности наблюдателя. Например, тот факт, что для обычной монеты (без жульнических подделок) вероятность выпадения орла составляет 0,5, является склонностью самой монеты. С этой точки зрения любые измерения эмпириков являются просто попытками наблюдать эти склонности. Большинство физиков согласны с тем, что квантовые явления объективно вероятностны, а неопределенность в макроскопическом масштабе — например, при подбрасывании монет — обычно возникает из-за незнания начальных условий и, по-видимому, не согласуется с представлением о склонности.

Сторонники ▶**субъективистского** подхода описывают вероятность как способ охарактеризовать убеждения агента, а не как что-то, имеющее некое внешнее физическое значение. Субъективный **байесовский** подход допускает любое самосогласованное приписывание априорных вероятностей высказываниям, однако затем настаивает на их надлежащем байесовском обновлении по мере поступления свидетельств.

Даже строгая эмпирическая позиция предполагает субъективность из-за проблемы ▶**эталонного класса**: пытаясь определить вероятность исхода *конкретного* эксперимента, эмпирик должен отнести его к эталонному классу "похожих" экспериментов с известными частотами исхода. Но как выбрать для него правильный класс? И.Дж. Гуд писал: "Каждое событие в жизни уникально, и каждая вероятность в реальной жизни, которую мы оцениваем на практике, — это событие, которое никогда не происходило ранее" (Гуд [894], 1983).

Например, в случае конкретного пациента эмпирик, желающий оценить вероятность наличия у него зубной полости, должен рассмотреть эталонный класс других пациентов, схожих с ним по некоторым важным аспектам — возрасту, симптомам, особенностям питания — и выяснить, какую часть из них составляли те, у кого имелась зубная полость. Если стоматолог примет во внимание все, что ему известно о пациенте, включая цвет волос, вес с точностью до грамма, девичью фамилию матери и так далее, эталонный класс в конечном счете окажется пустым. Эта ситуация была серьезной проблемой в философии науки.

Паскаль использовал вероятность в таких вычислениях, которые требовали не только ее объективной интерпретации как свойства мира, основанного на симметрии или относительных частотах событий, но и субъективной интерпретации, основанной на оценке степени уверенности. Первая интерпретация обнаруживается в проведенном Паскалем анализе вероятностей в играх с элементами случайности, а последняя — в знаменитых доводах "Спора с Паскалем", касающихся возможного существования Бога. Однако Паскаль недостаточно четко учитывал различие между этими двумя интерпретациями. Указанное различие было впервые наглядно подчеркнуто Джеймсом Бернулли (1654–1705).

Лейбниц ввел "классическое" понятие вероятности как доли перечислимых, равновероятных случаев, которое использовалось также Бернулли, но было полностью проанализировано Лапласом ([1353], 1816). Это понятие является противоречивым из-за наличия эмпирической (частотной) и субъективной интерпретации. События

могут рассматриваться как равновероятные либо из-за наличия естественной, физической симметрии между ними, либо просто из-за того, что мы не обладаем достаточными знаниями, которые позволили бы считать одно событие более вероятным, чем другое. Подход, предусматривающий использование последних, субъективных соображений, оправдывающих допустимость присваивания равных вероятностей, известен под названием ▶ **принцип безразличия** [792]. Этот принцип часто приписывают Лапласу ([1353], 1816), но он никогда не использовал это название явно, — впервые это сделал Кейнс ([1220], 1921). Джордж Буль и Джон Венн, оба ссылались на него как на ▶ **принцип недостаточной причины**.

Споры между сторонниками объективного и субъективного подходов еще более обострились в XX столетии. Колмогоров ([1272], 1963), Р.А. Фишер ([745], 1922) и Ричард фон Мизес ([745], 1922) были сторонниками относительной эмпирической (частотной) интерпретации. Приведенная в работе Карла Поппера [1812] (1959) (впервые опубликована на немецком языке в 1934 году) интерпретация "проявлений закономерностей" позволяет проследить истоки формирования относительных частот вплоть до основополагающих законов физической симметрии. Франк Рамсей ([1848], 1931), Бруно де Финетти ([553], 1937), Р.Т. Кокс ([489], 1946), Леонард Сэведж ([1984], 1954), Ричард Джеффри ([1129],1983) и И.Т. Джейнс (2003) интерпретировали вероятности как степени уверенности конкретных лиц. Их анализ степени уверенности был тесно связан с полезностями и с поведением, а именно — с готовностью субъекта делать те или иные ставки.

Рудольф Карнап предложил иную интерпретацию вероятности — не как определенной степени уверенности конкретного лица, а как степень уверенности, которую идеализированное рассуждающее лицо *должно* иметь в отношении истинности конкретного высказывания a, при заданном конкретном ряде свидетельств $\mathbf{e}$. Карнап попытался сделать это понятие степени **подтверждения** математически точным, как логическое отношение между a и $\mathbf{e}$. В настоящее время считается, что уникальной логики подобного типа не существует; скорее, любая подобная логика опирается на субъективное априорное распределение вероятностей, эффект которого уменьшается по мере сбора большего количества наблюдений.

Изучение этого отношения имело целью создание математической дисциплины, названной **индуктивной логикой** по аналогии с обычной дедуктивной логикой (Карнап [373], 1948; [374], 1950). Карнап не смог в достаточной степени расширить свою индуктивную логику за пределы пропозиционального случая, а Патнем ([1829], 1963) на состязательных аргументах показала, что ей присущи некоторые фундаментальные сложности. В более поздней работе Бакхуса, Гроува, Гальперна и Коллера ([99], 1992) метод Карнапа был расширен на теории первого порядка.

Первая строго аксиоматическая основа для теории вероятностей была предложена Колмогоровым ([1271], 1950) (впервые опубликована в Германии в 1933 году). Позднее Рени ([1873], 1970) дал аксиоматическое представление, использующее в качестве примитивов условные, а не абсолютные вероятности.

В дополнение к аргументам де Финетти в отношении обоснованности аксиом Кокс ([489], 1946) показал, что любая система неопределенных рассуждений, соответствующая его набору допущений, эквивалентна теории вероятностей. Это придало поклонникам вероятности новую уверенность, но остальные так и не были убеждены, возражая против допущения, что уверенность должна быть представлена единственным числом. Гальперн ([952], 1999) проанализировал допущения и указал на некоторые пробелы в первоначальной формулировке Кокса. Позднее Хорн ([1062], 2003) показал, как исправить эти трудности, а Джейнс (2003) привел аналогичный аргумент, который легче воспринимается.

Преподобный Томас Байес (1702–1761) сформулировал правило формирования рассуждений об условных вероятностях, которое позднее было названо в его честь (Байес [148], 1763). Но Байес рассматривал только случай равномерных априорных распределений, тогда как Лаплас независимо от него разработал теорию для общего случая. Байесовские вероятностные рассуждения использовались в приложениях ИИ уже с 1960-х годов, особенно в медицинской диагностике. Они использовались не только для постановки диагноза на основе имеющихся данных, но также для выбора необходимых дополнительных вопросов и тестов с использованием теории значения информации (раздел 16.6), когда имеющиеся доказательства были неубедительны (Горри [909], 1968; Горри и др. [910], 1973). Одна система даже превзошла людей-экспертов в диагностике острых брюшных заболеваний (де Домба и др. [550], 1974). В своей статье Лукас и соавт. ([1462], 2004) дали соответствующий обзор.

Однако эти ранние байесовские системы имели множество недостатков. Поскольку в них отсутствовали какие-либо теоретические модели диагностируемых ими условий, они были чувствительны к нерепрезентативным данным, встречающимся в тех ситуациях, когда были доступны лишь небольшие выборки (де Домба и др. [551], 1981). Еще более фундаментальным недостатком было то, что в этих системах не применялись лаконичные формальные средства (подобные тем, которые будут описаны в главе 13) для представления и использования информации об условной независимости информации. Поэтому успешная эксплуатация этих систем зависела от накопления, хранения и обработки громадных таблиц с вероятностными данными. Из-за этих сложностей в период с середины 1970-х до конца 1980-х годов интерес исследователей в области искусственного интеллекта к вероятностным методам решения задач в условиях неопределенности значительно снизился. Новые разработки в этой области, появившиеся лишь в конце 1980-х годов, будут рассмотрены в следующей главе.

Наивная байесовская модель для совместных распределений широко исследовалось в литературе по распознаванию образов уже с 1950-х годов (Дуда и Харт [659], 1973). Кроме того, такой способ представления использовался, часто непреднамеренно, в области выборки информации, начиная с работы Марона ([1496], 1961). Вероятностные основы этого метода, дополнительно рассматриваемые в упражнении 12.28, были исследованы Робертсоном и Спарком Джонсом

([1897], 1976). Домингос и Паццани ([631], 1997) объяснили причины поразительного успеха наивных байесовских рассуждений даже в тех проблемных областях, в которых они явно нарушали предположения о независимости.

По теории вероятностей есть много хороших вводных учебников, включая книги Бертсекаса и Цициклиса ([202], 2008), Росса ([1918], 2015) и Гринстеда и Снелла ([924], 1997). Де Грот и Шервиш ([593], 2001) выпустили объединенное введение в теорию вероятностей и статистику с байесовской точки зрения, а Уолпол и др. ([2289], 2016) предложили вводный курс для ученых и инженеров. Джейнс (2003) дал очень убедительное изложение байесовского подхода. Биллингсли ([215], 2012) и Венкатеш ([2268], 2012) придерживаются более математических методов изложения, включая обсуждение всех осложнений, связанных с непрерывными переменными, которое мы здесь опустили. Хакинг ([941], 1975) и Хелд ([946], 1990) рассматривают также раннюю историю концепции вероятности, а в статье Бенштейна ([192], 1996) приведен популярный обзор.

Упражнения

12.1. Исходя из основных принципов докажите, что $P(ab \wedge a) = 1$.

12.2. Воспользовавшись аксиомами вероятности, докажите, что любое распределение вероятностей дискретной случайной переменной должно в сумме составлять 1.

12.3. Для каждого из следующих высказываний либо докажите, что оно истинно, либо приведите контрпример.

а) Если $P(ab, c) = P(ba, c)$, то $P(ac) = P(bc)$

б) Если $P(ab, c) = P(a)$, то $P(bc) = P(b)$

в) Если $P(ab) = P(a)$, то $P(ab, c) = P(ac)$

12.4. Будет ли для агента рационально придерживаться трех убеждений: $P(A) = 0{,}4$; $P(B) = 0{,}3$ и $P(A \vee B) = 0{,}5$? Если это так, то какой диапазон вероятностей будет в этом случае рациональным для агента применительно к $A \wedge B$? Составьте таблицу, подобную приведенной на рис. 12.2, и покажите, подтверждает ли она ваши доводы в отношении рациональности. Затем составьте еще одну версию этой таблицы, в которой $P(A \vee B) = 0{,}7$. Объясните, почему рационально будет принять именно это значение вероятности, несмотря даже на то, что в данной таблице присутствует один случай, соответствующий проигрышу, и три случая с ничейным результатом. (*Подсказка.* Что агент 1 полагает относительно вероятности каждого из этих четырех случаев, в особенности того, когда имеет место проигрыш?)

12.5. Этот вопрос касается свойств возможных миров, определенных в разделе 12.2.2 как присваивание значений всем рассматриваемым случайным переменным. Мы будем работать с высказываниями, которые соответствуют точно одному возможному миру, поскольку они включают значения для всех переменных. В теории вероятностей такие высказывания называют **атомарными событиями**. Например, для булевых переменных X_1, X_2, X_3 высказывание $x_1 \wedge \neg x_2 \wedge \neg x_3$

определяет присваивание значений всех переменных, — на языке логики высказываний можно было бы сказать, что у него есть ровно одна модель.

а) Для случая n булевых переменных докажите, что любые два различных атомарных события являются взаимоисключающими, т.е. их конъюнкция эквивалентна значению *false*.

б) Докажите, что дизъюнкция всех возможных атомарных событий логически эквивалентна значению *true*.

в) Докажите, что любое высказывание логически эквивалентно дизъюнкции атомарных событий, которые влекут за собой его истинность.

12.6. Докажите уравнение (12.5) на основании уравнений (12.2) и (12.3).

12.7. Рассмотрим множество из всех возможных раздач по 5 карт при игре в покер со стандартной колодой в 52 карты, полагая, что раздача проводится честно.

а) Сколько атомарных событий будет в совместном распределении вероятностей (т.е. сколько существует различных раздач по пять карт)?

б) Какова вероятность каждого атомарного события?

в) Какова вероятность получения королевского флеш-стрита? А любого каре из всех возможных?

12.8. Исходя из полного совместного распределения, приведенного на рис. 12.3, рассчитайте следующее.

а) $\mathbf{P}(toothache)$

б) $\mathbf{P}(Cavity)$

в) $\mathbf{P}(Toothache \mid cavity)$

г) $\mathbf{P}(Cavity \mid toothache \vee catch)$

12.9. Исходя из полного совместного распределения, приведенного на рис. 12.3, рассчитайте следующее.

а) $\mathbf{P}(toothache)$

б) $\mathbf{P}(Catch)$

в) $\mathbf{P}(Cavity \mid catch)$

г) $\mathbf{P}(Cavity \mid toothache \vee catch)$

12.10. В своем письме от 24 августа 1654 года Паскаль попытался показать, как следует распределять денежные ставки, когда азартная игра должна закончиться преждевременно. Представьте себе игру, в которой каждый ход состоит из броска игральной кости. Игрок E получает очко, если выпадает четное число, а игрок O получает очко, если выпавшее число нечетное. Первый из игроков, набравший 7 очков, выигрывает банк. Предположим, что игра прерывается, когда игрок E ведет со счетом 4:2. Как в этом случае справедливо разделить деньги в банке между игроками? Какова будет общая формула? (Ферма и Паскаль сделали несколько ошибок, прежде чем решить проблему, но вы должны быть в состоянии сделать это правильно с первого раза.)

12.11. Решив использовать теорию вероятностей на практике, мы выбрали игровой автомат с тремя независимыми колесами, каждое из которых с равной вероятностью может отображать один из четырех символов: золотой слиток, колокольчик, лимон или вишню. Игровой автомат имеет следующую схему выплат для

ставки в одну монету (здесь "?" означает, что не имеет значения, что отображается на данном колесе):
– золотой слиток/золотой слиток/золотой слиток — выплата 20 монет
– колокольчик/колокольчик/колокольчик — выплата 15 монет
– лимон/лимон/лимон — выплата 5 монет
– вишня/вишня/вишня — выплата 3 монет
– вишня/вишня/? — выплата 2 монет
– вишня/?/? — выплата 1 монеты

а) Рассчитайте ожидаемый процент "окупаемости" автомата. Другими словами, какова ожидаемая выдача монет для каждой монеты, заплаченной за игру?
б) Вычислите вероятность того, что игра на этом игровом автомате приведет к выигрышу.
в) Оцените среднее и медианное количество игр, которое, как можно ожидать, будет сыграно до проигрыша всей имеющейся суммы, если начать с 10 монет. Можете запустить симуляцию, чтобы просто оценить эти значения, вместо того чтобы пытаться вычислить точный ответ.

12.12. Решив использовать теорию вероятностей на практике, мы выбрали игровой автомат с тремя независимыми колесами, каждое из которых с равной вероятностью может отображать один из четырех символов: золотой слиток, колокольчик, лимон или вишню. Игровой автомат имеет следующую схему выплат для ставки в одну монету (здесь "?" означает, что не имеет значения, что отображается на данном колесе):
– золотой слиток/золотой слиток/золотой слиток — выплата 21 монеты
– колокольчик/колокольчик/колокольчик — выплата 16 монет
– лимон/лимон/лимон — выплата 5 монет
– вишня/вишня/вишня — выплата 3 монет
– вишня/вишня/? — выплата 2 монет
– вишня/?/? — выплата 1 монеты

а) Рассчитайте ожидаемый процент "окупаемости" автомата. Другими словами, какова ожидаемая выдача монет для каждой монеты, заплаченной за игру?
б) Вычислите вероятность того, что игра на этом игровом автомате приведет к выигрышу.
в) Оцените среднее и медианное количество игр, которое, как можно ожидать, будет сыграно до проигрыша всей имеющейся суммы, если начать с 8 монет. Можете запустить симуляцию, чтобы просто оценить эти значения, вместо того чтобы пытаться вычислить точный ответ.

12.13. Необходимо передать n-битное сообщение агенту-получателю. В процессе передачи биты в сообщении независимо повреждаются (меняют значение на противоположное) с вероятностью ϵ для каждого. Используя дополнительный бит четности, отправляемый вместе с исходной информацией, получатель сможет восстановить сообщение, если повреждено не более одного бита во всем сообщении (включая бит четности). Предположим, нужно убедиться, что правильное сообщение будет получено с вероятностью не менее $1 - \delta$. Каким в этом

случае будет максимально допустимое значение n? Рассчитайте это значение для случая $\epsilon = 0{,}001$ и $\delta = 0{,}01$.

12.14. Необходимо передать n-битное сообщение агенту-получателю. В процессе передачи биты в сообщении независимо повреждаются (меняют значение на противоположное) с вероятностью ϵ для каждого. Используя дополнительный бит четности, отправляемый вместе с исходной информацией, получатель сможет восстановить сообщение, если повреждено не более одного бита во всем сообщении (включая бит четности). Предположим, нужно убедиться, что правильное сообщение будет получено с вероятностью не менее $1 - \delta$. Каким в этом случае будет максимально допустимое значение n? Рассчитайте это значение для случая $\epsilon = 0{,}002$ и $\delta = 0{,}01$.

12.15. Покажите, что три формы описания свойства независимости, приведенные в уравнении (12.11), являются эквивалентными.

12.16. Рассмотрим два медицинских теста на некоторый вирус, A и B. Тест A обладает эффективностью 95% при обнаружении вируса, когда он действительно присутствует, но дает 10% ложных положительных результатов (указывает, что вирус присутствует, когда на самом деле его нет). Тест B обладает эффективностью 90% при обнаружении вируса и дает 5% ложных положительных результатов. В этих двух тестах используются независимые методы идентификации вируса. Вирус присутствует у 1% всех людей. Пусть каждого человека проверяют на наличие вируса, используя только один из тестов, и для переносчиков вируса он дает положительный результат. Для какого из двух тестов получение положительного результата будет более показательным, если человек действительно является переносчиком вируса? Обоснуйте свой ответ математически.

12.17. Предположим, дана монета, которая падает орлом вверх с вероятностью x и решкой вверх — с вероятностью $1 - x$. Являются ли результаты последовательных бросков монеты независимыми друг от друга, если вы *знаете* значение x? Являются ли результаты последовательных бросков монеты независимыми друг от друга, если вы *не знаете* значение x? Обоснуйте свой ответ.

12.18. После ежегодного медицинского осмотра пациента у врача есть плохая новость и хорошая новость. Плохая новость состоит в том, что проверка на наличие серьезного заболевания оказалась положительной и что точность результатов этой проверки составляет 99% (т.е. вероятность получения положительного результата проверки, если пациент имеет это заболевание, равна 0,99, и такова же вероятность получения отрицательных результатов проверки, если пациент не имеет этого заболевания). Хорошая новость состоит в том, что это заболевание — редкое и поражает только одного из 10 тысяч людей того возраста, в котором находится пациент. Почему новость, что это заболевание редкое, следует считать хорошей? Каковы шансы на то, что пациент действительно имеет данное заболевание?

12.19. После ежегодного медицинского осмотра пациента у врача есть плохая новость и хорошая новость. Плохая новость состоит в том, что проверка на наличие серьезного заболевания оказалась положительной и что точность результатов этой проверки

составляет 99% (т.е. вероятность получения положительного результата проверки, если пациент имеет это заболевание, равна 0,99, и такова же вероятность получения отрицательных результатов проверки, если пациент не имеет этого заболевания). Хорошая новость состоит в том, что это заболевание — редкое и поражает только одного из 100 тысяч людей того возраста, в котором находится пациент. Почему новость, что это заболевание редкое, следует считать хорошей? Каковы шансы на то, что пациент действительно имеет данное заболевание?

12.20. Довольно часто полезно рассмотреть результаты некоторых конкретных высказываний в контексте некоторого общего фонового свидетельства, которое остается неизменным, а не действовать в условиях полного отсутствия информации. В приведенных ниже вопросах предлагается доказать более общие версии правила умножения вероятностей и правила Байеса применительно к некоторому фоновому свидетельству **e**.

а) Докажите версию общего правила умножения вероятностей для условных вероятностей:

$$\mathbf{P}(X, Y|\mathbf{e}) = \mathbf{P}(XY, \mathbf{e})\mathbf{P}(Y|\mathbf{e}).$$

б) Докажите версию правила Байеса с условными вероятностями из уравнения (12.13).

12.21. Покажите, что утверждение условной независимости

$$\mathbf{P}(X, Y|Z) = \mathbf{P}(X|Z)\mathbf{P}(Y|Z)$$

эквивалентно любому из следующих утверждений:

$$\mathbf{P}(X|Y, Z) = \mathbf{P}(X|Z) \quad \text{и} \quad \mathbf{P}(Y|X, Z) = \mathbf{P}(Y|Z).$$

12.22. Предположим, вам вручили мешок, содержащий n подлинных монет, и сообщили, что $n-1$ из этих монет являются нормальными, т.е. такими, что с одной стороны у них орел, с другой — решка, а одна монета — фальшивая: на обеих ее сторонах изображен орел.

а) Допустим, вы открыли мешок, случайным образом выбрали монету и подбросили ее, в результате чего выпал орел. Какова (условная) вероятность того, что выбранная вами монета является фальшивой?

б) Теперь предположим, что вы продолжаете подбрасывать эту монету в общей сложности k раз после того, как она была выбрана, и наблюдаете k выпадений орла. Какова теперь условная вероятность того, что вы выбрали фальшивую монету?

в) Наконец, предположим, что вы хотите принять решение, является ли выбранная монета фальшивой, подбросив ее k раз. Процедура принятия решения возвращает *fake* (фальшивая), если все k бросков приводят к выпадению орла, а в противном случае она возвращает значение *normal* (нормальная). Какова (безусловная) вероятность того, что эта процедура сделает ошибку?

12.23. В этом упражнении требуется вычислить коэффициент нормализации для примера с заболеванием менингитом. Вначале выберите подходящее значение для $P(s \mid \neg m)$ и примените его для вычисления ненормализованных значений $P(m \mid s)$ и $P(\neg m \mid s)$ (т.е. игнорируя терм $P(s)$ в выражении правила Байеса). Теперь нормализуйте эти значения таким образом, чтобы они в сумме составляли 1.

12.24. В этом упражнении исследуется то, как соотношения, касающиеся условной независимости, влияют на количество информации, требуемой для вероятностных вычислений.

а) Предположим, что необходимо рассчитать значение $P(h \mid e_1, e_2)$, а информация об условной независимости отсутствует. Какие из следующих множеств чисел являются достаточными для такого вычисления?

1. $\mathbf{P}(E_1, E_2)$, $\mathbf{P}(H)$, $\mathbf{P}(E_1 \mid H)$, $\mathbf{P}(E_2 \mid H)$

2. $\mathbf{P}(E_1, E_2)$, $\mathbf{P}(H)$, $\mathbf{P}(E_1, E_2 \mid H)$

3. $\mathbf{P}(H)$, $\mathbf{P}(E_1 \mid H)$, $\mathbf{P}(E_2 \mid H)$

б) Предположим, известно, что $\mathbf{P}(E_1 \mid H, E_2) = \mathbf{P}(E_1 \mid H)$ для всех значений H, E_1, E_2. Какое из этих трех множеств значений теперь будет достаточным?

12.25. Пусть X, Y, Z — случайные булевы переменные. Обозначьте восемь элементов совместного распределения $\mathbf{P}(X, Y, Z)$ буквами алфавита от a до h. Выразите утверждение, что X и Y являются условно независимыми при заданном Z в виде множества уравнений, связывающих элементы от a до h. Сколько среди них *неизбыточных* уравнений?

12.26. Предположим, что вы — свидетель ночного наезда на пешехода в Афинах с участием такси, которое скрылось с места происшествия. Все такси в Афинах покрашены в синий или зеленый цвет. Вы поклялись под присягой, что такси было синим, при этом результаты широких экспериментов показывают, что в условиях плохого освещения надежность распознавания синего и зеленого цветов составляет 75%.

а) Возможно ли рассчитать наиболее вероятный цвет этого такси? (*Подсказка.* Тщательно проведите различие между высказыванием, что такси — синего цвета, и высказыванием, что оно показалось вам синим.)

а) Что изменится после получения информации о том, что в Афинах 9 из 10 такси зеленого цвета?

12.27. Запишите общий алгоритм получения ответов на запросы в форме $\mathbf{P}(Cause \mid \mathbf{e})$, используя наивное байесовское распределение. Исходите из предположения, что свидетельство $\mathbf{e}$ может присваивать значения любому подмножеству переменных результата.

12.28. Категоризацией текста называется задача присваивания данному конкретному документу одной из фиксированного множества категорий на основе анализа его текста. Для решения этой задачи часто используются наивные байесовские модели, в которых переменной запроса является категория документа, а в качестве переменных "результата" рассматривается наличие или отсутствие каждого слова в "языке" категории. Основное предположение состоит в том, что слова в документах встречаются независимо друг от друга, а их частоты определяются категорией документа.

а) Дайте точное объяснение, как можно сформировать такую модель, получив в качестве "обучающих данных" множество документов, уже распределенных по категориям.

б) Дайте точное объяснение, как следует определять категорию нового документа.

в) Является ли указанное предположение о независимости обоснованным? Обсудите этот вопрос.

12.29. В проведенном в этой главе анализе мира вампуса использовался тот факт, что каждый квадрат содержит яму с вероятностью 0,2, независимо от содержимого других квадратов. Вместо этого примем предположение, что точно $N/5$ ям равномерно разбросаны случайным образом среди N квадратов, отличных от [1,1]. Останутся ли переменные $P_{i,j}$ и $P_{k,l}$ все еще независимыми? Каково теперь совместное распределение $\mathbf{P}(P_{1,1},..., P_{4,4})$? Заново выполните вычисление вероятностей наличия ям в квадратах [1,3] и [2,2].

12.30. Повторите расчет вероятности наличия ям в квадратах [1,3] и [2,2], полагая, что каждый квадрат содержит яму с вероятностью 0,01, независимо от других квадратов. Что в этом случае можно будет сказать об относительной эффективности логического и вероятностного агента?

12.31. Реализуйте гибридного вероятностного агента для мира вампуса, основываясь на гибридном агенте, представленном на рис. 7.20, и процедуре вероятностного вывода, рассмотренной в этой главе.

ГЛАВА 13

Вероятностные рассуждения

В этой главе объясняется, как строить эффективные сетевые модели для проведения рассуждений в условиях неопределенности в соответствии с законами теории вероятности и как отличить корреляцию от причинности.

В главе 12 рассматривались основные элементы теории вероятностей и отмечалась важность отношений независимости и условной независимости для упрощения вероятностных представлений о мире. В этой главе будет представлен систематический способ явного представления таких связей в форме **байесовских сетей**. В ней определены синтаксис и семантика этих сетей и показано, как они могут использоваться для представления неопределенных знаний естественным и эффективным способом. Далее будет продемонстрировано, что вероятностный вывод, хотя и не осуществимый в наихудшем случае с помощью вычислительных методов, может эффективно выполняться во многих практических ситуациях. Кроме того, будет описан целый ряд алгоритмов приближенного вероятностного вывода, которые часто могут применяться в тех случаях, когда точный вероятностный вывод неосуществим. В главе 15 с целью эффективного определения вероятностных моделей основные идеи байесовских сетей распространяются на более выразительные формальные языки.

13.1. Представление знаний в неопределенной проблемной области

В главе 12 было показано, что полное совместное распределение вероятностей позволяет получать ответы на любые вопросы о рассматриваемой проблемной области, но по мере увеличения количества переменных оно может приобретать столь большие размеры, что вычисления становятся невозможными. Более того, определение вероятностей для возможных миров по отдельности, одного за другим, является довольно неестественным и может оказаться весьма затруднительным.

Также в предыдущей главе было показано, что связи, определяющие независимость и условную независимость между переменными, позволяют существенно

сократить количество вероятностей, которые должны быть известны для определения полного совместного распределения. В этом разделе рассматривается структура данных, позволяющая представить подобные зависимости между переменными, которую называют ▶**байесовской сетью**.[1] Байесовские сети позволяют представить практически *любое* полное совместное распределение вероятностей и во многих случаях позволяют сделать это очень кратко.

Байесовская сеть — это ориентированный граф, в котором каждая вершина помечена количественной вероятностной информацией. Полная спецификация такой сети описана ниже.

1. Каждой вершине графа соответствует случайная переменная, которая может быть дискретной или непрерывной.
2. Направленные ребра или стрелки соединяют пары вершин. Если стрелка направлена от вершины X к вершине Y, то вершина X называется *родительской* вершиной вершины Y. Граф не имеет направленных циклов и, следовательно, является ориентированным ациклическим графом (или DAG — *Directed Acyclic Graph*).
3. Каждая вершина X_i характеризуется связанной с ней вероятностной информацией $\theta(X_i \mid Parents(X_i))$, количественно оценивающей влияние родительских вершин на эту вершину с использованием конечного числа ▶**параметров**.

Топология сети (множество ее вершин и ребер) определяет условную независимость связей, присутствующих в данной проблемной области, — некоторым образом, который вскоре будет точно сформулирован. *Интуитивно* смысл стрелки обычно состоит в том, что вершина X оказывает *прямое влияние* на вершину Y — на основании предположения, что вершины причин должны быть родительскими по отношению к вершинам следствий. Специалисту в некоторой проблемной области обычно несложно установить, какие непосредственные влияния в ней существуют, — как правило, это намного проще действительного определения самих вероятностей. Когда топология байесовской сети будет составлена, останется лишь указать локальную вероятностную информацию для каждой переменной в форме распределения условных вероятностей с учетом ее родительских переменных. Полное совместное распределение для всех переменных определяется топологией и локальной вероятностной информацией.

Вернемся к простому миру, описанному в главе 12 и состоящему из переменных *Toothache*, *Cavity*, *Catch* и *Weather*. В этой главе было показано, что

[1] В настоящее время это название (иногда сокращаемое до *байесова сеть*) применяется наиболее часто, но в 1980- и 1990-х годах их называли **сетями доверия** (*belief networks*). Название **причинно-следственная сеть** (*causal network*) относится к байесовским сетям с дополнительными ограничениями на смысл стрелок (см. раздел 13.5). Термин **графическая модель** относится к более широкому классу структур, включающему и байесовские сети.

переменная *Weather* не зависит от других переменных; более того, было доказано, что переменные *Toothache* и *Catch* являются условно независимыми, если задана переменная *Cavity*. На рис. 13.1 эти отношения представлены в виде структуры байесовской сети. Формально условная независимость переменных *Toothache* и *Catch*, если задана переменная *Cavity*, обозначается *отсутствием* связи между вершинами *Toothache* и *Catch*. Интуитивно понятно, что в этой сети представлен тот факт, что переменная *Cavity* является непосредственной причиной *Toothache* и *Catch*, тогда как между переменными *Toothache* и *Catch* нет прямой причинной связи.

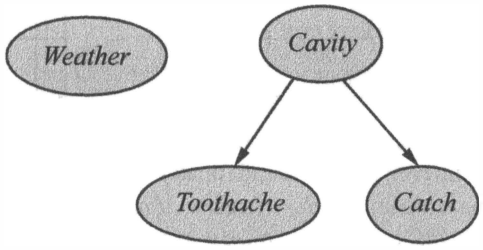


Рис. 13.1. Простая байесовская сеть, в которой переменная *Weather* независима от трех других переменных, а переменные *Toothache* и *Catch* являются условно независимыми, если задана переменная *Cavity*

Теперь рассмотрим следующий, немного более сложный пример. Предположим, что в доме была установлена новая система охранной сигнализации. Она достаточно надежно обнаруживает попытку взлома, но иногда срабатывает и на небольшие землетрясения. (Этим примером мы обязаны Джуди Перлу, проживающему в Лос-Анджелесе — на территории, подверженной частым землетрясениям.) У владельца дома есть два соседа, Джон и Мэри, которые обещали звонить ему на работу, услышав сигнал тревоги. Джон почти всегда звонит, услышав сигнал тревоги, но иногда путает с ним телефонный звонок в доме соседа и в таких случаях также звонит владельцу. С другой стороны, Мэри любит довольно громкую музыку и поэтому зачастую вообще пропускает сигнал тревоги соседской сигнализации. Принимая во внимание, кто из этих соседей звонил или не звонил, желательно в каждом случае правильно оценить вероятность взлома.

Байесовская сеть для этой проблемной области приведена на рис. 13.2. Структура сети говорит о том, что взлом и землетрясение непосредственно влияют на вероятность появления тревожного сигнала, в то время как звонки Джона и Мэри зависят только от тревожного сигнала. Поэтому сеть подтверждает наши предположения, что эти соседи самостоятельно не обнаруживают какие-либо попытки взлома, не замечают незначительных землетрясений и не совещаются друг с другом перед звонками.

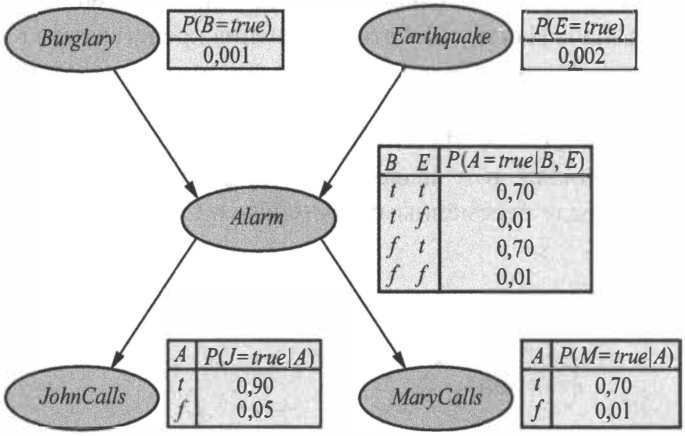


Рис. 13.2. Типичная байесовская сеть, для которой приведены и топология, и таблицы условных вероятностей (CPT). В таблицах CPT буквами *B*, *E*, *A*, *J* и *M* обозначены следующие переменные: *Burglary* (взлом), *Earthquake* (землетрясение), *Alarm* (сигнал тревоги), *JohnCalls* (звонки Джона) и *MaryCalls* (звонки Мэри)

Информация о локальной вероятности, приведенная для каждой вершины на рис. 13.2, представлена в виде ▶**таблицы условной вероятности** (*Conditional Probability Table* — ▶**CPT**). (Таблицы CPT могут использоваться только для дискретных переменных; другие представления, включая те, которые подходят для непрерывных переменных, описаны в разделе 13.2.) Каждая строка в таблице CPT содержит условную вероятность каждого значения вершины для ▶**обусловливающего случая** (*conditioning case*). Обусловливающий случай является просто одной из возможных комбинаций значений родительских вершин — его можно рассматривать как миниатюрное атомарное событие. Сумма элементов в каждой строке должна быть равна 1, поскольку элементы этой строки представляют собой исчерпывающее множество случаев для данной переменной. Для булевых переменных, если известно, что вероятность значения *true* равна p, вероятность значения *false* должна быть $1-p$, поэтому второе число часто просто опускается, как это сделано на рис. 13.2. В общем случае любая таблица для булевой переменной с k булевыми родительскими переменными содержит 2^k независимо определяемых вероятностей. Таблица для вершины без родительских вершин имеет только одну строку, представляющую априорные вероятности каждого возможного значения соответствующей переменной.

Обратите внимание, что в этой сети нет вершин, соответствующих ситуациям, когда Мэри в данный момент слушала бы громкую музыку или звонил бы соседский телефон, сбивая с толку Джона. Эти факторы подытожены в показателях неопределенности, связанных с ребрами, направленными от вершины *Alarm*

к вершинам *JohnCalls* и *MaryCalls*. Такой подход является примером проявления в действии как экономии усилий, так и недостатка знаний (см. раздел 12.1.1), поскольку потребовалось бы слишком много работы, чтобы узнать, по какой причине эти факторы могут оказаться более или менее вероятными в каждом конкретном случае; к тому же все равно отсутствует приемлемый способ получения релевантной информации.

Вероятности, показанные на рисунке, фактически подытоживают *потенциально бесконечное* множество обстоятельств, которые либо могут привести к нарушениям при выдаче сигнала тревоги (высокая влажность, отсутствие напряжения в сети электропитания, разряд аккумулятора, обрыв проводов, дохлая мышь, застрявшая внутри звонка, и т.д.), либо станут причиной того, что Джон или Мэри не смогут о нем сообщить (выйдут на обед, отправятся в отпуск, на время оглохнут, не расслышат сигнал тревоги из-за шума пролетающего вертолета и т.д.). Действуя таким образом, маленький агент получает возможность справиться со всем, что происходит в очень большом мире, по крайней мере приблизительно.

13.2. Семантика байесовских сетей

Синтаксис байесовской сети состоит из ориентированного ациклического графа с некоторой локальной вероятностной информацией, связанной с каждой вершиной. *Семантика* байесовской сети определяет, как синтаксис соответствует совместному распределению по переменным сети.

Предположим, что байесовская сеть включает n переменных, $X_1,\ldots,X_n$. Тогда типичная запись в совместном распределении будет иметь вид $P(X_1 = x_1 \wedge \ldots \wedge X_n = x_n)$ или $P(x_1,\ldots,x_n)$ для краткости. Семантика байесовских сетей определяет каждую запись в совместном распределении следующим образом:

$$P(x_1,\ldots,x_n) = \prod_{i=1}^{n} \theta(x_i \mid parents(X_i)), \qquad (13.1)$$

где $parents(X_i)$ обозначает конкретные значения переменных $Parents(X_i)$, которые появляются в $x_1,\ldots,x_n$. Следовательно, каждый элемент в совместном распределении представлен в виде произведения соответствующих элементов локальных условных распределений в байесовской сети.

Чтобы проиллюстрировать введенные выше понятия, рассчитаем вероятность того, что прозвучал сигнал тревоги, но не было ни взлома, ни землетрясения, а Мэри и Джон оба позвонили хозяину дома. Для этого просто перемножим соответствующие значения локальных условных распределений (и для краткости сократим имена этих переменных до одной буквы):

$$\begin{aligned} P(j, m, a, \neg b, \neg e) &= P(j \mid a)P(m \mid a)P(a \mid \neg b \wedge \neg e)P(\neg b)P(\neg e) = \\ &= 0{,}90 \times 0{,}70 \times 0{,}01 \times 0{,}999 \times 0{,}998 = 0{,}00628. \end{aligned}$$

В разделе 12.3 было показано, что полное совместное распределение может использоваться для получения ответа на любой запрос о данной проблемной области. Если байесовская сеть является представлением совместного распределения, то и она может использоваться для получения ответа на любой запрос посредством суммирования всех соответствующих значений вероятности совместного распределения, каждое из которых рассчитывается путем умножения вероятностей из локальных условных распределений. В разделе 13.3 этот вопрос рассматривается более подробно, а также описываются методы, которые являются гораздо более эффективными.

На данный момент необходимо дать дополнительные разъяснения по одному важному вопросу: каков смысл чисел, входящих в локальные условные распределения $\theta(x_i \mid parents(X_i))$? Оказывается, исходя из уравнения (13.1), можно доказать, что параметры $\theta(x_i \mid parents(X_i))$ в точности являются условными вероятностями $P(x_i \mid parents(X_i))$, следующими из совместного распределения. Вспомните, что условные вероятности могут быть вычислены из совместного распределения следующим образом:

$$P(x_i \mid parents(X_i)) \equiv \frac{P(x_i, parents(X_i))}{P(parents(X_i))} = \\ = \frac{\sum_{\mathbf{y}} P(x_i, parents(X_i), \mathbf{y})}{\sum_{x_i', \mathbf{y}} P(x_i', parents(X_i), \mathbf{y})}.$$

Здесь $\mathbf{y}$ представляет значения всех переменных, отличных от X_i и его родительских вершин. Исходя из этой последней строки, можно доказать, что $P(x_i \mid parents(X_i)) = \theta(x_i \mid parents(X_i))$ (см. упражнение 13.3). Следовательно, мы можем переписать уравнение (13.1) как

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i)). \qquad (13.2)$$

А это означает, что когда оцениваются значения для локальных условных распределений, они должны представлять собой фактические условные вероятности для переменной при заданных родительских переменных. Например, когда мы определяем $\theta(JohnCalls = true \mid Alarm = true) = 0{,}90$, это должно означать, что примерно в 90% случаев, когда звучит сигнал тревоги, Джон позвонит владельцу дома. Тот факт, что каждый параметр в байесовской сети имеет точный смысл в терминах лишь небольшого множества переменных, является критически важным для надежности и простоты определения моделей.

Метод построения байесовских сетей

Уравнение (13.2) определяет, что означает данная байесовская сеть. На следующем этапе необходимо выяснить, как *построить* байесовскую сеть таким

образом, чтобы результирующее совместное распределение являлось адекватным представлением данной проблемной области. Прежде всего покажем, что из уравнения (13.2) следуют определенные отношения условной независимости, которые могут использоваться инженером по знаниям как руководящие указания при определении топологии сети. Сначала перезапишем это совместное распределение в терминах условных вероятностей с использованием правила умножения вероятностей (см. раздел 12.2.1):

$$P(x_1,\dots,x_n) = P(x_n \mid x_{n-1},\dots,x_1)P(x_{n-1},\dots,x_1).$$

Затем повторим этот процесс, приводя каждую совместную вероятность к условной вероятности и совместной вероятности для меньшего множества переменных. В конечном итоге будет получено одно большое произведение:

$$P(x_1,\dots,x_n) = P(x_n \mid x_{n-1},\dots,x_1)P(x_{n-1} \mid x_{n-2},\dots,x_1))\dots P(x_2 \mid x_1)P(x_1) =$$
$$= \prod_{i=1}^{n} P(x_i \mid x_{i-1},\dots,x_1).$$

Это тождество называется ▶**цепным правилом** (*chain rule*). Оно справедливо для любого множества случайных переменных. Сравнивая его с уравнением (13.2), можно обнаружить, что эта спецификация совместного распределения эквивалентна общему утверждению, что для каждой переменной X_i в байесовской сети

$$\mathbf{P}(X_i \mid X_{i-1},\dots,X_1) = \mathbf{P}(X_i \mid Parents(X_i)), \tag{13.3}$$

при условии, что $Parents(X_i) \subseteq \{ X_{i-1},\dots,X_1 \}$. Это последнее условие можно легко выполнить, просто пронумеровав вершины графа в ▶**топологическом порядке**, т.е. в любом порядке, соответствующем структуре ориентированного графа. Например, узлы на рис. 13.2 могут быть упорядочены в порядке *B*, *E*, *A*, *J*, *M* или *E*, *B*, *A*, *M*, *J* и т.д.

Таким образом, уравнение (13.3) свидетельствует о том, что байесовская сеть будет правильным представлением проблемной области, только если каждая вершина в ней условно независима от ее предшественниц в конкретном упорядочении вершин при заданных ее родительских вершинах. Удовлетворить это условие можно, придерживаясь следующей методологии.

1. *Вершины*. Сначала определите множество переменных, необходимых для моделирования проблемной области, а затем тем или иным образом упорядочьте их: $\{X_1,\dots,X_n\}$. Подойдет любой порядок, но результирующая сеть будет более компактной, если переменные упорядочить так, чтобы причины предшествовали следствиям.
2. *Ребра*. Для всех *i*, от 1 до *n*, выполните следующее.
   - Для каждого X_i выберите из $X_1,\dots,X_{i-1}$ минимальное множество родительских вершин, — такое, что уравнение (13.3) будет выполняться.

- Для каждой родительской вершины добавьте ребро со стрелкой в направлении от родителя к X_i.
- *Таблицы CPT.* Для всех вершин укажите таблицы условных вероятностей, $\mathbf{P}(X_i \mid Parents(X_i))$.

Интуитивно понятно, что множество родительских вершин вершины X_i должно включать все те вершины в $X_1,\ldots,X_{i-1}$, которые ➨ *непосредственно влияют* на X_i. Например, предположим, что сеть, представленная на рис. 13.2, уже полностью построена и осталось лишь выбрать родительские вершины для узла *MaryCalls*. Безусловно, на вершину *MaryCalls* влияет, произошло ли событие *Burglary* или *Earthquake*, но это — не *непосредственное* влияние. Наши знания в данной проблемной области говорят о том, что эти события могут влиять на поведение Мэри в отношении звонков хозяину дома только через их воздействие на сигнал тревоги. Кроме того, при наличии сигнала тревоги звонок Джона никак не может повлиять на звонок Мэри. Говоря формально, при построении этой сети у нас есть полная уверенность в том, что справедливо следующее утверждение об условной независимости:

$$\mathbf{P}(MaryCalls \mid JohnCalls, Alarm, Earthquake, Burglary) = \mathbf{P}(MaryCalls \mid Alarm).$$

Следовательно, *Alarm* будет единственным родительским узлом для узла *MaryCalls*.

Поскольку каждый узел в создаваемой сети соединяется только с узлами-предшественниками в общем упорядочении, этот метод построения гарантированно исключает возможность появления в ней циклов. Другим важным свойством байесовских сетей является то, что они не содержат избыточных значений вероятностей. А там, где нет избыточности, нет и шансов для появления несовместимости: ➨ *у инженера по знаниям или специалиста по проблемной области нет возможности построить байесовскую сеть, в которой будут нарушаться аксиомы вероятности.*

Компактность сети и упорядочение вершин

Помимо того, что байесовская сеть является полным и неизбыточным представлением проблемной области, она часто оказывается намного более *компактной* по сравнению с полным совместным распределением. Именно благодаря этому свойству байесовские сети оказываются применимыми для представления проблемных областей со многими переменными. Компактность байесовских сетей является примером общего свойства ►**локально структурированных** (или, иначе, ►**разреженных**) систем. В локально структурированной системе каждый субкомпонент непосредственно взаимодействует только с ограниченным количеством других компонентов, независимо от их общего количества в системе. Локальная структура обычно ассоциируется с линейным, а не с экспоненциальным ростом сложности.

В случае байесовских сетей резонно предположить, что в большинстве проблемных областей на каждую случайную переменную оказывают непосредствен-

ное влияние самое большее k других переменных, где k — некоторая константа. Если для простоты предположить, что в такой сети представлено n булевых переменных, то количество информации, необходимое для задания каждой таблицы условных вероятностей, будет составлять не больше 2^k чисел, а всю сеть можно будет определить с помощью $2^k \cdot n$ числовых значений. В сравнении с этим соответствующее совместное распределение будет включать 2^n значений. В качестве конкретного примера предположим, что имеется $n = 30$ вершин и каждая из них имеет пять родительских вершин ($k = 5$). В этом случае для представления соответствующей байесовской сети потребуется 960 числовых значений, тогда как для полного совместного распределения — больше миллиарда.

Однако если на каждую переменную будут оказывать непосредственное влияние *все* другие переменные, т.е. эта сеть будет *полносвязной*, то для задания таблиц условных вероятностей потребуется такое же количество информации, как и для задания совместного распределения в табличной форме. По этой причине на практике часто пропускаются такие ребра, для которых существует лишь небольшая зависимость, — небольшой выигрыш в точности не стоит внесения дополнительной сложности в сеть. Например, можно критиковать предложенную выше структуру сети отслеживания взлома на том основании, что если бы имело место сильное землетрясение, то Джон и Мэри не позвонили бы хозяину дома, даже услышав сигнал тревоги, поскольку сочли бы, что его причиной стало землетрясение. Добавление ребер от вершины *Earthquake* к вершинам *JohnCalls* и *MaryCalls* (с соответствующим увеличением размеров таблиц CPT) зависит от важности получения более точных значений вероятностей в сравнении с издержками на обработку дополнительной информации.

Даже в локально структурированной проблемной области компактную байесовскую сеть можно будет получить только в том случае, если будет выбран правильный порядок узлов. А что произойдет, если выбранный порядок окажется неправильным? Еще раз вернемся к примеру со взломом. Предположим, что мы решили вводить в сеть вершины в порядке *MaryCalls*, *JohnCalls*, *Alarm*, *Burglary*, *Earthquake*. В таком случае будет получена немного более сложная сеть, показанная на рис. 13.3, *а*. При этом процесс введения вершин будет проходить следующим образом.

- Добавляем вершину *MaryCalls* — родительские вершины отсутствуют.
- Добавляем вершину *JohnCalls*. Если звонит Мэри, это, вероятно, означает, что раздался сигнал тревоги, а вероятность этого события, очевидно, будет выше, если позвонит также и Джон. Поэтому для вершины *JohnCalls* в качестве родительской следует использовать вершину *MaryCalls*.
- Добавляем вершину *Alarm*. Очевидно, что если позвонили оба соседа, вероятность того, что раздался сигнал тревоги, будет больше, чем только при одном звонке или вообще без звонков. Следовательно, для вершины *Alarm* в качестве родительских следует указать обе вершины, *MaryCalls* и *JohnCalls*.

- Добавляем вершину *Burglary*. Если известно состояние сигнала тревоги, то звонок от Джона или Мэри может дать владельцу дома лишь информацию о том, что звонил его телефон или Мэри слушала музыку, но не о взломе:

$$\mathbf{P}(Burglary \mid Alarm, JohnCalls, MaryCalls) = \mathbf{P}(Burglary \mid Alarm).$$

  Следовательно, использовать в качестве родительской следует только вершину *Alarm*.
- Добавляем вершину *Earthquake*. Если раздался сигнал тревоги, то, скорее всего, произошло землетрясение. (Этот тип охранной сигнализации можно считать своего рода детектором землетрясений.) Но если известно, что имел место взлом, то это объясняет появление сигнала тревоги, а вероятность землетрясения должна быть лишь немного выше нормальной. Значит, родительскими вершинами для этой вершины следует выбрать и *Alarm*, и *Burglary*.

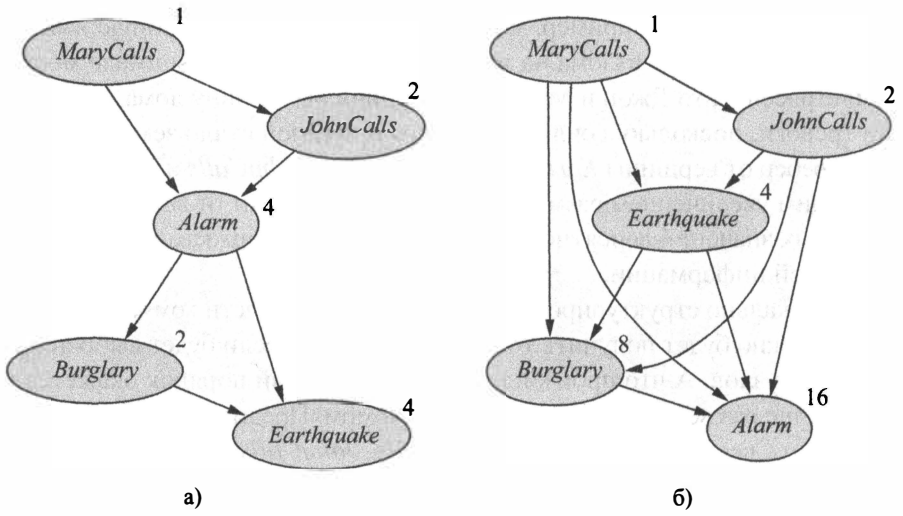


Рис. 13.3. Структура сети и количество параметров зависят от порядка введения вершин. **а)** Эта структура была получена при упорядочении вершин *M*, *J*, *A*, *B*, *E*. **б)** Данная структура была получена при упорядочении вершин *M*, *J*, *E*, *B*, *A*. Также на рисунке для каждой вершины указано количество требуемых параметров: в сумме их требуется 13 для варианта *а* и 31 — для варианта *б*. Для сети, представленной на рис. 13.2 необходимо только 10 параметров

Полученная сеть содержит на два ребра больше по сравнению с исходным вариантом сети, представленным на рис. 13.2, а также требует указания трех дополнительных вероятностей. Но что еще хуже, некоторые из ее ребер представляют надуманные отношения, требующие формирования сложных и неестественных

суждений о вероятностях, таких как оценки вероятности *Earthquake*, если даны *Burglary* и *Alarm*. Такой феномен является достаточно общим и связан с различием между **причинно-следственными** и **диагностическими** моделями, представленными в разделе 12.5.1 (см. также упражнение 13.5). ➨ *Если придерживаться причинно-следственной модели, то в конечном итоге потребуется задать меньше числовых значений, а сами эти значения определить, скорее всего, будет проще.* Например, для проблемной области медицинской диагностики Тверски и Канеман ([2239], 1982) показали, что опытные врачи предпочитают составлять вероятностные суждения для причинных, а не диагностических правил. В разделе 13.5 понятие причинно-следственных моделей исследуется более подробно.

На рис. 13.3, *б* показан совсем плохой вариант упорядочения вершин: *MaryCalls*, *JohnCalls*, *Earthquake*, *Burglary*, *Alarm*. Для этой сети требуется задать 31 отдельную вероятность — точно такое же количество, как и при использовании полного совместного распределения. Однако очень важно понимать, что любая из этих трех сетей может представлять *в точности одно и то же совместное распределение*. Просто в двух вариантах, приведенных на рис. 13.3, не удалось представить все отношения условной независимости и поэтому вместо них потребовалось ввести много ненужных числовых значений.

13.2.1. Отношения условной независимости в байесовских сетях

Из семантики байесовских сетей, как она была определена в уравнении (13.2), можно вывести ряд свойств условной независимости. Нам уже знакомо свойство, что переменная является условно независимой от своих предшественников при заданных родительских переменных. Также можно доказать более общее свойство "не потомков", которое формулируется следующим образом:

> *каждая переменная является условно независимой от переменных, не являющихся ее* ▶ ***потомками***, *при заданных родительских переменных.*

Например, на рис. 13.2 переменная *JohnCalls* независима от переменных *Burglary* и *Earthquake* и *MaryCalls*, если дано значение *Alarm*. Приведенное выше определение проиллюстрировано на рис. 13.4, *а*.

Оказывается, что свойства не-потомков в сочетании с интерпретацией сетевых параметров $\theta(X_i \mid Parents(X_i))$ как условных вероятностей $\mathbf{P}(X_i \mid Parents(X_i))$ вполне достаточно для восстановления полного совместного распределения, данного в уравнении (13.2). Другими словами, можно понимать семантику байесовских сетей иначе: вместо представления полного совместного распределения как произведения условных распределений сеть определяет множество условно независимых свойств. Полное совместное распределение может быть получено из этих свойств.

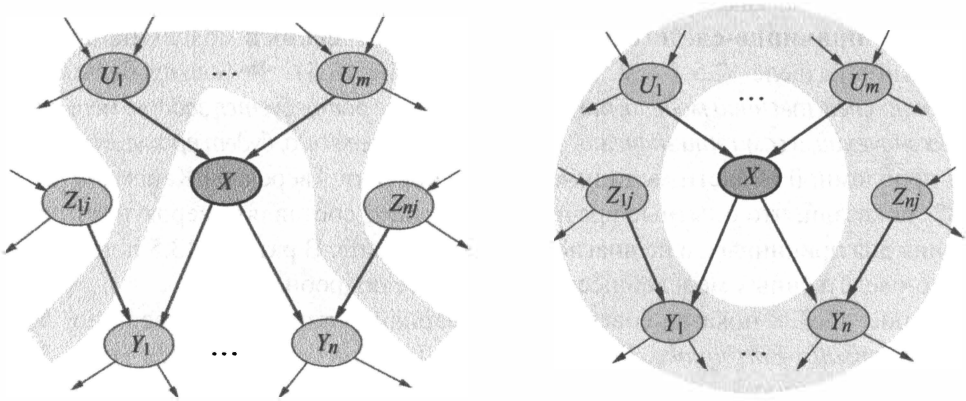


Рис. 13.4. а) Вершина X является условно независимой от вершин, не являющихся ее потомками (например, от вершин Z_{ij}), если даны ее родительские вершины (вершины U_i, находящиеся на сером фоне). **б)** Вершина X является условно независимой от всех других вершин в сети, если дано ее марковское покрытие (область с серым фоном)

Другое важное свойство независимости подразумевается из свойства не-потомков:

переменная является условно независимой от всех других вершин в сети, если даны ее родительские переменные, дочерние переменные и родительские переменные дочерних переменных, т.е. дано ее ►***марковское покрытие*** (*Markov blanket*).

(В упражнении 13.8 предлагается это доказать.) Например, переменная *Burglary* независима от переменных *JohnCalls* и *MaryCalls*, если даны *Alarm* и *Earthquake*. Это свойство проиллюстрировано на рис. 13.4, *б*. Свойство марковского покрытия делает возможным построение алгоритмов вероятностного вывода, использующих полностью локальные и распределенные стохастические процессы выборки, как объясняется в разделе 13.4.2.

Наиболее общий вопрос в отношении условной независимости, на который можно получить ответ в байесовской сети, следующий: является ли множество узлов **X** условно независимым от другого множества **Y** при заданном третьем множестве **Z**. Ответ можно достаточно просто найти за счет анализа байесовской сети с целью определения, обеспечивает ли **Z** ►**d-разделение** для **X** и **Y**. Процедура выполняется следующим образом.

1. В общем графе рассмотрим только ►**подграф предшествования**, включающий вершины из **X**, **Y**, **Z** и их предков.

2. Добавим ребра между любой парой несвязанных узлов, имеющих общий дочерний узел, и в результате получим так называемый ►**моральный граф**.
3. Заменим все направленные ребра ненаправленными ребрами.
4. Если в полученном графе вершины **Z** блокируют все пути между **X** и **Y**, то **Z** d-разделяет множества **X** и **Y**. В этом случае **X** условно не зависит от **Y** при заданном **Z**. В противном случае исходная байесовская сеть не требует условной независимости.

Если говорить кратко, то d-разделение означает разделение вершин в неориентированном, морализованном подграфе предшествования. Применяя это определение к байесовской сети, представленной на рис. 13.2, можно сделать вывод, что переменные *Burglary* и *Earthquake* являются независимыми при пустом заданном множестве (т.е. они абсолютно независимы), что они *необязательно* являются условно независимыми при заданном значении переменной *Alarm* и что переменные *JohnCalls* и *MaryCalls* являются условно независимыми при заданном значении *Alarm*. Также обратите внимание, что свойство марковского покрытия следует непосредственно из свойства d-разделения, поскольку марковское покрытие переменной d-отделяет ее от всех других переменных.

13.2.2. Эффективное представление условных распределений

Даже если максимальное количество родительских вершин k будет невелико, для заполнения таблицы CPT любой вершины потребуется задать вплоть до $O(2^k)$ числовых значений, а также, возможно, потребуется значительный объем опытных данных оценки всех возможных обусловливающих случаев. К тому же на практике иногда встречается наихудшая ситуация, в которой связь между родительскими вершинами и дочерней вершиной является полностью произвольной. Обычно такие отношения можно описать с помощью ►**канонического распределения**, которое соответствует некоторому стандартному образцу. В таких случаях полную таблицу можно определить, указав тип распределения и, возможно, введя несколько параметров.

Простейшим примером является наличие ►**детерминированных вершин**. Детерминированная вершина имеет значение, точно определяемое значениями ее родительских вершин, без какой-либо неопределенности. Отношение между вершинами может быть логическим, например отношение между родительскими вершинами *Canadian*, *US*, *Mexican* и дочерней вершиной *NorthAmerican*, где значение дочерней вершины представляет собой дизъюнкцию значений родительских вершин. Отношение также может быть числовым, например переменная *BestPrice* для автомобиля может представлять ее минимальную цену у каждого дилера в регионе, тогда как значением переменной *WaterStored* для водоема на конец года будет разность между всеми поступлениями в него воды (включая исходное значение на начало года и приток от впадающих рек, береговых стоков,

осадков) и всего расхода (включая сток через вытекающие реки, испарение, просачивание в почву).

Многие системы реализации байесовских сетей предоставляют пользователю возможность задать детерминированные функции, воспользовавшись языком программирования общего назначения. Такой подход позволяет включать в вероятностную модель сложные элементы, например модель глобального климата или эмулятор силовых электросетей.

Другой важный случай, который часто встречается на практике, — это ▶**контекстно специфическая независимость** (CSI — *context-specific independence*). Условное распределение проявляет контекстно специфическую независимость, если некоторая переменная оказывается условно независимой от некоторых из своих родительских переменных при *определенных значениях* других. Например, предположим, что для автомобиля переменная *Damage* (*повреждение*) зависит от переменной *Ruggedness* (*прочность*) этого автомобиля и наличия случаев *Accident* (*авария*). Понятно, что если переменная *Accident* имеет значение *false*, то переменная *Damage*, отмечающая повреждения, если таковые имеются, не зависит от переменной *Ruggedness*. (Может иметь место повреждение лакокрасочного покрытия или окон автомобиля по причине вандализма, но мы полагаем, что все автомобили в одинаковой степени страдают от повреждений такого рода.) Можно сказать, что переменная *Damage* является контекстно специфически независимой от переменной *Ruggedness* при значении переменной $Accident = false$. В системах построения байесовских сетей отношения CSI обычно реализуются с использованием синтаксиса `if-then-else` для определения условных распределений, например можно написать следующее:

$$\mathbf{P}(Damage \mid Ruggedness, Accident) = \\ = \textbf{if } (Accident = false) \textbf{ then } d_1 \textbf{ else } d_2(Ruggedness),$$

где d_1 и d_2 представляют произвольные распределения. Как и в случае детерминизма, присутствие в сети отношений CSI может способствовать повышению эффективности вероятностного вывода. Все точные алгоритмы вывода, упоминаемые в разделе 13.3, легко могут быть изменены с целью использования преимуществ отношений CSI для ускорения вычислений.

Неопределенные отношения можно также часто охарактеризовать с использованием так называемых **зашумленных** логических отношений. Стандартным примером является отношение ▶**зашумленного OR**, представляющего собой обобщение логического отношения OR. В логике высказываний можно сделать утверждение, что высказывание *Fever* (*жар, лихорадка*) является истинным тогда и только тогда, когда истинны высказывания *Cold* (*простуда*), *Flu* (*грипп*) и *Malaria* (*малярия*). Модель зашумленного OR позволяет учитывать неопределенность знаний в отношении способности каждой из родительских вершин вызывать присваивание истинного значения дочерней вершине, поскольку причинная связь

между родительской и дочерней вершинами может быть *заблокирована* и поэтому иногда пациент бывает простужен, но жар у него отсутствует.

В этой модели приняты два допущения. Во-первых, предполагается, что учтены все возможные причины. (Если некоторые пропущены, всегда можно добавить так называемую ▶**вершину утечки** (*leak node*), покрывающую все "прочие причины".) Во-вторых, предполагается, что блокирование каждой родительской вершины не зависит от блокирования любых других родительских вершин, например та причина, которая блокирует появление жара под действием вершины *Malaria*, не зависит от причин, блокирующих появление жара под действием вершины *Flu*. С учетом этих предположений вершина *Fever* будет иметь значение *false* тогда и только тогда, когда будут заблокированы все ее родительские вершины, имеющие значение *true*. Вероятность такой ситуации представляет собой произведение вероятностей блокирования каждой родительской вершины. Предположим, что такие отдельные вероятности блокирования выражаются следующим образом:

$$q_{cold} = P(\neg fever \mid cold, \neg flu, \neg malaria) = 0{,}6;$$
$$q_{flu} = P(\neg fever \mid \neg cold, flu, \neg malaria) = 0{,}2;$$
$$q_{malaria} = P(\neg fever \mid \neg cold, \neg flu, malaria) = 0{,}1.$$

Затем, исходя из этой информации и допущений зашумленного OR, можно будет построить всю таблицу условных вероятностей (CPT). Общее правило таково, что

$$P(x_i \mid parents(X_i)) = 1 - \prod_{\{j:X_j = true\}} q_j,$$

где перемножаются те родительские вершины, которые принимают значение *true* для данной строки CPT. Пример подобных вычислений приведен на рис. 13.5.

Cold	*Flu*	*Malaria*	$P(fever \mid \cdot)$	$P(\neg fever \mid \cdot)$
f	*f*	*f*	0,0	1,0
f	*f*	*t*	0,9	**0,1**
f	*t*	*f*	0,8	**0,2**
f	*t*	*t*	0,98	0,02 = 0,2 × 0,1
t	*f*	*f*	0,4	**0,6**
t	*f*	*t*	0,94	0,06 = 0,6 × 0,1
t	*t*	*f*	0,88	0,12 = 0,6 × 0,2
t	*t*	*t*	0,988	0,012 = 0,6 × 0,2 × 0,1

Рис. 13.5. Полная таблица условных вероятностей для **P**(*Fever* | *Cold*, *Flu*, *Malaria*) при условии, что модель включает отношения зашумленного OR для трех *q*-значений, выделенных полужирным шрифтом

В общем случае зашумленные логические отношения, в которых некоторая переменная зависит от k родительских переменных, могут быть описаны с использованием $O(k)$ параметров вместо $O(2^k)$ параметров, необходимых для определения полной таблицы условных вероятностей. В результате задача присваивания значений или обучения намного упрощается. Например, в сети CPCS (Прадхан и др. [1819], 1994) распределения зашумленного OR и зашумленного MAX успешно использовались для моделирования отношений между заболеваниями и симптомами в диагностике внутренних органов. При наличии 448 вершин и 906 ребер в этой сети потребовалось определить всего лишь 8254 значения вместо 133 931 430 значений для случая с полными таблицами CPT.

13.2.3. Байесовские сети с непрерывными переменными

Многие реальные задачи включают непрерывные величины, такие как высота, масса, температура или денежная сумма. По определению непрерывные величины имеют бесконечное количество возможных значений, поэтому невозможно явно задать условные вероятности для каждого значения. Один из возможных способов обработки непрерывных переменных состоит в избавлении от них с помощью ►**дискретизации**, т.е. распределения всех возможных значений в фиксированное множество интервалов. Например, значения температуры могут быть разделены на три интервала: (< 0°C), (0°C – 100°C) и (> 100°C). Выбор количества категорий предполагает компромисс между потерей точности и использованием больших таблиц CPT, что ведет к увеличению времени выполнения расчетов.

Другой подход заключается в определении непрерывной переменной с использованием одного из стандартных семейств функций распределения плотности вероятности (см. приложение А). Например, гауссово (или нормальное) распределение $\mathcal{N}(x, \mu, \sigma^2)$ задается только двумя параметрами: средним значением μ и дисперсией σ^2. Еще одно возможное решение — иногда его также называют ►**непараметрическим** представлением — заключается в определении условного распределения неявным образом, с использованием набора экземпляров, каждый из которых включает конкретные значения родительских и дочерних переменных. Подробнее этот подход будет рассмотрен позже, в главе 19.

Сеть, в которой имеются и дискретные, и непрерывные переменные, называется ►**гибридной байесовской сетью**. Для описания гибридной сети необходимо определить два новых типа распределений: условное распределение для непрерывной переменной, при заданных дискретных или непрерывных родительских переменных, и условное распределение для дискретной переменной, при заданных непрерывных родительских переменных. Рассмотрим простой пример, приведенный на рис. 13.6, в котором клиент покупает те или иные фрукты в зависимости от их стоимости, которая, в свою очередь, зависит от размера урожая (переменная *Harvest*) и от того, применяется ли правительственная программа субсидий (переменная *Subsidy*). Переменная *Cost* (*стоимость*) является непрерывной и имеет

непрерывные и дискретные родительские переменные; переменная *Buys* (*покупает*) является дискретной и имеет непрерывную родительскую переменную.

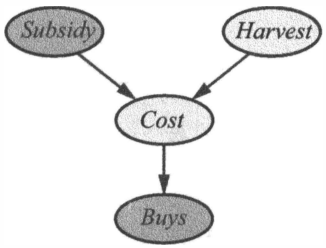


Рис. 13.6. Простая сеть с дискретными переменными (*Subsidy* и *Buys*) и непрерывными переменными (*Harvest* и *Cost*)

Для переменной *Cost* необходимо задать распределение **P**(*Cost* | *Harvest*, *Subsidy*). Это дискретное родительское значение учитывается с помощью явного перечисления, т.е. определения как значения **P**(*Cost* | *Harvest*, *subsidy*), так и значения **P**(*Cost* | *Harvest*, ¬*subsidy*). Чтобы работать с переменной *Harvest*, необходимо определить, как распределение по стоимости *c* зависит от значения *h* непрерывной переменной *Harvest*. Иными словами, требуется определить *параметры* распределения стоимости как функции от *h*. Чаще всего применяемым вариантом является ▶ **линейное гауссово** условное распределение, в котором дочерняя переменная имеет гауссово распределение со средним μ, изменяющимся линейно в зависимости от значения родительской переменной, и с постоянным среднеквадратичным отклонением σ. Необходимо иметь два распределения: одно — для значения *subsidy* и одно — для значения ¬*subsidy*, с разными параметрами:

$$P(c \mid h, subsidy) = \mathcal{N}(c; a_t h + b_t, \sigma_t^2) = \frac{1}{\sigma_t\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{c-(a_t h+b_t)}{\sigma_t}\right)^2},$$

$$P(c \mid h, \neg subsidy) = \mathcal{N}(c; a_f h + b_f, \sigma_f^2) = \frac{1}{\sigma_f\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{c-(a_f h+b_f)}{\sigma_f}\right)^2}.$$

Таким образом, в данном примере условное распределение для переменной *Cost* было задано посредством обозначения его как линейного гауссова распределения и предоставления параметров a_t, b_t, σ_t и a_f, b_f, σ_f. На рис. 13.7, *а* и *б* представлены оба эти отношения; обратите внимание, что в каждом случае наклон кривой *c* относительно *h* является отрицательным, поскольку цена уменьшается по мере увеличения урожая. (Безусловно, из предположения о линейности следует, что цена в некоторый момент станет отрицательной; линейная модель является приемлемой, только если размер урожая ограничен каким-то узким диапазоном.)

На рис. 13.7, *в* показано распределение $P(c \mid h)$, усредненное по двум возможным значениям переменной *Subsidy* в предположении, что каждое из двух ее возможных значений имеет априорную вероятность 0,5. Этот пример показывает, что с помощью даже очень простых моделей вполне возможно представить довольно интересные распределения.

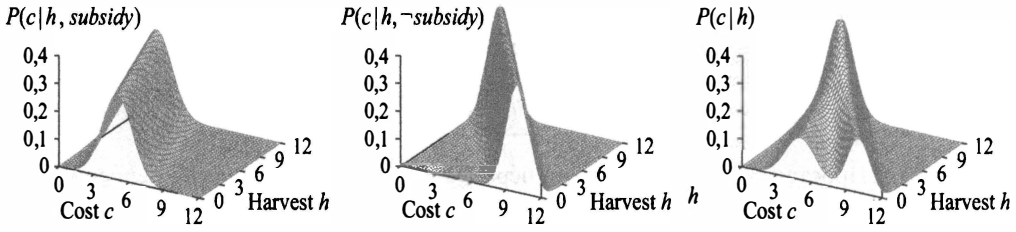


Рис. 13.7. а) и б) На графиках показаны распределения вероятностей для значения стоимости *Cost* как функции от размера урожая *Harvest* при условии, что переменная *Subsidy* имеет соответственно истинное и ложное значения. **в)** На графике показано распределение $P(Cost \mid Harvest)$, полученное путем суммирования по двум случаям, связанным с предоставлением и не предоставлением субсидии

Линейное гауссово условное распределение обладает некоторыми особыми свойствами. Сеть, содержащая только непрерывные переменные с линейными гауссовыми распределениями, имеет совместное распределение, представляющее собой многомерное гауссово распределение (см. приложение А) по всем переменным (упражнение 13.11). Более того, апостериорное распределение при любом свидетельстве также обладает этим свойством.[2] Когда дискретные переменные добавляются в сеть в качестве родительских (но не дочерних) для непрерывных переменных, то сеть определяет ▶**условное гауссово распределение**, или распределение CG (*conditional Gaussian*): если дано любое присваивание дискретным переменным, то распределение по непрерывным переменным является многомерным гауссовым распределением.

Теперь обратимся к распределениям для дискретных переменных с непрерывными родительскими переменными. Например, рассмотрим вершину *Buys* на рис. 13.6. Представляется обоснованным предположение, что клиент сделает покупку, если цена низкая, и не сделает покупку, если она высокая, а также, что вероятность покупки плавно изменяется в некоторой промежуточной области.

[2] Из этого следует, что вероятностный вывод в линейных гауссовых сетях в наихудшем случае требует $O(n^3)$ времени независимо от топологии сети. В разделе 13.3 будет показано, что вероятностный вывод в сетях с дискретными переменными является NP-трудным.

Другими словами, условное распределение будет напоминать "мягкую" пороговую функцию. Один из способов определения мягких пороговых функций состоит в использовании *интеграла* стандартного нормального распределения:

$$\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(s; 0, 1) ds.$$

Здесь $\Phi(x)$ является возрастающей функцией от x, тогда как вероятность покупки уменьшается с увеличением цены, т.е. эту функцию нужно перевернуть:

$$P(buys \mid Cost = c) = 1 - \Phi((c - \mu)/\sigma),$$

а это означает, что пороговое значение цены находится в районе μ и ширина этой области пропорциональна σ, а вероятность покупки уменьшается по мере возрастания цены. Такое распределение вероятностей называется ▶**пробит-распределением** (*probit* — сокращение от *probability unit*) и показано на рис. 13.8, *а*. Применение распределения с такой формой можно обосновать тем, что положенный в основу процесс принятия решения предполагает наличие четкого порога, но точное местонахождение этого порогового значения подвержено воздействию случайного гауссова шума.

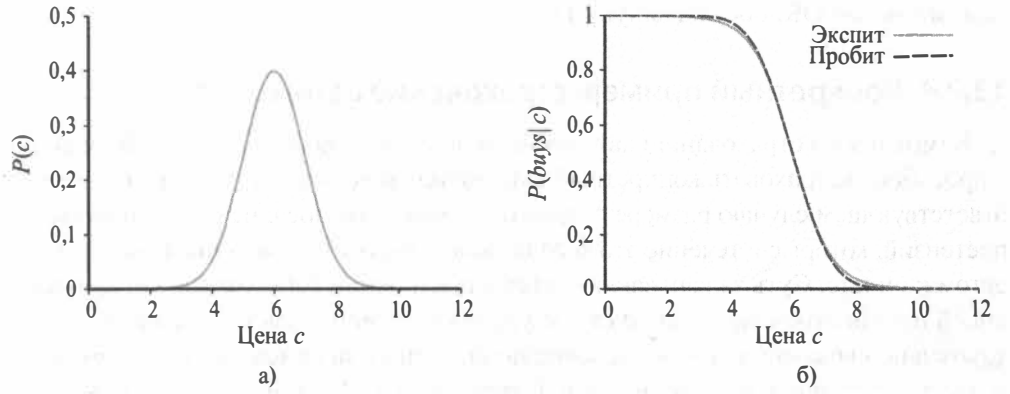


Рис. 13.8. а) Нормальное (гауссово) распределение для порога цены с центром в $\mu = 6{,}0$ и стандартным отклонением $\sigma = 1{,}0$. **б)** Экспит- и пробит-модели для вероятности покупок *buys* при заданной цене *cost* и значениях параметров $\mu = 6{,}0$ и $\sigma = 1{,}0$

Альтернативным по отношению к пробит-распределению является ▶**экспит-распределение** (*expit distribution*) или ▶**инверсное логит-распределение** (*inverse logit distribution*), в котором для формирования мягкого порога используется логистическая (сигмоидальная) функция $1/(1 + e^{-x})$, — она отображает любое значение x на некоторое значение между 0 и 1. И вновь, для нашего примера ее необходимо перевернуть, чтобы получить убывающую, а не возрастающую

функцию. Также дополнительно масштабируем показатель степени на $4/\sqrt{2\pi}$, чтобы обеспечить соответствие наклону кривой пробит-модели в среднем значении:

$$P(buys \mid Cost = c) = 1 - \frac{1}{1 + exp\left(-\frac{4}{\sqrt{2\pi}} \cdot \frac{c-\mu}{\sigma}\right)}.$$

Это распределение показано на рис. 13.8, *б*. Два распределения внешне выглядят одинаковыми, но в действительности экспит-распределение имеет гораздо более длинные "хвосты". Пробит-распределения часто лучше подходят в реальных ситуациях, а экспит-распределения иногда проще поддаются математической обработке. В настоящее время последние очень широко используются в машинном обучении. Обе модели могут быть обобщены для учета многих непрерывных родительских значений посредством линейной комбинации этих родительских значений. Такой подход также работает для дискретных родительских переменных, если их значения целочисленные. Например, при k булевых родительских переменных, каждая из которых рассматривается как имеющая значение 0 или 1, входным значением для экспит- или пробит-распределения может быть взвешенная линейная комбинация с k параметрами, что дает модель, очень похожую на модель зашумленного OR, обсуждавшуюся выше.

13.2.4. Конкретный пример: страхование автомобиля

Компания по страхованию автомобилей получает заявку от владельца машины с просьбой застраховать конкретный автомобиль и должна принять решение о соответствующем случаю размере годового страхового взноса, исходя из ожидаемых претензий, которые в течение этого срока могут быть предъявлены данным кандидатом к оплате. Суть задачи заключается в построении байесовской сети, отражающей причинно-следственную структуру проблемной области и дающей точное, тщательно выверенное распределение по выходным переменным с учетом свидетельств, доступных из заполненной формы заявки.[3] Сеть Байеса будет включать ►**скрытые переменные**, которые не являются ни входными, ни выходными, но необходимы для структурирования сети, обеспечивая ее разумную разреженность с управляемым количеством параметров. На рис. 13.9 скрытые переменные выделены темно-серым цветом.

[3] Сеть, показанная на рис. 13.9, не предназначена для практического использования, но ее общая структура была одобрена страховыми экспертами. На практике информация, запрашиваемая в формах заявки, варьирует в зависимости от компании и юридических особенностей локации — например, иногда имеет место вопрос о поле клиента, — и эта модель, безусловно, может быть сделана более подробной и сложной.

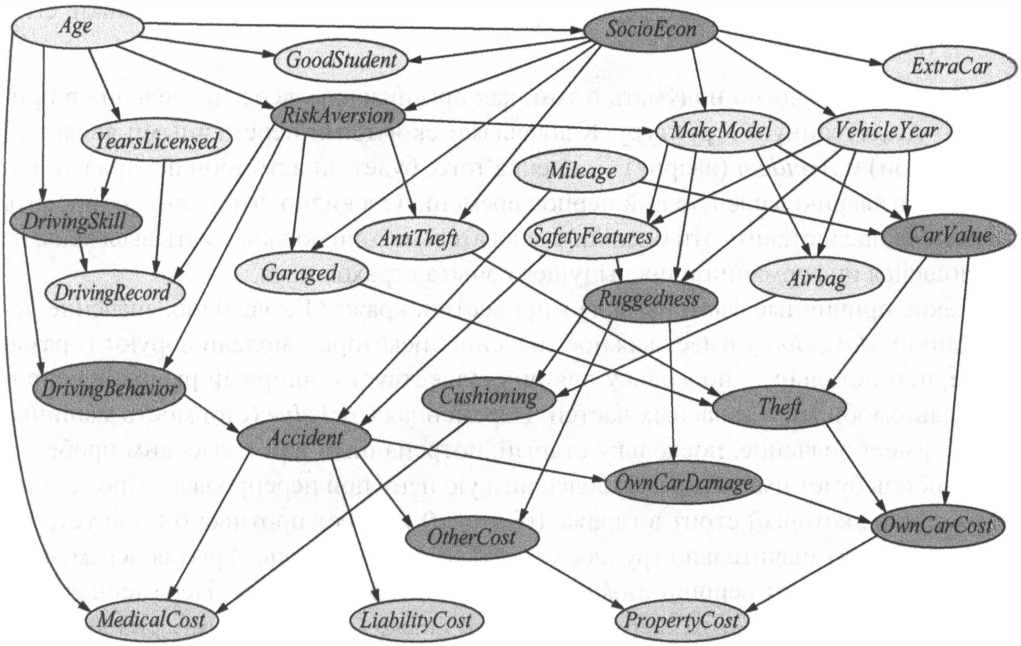


Рис. 13.9. Байесовская сеть для оценки заявок на страхование автомобиля

Требования, которые могут быть предъявлены к оплате, бывают трех видов (нижняя строка на рис. 13.9): *MedicalCost* (медицинские издержки) в случае любых травм, которые мог получить заявитель; *LiabilityCost* (стоимость претензий) для покрытия исков, предъявленных другими сторонами к заявителю и самой компании; и *PropertyCost* (стоимость имущества) для компенсации ущерба, нанесенного транспортному средству любой из сторон, или его утраты в результате хищения. В форму заявки требуется внести следующую входную информацию (на рис. 13.9 эти узлы выделены светло-серым цветом).

- **Информация о заявителе.** Вершины *Age* (возраст); *YearsLicensed* (дата выдачи прав) — сколько времени прошло с момента получения водительских прав; *DrivingRecord* (сведения о вождении) — краткое изложение истории вождения, возможно, построенной на "очках" за недавние аварии и нарушения правил дорожного движения; и (для студентов) *GoodStudent* (хороший студент) — показатель среднего балла 3,0 (*B*) по 4-балльной шкале.
- **Информация об автомобиле.** Вершины *MakeModel* (модель) и *VehicleYear* (год выпуска); имеется ли *Airbag* (подушка безопасности); а также некоторые общие сведения *SafetyFeatures* (функции безопасности), такие как наличие антиблокировочной системы экстренного торможения или системы предупреждения о столкновении.

- **Показатели эксплуатации**. Вершины *Mileage* (пробег) — годовой пробег, *Garaged* (гараж) — степень защищенности места постоянной стоянки, если таковое имеется.

Теперь необходимо подумать о том, как организовать все эти сведения в причинно-следственную структуру. Ключевыми скрытыми переменными являются *Theft* (угон) и *Accident* (авария) — оценка того, будет ли автомобиль украден или попадет в аварию в следующий период времени. Очевидно, что нельзя попросить заявителя предоставить эти оценки; следовательно, они должны быть выведены из имеющейся информации и предыдущего опыта страховщика.

Какие причинные факторы могут привести к краже? Безусловно, значение переменной *MakeModel* имеет важное значение: некоторые модели воруют гораздо чаще, чем остальные, поскольку для них существует обширный рынок перепродажи автомобилей и запасных частей. Переменная *CarValue* (стоимость машины) также имеет значение, поскольку старый, потрепанный или с высоким пробегом автомобиль будет иметь гораздо более низкую цену при перепродаже. Кроме того, автомобиль, который стоит в гараже (*Garaged*) и имеет противоугонное устройство (*AntiTheft*) значительно труднее украсть. В свою очередь, скрытая переменная *CarValue* зависит от вершин *MakeModel*, *VehicleYear* и *Mileage*. Переменная *CarValue* также определяет размер убытков при краже (*Theft*), поэтому она является одной из родительских вершин переменной *OwnCarCost* (собственная стоимость машины) (при возникновении иных ситуаций, о которых вскоре пойдет речь).

Обычно в моделях такого типа вводится еще одна иная скрытая переменная, *SocioEcon* (общественная категория), определяющая социально-экономическую категорию заявителя. Считается, что это значение оказывает влияние на широкий спектр особенностей его поведения и других характеристик. В нашей модели нет никаких *прямых* свидетельств в виде переменных с данными о заявленных доходах или занимаемой должности;[4] но вершина *SocioEcon* является родительской для переменных *MakeModel* и *VehicleYear*, а также влияет на переменные *ExtraCar* и *GoodStudent*, но при этом в нашей модели зависит только от переменной *Age* (возраст).

Для любой страховой компании наиболее важной скрытой переменной, вероятно, является *RiskAversion* (неприятие риска): люди, не склонные к риску, очень привлекательны с точки зрения страховых рисков! Для переменной *RiskAversion* родительскими являются вершины *Age* и *SocioEcon* и их "симптомы" включают

[4] Некоторые страховые компании также приобретают кредитную историю заявителя, полагая, что это может помочь в оценке риска, поскольку дает значительно больше информации о его социально-экономической категории. Всякий раз при использовании скрытых переменных подобного рода нужно проявлять осторожность в отношении того, чтобы они случайно не оказались скрытыми проводниками для таких характеристик, как раса, которые не могут использоваться при принятии страховых решений. Методы предотвращения подобных ошибок описаны в главе 19.

выбор заявителя в отношении того, будет ли автомобиль *Garaged* и будут ли на нем установлены средства *AntiTheft* и *SafetyFeatures*.

В прогнозировании будущих аварий ключевым фактором является оценка будущих значений переменной *DrivingBehavior* (особенности вождения), для которой родительскими вершинами являются *RiskAversion* и *DrivingSkill* (водительские навыки), а последняя, в свою очередь, зависит от вершин *Age* и *YearsLicensed*. Характеристики прежних особенностей вождения заявителя находят свое отражение в переменной *DrivingRecord*, которая также зависит от переменных *RiskAversion* и *DrivingSkill*, а также *YearsLicensed* (поскольку у того, кто начал водить машину лишь недавно, пока не было достаточно времени, чтобы накопить показательный перечень аварий и нарушений). В этом же смысле вершина *DrivingRecord* предоставляет свидетельства для переменных *RiskAversion* и *DrivingSkill*, что, в свою очередь, помогает сделать предположения о будущих значениях переменной *DrivingBehavior*.

Можно рассматривать переменную *DrivingBehavior* как оцениваемую в милях склонность заявителя к вождению в стиле, способствующем возникновению аварий. Действительно ли произойдет *Accident* за установленный промежуток времени, зависит также от годового пробега *Mileage* и характеристик *SafetyFeatures* транспортного средства. Если *Accident* будет иметь место, то страховая компания может оказаться вынужденной сделать три вида выплат. Первая, *MedicalCost* в пользу заявителя, в зависимости от переменных *Age* и *Cushioning* (амортизация), значение которых, в свою очередь, зависит от вершины *Ruggedness* (прочность) для данной машины, а также наличия в ней средства безопасности *Airbag*. Вторая выплата, *LiabilityCost* делается в пользу другого водителя как компенсация за боль и страдания и возмещение расходов на лечение, потерь от сокращения доходов и т.п. Третья выплата, *PropertyCost* делается в пользу заявителя и другого водителя, причем для каждого из них размер ее (различным образом) зависит от вершин *Ruggedness* и *CarValue* автомобиля заявителя.

Выше был проиллюстрирован тот тип рассуждений, который связан с разработкой топологии и выбором скрытых переменных в байесовской сети. Далее также потребуется указать диапазоны и условные распределения для каждой переменной. Для диапазонов основное решение часто состоит в том, какой сделать эту переменную, — дискретной или непрерывной. Например, для автомобиля переменная *Ruggedness* может быть непрерывной и изменяться в пределах от 0 до 1, но может быть и дискретной с возможными значениями {*TinCan*, *Normal*, *Tank*} (*консервная банка*, *норма*, *танк*).

Непрерывные переменные обеспечивают большую точность, но при этом делают невозможным точный вывод за исключением нескольких особых случаев. Дискретная переменная с множеством возможных значений может сделать весьма утомительной задачу заполнения значениями обширных таблиц условной вероятности, а также способствовать тому, что точный вероятностный вывод будет более затратным за исключением случая, когда значение этой переменной постоянно

наблюдается. Например, переменная *MakeModel* в реальной системе будет иметь тысячи возможных значений и это приведет к тому, что ее дочерняя вершина *CarValue* будет иметь огромную CPT, которую потребуется заполнить значениями, взятыми из баз данных отрасли. Однако тот факт, что значение переменной *MakeModel* всегда наблюдаемо, исключает возможность усложнения вывода: в действительности всегда наблюдаемые значения всех трех родительских вершин позволяют сразу выбрать единственную, соответствующую случаю строку в CPT вершины *CarValue*.

Условные распределения для данной модели приведены в репозитории кода для этой книги, — читателю предоставляется версия только с дискретными переменными, обеспечивая возможность выполнения точного вывода. На практике многие из переменных модели будут непрерывными, а их условные распределения потребуется определять на основании исторических данных по заявителям и их страховым претензиям. Как обучать модели на основе байесовской сети с использованием реальных данных, обсуждается в главе 20.

И последний вопрос, конечно же, состоит в том, как выполнять вероятностный вывод в сети, чтобы получать прогнозы. Именно его рассмотрением мы и займемся в следующем разделе. Для каждого метода вероятностного вывода, который будет обсуждаться ниже, пример сети страховой компании будет использован для оценки этого метода с точки зрения затрат времени и требований к объему памяти.

13.3. Точный вывод в байесовских сетях

Основной задачей для любой системы вероятностного вывода является вычисление распределения апостериорных вероятностей для множества **переменных запроса**, если дано некоторое наблюдаемое **►событие**; обычно это связано с выполнением некоторого присваивания значений множеству **переменных свидетельства**.[5] Для упрощения представления здесь мы будем рассматривать запросы только с единственной переменной, — все обсуждаемые алгоритмы можно легко расширить для обработки запросов со многими переменными. (Например, можно получить ответ на запрос $\mathbf{P}(U, V \mid \mathbf{e})$ путем перемножения $\mathbf{P}(V \mid \mathbf{e})$ и $\mathbf{P}(U \mid V, \mathbf{e})$.) Мы будем использовать систему обозначений, введенную в главе 12: X обозначает переменную запроса; $\mathbf{E}$ — множество переменных свидетельства $E_1, \ldots, E_m$; $\mathbf{e}$ — конкретное наблюдаемое событие; $\mathbf{Y}$ — скрытые (отличные от переменных свидетельства и запроса) переменные $Y_1, \ldots, Y_\ell$. Таким образом, полным множеством переменных является $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$. Типичный запрос предполагает определение распределения апостериорных вероятностей $\mathbf{P}(X \mid \mathbf{e})$.

[5] Другой широко изучаемой задачей является нахождение **наиболее вероятного объяснения** для некоторого наблюдаемого свидетельства. Эта и другие задачи обсуждаются в разделе "Библиографические и исторические заметки" в конце главы.

В байесовской сети системы защиты от взлома может наблюдаться событие, в котором переменные *JohnCalls* = *true* и *MaryCalls* = *true*. В таком случае можно ввести запрос, скажем, для определения вероятности того, что произошел взлом:

$$\mathbf{P}(Burglary \mid JohnCalls = true, MaryCalls = true) = \langle 0{,}284;\ 0{,}716 \rangle .$$

В этом разделе рассматриваются точные алгоритмы вычисления апостериорных вероятностей, а также оценивается сложность такой задачи. Как оказалось, в общем случае подобные задачи являются неразрешимыми, поэтому в разделе 13.4 рассматриваются методы приближенного вероятностного вывода.

13.3.1. Вероятностный вывод посредством перебора

В главе 12 было показано, что любую условную вероятность можно вычислить, суммируя элементы из полного совместного распределения. Говоря более конкретно, ответ на запрос $\mathbf{P}(X \mid \mathbf{e})$ можно получить с использованием уравнения (12.9), которое еще раз приводится здесь для удобства:

$$\mathbf{P}(X \mid \mathbf{e}) = \alpha \mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y}).$$

Нам уже известно, что любая байесовская сеть дает исчерпывающее представление полного совместного распределения. Если говорить конкретнее, в уравнении (13.2) (см. раздел 13.2) показано, что термы $P(x, \mathbf{e}, \mathbf{y})$ в совместном распределении можно записать в виде произведений условных вероятностей, взятых из сети. Поэтому ➔ *ответ на любой запрос можно найти с помощью байесовской сети, вычисляя суммы произведений условных вероятностей из этой сети.*

Рассмотрим запрос $\mathbf{P}(Burglary \mid JohnCalls = true, MaryCalls = true)$. Для этого запроса скрытыми переменными являются *Earthquake* и *Alarm*. Из уравнения (12.9), используя начальные символы имен переменных для сокращения длины выражений, можно получить

$$\mathbf{P}(B \mid j, m) = \alpha \mathbf{P}(B, j, m) = \alpha \sum_e \sum_a \mathbf{P}(B, j, m, e, a) .$$

Семантика байесовских сетей (уравнение (13.2)) предоставляет возможность записать выражение в терминах значений CPT. Для простоты здесь это сделано только для переменной *Burglary* = *true*:

$$P(b \mid j, m) = \alpha \sum_e \sum_a P(b)P(e)P(a \mid b, e)P(j \mid a)P(m \mid a) . \qquad (13.4)$$

Для вычисления этого выражения необходимо сложить четыре терма, каждый из которых вычисляется путем умножения пяти чисел. В наихудшем случае, когда требуется суммировать почти все переменные, в сумме будет $O(2^n)$ слагаемых, каждое из которых будет представлять собой произведение $O(n)$ значений вероятности. Поэтому простая реализация этого алгоритма будет иметь сложность $O(n2^n)$.

Здесь можно достигнуть упрощения до $O(2^n)$, воспользовавшись преимуществами вложенной структуры вычислений. В символическом представлении для выражений, таких как уравнение (13.4), это означает перемещение сумм как можно дальше внутрь. Поступить так можно потому, что не все множители в произведении вероятностей зависят от всех переменных. Таким образом, мы имеем

$$P(b \mid j, m) = \alpha P(b) \sum_e P(e) \sum_a P(a \mid b, e) P(j \mid a) P(m \mid a) . \qquad (13.5)$$

Это выражение можно вычислить, последовательно обрабатывая в цикле его переменные и перемножая по ходу элементы CPT. При каждом суммировании необходимо также выполнить цикл по возможным значениям переменной. Структура этих вычислений представлена в виде дерева на рис. 13.10. Используя числовые значения, приведенные на рис. 13.2, получим выражение $P(b \mid j, m) = \alpha \times 0{,}00059224$. Соответствующие вычисления для $\neg b$ приводят к выражению $\alpha \times 0{,}0014919$, следовательно:

$$\mathbf{P}(B \mid j, m) = \alpha \langle 0{,}00059224;\ 0{,}0014919 \rangle \approx \langle 0{,}284;\ 0{,}716 \rangle .$$

Таким образом, вероятность взлома, учитывая, что поступили звонки обоих соседей, составляет около 28%.

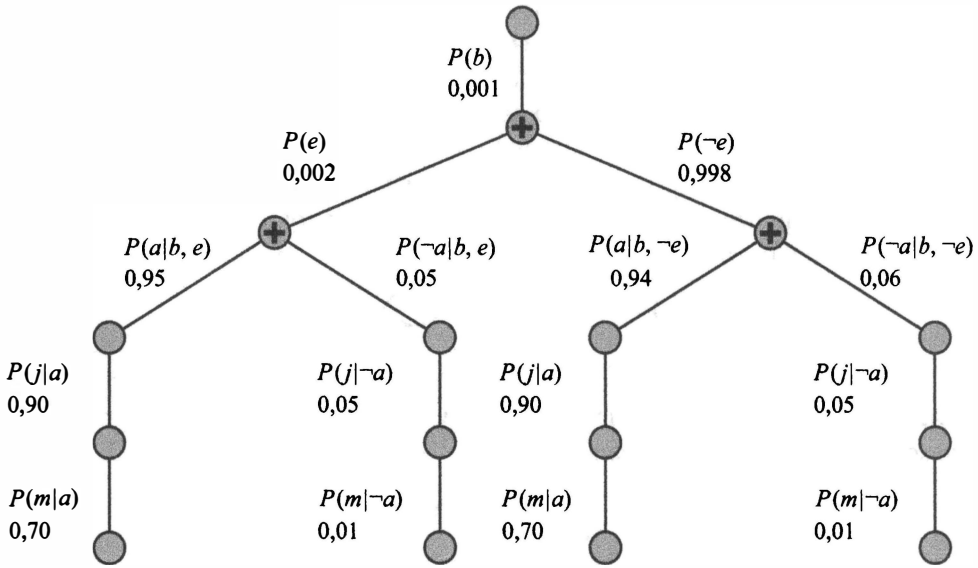


Рис. 13.10. Структура выражения, приведенного в уравнении (13.5). Процесс вычисления осуществляется сверху вниз; при этом значения вдоль каждого пути умножаются и суммируются в узлах, отмеченных знаком "+". Обратите внимание на то, что пути для j и m повторяются

Алгоритм ENUMERATION-ASK, представленный на рис. 13.11, вычисляет эти деревья выражений с использованием рекурсии в глубину, слева направо. По своей структуре этот алгоритм очень похож на алгоритм поиска с возвратами для решения задач УО (рис. 6.5) и на алгоритм DPLL для проверки выполнимости (рис. 7.17). Его пространственная сложность зависит от количества переменных только линейно: по сути, этот алгоритм вычисляет суммы по полному совместному распределению, даже не формируя его явно. К сожалению, временная сложность этого алгоритма для сети с n булевыми переменными (не считая переменных свидетельств) всегда составляет $O(2^n)$, — она лучше по сравнению с оценкой $O(n2^n)$ для простого подхода, описанного выше, но все еще довольно велика. Для сети страховой компании, представленной на рис. 13.9, которую следует считать относительно небольшой, точный вероятностный вывод по методу перебора потребует выполнения около 227 млн арифметических операций при обработке типичного запроса в отношении переменных стоимости.

---

function ENUMERATION-ASK(X, **e**, bn) **returns** распределение по X
  inputs: X, переменная запроса
    e, наблюдаемые значения переменных **E**
    bn, байесовская сеть с переменными *vars*

  $\mathbf{Q}(X) \leftarrow$ распределение по X, первоначально пустое
  for each значение x_i переменной X **do**
    $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL(*vars*, $\mathbf{e}_{x_i}$)
      где $\mathbf{e}_{x_i}$ есть **e**, дополненное значением $X = x_i$
  return NORMALIZE($\mathbf{Q}(X)$)

function ENUMERATE-ALL(*vars*, **e**) **returns** действительное число
  if EMPTY?(*vars*) **then return** 1,0
  $V \leftarrow$ FIRST(*vars*)
  if V является переменной свидетельства со значением v в **e**
    then return $P(v \mid parents(V)) \times$ ENUMERATE-ALL(REST(*vars*), **e**)
    else return $\sum_v P(v \mid parents(V)) \times$ ENUMERATE-ALL(REST(*vars*), $\mathbf{e}_v$)
      где $\mathbf{e}_v$ есть **e**, дополненное значением $V = v$

Рис. 13.11. Алгоритм с перебором для точного вероятностного вывода в байесовских сетях

Однако если внимательно присмотреться к дереву, приведенному на рис. 13.10, то можно заметить, что оно содержит *повторяющиеся подвыражения*. Произведения $P(j \mid a)P(m \mid a)$ и $P(j \mid \neg a)P(m \mid \neg a)$ вычисляются дважды, по одному разу для каждого значения E. Ключ к эффективному вероятностному выводу в байесовских сетях кроется в исключении подобных избыточных вычислений. В следующем разделе описан общий метод, позволяющий этого добиться.

13.3.2. Алгоритм устранения переменной

Алгоритм перебора можно существенно улучшить, исключив повторные вычисления такого типа, как показано на рис. 13.10. Сама идея очень проста: выполнить расчет один раз и сохранить результаты для дальнейшего использования. Это одна из форм динамического программирования. Существует несколько версий подобного подхода; в данной главе будет представлен алгоритм ▶**устранения переменной** (*variable elimination*), который является самым простым. Устранение переменной осуществляется посредством вычисления выражений, подобных представленному в уравнении (13.5), в порядке *справа налево* (т.е. *сверху вниз*; см. рис. 13.10). Промежуточные результаты сохраняются, и операции суммирования по каждой переменной выполняются только для тех частей выражения, которые зависят от этой переменной.

Проиллюстрируем этот процесс на примере сети системы защиты от взлома. В этом случае необходимо вычислить выражение

$$\mathbf{P}(B \mid j, m) = \alpha \underbrace{\mathbf{P}(B)}_{\mathbf{f}_1(B)} \sum_e \underbrace{P(e)}_{\mathbf{f}_2(E)} \sum_a \underbrace{\mathbf{P}(a \mid B, e)}_{\mathbf{f}_3(A,B,E)} \underbrace{P(j \mid a)}_{\mathbf{f}_4(A)} \underbrace{P(m \mid a)}_{\mathbf{f}_5(A)}.$$

Обратите внимание, что каждая часть этого выражения была помечена именем соответствующего ▶**фактора**. Каждый фактор представляет собой матрицу, проиндексированную значениями ее переменных-аргументов. Например, факторы $\mathbf{f}_4(A)$ и $\mathbf{f}_5(A)$, соответствующие распределениям $P(j \mid a)$ и $P(m \mid a)$, зависят только от A, поскольку J и M фиксированы по запросу. Поэтому эти факторы являются двухэлементными векторами

$$\mathbf{f}_4(A) = \begin{pmatrix} P(j \mid a) \\ P(j \mid \neg a) \end{pmatrix} = \begin{pmatrix} 0{,}90 \\ 0{,}05 \end{pmatrix} \qquad \mathbf{f}_5(A) = \begin{pmatrix} P(m \mid a) \\ P(m \mid \neg a) \end{pmatrix} = \begin{pmatrix} 0{,}70 \\ 0{,}01 \end{pmatrix}.$$

Фактор $\mathbf{f}_3(A, B, E)$ представляет собой трехмерную матрицу $2 \times 2 \times 2$, которую сложно представить на странице книги. (Здесь "первый" элемент задается распределением $P(a \mid b, e) = 0{,}95$, а "последний" — распределением $P(\neg a \mid \neg b, \neg e) = 0{,}999$.) В терминах факторов выражение запроса можно переписать в виде

$$\mathbf{P}(B \mid j, m) = \alpha \, \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \times \sum_a \mathbf{f}_3(A, B, E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A).$$

Здесь оператор "$\times$" — это не обычное матричное умножение, а операция ▶**точечного произведения** (*pointwise product*), которая будет кратко описана ниже.

В процессе вычислений из точечных произведений факторов устраняются переменные (справа налево) с целью получения новых факторов и в конечном счете получения фактора, образующего решение, т.е. апостериорного распределения по переменной запроса. Поставленная цель достигается выполнением следующих этапов.

- На первом этапе устраняем переменную A из произведения $\mathbf{f}_3$, $\mathbf{f}_4$ и $\mathbf{f}_5$. В результате получаем новый фактор размерностью 2×2 $\mathbf{f}_6(B, E)$, индексы которого пробегают только по B и E:

$$\mathbf{f}_6(B, E) = \sum_a \mathbf{f}_3(A, B, E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A) =$$
$$= (\mathbf{f}_3(a, B, E) \times \mathbf{f}_4(a) \times \mathbf{f}_5(a)) + (\mathbf{f}_3(\neg a, B, E) \times \mathbf{f}_4(\neg a) \times \mathbf{f}_5(\neg a)).$$

  Теперь у нас осталось лишь выражение

$$\mathbf{P}(B \mid j, m) = \alpha\, \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \times \mathbf{f}_6(B, E).$$

- На втором этапе аналогичным образом устраняем переменную E из произведения $\mathbf{f}_2$ и $\mathbf{f}_6$:

$$\mathbf{f}_7(B) = \sum_e \mathbf{f}_2(E) \times \mathbf{f}_6(B, E) =$$
$$= \mathbf{f}_2(e) \times \mathbf{f}_6(B, e) + \mathbf{f}_2(\neg e) \times \mathbf{f}_6(B, \neg e).$$

  В результате у нас остается выражение

$$\mathbf{P}(B \mid j, m) = \alpha\, \mathbf{f}_1(B) \times \mathbf{f}_7(B),$$

  которое можно вычислить, получив точечное произведение и нормализовав результат.

Анализируя приведенную выше последовательность этапов, можно убедиться, что здесь требуется выполнение только двух основных вычислительных операций: получение точечного произведения пары факторов и исключение некоторой переменной из произведения факторов путем суммирования. В следующем подразделе каждая из этих операций описывается подробнее.

Операции над факторами

Результатом точечного перемножения двух факторов, **f** и **g**, является новый фактор **h**, переменные которого представляют собой объединение переменных из **f** и **g**, а элементы задаются произведением соответствующих элементов в двух исходных факторах. Предположим, что два фактора имеют общие переменные $Y_1, \ldots, Y_k$. В таком случае получим

$$\mathbf{f}(X_1, \ldots X_j, Y_1, \ldots Y_k) \times \mathbf{g}(Y_1, \ldots Y_k, Z_1, \ldots Z_\ell) = \mathbf{h}(X_1, \ldots X_j, Y_1, \ldots Y_k, Z_1, \ldots Z_\ell).$$

Если все переменные являются бинарными, то факторы **f** и **g** имеют 2^{j+k} и $2^{k+\ell}$ элементов соответственно, а их точечное произведение будет иметь $2^{j+k+\ell}$ элементов. Например, при заданных двух факторах $\mathbf{f}(X, Y)$ и $\mathbf{g}(Y, Z)$ точечное произведение $\mathbf{f} \times \mathbf{g} = \mathbf{h}(X, Y, Z)$ имеет $2^{1+1+1} = 8$ записей, как показано на рис. 13.12. Обратите внимание, что фактор, полученный в результате точечного произведения, может

содержать больше переменных, чем любой из перемножаемых факторов, и что размер фактора растет по экспоненциальному закону в зависимости от числа переменных. Именно это является причиной быстрого роста пространственной и временной сложности в алгоритме устранения переменной.

X	Y	$\mathbf{f}(X, Y)$	Y	Z	$\mathbf{g}(Y, Z)$	X	Y	Z	$\mathbf{h}(X, Y, Z)$
t	*t*	0,3	*t*	*t*	0,2	*t*	*t*	*t*	0,3 × 0,2 = 0,06
t	*f*	0,7	*t*	*f*	0,8	*t*	*t*	*f*	0,3 × 0,8 = 0,24
f	*t*	0,9	*f*	*t*	0,6	*t*	*f*	*t*	0,7 × 0,6 = 0,42
f	*f*	0,1	*f*	*f*	0,4	*t*	*f*	*f*	0,7 × 0,4 = 0,28
						f	*t*	*t*	0,9 × 0,2 = 0,18
						f	*t*	*f*	0,9 × 0,8 = 0,72
						f	*f*	*t*	0,1 × 0,6 = 0,06
						f	*f*	*f*	0,1 × 0,4 = 0,04

Рис. 13.12. Пример точечного произведения факторов: $\mathbf{f}(X, Y) \times \mathbf{g}(Y, Z) = \mathbf{h}(X, Y, Z)$

Операция устранения некоторой переменной из произведения факторов выполняется посредством устранения подматриц, образованных путем фиксации переменной к каждому из ее значений по очереди. Например, для устранения переменной X из фактора $\mathbf{h}(X, Y, Z)$ запишем

$$\mathbf{h}_2(Y,Z) = \sum_x \mathbf{h}(X,Y,Z) = \mathbf{h}(x,Y,Z) + \mathbf{h}(\neg x,Y,Z) =$$

$$= \begin{pmatrix} 0,06 & 0,24 \\ 0,42 & 0,28 \end{pmatrix} + \begin{pmatrix} 0,18 & 0,72 \\ 0,06 & 0,04 \end{pmatrix} = \begin{pmatrix} 0,24 & 0,96 \\ 0,48 & 0,32 \end{pmatrix}.$$

Единственная хитрость состоит в том, что в этой операции следует учитывать, что любой фактор, *не зависящий* от переменной, подлежащей устранению, может быть вынесен за пределы выражения суммирования. Например, для устранения суммированием переменной X из точечного произведения факторов $\mathbf{f}$ и $\mathbf{g}$ можно вынести фактор $\mathbf{g}$ за пределы выражения суммирования:

$$\sum_x \mathbf{f}(X,Y) \times \mathbf{g}(Y,Z) = \mathbf{g}(Y,Z) \times \sum_x \mathbf{f}(X,Y).$$

Потенциально это намного более эффективно, чем вычисление большего точечного произведения $\mathbf{h}$, а затем устранения из него переменной X.

Обратите внимание на то, что матрицы *не умножаются* до тех пор, пока не потребуется устранить суммированием некоторую переменную из накопленного произведения. В этот момент умножаются только те матрицы, которые включают переменную, подлежащую устранению. Если даны функции получения точечного

произведения и устранения суммированием, то алгоритм устранения переменной может быть записан весьма просто, как показано на рис. 13.13.

---

function ELIMINATION-ASK(X, **e**, bn) **returns** распределение по X
  inputs: X, переменная запроса
    e, наблюдаемые значения переменных **E**
    bn, байесовская сеть с переменными *vars*

  factors ← [ ]
  for each V **in** ORDER(*vars*) **do**
    factors ← [MAKE-FACTOR(V, **e**)] + *factors*
    if V является скрытой переменной **then** *factors* ← SUM-OUT(V, *factors*)
  return NORMALIZE(POINTWISE-PRODUCT(*factors*))

Рис. 13.13. Алгоритм устранения переменной, предназначенный для выполнения точного вероятностного вывода в байесовских сетях

Упорядочение переменных и релевантность переменной

Алгоритм, представленный на рис. 13.13, включает некоторую неуточненную функцию ORDER, предназначенную для выбора упорядоченности переменных. Выбор любой упорядоченности приводит к допустимому алгоритму, но разная упорядоченность приводит к тому, что в процессе вычислений генерируются разные промежуточные факторы. Например, в приведенном выше расчете переменная A устраняется раньше переменной E; но если поступить наоборот, выполняемые вычисления изменятся на

$$\mathbf{P}(B \mid j,m) = \alpha\, \mathbf{f}_1(B) \times \sum_a \mathbf{f}_4(A) \times \mathbf{f}_5(A) \times \sum_e \mathbf{f}_2(E) \times \mathbf{f}_3(A,B,E),$$

при выполнении которых будет сгенерирован другой фактор $\mathbf{f}_6(A, B)$.

В общем случае временная и пространственная сложность процедуры устранения переменной определяется в основном размером наибольшего фактора, создаваемого в процессе работы алгоритма. А это, в свою очередь, определяется выбранным порядком устранения переменных и структурой сети. Оказывается, очень трудно определить оптимальный порядок, но есть несколько хороших эвристических методов. Один довольно эффективный метод можно характеризовать как "жадный": устраняется та переменная, для которой размер следующего создаваемого фактора будет минимальным.

Рассмотрим еще один запрос: $\mathbf{P}(JohnCalls \mid Burglary = true)$. Как обычно (см. уравнение (13.5)), на первом этапе необходимо записать вложенное выражение суммирования:

$$\mathbf{P}(J \mid b) = \alpha P(b) \sum_e P(e) \sum_a P(a \mid b, e) \mathbf{P}(J \mid a) \sum_m P(m \mid a) .$$

Если вычислять это выражение справа налево, то можно заметить нечто интересное: терм $\Sigma_m P(m \mid a)$ равен 1 по определению! А значит, необходимость учитывать это выражение просто отпадает: переменная *M не имеет отношения* к данному запросу. Эту мысль можно выразить и иным образом: результат запроса $P(JohnCalls \mid Burglary = true)$ не изменится, если из сети вообще исключить переменную *MaryCalls*. Вообще говоря, из сети может быть удалена любая листовая вершина, если она не является переменной запроса или переменной свидетельства. После ее удаления в сети могут оставаться еще некоторые листовые вершины, которые также могут не иметь отношения к данному запросу. Продолжая указанный процесс, мы в конечном итоге обнаружим, что ➔ *любая переменная, не являющаяся потомком переменной запроса или переменной свидетельства, не имеет отношения к запросу.* Поэтому алгоритм устранения переменной позволяет сразу удалить из рассмотрения все эти переменные, прежде чем приступить к вычислению ответа на запрос.

Если применить сказанное к примеру сети страховой компании, представленной на рис. 13.9, то метод устранения переменной дает значительное улучшение в сравнении с показателями алгоритма простого перебора. При выборе обратного топологического порядка для переменных точный вероятностный вывод с использованием устранения переменной работает примерно в 1000 раз быстрее, чем алгоритм перебора.

13.3.3. Сложность точного вероятностного вывода

Сложность точного вероятностного вывода в байесовских сетях сильно зависит от структуры сети. Сеть системы защиты от взлома, приведенная на рис. 13.2, принадлежит к семейству сетей, в которых существует не более одного ненаправленного (т.е. игнорирующего направление стрелок) пути между любыми двумя узлами в сети. Такие сети называют ▶**односвязными** или ▶**полидеревьями**, они обладают особенно хорошим свойством: ➔ *временная и пространственная сложность точного вывода в полидеревьях линейно зависит от размера сети.* В нашем случае размер определяется как количество записей в CPT; если количество родительских вершин каждой вершины ограничено некоторой константой, то сложность также будет линейно зависеть от количества вершин. Эти результаты верны для любого упорядочивания, совместимого с топологическим порядком сети (см. упражнение 13.18).

Для ▶**многосвязных** сетей, таких как сеть страховой компании, представленная на рис. 13.9, метод устранения переменной в худшем случае может иметь экспоненциальную временную и пространственную сложность, даже если количество родительских вершин для любой вершины ограниченно. И это не удивительно,

если учесть тот факт, что ➨*вероятностный вывод в байесовских сетях — поскольку вывод в логике высказываний можно рассматривать как его частный случай — является задачей NP-трудной.* Чтобы доказать это утверждение, необходимо найти способ закодировать задачу пропозициональной выполнимости в виде байесовской сети таким образом, чтобы, выполнив вероятностный вывод в этой сети, можно было узнать, выполнимы ли исходные пропозициональные высказывания. (На языке теории сложности это позволит ➨**свести** задачи выполнимости к задачам вероятностного вывода в байесовских сетях.) Как оказалось, сделать это довольно просто. На рис. 13.14 показано, как можно закодировать конкретную задачу 3-SAT. Пропозициональные переменные используются в качестве корневых переменных сети, каждая с априорной вероятностью 0,5. Следующий уровень вершин соответствует выражениям, причем каждая переменная выражения C_j связывается с соответствующими переменными как родительскими вершинами. Условное распределение для переменной выражения задается как детерминированная дизъюнкция — с отрицанием, если это потребуется, так что каждая переменная выражения будет иметь значение *true* тогда и только тогда, когда присваивание значений ее родительским узлами удовлетворяет это выражение. И наконец, переменная S является конъюнкцией переменных выражений.

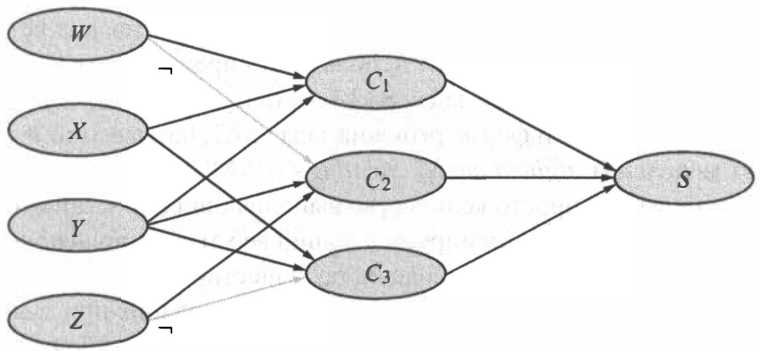


Рис. 13.14. Байесовская сеть, в которой закодировано высказывание 3-CNF $(W \vee X \vee Y) \wedge (\neg W \vee Y \vee Z) \wedge (X \vee Y \vee \neg Z)$

Чтобы выяснить, удовлетворяется ли исходное высказывание, достаточно просто вычислить $P(S = true)$. Если высказывание *выполнимо*, то существуют некоторые возможные присваивания для логических переменных, при которых S будет иметь значение *true*. В байесовской сети это означает, что существует возможный мир с ненулевой вероятностью, в котором корневые переменные имеют такое присваивание, что переменные выражений будут иметь значение *true* и переменная S также будет иметь значение *true*. Следовательно, $P(S = true) > 0$ для выполнимых высказываний. И наоборот, $P(S = true) = 0$ для невыполнимых высказываний:

все миры с $S = true$ имеют вероятность 0. Таким образом, вероятностный вывод в байесовских сетях можно использовать для решения задач 3-SAT, а из этого можно сделать заключение, что вероятностный вывод в байесовских сетях является NP-сложной задачей.

На самом деле можно сделать даже больше. Обратите внимание, что вероятность каждого выполняющего присваивания составляет 2^{-n} для задачи с n переменными. Следовательно, *количество* выполняющих присваиваний равно $P(S = true)/(2^{-n})$. Поскольку вычисление *количества* выполняющих присваиваний для задачи 3-SAT является #P-полным (читается как "шарп-P-полным"), это означает, что вероятностный вывод в байесовской сети является задачей #P-трудной, т.е. строго сложнее, чем NP-полные задачи.

Существует тесная связь между сложностью вероятностного вывода в байесовской сети и сложностью задач удовлетворения ограничений (УО). Как было показано в главе 6, трудность решения дискретной задачи УО зависит от того, насколько "древовидным" является ее граф ограничений. Такие показатели, как **ширина гипердерева**, устанавливающие пределы сложности решения задачи УО, могут также применяться непосредственно к байесовским сетям. Более того, можно обобщить алгоритм устранения переменной таким образом, чтобы он позволял находить решения не только в байесовских сетях, но и в задачах УО.

Подобно сведению задач выполнимости к вероятностному выводу в байесовских сетях, можно свести сам вероятностный вывод в байесовских сетях к определению выполнимости, что позволит использовать преимущества мощных приложений, созданных для решения задач SAT (см. главу 7). В этом случае сведение выполняется к определенной форме решения задач SAT, называемой ▶ **подсчетом взвешенных моделей** (*weighted model counting* — WMC). При обычном подсчете моделей подсчитывается просто количество выполняющих присваиваний для выражения SAT. В методе WMC суммируется общий вес этих выполняющих присваиваний, где для данного случая вес модели, по существу, является произведением условных вероятностей для каждой переменной присваивания при заданных значениях родительских вершин. (Подробности приведены в упражнении 13.19.) Отчасти потому, что технология решателей SAT была столь хорошо оптимизирована именно для крупномасштабных приложений, вероятностный вывод в байесовских сетях через WMC оказался вполне конкурентоспособным, а иногда и превосходящим эффективность других точных алгоритмов для байесовских сетей с большой шириной дерева.

13.3.4. Алгоритмы кластеризации

Обсуждавшийся выше алгоритм устранения переменной является простым и эффективным средством получения ответов на отдельные запросы. Однако, если потребуется вычислить апостериорные вероятности для всех переменных в сети, этот алгоритм может оказаться менее эффективным. Например, в сети с

полидревовидной структурой для этого потребуется выдать $O(n)$ запросов с затратами $O(n)$ каждый, что в сумме потребует затрат времени $O(n^2)$. Использование алгоритмов ►**кластеризации** (известных также под названием алгоритмов ►**дерева соединения** (*join tree*), позволяет сократить это время до $O(n)$. По этой причине данные алгоритмы широко используются в коммерческих системах работы с байесовскими сетями.

Основная идея кластеризации состоит в объединении отдельных узлов сети для формирования кластерных вершин таким образом, чтобы результирующая сеть стала полидеревом. Например, многосвязная сеть, показанная на рис. 13.15, *а*, может быть преобразована в полидерево путем объединения вершин *Sprinkler* (дождеватель) и *Rain* (дождь) в кластерную вершину с названием *Sprinkler+Rain*, как показано на рис. 13.15, *б*. В результате эти две булевы вершины заменяются ►**мегавершиной**, принимающей четыре возможных значения: *tt*, *tf*, *ft* и *ff*. Эта мегавершина имеет только одну родительскую вершину — булеву переменную *Cloudy*, поэтому для нее количество обусловливающих случаев равно двум. Хотя данный пример этого не демонстрирует, процесс кластеризации часто приводит к формированию мегавершин, разделяющих некоторые переменные.

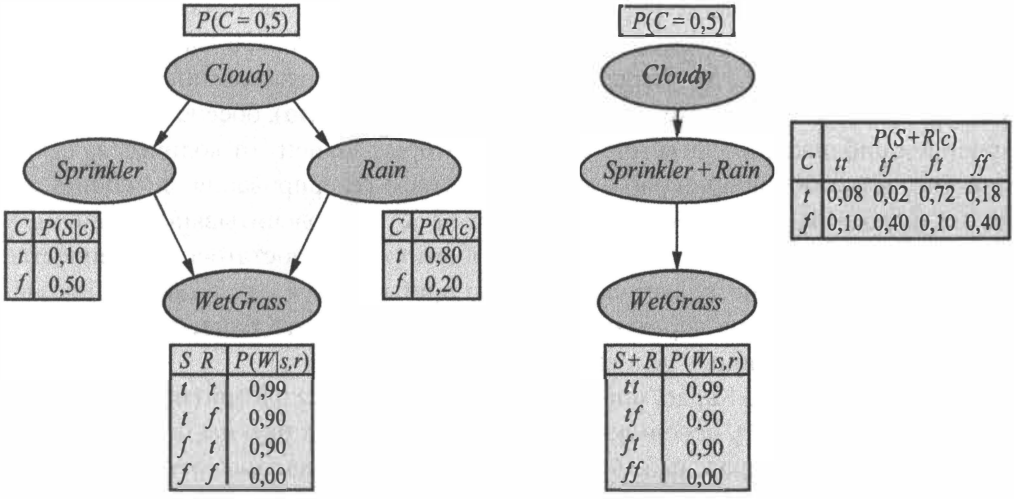


Рис. 13.15. а) Многосвязная сеть, описывающая процедуру ежедневного ухода за газоном Мэри. Каждое утро она проверяет погоду и, если облачно (*cloudy)*, обычно не включает дождеватель системы полива. Если дождеватель был включен или в течение дня шел дождь, то трава будет влажной (*wetgrass*). Следовательно, вершина *Cloudy* влияет на вершину *WetGrass* двумя различными путями причинно-следственных связей. **б)** Кластеризованный эквивалент исходной многосвязной сети

После приведения сети к форме полидерева применяется алгоритм вероятностного вывода специального назначения, поскольку обычные методы не позволяют обрабатывать мегаузлы, включающие общие переменные. По сути, этот алгоритм представляет собой одну из форм алгоритма распространения ограничений (см. главу 6), где ограничения обеспечивают согласование соседних кластеров по апостериорной вероятности любых переменных, которые являются у них общими. При наличии тщательно продуманных средств учета этот алгоритм позволяет вычислять апостериорные вероятности для всех вершин в сети, отличных от вершин свидетельства, за время, *линейно* зависящее от размера кластеризованной сети. Тем не менее NP-трудность решаемой задачи никуда не исчезает: если сеть требует экспоненциальных затрат времени и пространства при устранении переменных, то экспоненциальные затраты времени и пространства потребуются и для построения таблиц условной вероятности (CPT) в кластеризованной сети.

13.4. Приближенный вероятностный вывод в байесовских сетях

Принимая во внимание неразрешимость точного вероятностного вывода в больших байесовских сетях, далее мы рассмотрим приближенные методы вероятностного вывода. В этом разделе описываются рандомизированные алгоритмы выборки (называемые также алгоритмами ►**Монте-Карло**), обеспечивающие получение приближенных ответов, точность которых зависит от количества сформированных выборок. Они работают посредством генерирования случайных событий, исходя из вероятностей в байесовской сети и подсчитывания различных ответов, найденных при этих случайных событиях. При достаточном количестве выборок этот метод позволяет сколь угодно приблизиться к восстановлению истинного распределения вероятностей — при условии, что байесовская сеть не имеет детерминированных условных распределений.

Алгоритмы Монте-Карло, примером которых является алгоритм эмуляции отжига (см. раздел 4.1.2), используются во многих отраслях науки для оценки величин, которые трудно рассчитать точно. В этом разделе нас интересует выборка, применяемая для вычисления апостериорных вероятностей в байесовских сетях. Здесь описывается два семейства алгоритмов: непосредственная выборка и выборка с помощью цепи Маркова. Несколько других подходов к реализации приблизительного вероятностного вывода упоминаются в разделе "Библиографические и исторические заметки" в конце главы.

13.4.1. Методы непосредственной выборки

Примитивным элементом в любом алгоритме выборки является формирование выборок из известного распределения вероятностей. Например, обычная монета может рассматриваться как случайная переменная *Coin* со значениями ⟨*heads*, *tails*⟩ и априорным распределением $\mathbf{P}(Coin) = \langle 0{,}5;\ 0{,}5\rangle$. Операция получения выборки из этого распределения полностью аналогична подбрасыванию монеты: с вероятностью 0,5 эта операция будет возвращать значение *heads* (орел) и с такой же вероятностью — значение *tails* (решка). Если имеется источник случайных чисел r, равномерно распределенных в диапазоне [0, 1], совсем несложно сформировать любое распределение по одной переменной, как дискретное, так и непрерывное. Это выполняется путем построения кумулятивного распределения для переменной и возврата первого значения, кумулятивная вероятность которого превышает r (см. упражнение 13.20).

Начнем с процесса построения случайной выборки для байесовской сети, не имеющей связанных с этой выборкой свидетельств. Идея состоит в том, что выборка должна формироваться последовательно по каждой переменной, в топологическом порядке. Распределение вероятностей, из которого берется выборка значения, обусловливается значениями, уже присвоенными родительским переменным текущей переменной. (Поскольку выборка выполняется в топологическом порядке, родительские вершины гарантированно уже будут иметь значения.) Этот алгоритм приведен на рис. 13.16. Применяя его к сети, представленной на рис 13.15, *a*, с упорядоченностью переменных *Cloudy*, *Sprinkler*, *Rain*, *WetGrass*, случайное событие можно создать следующим образом.

1. Выполняем выборку из распределения $\mathbf{P}(Cloudy) = \langle 0{,}5;\ 0{,}5\rangle$; предположим, что она возвращает *true*.
2. Выполняем выборку из распределения $\mathbf{P}(Sprinkler \mid Cloudy = true) = \langle 0{,}1;\ 0{,}9\rangle$; предположим, что она возвращает *false*.
3. Выполняем выборку из распределения $\mathbf{P}(Rain \mid Cloudy = true) = \langle 0{,}8;\ 0{,}2\rangle$; предположим, что она возвращает *true*.
4. Выполняем выборку из распределения $\mathbf{P}(WetGrass \mid Sprinkler = false, Rain = true) = \langle 0{,}9;\ 0{,}1\rangle$; предположим, что она возвращает *true*.

В данном случае алгоритм PRIOR-SAMPLE возвращает событие [*true*, *false*, *true*, *true*].

Легко показать, что алгоритм PRIOR-SAMPLE генерирует выборки на основе априорного совместного распределения, заданного в рассматриваемой сети. Итак, пусть $S_{PS}(x_1, \ldots, x_n)$ представляет собой вероятность того, что конкретное событие сгенерировано алгоритмом PRIOR-SAMPLE. Просто взглянув на процесс формирования выборки, можно утверждать, что

$$S_{PS}(x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i)),$$

поскольку каждый этап процедуры зависит только от родительских значений. Это выражение должно показаться читателю весьма знакомым, так как оно определяет также вероятность события в соответствии с представлением совместного распределения в байесовской сети, как указано в уравнении (13.2). Поэтому получаем следующее:

$$S_{PS}(x_1,,\cdots,x_n) = P(x_1,,\cdots,x_n).$$

Этот простой факт позволяет легко получать ответы на вопросы с помощью выборок.

---

function PRIOR-SAMPLE(*bn*) **returns** событие, полученное путем применения операции формирования выборки к априорному распределению, заданному в виде сети *bn*
inputs: *bn*, байесовская сеть, задающая совместное распределение $\mathbf{P}(X_1,\ldots,X_n)$

x ← событие с *n* элементами
for each переменная X_i **in** $X_1,\ldots,X_n$ **do**
  x[*i*] ← случайная выборка из $\mathbf{P}(X_i \mid parents(X_i))$
return x

Рис. 13.16. Алгоритм выборки, генерирующий события на основании байесовской сети. Каждая переменная получает значение в соответствии с ее условным распределением при уже заданных значениях ее родительских переменных

В любом алгоритме формирования выборки ответы вычисляются подсчетом фактически сформированных выборок. Предположим, что общее количество выборок, полученных с помощью алгоритма PRIOR-SAMPLE, равно N, и пусть $N_{PS}(x_1,\cdots,x_n)$ — количество раз, когда конкретное событие $x_1,\cdots,x_n$ появлялось в множестве выборок. Мы рассматриваем это количество как часть от целого, которая в пределе сойдется к ожидаемому значению, соответствующему вероятности выборки:

$$\lim_{N\to\infty} \frac{N_{PS}(x_1,\ldots,x_n)}{N} = S_{PS}(x_1,\cdots,x_n) = P(x_1,\cdots,x_n). \qquad (13.6)$$

Например, рассмотрим событие, полученное ранее: [*true*, *false*, *true*, *true*]. Вероятность формирования выборки для этого события такова:

$$S_{PS}(true, false, true, true) = 0{,}5 \times 0{,}9 \times 0{,}8 \times 0{,}9 = 0{,}324.$$

Следовательно, в пределе при очень больших значениях N около 32,4% выборок будут относиться к этому событию.

В дальнейшем всякий раз, когда будет использован символ приближенного равенства ($\approx$), мы будем придавать ему именно этот смысл, — что оцениваемая

вероятность становится точной в пределе при больших количествах выборок. Такая оценка называется ▶**согласованной**. Например, можно получить согласованную оценку вероятности любого частично заданного события $x_1,\ldots,x_m$, где $m \leq n$, следующим образом:

$$P(x_1,\ldots,x_m) \approx N_{PS}(x_1,\ldots,x_m)/N. \tag{13.7}$$

Это означает, что вероятность события можно оценить с помощью деления количества выборок частично заданного события на количество завершенных событий, полученных в процессе формирования выборок. Мы будем использовать запись $\hat{P}$ (произносится как "P-шапочка") для обозначения предполагаемой вероятности. Например, если для сети полива лужайки (см. рис. 13.15, *а*) будет сгенерирована 1000 выборок и 511 из них будут включать выражение $Rain = true$, то оценка вероятности дождя, которая записывается как $\hat{P}(Rain = true)$, будет равна 0,511.

Формирование выборок с отклонением в байесовских сетях

Формирование ▶**выборок с отклонением** представляет собой общий метод получения выборок на основании распределения, для которого трудно получить выборки, если дано распределение, позволяющее легко сформировать выборки. В своей простейшей форме этот метод может использоваться для вычисления условных вероятностей, т.е. для определения вероятностей $P(X \mid e)$. Алгоритм REJECTION-SAMPLING приведен на рис. 13.17. Вначале он формирует выборки из априорного распределения, определяемого сетью, а затем исключает все те выборки, которые не соответствуют свидетельству. Наконец, формируется оценка $\hat{P}(X=x \mid \mathbf{e})$ путем подсчета, насколько часто событие $X=x$ встречалось в оставшихся выборках.

Пусть $\hat{P}(X \mid \mathbf{e})$ — оцениваемое распределение, которое было возвращено алгоритмом; это распределение вычислено посредством нормализации $\mathbf{N}_{PS}(X, \mathbf{e})$, вектора количества выборок для каждого значения X, где выборка согласуется со свидетельством $\mathbf{e}$:

$$\hat{\mathbf{P}}(X \mid \mathbf{e}) = \alpha\, \mathbf{N}_{PS}(X, \mathbf{e}) = \frac{\mathbf{N}_{PS}(X, \mathbf{e})}{N_{PS}(\mathbf{e})}.$$

На основании уравнения (13.7) это соотношение может быть преобразовано в следующее:

$$\hat{\mathbf{P}}(X \mid \mathbf{e}) \approx \frac{\mathbf{P}(X, \mathbf{e})}{P(\mathbf{e})} = \mathbf{P}(X \mid \mathbf{e}).$$

Таким образом, алгоритм формирования выборок с отклонением вырабатывает согласованную оценку истинной вероятности.

function REJECTION-SAMPLING(X, **e**, bn, N) **returns** оценка значения $\mathbf{P}(X \mid \mathbf{e})$
  inputs: X, переменная запроса
    e, наблюдаемые значения для переменных **E**
    bn, байесовская сеть
    N, общее количество выборок, которые должны быть сгенерированы
  local variables: **C**, вектор результатов подсчетов количества для каждого значения X, исходно равен нулю

  for $j = 1$ **to** N **do**
    x ← PRIOR-SAMPLE(bn)
    if **x** согласуется со свидетельством **e** **then**
      C[j] ← **C**[j] + 1, где x_j есть значение переменной X в множестве **x**
  return NORMALIZE(C)

Рис. 13.17. Алгоритм формирования выборок с отклонением для получения ответов на запросы при заданных свидетельствах в байесовской сети

Вновь обратившись к примеру, приведенному на рис. 13.15, *а*, предположим, что необходимо оценить вероятность $\mathbf{P}(Rain \mid Sprinkler = true)$ на основании 100 выборок. Предположим, что из 100 сформированных выборок 73 включают событие $Sprinkler = false$ и исключаются из рассмотрения, а в 27 оставшихся выборках имеет место событие $Sprinkler = true$, причем в 8 из этих 27 выборок наблюдается событие $Rain = true$, а в оставшихся 19 — $Rain = false$. Следовательно,

$$\mathbf{P}(Rain \mid Sprinkler = \text{true}) \approx \text{NORMALIZE}(\langle 8; 19\rangle) = \langle 0{,}296; 0{,}704\rangle.$$

Точным ответом является $\langle 0{,}3; 0{,}7\rangle$. По мере дальнейшего накопления все большего количества выборок получаемая оценка сходится к правильному ответу. Среднеквадратичное отклонение ошибки в каждой оценке вероятности будет пропорционально $1/\sqrt{n}$, где n — количество выборок, используемых для получения оценки.

Теперь, когда мы знаем, что результат алгоритма формирования выборок с отклонением сходится к правильному ответу, следующим вопросом будет, насколько быстро это происходит? Говоря точнее, как много потребуется сгенерировать выборок, прежде чем станет известно, что полученные оценки с высокой вероятностью близки к правильным ответам? В то время как сложность точных алгоритмов в значительной степени зависит от топологии сети — деревья просчитать легко, а плотно связанные сети трудно, — сложность выборки с отклонением в первую очередь зависит от доли выборок, которые принимаются. Эта доля в точности равна априорной вероятности свидетельства $P(\mathbf{e})$. К сожалению, для сложных задач со многими переменными свидетельства эта доля исчезающе мала. При применении к дискретной версии сети страховой компании, приведенной на рис. 13.9, доля

выборок, согласованных с типичными вариантами вектора свидетельств, выбранных из самой сети, обычно находится между одной тысячной и одной десятитысячной. Схождение к правильному результату в этом случае является чрезвычайно медленным (обратитесь к рис. 13.19, приведенному ниже).

Можно полагать, что доля выборок, согласованных со свидетельством **e**, будет экспоненциально уменьшаться по мере увеличения количества переменных свидетельства, поэтому данная процедура становится неприменимой для решения сложных задач. Для нее также характерны трудности в отношении переменных свидетельства с непрерывными значениями, так как вероятность получения выборки, согласованной с такими свидетельствами, равна нулю (если переменная действительно является непрерывной) или бесконечно мала (если это просто число с плавающей точкой конечной точности).

Обратите внимание на то, что процесс формирования выборок с отклонением очень похож на процесс оценки условных вероятностей по данным, полученным из реального мира. Например, чтобы оценить условную вероятность того, что какие-то люди выживут после падения на Землю астероида диаметром 1 км, можно просто подсчитать, как часто какие-то люди выживали после падения на Землю астероида диаметром 1 км, исключив из рассмотрения все остальные дни, когда такого события не происходило. (В этом случае роль алгоритма формирования выборок играет сама Вселенная.) Чтобы получить достойную оценку, может потребоваться дождаться, пока произойдет 100 подобных событий. Очевидно, что на это потребуется очень много времени, и именно в этом и состоит основной недостаток процедуры формирования выборок с отклонением.

Выборка по значимости

Общий статистический метод ►**выборки по значимости** нацелен на имитацию эффекта выборки из распределения P с использованием выборок из другого распределения Q. Гарантируется, что в пределе полученные ответы будут верны за счет применения поправочного коэффициента $P(\mathbf{x})/Q(\mathbf{x})$, называемого **весовым**, к каждой выборке **x** при подсчете выборок.

Причина использования выборки по значимости в байесовских сетях очень проста: хотелось бы делать выборки из истинного апостериорного распределения, обусловленного по всем имеющимся свидетельствам, но, как правило, это слишком сложно.[6] Так что вместо этого выборка выполняется из простого распределения и к ней применяются необходимые поправки. Причина, по которой выборка по значимости работает, также проста. Обозначим множество переменных, не имеющих свидетельства, как **Z**. Если бы у нас была возможность делать выборку непосредственно из $P(\mathbf{z} \mid \mathbf{e})$, можно было бы получить такие оценки:

[6] Если бы это было просто, то можно было бы аппроксимировать желаемую вероятность до произвольной точности при полиномиальном количестве выборок. Однако можно показать, что подобной схемы аппроксимации за полиномиальное время не существует.

$$\hat{P}(\mathbf{z} \mid \mathbf{e}) = \frac{N_P(\mathbf{z})}{N} \approx P(\mathbf{z} \mid \mathbf{e}),$$

где $N_P(\mathbf{z})$ представляет собой количество выборок с $\mathbf{Z} = \mathbf{z}$ при выполнении выборки из P. Теперь предположим, что вместо этого мы выбираем из $Q(\mathbf{z})$. Оценка в этом случае будет включать поправочные коэффициенты:

$$\hat{P}(\mathbf{z} \mid \mathbf{e}) = \frac{N_Q(\mathbf{z})}{N} \frac{P(\mathbf{z} \mid \mathbf{e})}{Q(\mathbf{z})} \approx Q(\mathbf{z}) \frac{P(\mathbf{z} \mid \mathbf{e})}{Q(\mathbf{z})} = P(\mathbf{z} \mid \mathbf{e}).$$

Следовательно, оценка сходится к правильному значению *независимо от того, какое выборочное распределение Q используется*. (Единственное техническое требование состоит в том, что $Q(\mathbf{z})$ не должно быть равно нулю для любого $\mathbf{z}$, где $P(\mathbf{z} \mid \mathbf{e})$ отлично от нуля.) Интуитивно понятно, что поправочный коэффициент компенсирует перевыборку или недовыборку. Например, если для некоторого $\mathbf{z}$ распределение $Q(\mathbf{z})$ значительно больше, чем $P(\mathbf{z} \mid \mathbf{e})$, то для этого $\mathbf{z}$ будет намного больше выборок, чем должно быть, но каждая из них будет иметь небольшой вес, поэтому конечный результат будет таким же, как если бы их было правильное количество.

Что касается того, какое распределение Q следует использовать, то желательно выбрать такое, из которого легко делать выборки и которое как можно ближе к истинному апостериорному распределению $P(\mathbf{z} \mid \mathbf{e})$. Наиболее общепринятый подход называют ▶ **взвешиванием по правдоподобию** (*likelihood weighting*) — по причинам, которые скоро станут понятны. Как представлено в функции WEIGHTED-SAMPLE на рис. 13.18, алгоритм фиксирует значения переменных свидетельства $\mathbf{E}$ и делает выборку по всем переменным, не имеющим свидетельства, в топологическом порядке, каждая из которых обусловливается своими родительскими вершинами. Это гарантирует, что каждое сгенерированное событие будет согласовано со свидетельством.

Пусть Q_{WS} — выборочное распределение, сформированное этим алгоритмом. Если множеством переменных, отличных от переменных свидетельства, является $\mathbf{Z} = \{Z_1, \ldots, Z_l\}$, то имеем

$$Q_{WS}(\mathbf{z}) = \prod_{i=1}^{l} P(z_i \mid parents(Z_i)), \tag{13.8}$$

поскольку для каждой переменной выборка формируется с учетом значений ее родительских узлов. Чтобы полностью завершить этот алгоритм, необходимо знать, как вычислить весовой коэффициент для каждой выборки, сформированной из Q_{WS}. Согласно общей схеме формирования выборки по значимости, весовой коэффициент должен быть

$$w(\mathbf{z}) = P(\mathbf{z} \mid \mathbf{e}) / Q_{WS}(\mathbf{z}) = \alpha P(\mathbf{z}, \mathbf{e}) / Q_{WS}(\mathbf{z}),$$

где константа нормализации $\alpha = 1/P(\mathbf{e})$ будет одной и той же для всех выборок. Теперь **z** и **e** вместе охватывают все переменные в байесовской сети, поэтому $P(\mathbf{z}, \mathbf{e})$ является просто произведением всех условных вероятностей (см. уравнение (13.2), раздел 13.2). Можно записать его как произведение условных вероятностей для переменных, не являющихся переменными свидетельства, умноженное на произведение условных вероятностей для переменных свидетельства:

$$\begin{aligned} w(\mathbf{z}) &= \alpha \frac{P(\mathbf{z}, \mathbf{e})}{Q_{WS}(\mathbf{z})} = \\ &= \alpha \frac{\prod_{i=1}^{l} P(z_i \mid parents(Z_i)) \prod_{i=1}^{m} P(e_i \mid parents(E_i))}{\prod_{i=1}^{l} P(z_i \mid parents(Z_i))} = \\ &= \alpha \prod_{i=1}^{m} P(e_i \mid parents(E_i)) . \end{aligned} \tag{13.9}$$

---

function LIKELIHOOD-WEIGHTING(X, **e**, bn, N) **returns** оценка значения $\mathbf{P}(X \mid \mathbf{e})$
  inputs: X, переменная запроса
    e, свидетельство, определяемое как некоторое событие в **E**
    bn, байесовская сеть, задающая совместное распределение $\mathbf{P}(X_1, \ldots, X_n)$
    N, общее количество выборок, которые должны быть сформированы
  local variables: **W**, вектор взвешенных результатов подсчетов для каждого значения X, исходно равен нулю

  for $j = 1$ **to** N **do**
    x, $w \leftarrow$ WEIGHTED-SAMPLE(bn, **e**)
    $\mathbf{W}[j] \leftarrow \mathbf{W}[j] + w$, где x_j является значением переменной X в множестве **x**
  return NORMALIZE(**W**)

function WEIGHTED-SAMPLE(bn, **e**) **returns** событие **x** и вес w

  $w \leftarrow 1$; **x** $\leftarrow$ событие с n элементами, значения которых определяются из **e**
  for $i = 1$ **to** n **do**
    if X_i является переменной свидетельства со значением x_{ij} в **e**
      then $w \leftarrow w \times P(X_i = x_{ij} \mid parents(X_i))$
      else $\mathbf{x}[i] \leftarrow$ случайная выборка из распределения $\mathbf{P}(X_i \mid parents(X_i))$
  return **x**, w

Рис. 13.18. Алгоритм взвешивания по правдоподобию для вероятностного вывода в байесовских сетях. В функции WEIGHTED-SAMPLE для каждой переменной, не являющейся переменной свидетельства, формируется выборка в соответствии с условным распределением при заданных значениях родительских переменных, выборка для которых уже была сформирована, в то время как весовое значение накапливается на основе вероятности для каждой переменной свидетельства

Таким образом, весовой коэффициент является произведением условных вероятностей для переменных свидетельства при заданных значениях родительских переменных. (Вероятности свидетельства обычно называют **правдоподобием**, отсюда и название метода.) Вычисление весовых коэффициентов в функции WEIGHTED-SAMPLE выполняется пошагово, умножением текущего значения (исходно это 1) на условную вероятность каждый раз, когда встречается очередная переменная свидетельства. Нормализация весовых коэффициентов выполняется в конце работы основного алгоритма, перед возвращением результата запроса.

Применим этот алгоритм к сети, представленной на рис. 13.15, *a*, на примере получения ответа на запрос $\mathbf{P}(Rain \mid Cloudy = true, WetGrass = true)$ и упорядоченности вершин *Cloudy*, *Sprinkler*, *Rain*, *WetGrass*. (Может быть использовано любое топологическое упорядочение.) Этот процесс происходит следующим образом: вначале вес *w* устанавливается равным 1,0, а затем генерируется событие, как описано ниже.

1. *Cloudy* — это переменная свидетельства со значением *true*. Поэтому мы устанавливаем
$$w \leftarrow w \times P(Cloudy = true) = 0{,}5.$$
2. Переменная *Sprinkler* не является переменной свидетельства, поэтому выборка из $\mathbf{P}(Sprinkler \mid Cloudy = true) = \langle 0{,}1;\ 0{,}9 \rangle$. Предположим, что возвращается значение *false*.
3. Переменная *Rain* не является переменной свидетельства, поэтому выборка из $\mathbf{P}(Rain \mid Cloudy = true) = \langle 0{,}8;\ 0{,}2 \rangle$. Предположим, что возвращается значение *true*.
4. Переменная *WetGrass* — это переменная свидетельства со значением *true*, поэтому мы устанавливаем
$$w \leftarrow w \times P(WetGrass = true \mid Sprinkler = false, Rain = true) = \\ = 0{,}5 \times 0{,}9 = 0{,}45.$$

В этом случае алгоритм WEIGHTED-SAMPLE возвращает событие [*true*, *false*, *true*, *true*] с весовым коэффициентом 0,45, и данные об этом событии подытоживаются с учетом условия *Rain* = *true*.

Обратите внимание, что в число переменных $Parents(Z_i)$ могут входить как скрытые переменные, не являющиеся переменными свидетельства, так и переменные свидетельства. В отличие от априорного распределения $P(\mathbf{z})$, в распределении Q_{WS} некоторое внимание уделено свидетельству: для каждой переменной Z_i на значения сформированных выборок среди других предков Z_i оказывает влияние и свидетельство. Например, при формировании выборочных значений для переменной *Sprinkler* алгоритм обращает внимание на свидетельство *Cloudy* = *true* в ее родительских переменных. С другой стороны, в распределении Q_{WS} свидетельству уделяется меньше внимания, чем в истинном апостериорном распределении

$P(\mathbf{z} \mid \mathbf{e})$, поскольку в значениях сформированных выборок для каждой переменной Z_i *игнорируются* свидетельства, относящиеся к переменным, не являющимся предками Z_i. Например, при формировании выборок для переменных *Sprinkler* и *Rain* алгоритм игнорирует свидетельство дочерней переменной *WetGrass* = *true*, а это означает, что он сформирует много выборок с переменными *Sprinkler* = *false* и *Rain* = *false*, несмотря на тот факт, что свидетельство фактически исключает этот случай. Эти выборки будут иметь нулевой вес.

Поскольку в алгоритме взвешивания по правдоподобию используются все сформированные выборки, он может оказаться гораздо более эффективным по сравнению с алгоритмом формирования выборок с отклонением. Тем не менее он также имеет недостаток: снижение производительности по мере увеличения количества переменных свидетельства. Это связано с тем, что в таких случаях большинство выборок будут иметь очень малые весовые коэффициенты и, следовательно, во взвешенной оценке будет доминировать крошечная доля выборок, которые согласуются со свидетельством с правдоподобием, большим бесконечно малого. Эта проблема усугубляется, если переменные свидетельства занимают последние места в упорядочении переменных, поскольку в этом случае переменные, не являющиеся переменными свидетельства, не получат никаких свидетельств от своих родительских переменных и других предков, которые направляли бы процесс формирования выборок. Фактически в этом случае процесс формирования выборки будет представлять собой некую "галлюцинацию" — моделирование ситуации, имеющей мало сходства с реальностью, определяемой свидетельством.

При применении к дискретной версии сети страховой компании, представленной на рис. 13.9, алгоритм взвешивания по правдоподобию оказывается значительно более эффективным в сравнении с алгоритмом выборки с отклонением (рис. 13.19). Сеть страховой компании является относительно благоприятным случаем для алгоритма взвешивания по правдоподобию, поскольку большая часть переменных свидетельства находится в начале последовательности упорядочения переменных, а переменные запроса представляют собой конечные узлы сети.

13.4.2. Вероятностный вывод по методу моделирования цепи Маркова

Алгоритмы ►**Монте-Карло с использованием цепи Маркова** (*Markov chain Monte Carlo* — **MCMC**) работают не так, как алгоритмы выборки с отклонением или взвешивания по правдоподобию. Вместо формирования каждой выборки с нуля алгоритмы MCMC формируют очередную выборку посредством внесения случайного изменения в предыдущую выборку. Работу алгоритма MCMC можно представить так: находясь в некотором определенном *текущем состоянии*, в котором каждой переменной присвоено некоторое значение, алгоритм формирует *следующее состояние* за счет внесения случайных изменений в текущее состояние.

Рис. 13.19. Производительность алгоритмов выборки с отклонением и взвешивания по правдоподобию применительно к сети страховой компании. Ось *x* представляет количество сформированных выборок, а ось *y* — максимальную абсолютную ошибку любых значений вероятности для запроса о переменной *PropertyCost*

Термин ►**цепь Маркова** определяет случайный процесс, в котором формируется некоторая последовательность состояний. (Концепция цепей Маркова также занимает видное место в материале глав 14 и 17; алгоритмы моделирования отжига из главы 4 и WALKSAT из главы 7 также являются членами семейства MCMC.) Мы начнем с обсуждения определенной формы алгоритмов MCMC, называемой ►**выборкой Гиббса**, которая особенно хорошо подходит для байесовских сетей. Затем будет рассмотрен более общий алгоритм ►**Метрополиса–Гастингса**, обеспечивающий гораздо больше гибкости в процессе формирования выборок.

Выборка Гиббса в байесовских сетях

Алгоритм выборки Гиббса для сетей Байеса начинает работу с произвольного состояния (при этом наблюдаемые значения переменных свидетельства являются зафиксированными) и формирует следующее состояние посредством случайной выборки значения для одной из переменных X_i, отличной от переменных свидетельства. Напомним из раздела 13.2.1, что переменная X_i является независимой от всех других переменных при заданных значениях переменных ее марковского покрытия (ее родительских переменных, дочерних переменных и других родительских переменных ее дочерних переменных). Следовательно, выборка Гиббса для переменной X_i означает выборку, *обусловленную текущими значениями переменных в ее марковском покрытии*. Таким образом, алгоритм предполагает случайное блуждание в пространстве состояний (пространстве возможных полных присваиваний), при котором каждый раз изменяется значение одной переменной, но

значения переменных свидетельства остаются зафиксированными. Полный текст алгоритма представлен на рис. 13.20.

---

function GIBBS-ASK(X, **e**, bn, N) **returns** оценка значения $\mathbf{P}(X|\mathbf{e})$
  local variables: **C**, вектор результатов подсчетов по X, исходно равен нулю
    Z, переменные в сети bn, отличные от переменных свидетельства
    x, текущее состояние сети, первоначально скопированное из **e**

  инициализировать **x** случайными значениями переменных из **Z**
  for $k = 1$ **to** N **do**
    choose значение любой переменной Z_i из **Z** в соответствии с любым распределением $\rho(i)$
    установить значение Z_i в **x** посредством выборки из $\mathbf{P}(Z_i|mb(Z_i))$
    $\mathbf{C}[j] \leftarrow \mathbf{C}[j] + 1$, где x_j является значением переменной X в **x**
return NORMALIZE(C)

Рис. 13.20. Алгоритм выборки Гиббса для приближенного вывода в байесовских сетях. В представленной версии переменные выбирают случайным образом, но вариант с циклическим перебором переменных также будет работать

Рассмотрим запрос $\mathbf{P}(Rain \mid Sprinkler = true, WetGrass = true)$ для сети, приведенной на рис. 13.15, *а*. Значения переменных свидетельства *Sprinkler* и *WetGrass* зафиксированы и равны их наблюдаемым значениям (обе имеют значение *true*), а переменные *Cloudy* и *Rain*, отличные от переменных свидетельства, инициализируются случайным образом, — скажем, *true* и *false* соответственно. Таким образом, начальным состоянием сети является [*true*, ***true***, *false*, ***true***], где фиксированные значения переменных свидетельства выделены полужирным шрифтом. Далее для переменных Z_i, отличных от переменных свидетельства, многократно формируются выборки, при этом изменяемая переменная определяется в некотором случайном порядке, соответствующем вероятностному распределению $\rho(i)$. Вот конкретный пример.

1. В качестве изменяемой выбирается переменная *Cloudy* и формируется выборка с учетом текущих значений переменных ее марковского покрытия: в данном случае выборка берется из $\mathbf{P}(Cloudy \mid Sprinkler = true, Rain = false)$. Предположим, что результатом является $Cloudy = false$, и в этом случае новым текущим состоянием становится [*false*, ***true***, *false*, ***true***].
2. Выбирается переменная *Rain* и формируется выборка с учетом текущих значений переменных ее марковского покрытия: в данном случае выборка формируется из $\mathbf{P}(Rain \mid Cloudy = false, Sprinkler = true, WetGrass = true)$. Предположим, что результатом является $Rain = true$. Новым текущим состоянием становится [*false*, ***true***, *true*, ***true***].

Одной из еще не рассмотренных деталей является метод расчета распределения марковского покрытия $\mathbf{P}(X_i \mid mb(X_i))$, где через $mb(X_i)$ обозначены значения переменных в марковском покрытии $MB(X_i)$ переменной X_i. К счастью, он не предполагает каких-либо сложных логических выводов. Как показано в упражнении 13.8, это распределение задается как

$$P(x_i \mid mb(X_i)) = \alpha P(x_i \mid parents(X_i)) \prod_{Y_j \in Children(X_i)} P(y_j \mid parents(Y_j)). \quad (13.10)$$

Иначе говоря, для каждого значения x_i вероятность задается перемножением вероятностей из таблиц условных вероятностей (CPT) для переменной X_i и ее дочерних переменных. Например, на первом этапе показанной выше последовательности действий выборка выполнялась из распределения $\mathbf{P}(Cloudy \mid Sprinkler = true, Rain = false)$. По уравнению (13.10), воспользовавшись сокращенными именами переменных, получим

$$P(c \mid s, \neg r) = \alpha P(c)P(s \mid c)P(\neg r \mid c) = \alpha\, 0{,}5 \cdot 0{,}1 \cdot 0{,}2,$$
$$P(\neg c \mid s, \neg r) = \alpha P(\neg c)P(s \mid \neg c)P(\neg r \mid \neg c) = \alpha\, 0{,}5 \cdot 0{,}5 \cdot 0{,}8,$$

поэтому распределение выборки $\alpha\langle 0{,}001;\ 0{,}020\rangle \approx \langle 0{,}048;\ 0{,}952\rangle$.

На рис. 13.21, *а* показана полная марковская цепь для случая, когда переменные выбирались при равномерном распределении, т.е. $\rho(Cloudy) = \rho(Rain) = 0{,}5$. Алгоритм будет просто блуждать по этому графу, выбирая ребра для очередного перехода с указанными вероятностями. Каждое состояние, посещенное в ходе этого процесса, представляет собой выборку, которая вносит свой вклад в оценку значения переменной запроса *Rain*. Если процесс посетит 20 состояний, в которых переменная *Rain* имеет значение *true*, и 60 состояний, в которых переменная *Rain* будет иметь значение *false*, то ответом на запрос будет $\text{NORMALIZE}(\langle 20;\ 60\rangle) = \langle 0{,}25;\ 0{,}75\rangle$.

Анализ цепей Маркова

Выше уже упоминалось, что выборка Гиббса работает посредством случайного блуждания по пространству состояний в процессе генерации выборок. Чтобы объяснить, почему выборка Гиббса работает *правильно* — т.е. почему ее оценки в пределе сходятся к правильным значениям, — нам потребуется некоторый тщательный анализ. (Этот раздел несколько математический и может быть пропущен при первом чтении.)

Начнем с некоторых основных понятий из области анализа цепей Маркова в целом. Любая такая цепь определяется своим начальным состоянием и ▶**ядром перехода** $k(\mathbf{x} \rightarrow \mathbf{x}')$ — вероятностью перехода в состояние $\mathbf{x}'$ из состояния $\mathbf{x}$. Теперь предположим, что мы запускаем цепь Маркова для выполнения t этапов, и пусть $\pi_t(\mathbf{x})$ — вероятность того, что система будет в состоянии $\mathbf{x}$ в момент времени t. Аналогичным образом пусть $\pi_{t+1}(\mathbf{x}')$ — вероятность того, что система будет в состоянии $\mathbf{x}'$ в момент времени $t+1$. Если дано значение $\pi_t(\mathbf{x})$, то значение $\pi_{t+1}(\mathbf{x}')$ можно вычислить путем суммирования по всем состояниям $\mathbf{x}$, в которых система

может находиться во время t, вероятностей пребывания в этом состоянии, умноженных на вероятности осуществления перехода в состояние $\mathbf{x}'$:

$$\pi_{t+1}(\mathbf{x}') = \sum_{\mathbf{x}} \pi_t(\mathbf{x}) k(\mathbf{x} \rightarrow \mathbf{x}') .$$

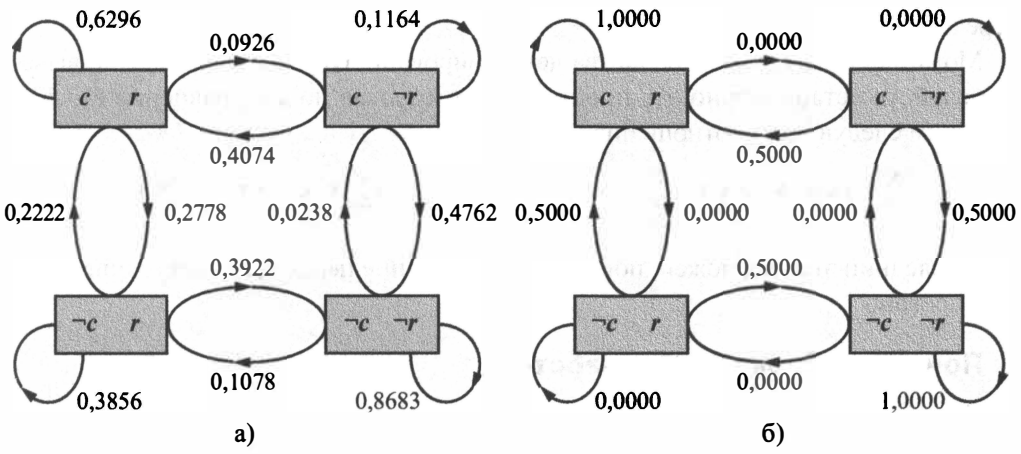


Рис. 13.21. а) Состояния и вероятности перехода марковской цепи для запроса $\mathbf{P}(Rain \mid Sprinkler = true, WetGrass = true)$. Обратите внимание на петли для отдельных состояний: состояние остается тем же самым, когда выбирается любая переменная, а затем для нее вновь выбирается то же значение, которое она уже имеет. **б)** Вероятности перехода, когда таблица условных вероятностей переменной *Rain* ограничивает ее, вынуждая иметь такое значение, как у переменной *Cloudy*

Цепь называют достигшей своего ▶**стационарного распределения** (*stationary distribution*), когда $\pi_t = \pi_{t+1}$. Обозначим это стационарное распределение как π, — оно определяется следующим уравнением:

$$\pi(\mathbf{x}') = \sum_{\mathbf{x}} \pi(\mathbf{x}) k(\mathbf{x} \rightarrow \mathbf{x}') \qquad \text{для всех } \mathbf{x}'. \tag{13.11}$$

При условии, что ядро перехода является ▶**эргодическим** — т.е. любое состояние достижимо из любого другого и в сети нет строго периодических циклов, — существует одно и только одно распределение π, удовлетворяющее данному уравнению при любом заданном k.

Уравнение (13.11) можно трактовать как утверждение, что ожидаемый "отток" из каждого состояния (т.е. его текущее "население") равен ожидаемому "притоку" из всех других состояний. Один из очевидных способов удовлетворения этого отношения состоит в достижении того, чтобы ожидаемый поток между любыми парами состояний был одинаковым в обоих направлениях, иначе говоря

$$\pi(\mathbf{x})k(\mathbf{x}\rightarrow\mathbf{x}') = \pi(\mathbf{x}')k(\mathbf{x}'\rightarrow\mathbf{x}) \quad \text{для всех } \mathbf{x}, \mathbf{x}'. \tag{13.12}$$

Когда эти уравнения выполнены, говорят, что $k(\mathbf{x}\rightarrow\mathbf{x}')$ находится в ►**детализированном равновесии** с $\pi(\mathbf{x})$. Одним из частных случаев является петля $\mathbf{x}=\mathbf{x}'$, т.е. переход из некоторого состояния в него же. В этом случае условие детализированного равновесия преобразуется в $\pi(\mathbf{x})k(\mathbf{x}\rightarrow\mathbf{x}) = \pi(\mathbf{x})k(\mathbf{x}'\rightarrow\mathbf{x})$, что, конечно же, является тождеством для любого стационарного распределения π и любого ядра перехода k.

Можно показать, что из свойства детализированного равновесия можно вывести свойство стационарности, просто просуммировав по $\mathbf{x}$ в уравнении (13.12). Получим следующее соотношение:

$$\sum_{\mathbf{x}}\pi(\mathbf{x})k(\mathbf{x}\rightarrow\mathbf{x}') = \sum_{\mathbf{x}}\pi(\mathbf{x}')k(\mathbf{x}'\rightarrow\mathbf{x}) = \pi(\mathbf{x}')\sum_{k}k(\mathbf{x}'\rightarrow\mathbf{x}) = \pi(\mathbf{x}'),$$

где последний этап возможен, поскольку выполнение перехода из состояния $\mathbf{x}'$ гарантировано.

Почему выборка Гиббса работает

Теперь покажем, что выборка Гиббса возвращает совместимые оценки для апостериорной вероятности. Основное выдвигаемое положение просто: ►*стационарное распределение процесса выборки Гиббса — это в точности апостериорное распределение переменных, отличных от переменных свидетельства, обусловленных свидетельством.* Это замечательное свойство следует из особого способа, которым процесс выборки Гиббса переходит из состояния в состояние.

Общее определение выборки Гиббса состоит в том, что сначала выбирается переменная X_i, а затем для нее формируется выборочное значение, обусловленное текущими значениями *всех* других переменных. (При применении конкретно к байесовским сетям мы просто используем тот дополнительный факт, что обусловленность выборки текущими значениями всех переменных здесь эквивалентна обусловленности выборки лишь текущими значениями переменных марковского покрытия, как было показано в разделе 13.2.1.) Мы будем использовать нотацию $\overline{\mathbf{X}_i}$ для ссылок на эти другие переменные (за исключением переменных свидетельства), и $\overline{\mathbf{x}_i}$ для ссылок на их значения в текущем состоянии.

Чтобы записать ядро перехода $k(\mathbf{x}\rightarrow\mathbf{x}')$ для выборки Гиббса, необходимо рассмотреть три случая:

1. Состояния $\mathbf{x}$ и $\mathbf{x}'$ различаются двумя или более переменными. В этом случае $k(\mathbf{x}\rightarrow\mathbf{x}') = 0$, поскольку выборка Гиббса изменяет только одну переменную.
2. Состояния $\mathbf{x}$ и $\mathbf{x}'$ различаются ровно одной переменной X_i, которая изменила свое значение с x_i на x_i'. Вероятность такой ситуации равна

$$k(\mathbf{x}\rightarrow\mathbf{x}') = k((x_i,\overline{\mathbf{x}_i})\rightarrow(x_i',\overline{\mathbf{x}_i})) = \rho(i)P(x_i'\mid\overline{\mathbf{x}_i}). \tag{13.13}$$

3. Состояния одинаковы: $\mathbf{x} = \mathbf{x}'$. В этом случае может быть выбрана *любая* переменная, но процесс выборки сформирует то же значение, которое эта переменная уже имеет. Вероятность такой ситуации равна

$$k(\mathbf{x} \to \mathbf{x}) = \sum_i \rho(i) k((x_i, \overline{\mathbf{x}_i}) \to (x_i, \overline{\mathbf{x}_i})) = \sum_i \rho(i) P(x_i \mid \overline{\mathbf{x}_i}) .$$

Теперь покажем, что это общее определение выборки Гиббса находится в детализированном равновесии со стационарным распределением, равным $P(\mathbf{x} \mid \mathbf{e})$, т.е. истинным апостериорным распределением по переменным, отличным от переменных свидетельства. Иначе говоря, покажем, что $\pi(\mathbf{x})k(\mathbf{x} \to \mathbf{x}') = \pi(\mathbf{x}')k(\mathbf{x}' \to \mathbf{x})$, где $\pi(\mathbf{x}) = P(\mathbf{x} \mid \mathbf{e})$, для всех состояний $\mathbf{x}$ и $\mathbf{x}'$.

Для первого и третьего случаев из числа приведенных выше детализированное равновесие будет всегда соблюдаться: если два состояния различаются значениями двух или более переменных, вероятность перехода в обоих направлениях равна нулю. Если $\mathbf{x} \neq \mathbf{x}'$, то из уравнения (13.13) имеем

$$\begin{aligned}
\pi(\mathbf{x})k(\mathbf{x} \to \mathbf{x}') &= P(\mathbf{x} \mid \mathbf{e})\rho(i)P(x_i' \mid \overline{\mathbf{x}_i}, \mathbf{e}) = \rho(i)P(x_i, \overline{\mathbf{x}_i} \mid \mathbf{e})P(x_i' \mid \overline{\mathbf{x}_i}, \mathbf{e}) = \\
&= \rho(i)P(x_i \mid \overline{\mathbf{x}_i}, \mathbf{e})P(\overline{\mathbf{x}_i} \mid \mathbf{e})P(x_i' \mid \overline{\mathbf{x}_i}, \mathbf{e}) = && \text{(используя цепное правило после первого терма)} \\
&= \rho(i)P(x_i \mid \overline{\mathbf{x}_i}, \mathbf{e})P(x_i', \overline{\mathbf{x}_i} \mid \mathbf{e}) = && \text{(используя цепное правило для перехода в обратном направлении)} \\
&= \pi(\mathbf{x}')k(\mathbf{x}' \to \mathbf{x}).
\end{aligned}$$

Заключительной частью головоломки является эргодичность цепи, т.е. каждое ее состояние должно быть достижимо из каждого другого и не существует ни одного периодического цикла. Оба условия будут удовлетворены, если таблицы условных вероятностей (CPT) не будут содержать вероятностей 0 или 1. Достижимость определяется тем фактом, что одно состояние можно преобразовать в другое, изменяя по одной переменной за раз, а отсутствие периодических циклов определяется тем, что каждое состояние имеет петлю с ненулевой вероятностью. Следовательно, при этих условиях k является эргодическим, а это означает, что выборки, сформированные алгоритмом выборки Гиббса, в конечном итоге будут взяты из истинного апостериорного распределения.

Сложность выборки Гиббса

Прежде всего, хорошая новость: на каждом этапе работы алгоритма выборки Гиббса требуется вычислить распределение марковского покрытия для выбранной переменной X_i, что предполагает выполнение нескольких операций умножения, количество которых пропорционально числу дочерних переменных X_i и размеру ее области определения. Это очень важно, поскольку означает, что ➔ *объем работы, необходимый для формирования каждой выборки, не зависит от размера сети.*

Теперь не обязательно плохая новость: сложность алгоритма выборки Гиббса гораздо труднее анализировать по сравнению с анализом сложности алгоритмов выборки с отклонением или взвешивания по правдоподобию. Первое, что необходимо отметить, — выборка Гиббса, в отличие от взвешивания по правдоподобию, *действительно* обеспечивает распространение свидетельства. Информация распространяется от узлов свидетельства во всех направлениях: сначала для любых соседей узлов свидетельства формируются выборочные значения, отражающие данные свидетельства в этих узлах; затем то же самое происходит с *их* соседями и т.д. Следовательно, можно ожидать, что алгоритм выборки Гиббса покажет лучшие результаты по сравнению с алгоритмом взвешивания по правдоподобию, когда данные свидетельства присутствуют в основном в узлах нижней части сети. И действительно, именно это можно отметить на рис. 13.22.

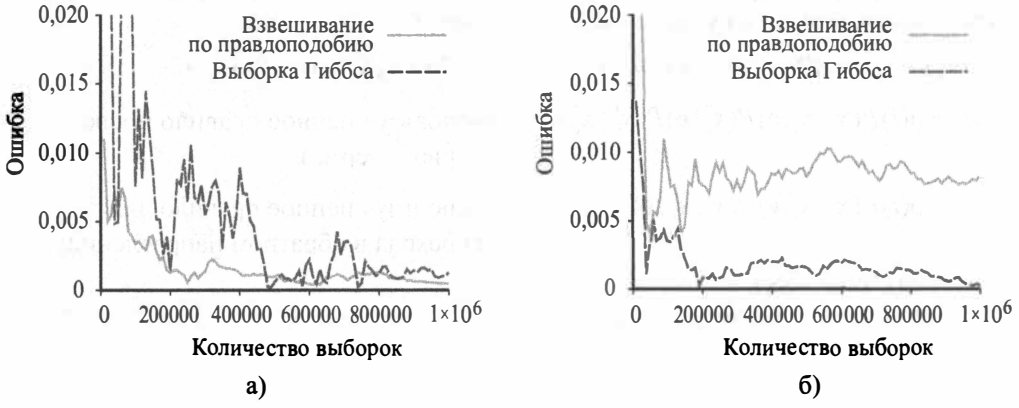


Рис. 13.22. Производительность алгоритма выборки Гиббса по сравнению с алгоритмом взвешивания по правдоподобию для сети страховой компании. **а)** Для стандартного запроса по переменной *PropertyCost*. **б)** Для случая, когда значения выходных переменных наблюдаются, а переменная запроса — *Age*

Скорость сходимости для алгоритма выборки Гиббса — ▶**скорость смешивания** марковской цепи, определяемая алгоритмом — очень сильно зависит от количественных свойств условных распределений в сети. Чтобы увидеть это, рассмотрим, что произойдет в сети, представленной на рис. 13.15, *а*, если CPT для переменной *Rain* станет детерминированной: дождь идет *тогда и только тогда*, когда небо облачное. В этом случае истинное апостериорное распределение для запроса $\mathbf{P}(Rain \mid sprinkler, wetGrass)$ будет примерно $\langle 0{,}18;\ 0{,}82\rangle$, но выборка Гиббса никогда не достигнет этого значения. Проблема состоит в том, что единственными двумя состояниями для переменных *Cloudy* и *Rain*, имеющими ненулевую вероятность, являются [*true*, *true*] и [*false*, *false*]. Начав работу с состояния

[*true*, *true*], цепь никогда не сможет достичь состояния [*false*, *false*], потому что переходы в требуемые промежуточные состояния имеют нулевую вероятность (см. рис. 13.21, *б*). В итоге, если работа начинается с состояния [*true*, *true*], результатом для запроса всегда будет апостериорная вероятность $\langle 1{,}0;\ 0{,}0\rangle$, а если работа начинается с состояния [*false*, *false*], то результатом всегда будет апостериорная вероятность $\langle 0{,}0;\ 1{,}0\rangle$.

В этом случае алгоритм выборки Гиббса терпит неудачу, потому что детерминистическая связь между переменными *Cloudy* и *Rain* нарушает свойство эргодичности, необходимое для сходимости. Однако, если сделать отношения *почти* детерминированными, то сходимость восстанавливается, но достигается произвольно медленно. Существует несколько вариантов исправлений, помогающих алгоритмам МСМС достигать смешивания быстрее. Одним из них является ▶**блочная выборка**, т.е. выборка нескольких переменных одновременно. В этом случае можно попробовать совместно сформировать выборочные значения для переменных *Cloudy* и *Rain* при условии объединения их марковских покрытий. Другой способ — генерировать очередные состояния более разумно, как будет показано в следующем подразделе.

Выборка Метрополиса–Гастингса

Метод выборки Метрополиса–Гастингса, или МН — это, вероятно, наиболее широко используемый алгоритм МСМС. Подобно выборке Гиббса, алгоритм МН предназначен для генерации выборок $\mathbf{x}$ (в конечном итоге) в соответствии с целевыми вероятностями $\pi(\mathbf{x})$. В случае вероятностного вывода в байесовских сетях требуется, чтобы $\pi(\mathbf{x}) = P(\mathbf{x} \mid \mathbf{e})$. Как и алгоритм имитации отжига (раздел 4.1.2), на каждой итерации процесса выборки алгоритм МН выполняется в два этапа.

1. Формируется новое состояние $\mathbf{x}'$ из ▶**вспомогательного распределения** $q(\mathbf{x}' \mid \mathbf{x})$ при заданном текущем состоянии $\mathbf{x}$.
2. Состояние $\mathbf{x}'$ принимается или отклоняется в соответствии с ▶**вероятностью успешного приема**

$$a(\mathbf{x}' \mid \mathbf{x}) = \min\left(1, \frac{\pi(\mathbf{x}')q(\mathbf{x} \mid \mathbf{x}')}{\pi(\mathbf{x})q(\mathbf{x}' \mid \mathbf{x})}\right).$$

3. Если предложение $\mathbf{x}'$ отклонено, сохраняется состояние $\mathbf{x}$.

Для алгоритма МН ядро перехода состоит из этого двухэтапного процесса. Обратите внимание, что если предложение отклонено, цепь остается в том же состоянии.

Именно вспомогательное распределение отвечает за то, каким будет следующее предлагаемое состояние $\mathbf{x}'$. Например, $q(\mathbf{x}' \mid \mathbf{x})$ можно определить следующим образом.

- Для генерации $\mathbf{x}'$ с вероятностью 0,95 использовать на этом этапе алгоритм выборки Гиббса.
- В противном случае для генерации $\mathbf{x}'$ использовать алгоритм WEIGHTED-SAMPLE, приведенный на рис. 13.18.

Это вспомогательное распределение вынудит алгоритм МН выполнить около 20 этапов, применяя выборку Гиббса, а затем "перезапустить" процесс из нового состояния (при условии, что оно будет принято), сформированного "с нуля". Такая уловка позволяет алгоритму МН обойти свойственную выборке Гиббса проблему "застревания" в одной части пространства состояний без возможности добраться до других его частей.

Может возникнуть вопрос, откуда, в конце концов, берется такая уверенность, что алгоритм МН с таким странным выбором предлагаемых состояний вообще сойдется в конечном счете к правильному результату? Замечательная особенность алгоритма МН состоит в том, что ➨ *сходимость к правильному стационарному распределению гарантируется для любого вспомогательного распределения* — при условии, что результирующее ядро перехода будет эргодическим.

Это свойство следует из способа, которым была определена вероятность успешного приема. Как и в случае выборки Гиббса, наличие петли с $\mathbf{x} = \mathbf{x}'$ автоматически удовлетворяет детализированному равновесию, поэтому сосредоточимся только на случае, когда $\mathbf{x} \neq \mathbf{x}'$. Очевидно, что это может иметь место, только если предложение принято. Вероятность такого перехода следующая:

$$k(\mathbf{x} \rightarrow \mathbf{x}') = q(\mathbf{x}' \mid \mathbf{x})a(\mathbf{x}' \mid \mathbf{x}).$$

Как и в случае выборки Гиббса, доказательство детализированного равновесия означает, что поток от $\mathbf{x}$ до $\mathbf{x}'$, $\pi(\mathbf{x})k(\mathbf{x} \rightarrow \mathbf{x}')$, совпадает с потоком от $\mathbf{x}'$ до $\mathbf{x}$, $\pi(\mathbf{x}')k(\mathbf{x}' \rightarrow \mathbf{x})$. После подстановки вместо $k(\mathbf{x} \rightarrow \mathbf{x}')$ приведенного выше выражения доказательство будет довольно простым.

$$\begin{aligned}
\pi(\mathbf{x})q(\mathbf{x}'|\mathbf{x})a(\mathbf{x}'|\mathbf{x}) &= \pi(\mathbf{x})q(\mathbf{x}'|\mathbf{x})\min\left(1, \frac{\pi(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}{\pi(\mathbf{x})q(\mathbf{x}'|\mathbf{x})}\right) = && \text{(определение } a(\cdot|\cdot)\text{)} \\
&= \min(\pi(\mathbf{x})q(\mathbf{x}'|\mathbf{x}), \pi(\mathbf{x}')q(\mathbf{x}|\mathbf{x}') = && \text{(выполняем умножение)} \\
&= \pi(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')\min\left(\frac{\pi(\mathbf{x})q(\mathbf{x}'|\mathbf{x})}{\pi(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}, 1\right) = && \text{(делим на второй терм)} \\
&= \pi(\mathbf{x}')q(\mathbf{x} \mid \mathbf{x}')a(\mathbf{x} \mid \mathbf{x}')
\end{aligned}$$

Оставив математические свойства в стороне, сосредоточим внимание на важной части алгоритма МН: отношении $\pi(\mathbf{x}')/\pi(\mathbf{x})$ в формуле вероятности успешного приема. Оно говорит о том, что если предлагаемое следующее состояние является *более* вероятным, чем текущее состояние, оно, определенно, будет принято. (Мы пока не обращаем внимания на терм $q(\mathbf{x} \mid \mathbf{x}')/q(\mathbf{x}' \mid \mathbf{x})$, который присутствует

там, чтобы обеспечить детализированное равновесие и во многих пространствах состояний является равным 1 вследствие симметрии.) Если предлагаемое новое состояние менее вероятно, чем текущее состояние, вероятность его принятия пропорционально уменьшается.

Таким образом, один из руководящих принципов при разработке вспомогательного распределения состоит в необходимости убедиться, что предлагаемые на его основе новые состояния будут достаточно вероятными. Алгоритм выборки Гиббса обеспечивает это автоматически: здесь в качестве вспомогательного используется распределение Гиббса $P(X_i \mid \overline{\mathbf{x}_i})$, а это означает, что вероятность формирования некоторого конкретного нового значения для X_i прямо пропорциональна его вероятности. (В упражнении 13.23 предлагается показать, что выборка Гиббса — это особый случай алгоритма МН с вероятностью принятия, равной 1).

Другая важная рекомендация состоит в необходимости убедиться, что марковская цепь хорошо смешивается, — в том смысле, что иногда должны предлагаться большие перемещения, ведущие в отдаленные области пространства состояний. В приведенном выше примере случайное использование алгоритма WEIGHTED-SAMPLE для перезапуска цепи из нового состояния служит именно этой цели.

Помимо почти полной свободы в разработке вспомогательного распределения, у алгоритма МН есть два дополнительных свойства, которые делают его практичным. Во-первых, апостериорная вероятность $\pi(\mathbf{x}) = P(\mathbf{x} \mid \mathbf{e})$ появляется в расчетах возможности принятия только в виде отношения $\pi(\mathbf{x}')/\pi(\mathbf{x})$, что очень удачно. Вычисление $P(\mathbf{x} \mid \mathbf{e})$ напрямую — это то же самое вычисление, которое мы пытаемся аппроксимировать, используя алгоритм МН, поэтому не имеет смысла делать подобные расчеты для каждого образца! Вместо этого мы используем следующий прием:

$$\frac{\pi(\mathbf{x}')}{\pi(\mathbf{x})} = \frac{P(\mathbf{x}' \mid \mathbf{e})}{P(\mathbf{x} \mid \mathbf{e})} = \frac{P(\mathbf{x}', \mathbf{e})}{P(\mathbf{e})} \frac{P(\mathbf{e})}{P(\mathbf{x}, \mathbf{e})} = \frac{P(\mathbf{x}', \mathbf{e})}{P(\mathbf{x}, \mathbf{e})} .$$

Термы в этом соотношении являются полными совместными распределениями вероятности, т.е. произведениями условных вероятностей в байесовской сети. Второе полезное свойство этого отношения состоит в том, что, пока вспомогательное распределение вносит лишь локальные изменения в $\mathbf{x}$ для получения $\mathbf{x}'$, только небольшое число членов в произведении условных вероятностей будет различаться. Все условные вероятности, включающие переменные, значения которых не изменяются, в этом соотношении будут скомпенсированы. Таким образом, как и в случае выборки Гиббса, работа, необходимая для формирования каждой выборки, не зависит от размера сети, пока изменения состояния являются локальными.

13.4.3. Вычисления при приближенном вероятностном выводе

Алгоритмы выборки, приведенные на рис. 13.17, 13.18 и 13.20, имеют общее свойство: они работают в байесовской сети, представленной в виде структуры данных. Это кажется вполне естественным: в конце концов, байесовская сеть является направленным ациклическим графом, как же иначе его можно представить? Проблема с этим подходом состоит в том, что для доступа к этим структурам данных необходимы определенные операции — например, поиск родительских узлов для заданного узла, — которые выполняются вновь и вновь тысячи или даже миллионы раз, пока продолжается выполнение алгоритма, и *все эти вычисления являются совершенно ненужными.*

Структура сети и условные вероятности остаются неизменными на всем протяжении вычислений, поэтому есть возможность *компиляции* сети в специфический для модели код логического вывода, который выполняет только те вычисления вероятностного вывода, которые необходимы именно для этой конкретной сети. (Возможно, это звучит знакомо, — та же идея использовалась при компиляции логических программ в главе 9.) Например, предположим, что требуется сформировать выборочное значение переменной *Earthquake* в сети защиты от взлома, приведенной на рис. 13.2, используя алгоритм выборки Гиббса. В соответствии с алгоритмом GIBBS-ASK, представленном на рис. 13.20, потребуется выполнить следующие вычисления:

установить значение переменной *Earthquake* в **x**, сформировав выборку из распределения $\mathbf{P}(Earthquake \mid mb(Earthquake))$,

где это распределение вычисляется в соответствии с уравнением (13.10), повторно приведенным ниже:

$$P(x_i \mid mb(X_i)) = \alpha P(x_i \mid parents(X_i)) \prod_{Y_j \in Children(X_i)} P(y_j \mid parents(Y_j)).$$

Это вычисление, в свою очередь, потребует выполнить поиск родительских и дочерних узлов переменной *Earthquake* в структуре байесовской сети и поиск их текущих значений; использовать эти значения для индексации в соответствующих таблицах CPT (которые также необходимо найти в байесовской сети) с последующим перемножением всех соответствующих строк из этих таблиц для формирования нового распределения для выборки. Наконец, как было отмечено в разделе 13.4.1, на самом этапе формирования выборочного значения потребуется построить накопительную версию дискретного распределения, а затем найти в ней значение, которое будет соответствовать случайному числу, выбранному в интервале [0, 1].

Если вместо всего этого скомпилировать сеть, можно получить специфический для конкретной модели код выборки значения переменной *Earthquake*, который будет выглядеть следующим образом:

```
r ← равномерная случайная выборка из [0, 1]
if Alarm = true
    then if Burglary = true
        then return [r < 0.0020212]
        else return [r < 0.36755]
    else if Burglary = true
        then return [r < 0.0016672]
        else return [r < 0.0014222]
```

Здесь переменные байесовской сети *Alarm*, *Burglary* и так далее становятся обычными программными переменными со значениями, представляющими текущее состояние цепи Маркова. Числовые пороговые выражения позволяют в числовом виде выразить истинность или ложность и представляют собой предвычисленные распределения Гиббса для каждой комбинации значений в марковском покрытии переменной *Earthquake*. Этот код нельзя назвать совершенным — в типичном случае он будет примерно так же велик, как и байесовская сеть сама по себе, — но он является невероятно эффективным. По сравнению с алгоритмом GIBBS-ASK, скомпилированный код будет работать, как правило, на 2–3 порядка быстрее. На обычном ноутбуке он позволяет выполнять десятки миллионов этапов выборки в секунду, и скорость его работы в значительной степени ограничивается вычислительными затратами на генерацию случайных чисел.

13.5. Причинно-следственные байесовские сети

Выше уже обсуждалось несколько важных преимуществ поддержания в байесовских сетях упорядоченности узлов, совместимой с направлением причинно-следственных связей. В частности, была отмечена легкость, с которой могут быть оценены условные вероятности, когда такое упорядочение поддерживается, а также компактность результирующей структуры сети. Однако было отмечено, что в принципе любая упорядоченность узлов обеспечивает структуру сети, пригодную для представления функции совместного распределения. Это было продемонстрировано на рис. 13.3, где изменение порядка узлов приводило к получению сетей, которые были более запутанными и намного менее естественными, чем исходная сеть, приведенная на рис. 13.2, но позволяли представить то же самое распределение по всем переменным.

В этом разделе описываются ▶ **причинно-следственные сети**, ограниченный класс байесовских сетей, в которых запрещены все совместимые упорядоченности за исключением тех, которые отражают отношения причинности. Мы рассмотрим, как строятся такие сети, какие преимущества дает подобная структура сети и как использовать эти преимущества при решении задач принятия решений.

Рассмотрим простейшую из возможных байесовских сетей с единственным ребром: $Fire \rightarrow Smoke$. Она говорит нам, что переменные *Fire* (огонь) и *Smoke* (дым) могут быть зависимы, поэтому необходимо определить априорную вероятность

$P(Fire)$ и условную вероятность $P(Smoke \mid Fire)$, чтобы задать совместное распределение $P(Fire, Smoke)$. Однако такое распределение может быть одинаково хорошо представлено и при развороте стрелки в обратную сторону: $Fire \leftarrow Smoke$ с использованием соответствующих вероятностей $P(Smoke)$ и $P(Fire \mid Smoke)$, вычисленных по правилу Байеса. Мысль о том, что эти две сети эквивалентны и, следовательно, представляют одну и ту же информацию, вызывает у большинства людей дискомфорт и даже внутреннее сопротивление. Как они могут представлять ту же информацию, если мы знаем, что огонь вызывает дым, а не наоборот?

Другими словами, из нашего опыта и научных соображений мы знаем, что простое удаление дыма не погасит огонь, тогда как, погасив огонь, мы прекратим и поступление дыма. Поэтому мы ожидаем, что именно эта асимметрия представляется направлением стрелки между вершинами графа. Но если обращение стрелки позволяет получить лишь эквивалентные результаты, каким образом можно представить эту важную информацию формально?

Причинно-следственные байесовские сети — иногда их называют причинно-следственными диаграммами — были разработаны, чтобы позволить нам представлять причинную асимметрию и использовать ее для рассуждений в отношении причинно-следственной информации. Идея состоит в том, чтобы выбирать направление стрелки по соображениям, которые выходят за пределы вероятностной зависимости и требуют совершенно другого типа суждений. Вместо того чтобы спросить у эксперта, являются ли дым и огонь вероятностно зависимыми — что и делается в обычной байесовской сети, — мы теперь спрашиваем, что за что несет ответственность: дым за огонь или огонь за дым?

Это фраза может показаться немного мистической, но ее можно уточнить с помощью понятия "присваивания", аналогичного оператору присваивания в языках программирования. Если природа присваивает значение вершине *Smoke* на основе того, что она знает о вершине *Fire*, мы можем нарисовать стрелку в направлении от *Smoke* к *Fire*. Более важно другое: если предполагается, что природа назначает переменной *Fire* значение *truth* в зависимости от значений других переменных, но не от значения переменной *Smoke*, то следует воздержаться от рисования стрелки в направлении $Fire \leftarrow Smoke$. Другими словами, значение x_i каждой переменной X_i определяется уравнением $x_i = f_i(OtherVariables)$, а стрелка $X_j \rightarrow X_i$ рисуется тогда и только тогда, когда переменная X_j является одним из аргументов функции f_i.

Уравнение $x_i = f_i(\cdot)$ называют ▶**структурным уравнением**, поскольку оно описывает устойчивый механизм в природе, который, в отличие от вероятностей, лишь количественно определяющих байесовскую сеть, остается инвариантным к измерениям и локальным изменениям в окружающей среде.

Чтобы оценить эту устойчивость к локальным изменениям, обратимся к рис. 13.23, *а*, на котором представлена слегка измененная версия сети задачи о поливе газона (см. рис. 13.15). Например, для того, чтобы представить выключенный

дождеватель, из сети достаточно просто удалить все ребра, направленные на вершину *Sprinkler*. Чтобы представить газон, покрытый палаткой, потребуется лишь удалить стрелку *Rain* → *WetGrass*. Следовательно, любое локальное изменение конфигурации механизмов в окружающей среде может быть представлено относительно небольшой модификацией в виде изоморфной реконфигурации топологии сети. Однако потребуются гораздо более сложные преобразования, если сеть была построена с нарушением причинно-следственной упорядоченности. Эта локальная стабильность особенно важна для представления действий или вмешательств, — темы, обсуждаемой в следующем разделе.

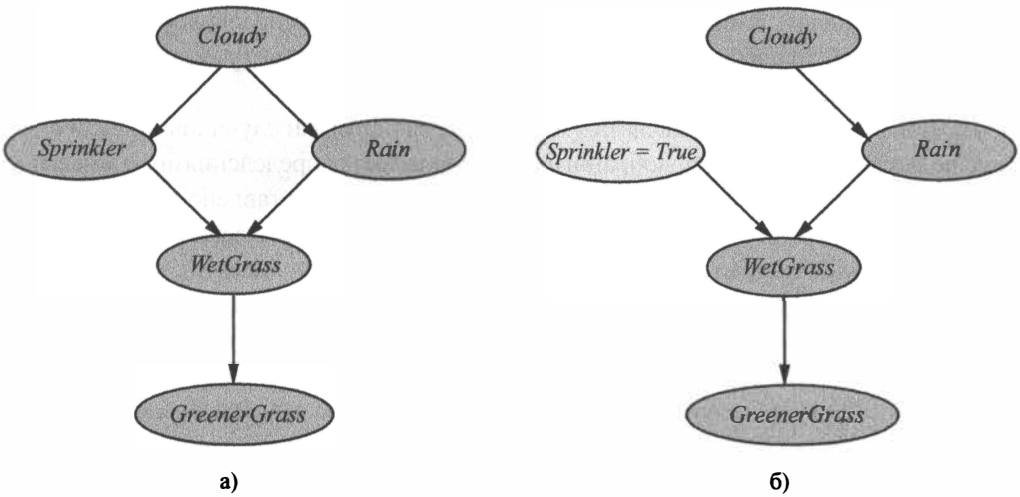


Рис. 13.23. а) Причинно-следственная байесовская сеть, представляющая причинно-следственные отношения между пятью переменными. **б)** Эта же сеть после выполнения действия "включить дождеватель"

13.5.1. Представление действий: оператор do

Еще раз обратимся к примеру причинно-следственной сети для задачи о поливе газона, представленной на рис. 13.23, *а*. В соответствии со стандартной семантикой байесовских сетей совместное распределение для ее пяти переменных определяется как произведение пяти условных распределений:

$$P(c, r, s, w, g) = P(c)\, P(r \mid c)\, P(s \mid c)\, P(w \mid r, s)\, P(g \mid w). \qquad (13.14)$$

(Здесь имя каждой переменной сокращено до его первой буквы.) Как система структурных уравнений соответствующая модель выглядит следующим образом.

$$\begin{aligned} C &= f_C(U_C) \\ R &= f_R(C, U_R) \\ S &= f_S(C, U_S) \\ W &= f_W(R, S, U_W) \\ G &= f_G(W, U_G) \end{aligned} \tag{13.15}$$

Здесь, без потери общности, f_C может быть функцией тождества. U-переменные в этих уравнениях представляют ▶**случайные составляющие** (*unmodeled variables*), также называемые **ошибками** (*error terms*) или **возмущениями** (*disturbances*), которые вносят искажения в функциональную зависимость между каждой переменной и ее родительскими переменными. Например, переменная U_W может представлять собой еще один потенциальный источник влажности травы на газоне, помимо дождевателя и дождя — скажем, *MorningDew* (утренняя роса) или *FirefightingHelicopter* (пожарный вертолет).

Если все U-переменные являются взаимно независимыми случайными величинами с подходящим образом подобранными априорными распределениями, совместное распределение в уравнении (13.14) может быть точно представлено структурными уравнениями в уравнении (13.16). Это значит, что систему стохастических отношений можно будет охватить системой детерминированных отношений, каждое из которых подвержено внешним возмущениям. Однако система структурных уравнений дает нам нечто большее: она позволяет предсказать, какое влияние *внешние воздействия* окажут на работу всей системы и, следовательно, на наблюдаемые следствия этих воздействий. Это невозможно, если дано только совместное распределение.

Например, предположим, что *дождеватель был включен*, т.е. некто (кто по определению не является частью причинных процессов, описываемых моделью) оказал на систему некоторое *внешнее воздействие*, результатом которого стало условие $Sprinkler = true$. В обозначениях ▶**do-исчисления**, которое является ключевой частью теории причинно-следственных сетей, это записывается как $do(Sprinkler = true)$. В результате такого внешнего воздействия переменная *Sprinkler* уже не будет зависеть от того, облачный ли сегодня день. Поэтому уравнение $S = f_S(C, U_S)$ удаляется из системы структурных уравнений и заменяется уравнением $S = true$, что приводит модель к такому виду.

$$\begin{aligned} C &= f_C(U_C) \\ R &= f_R(C, U_R) \\ S &= true \\ W &= f_W(R, S, U_W) \\ G &= f_G(W, U_G) \end{aligned} \tag{13.16}$$

Из этих уравнений можно получить новое совместное распределение для оставшихся переменных, обусловленное действием $do(Sprinkler = true)$:

$$P(c, r, w, g \mid do(S = true)) = P(c)\, P(r \mid c)\, P(w \mid r, s = true)\, P(g \mid w). \tag{13.17}$$

Это уравнение соответствует "искаженной" сети, представленной на рис. 13.23, *б*. Из уравнения (13.17) следует, что единственными переменными, вероятности которых изменились, являются переменные *WetGrass* и *GreenerGrass*, т.е. потомки подвергшейся внешнему воздействию переменной *Sprinkler*.

Обратите внимание на разницу между обусловленностью для *действия* $do(Sprinkler = true)$ в исходной сети и обусловленностью для *наблюдения* $Sprinkler = true$. Исходная сеть указывает, что дождеватель будет включен с меньшей вероятностью, когда погода облачная, поэтому, если *наблюдения свидетельствуют*, что дождеватель включен, это снижает вероятность того, что погода облачная. Но здравый смысл подсказывает, что если некто (действующий, так сказать, за пределами мира) подойдет и включит дождеватель, это не окажет влияния на погоду и не предоставит новой информации о том, какая погода в этот день. Как показано на рис. 13.23, *б*, данное воздействие нарушает нормальную причинную связь между погодой и дождевателем, а также препятствует любому влиянию в обратном направлении, от переменной *Sprinkler* к переменной *Cloudy*. Таким образом, обусловленность действия $do(Sprinkler = true)$ в исходном графе эквивалентна обусловленности наблюдения $Sprinkler = true$ в искаженном графе.

Аналогичный подход можно применить к анализу эффекта от действия $do(X_j = x_{jk})$ в обобщенной причинно-следственной сети с переменными $X_1, \ldots, X_n$. Эта сеть соответствует совместному распределению, определенному обычным способом (см. уравнение (13.2)):

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i)). \tag{13.18}$$

После применения действия $do(X_j = x_{jk})$ в новом совместном распределении $P_{x_{jk}}$ будет просто опущен фактор для X_j.

$$P_{x_{jk}}(x_1, \ldots, x_n) = \begin{cases} \prod_{i \neq j} P(x_i \mid parents(X_i)) = \dfrac{P(x_1, \ldots x_n)}{P(x_j \mid parents(X_j)} & \text{если } x_j = x_{jk} \\ 0 & \text{если } x_j \neq x_{jk} \end{cases} \tag{13.19}$$

Этот вывод следует из того факта, что присваивание переменной X_j определенного значения x_{jk} соответствует удалению уравнения $X_j = f_j(Parents(X_j), U_j)$ из системы структурных уравнений и замене его на $X_j = x_{jk}$. После небольших дополнительных алгебраических манипуляций можно вывести формулу для эффекта, оказываемого присваиванием значения переменной X_j на любую другую переменную X_i:

$$P(X_i = x_i \mid do(x_j = x_{jk})) = P_{x_{jk}}(X_i = x_i) = \sum_{parents(X_j)} P(x_i \mid x_{jk}, parents(X_j)) P(parents(X_j)). \tag{13.20}$$

Вероятностные слагаемые в сумме получаются посредством вычислений в исходной сети с использованием любого стандартного алгоритма вероятностного вывода. Это уравнение известно как ►**формула корректировки** (*adjustment formula*). Оно характеризует вероятностно-взвешенное среднее влияние переменной X_j и ее родительских переменных на X_i, где веса являются априорными вероятностями значений родительских переменных. Эффект от воздействия сразу на несколько переменных может быть вычислен, если предположить, что эти отдельные воздействия происходили последовательно, одно за другим, каждый раз вызывая удаление причинных влияний на очередную переменную и получение новой, искаженной модели.

13.5.2. Критерий косвенного влияния

Возможность прогнозировать эффект от любого внешнего воздействия является замечательным результатом, но при этом нам требуется точно знать необходимые условные распределения в модели, в частности $P(x_j \mid parents(X_j))$. Однако во многих реальных ситуациях потребуется слишком много усилий, чтобы получить эти сведения. Например, мы знаем, что "генетические факторы" играют определенную роль в ожирении, но пока ничего не известно о том, какие именно гены за это ответственны, как и о точном характере их воздействия. Даже в простой задаче о принятии решений в отношении полива газона Мэри (см. рис. 13.15 и 13.23, *а*) нам может быть известно, что она обращает внимание на погоду, прежде чем принять решение о включении дождевателя, но мы можем не знать, *как именно* она принимает свое решение.

Конкретная причина, вызывающая в данном случае проблемы, заключается в том, что желательно было бы предсказать эффект от включения дождевателя, оказываемый на выходные переменные, такие как *GreenerGrass*, но тогда формула корректировки (уравнение (13,20)) должна принимать во внимание не только воздействие по прямому пути от переменной *Sprinkler*, но и воздействие по другому, "обходному", пути через переменные *Cloudy* и *Rain*. Если значение переменной *Rain* известно, то этот другой путь будет заблокирован, и это дает нам подсказку, что может существовать способ записать формулу корректировки, которая будет обусловливать переменную *Rain* вместо *Cloudy*. И действительно, это вполне возможно:

$$P(g \mid do(S = true)) = \sum_r P(g \mid S = true, r)P(r). \qquad (13.21)$$

В общем, если требуется определить эффект от действия $do(X_j = x_{jk})$ на переменную X_i, ►**критерий косвенного влияния** (*back-door criterion*) позволяет написать формулу корректировки, которая обусловливает любое множество переменных $\mathbf{Z}$, которое охватывает все возможные косвенные влияния. Говоря более формально, необходимо, чтобы множество $\mathbf{Z}$ было таким, чтобы переменная X_i была условно

независима от $Parents(X_j)$ при заданных X_j и $\mathbf{Z}$. Это прямое применение метода d-разделения (см. раздел 13.2.1).

Критерий косвенного влияния является основным строительным блоком в теории причинно-следственных рассуждений, появившейся в последние два десятилетия. Эта теория предоставляет способ оспорить положения вековой статистической догмы, утверждающей, что причинную информацию нам может предоставить только ▶ **рандомизированное контролируемое испытание**. Новая теория предоставляет концептуальные инструменты и алгоритмы для причинно-следственного анализа в широком спектре не экспериментального и квазиэкспериментального окружения; для вычисления вероятностей на встречных фактических утверждениях ("Если бы вместо того произошло это, какой была бы вероятность?"), для определения, когда результаты в одной популяции могут быть перенесены на другую, и для обработки всех форм отсутствующих данных при изучении вероятностных моделей.

Резюме

В этой главе рассматривались **байесовские сети** — тщательно разработанное представление для неопределенных знаний. Байесовские сети играют роль, аналогичную той, которую выполняет логика высказываний применительно к определенным знаниям.

- Байесовская сеть — это ориентированный ациклический граф, вершины которого соответствуют случайным переменным и с каждой вершиной связано распределение условных вероятностей для этой вершины с учетом ее родительских вершин.
- Байесовские сети обеспечивают удобный способ представления отношений **условной независимости** в рассматриваемой проблемной области.
- Любая байесовская сеть определяет полное совместное распределение по своим переменным. Вероятность любого заданного присваивания значений всем переменным определяется как произведение соответствующих элементов в локальных условных распределениях. Байесовская сеть часто позволяет экспоненциально уменьшить размеры вероятностного представления по сравнению с полным совместным распределением.
- Многие условные распределения могут быть представлены компактно с помощью канонических семейств распределений. Целый ряд канонических распределений используется в **гибридных байесовских сетях**, которые могут включать как дискретные, так и непрерывные переменные.
- Под вероятностным выводом в байесовских сетях подразумевается вычисление распределения вероятностей множества переменных запроса, если дано множество переменных свидетельства. Алгоритмы точного вероятностного вывода, такие как алгоритм **устранения переменной**, позволяют вычислять

суммы произведений условных вероятностей настолько эффективно, насколько это возможно.

- В **полидеревьях** (односвязных сетях) точный вероятностный вывод требует времени, линейно зависящего от размера сети. А в общем случае проблема такого вывода неразрешима.
- Методы случайной выборки, такие как **выборка по значимости** и **метод Монте-Карло на основе цепи Маркова**, позволяют получить приемлемые оценки истинных апостериорных вероятностей в сети и позволяют достичь желаемых результатов для гораздо более крупных сетей по сравнению с точными алгоритмами вероятностного вывода.
- В то время как байесовские сети представляют вероятностные влияния, **причинно-следственные сети** отображают причинно-следственные связи между переменными и позволяют прогнозировать эффекты от внешних воздействий в той же мере, что и наблюдения.

Библиографические и исторические заметки

Использование сетей для представления вероятностной информации началось в первой четверти XX века с работы Сьюэлла Райта по вероятностному анализу генетического наследования и показателей роста животных (Райт [2387], 1921; [2389], 1934). И. Дж. Гуд ([891], 1961) в сотрудничестве с Аланом Тьюрингом разработал вероятностные представления и методы байесовского вероятностного вывода, которые можно рассматривать как предшествующие современным байесовским сетям, хотя указанная статья не часто цитируется в данном контексте.[7] Эта же статья является оригинальным источником с описанием модели зашумленного OR.

Форма представления задач принятия решений с помощью **диаграммы влияния**, встроенная в представление DAG для случайных переменных, использовалась в области анализа принятия решений с конца 1970-х годов (см. главу 16), но для оценки применялись только методы перебора. Джуди Перл разработал метод передачи сообщений для осуществления вероятностного вывода в древовидных сетях (Перл [1749], 1982) и ввел понятие полидеревьев сетей (Ким и Перл [1225], 1983), а также объяснил важность составления причинных, а не диагностических вероятностных моделей. Первой экспертной системой, в которой использовались байесовские сети, была CONVINCE (Ким [1224], 1983).

Как упоминалось в историческом обзоре в главе 1, в середине 1980-х годов возник настоящий бум экспертных систем на основе правил, в которые были

[7] И. Дж. Гуд был главным статистиком в группе Тьюринга, занимавшейся раскрытием шифров во время Второй мировой войны. В книге "2001: Космическая одиссея" (Кларк [449], 1968) выражена благодарность Гуду и Мински за их вклад в научный прорыв, приведший к разработке компьютера HAL 9000.

включены специальные методы для работы с неопределенностью, — вероятность в качестве основы для рассуждений расценивалась и как непрактичная, и как "познавательно неправдоподобная". Воинственная статья Питера Чизмена "В защиту вероятности" ([405], 1985) и более поздняя его статья "Исследование компьютерного понимания" ([406], 1988, с комментариями) помогли изменить это отношение.

Однако возрождение интереса к вероятности определялось главным образом разработкой Перлом теории байесовских сетей и вызванного этим широкого применения вероятностного подхода в области ИИ, как это было описано в его книге *Probabilistic Reasoning in Intelligent Systems* (Перл [1755], 1988). В книге рассматривались как вопросы представления, включая отношения условной независимости и критерий d-разделения, так и алгоритмические подходы. Гейгер и соавт. ([825], 1990), а также Тиан и соавт. ([2215], 1998) представили ключевые вычислительные результаты по эффективному обнаружению d-разделения.

Представить идеи Перла исследователям в области искусственного интеллекта помог Юджин Чарняк — благодаря популярной статье "Байесовские сети без слез" ([398], 1991)[8] и книге [393] (1993). Книга Дина и Веллмана [572] (1991) также способствовала ознакомлению исследователей в области ИИ с байесовскими сетями. Позднее Шахтер ([2029], 1998) представил упрощенный способ определения d-разделения, названный алгоритмом "байесовского шара".

По мере разработки приложений с применением байесовских сетей исследователи столкнулись с необходимостью выйти за пределы базовой модели, допускающей лишь дискретные переменные в таблицах условной вероятности (CPT). Например, в медицинской системе CPCS (Прадхан и др. [1819], 1994), использовалась байесовская сеть, насчитывающая 448 вершин и 906 ребер, — в ней широко применялись зашумленные логические операторы, предложенные Гудом ([891], 1961). Бутилье и соавт. ([270], 1996) проанализировали алгоритмические преимущества независимости от контекста. Включение непрерывных случайных величин в байесовские сети рассматривалось Перлом ([1755], 1988) и Шахтером и Кенли ([2031], 1989); в этих работах обсуждались сети, содержащие только непрерывные переменные с линейным распределением Гаусса.

Гибридные сети как с дискретными, так и с непрерывными переменными исследовались Лориценом и Вермутом ([1362], 1989) и были реализованы в системе cHUGIN (Олесен [1710], 1993). Дальнейший анализ линейно-гауссовых моделей с привязкой ко многим другим моделям, используемым в статистике, был представлен Роуисом и Гахрамани ([1923], 1999), а Лернер ([1388], 2002) дал очень подробное обсуждение их использования в гибридных байесовских сетях. Разработку пробит-распределения обычно приписывают Гэддуму ([803], 1933) и Блиссу

[8] Оригинальная версия статьи имела другое название: "Pearl for swine", т.е. "Жемчуг для свиней" — здесь обыгрывается дословный перевод с английского фамилии Джуди Перла: *pearl* означает "жемчужина".

([229], 1934), хотя оно несколько раз упоминалось еще в XIX веке. Позднее работа Блисса была значительно расширена Финни ([742], 1947). Пробит-распределение широко использовалось для моделирования процессов дискретного выбора и может быть расширено для работы с более чем двумя выборами (Даганцо [514], 1979). Модель экспит-распределения (или инверсного логит-распределения) была представлена Берксоном ([182], 1944), — в начале к ней относились насмешливо, но в итоге она стало более популярной, чем модель пробит-распределения. Бишоп ([221], 1995) дал простое обоснование его использования.

К ранним приложениям на основе байесовских сетей в медицине относятся система MUNIN, предназначенная для диагностики нервно-мышечных нарушений (Андерсен и др. [46], 1989), и система PATHFINDER, предназначенная для работы с патологиям (Хекерман [999], 1991). К приложениям в области инженерии относятся разработки института *Electric Power Research* по мониторингу силовых генераторов (Морхарья и др. [1625], 1995), разработки НАСА по отображению информации, критической по времени, в Центре управления полетами в Хьюстоне (Хорвиц и Барри [1066], 1995), а также общее поле **сетевой томографии**, направленной на вывод ненаблюдаемых локальных свойств узлов и связей в Интернете на основе наблюдений производительности передачи сообщений от источника к получателю (Кастро и др. [380], 2004). Пожалуй, наиболее широко используемыми системами с байесовскими сетями стали модули диагностики и восстановления (например, модуль Printer Wizard) в операционной системе Microsoft Windows (Бриз и Хекерман [297], 1996) и приложение Office Assistant в пакете Microsoft Office (Хорвиц и др. [1067], 1998).

Другой важной областью применения является биология: математические модели, используемые для анализа генетического наследования в генеалогических деревьях (так называемый ►**анализ родословной**), в действительности представляют собой специальную форму байесовских сетей. Этот метод точного вероятностного вывода, используемый в анализе родословных и напоминающий метод устранения переменной, был разработан в 1970-х годах (Каннингс и др. [362], 1978). Также байесовские сети использовались для идентификации генов человека посредством ссылок на гены мыши (Чжанг и др. [2431], 2003), при исследовании клеточных сетей (Фридман [793], 2004), в анализе генетических связей с целью определения местонахождения генов, связанных с определенным заболеванием (Зильберштейн и др. [2060], 2013), а также для решения многих других задач в области биоинформатики. Мы могли бы продолжить, но вместо этого отсылаем читателя к работе Пурре и соавт. [1818] (2008) — руководству по применению байесовских сетей объемом в 400 страниц. В последнем десятилетии количество публикаций о разработке подобных приложений измеряется десятками тысяч, от стоматологии до глобальных климатических моделей.

Джуди Перл ([1752], 1985) в первой статье, в которой использовался термин "байесовские сети", кратко описал алгоритм вероятностного вывода для сетей общего вида, основанный на идее обусловленности сечением, представленной в главе 6. Независимо от него Росс Шахтер ([2028], 1986), работавший в сообществе

исследователей диаграмм влияния, разработал полный алгоритм, основанный на целенаправленном сокращении сети с использованием апостериорных преобразований.

Перл ([1753], 1986) разработал алгоритм кластеризации для точного вероятностного вывода в байесовских сетях общего вида, используя метод преобразования в ориентированное полидерево кластеров, в котором для достижения согласованности по переменным, разделяемым между кластерами, использовалась передача сообщений. Аналогичный подход, разработанный статистиками Дэвидом Шпигельхальтером и Штеффеном Лауритценом ([1361], 1988), основан на преобразовании графической модели в неориентированную форму, называемую **цепью Маркова**. Этот подход реализован в системе HUGIN — эффективном и широко применяемом инструментальном средстве формирования рассуждений в условиях неопределенности (Андерсен и др. [46], 1989).

Основная идея метода устранения переменной — что повторных вычислений в пределах выражений, включающих суммирование произведений, можно избежать за счет кеширования — появилась в алгоритме символического вероятностного вывода (*Symbolic Probabilistic Inference*, SPI) (Шахтер и соавт. [2030], 1990). Приведенный в данной книге вариант больше всего напоминает алгоритм, разработанный Чжангом и Пулом ([2432], 1994). Критерии отсечения нерелевантных переменных были разработаны Гейгером и соавт. ([826], 1990), а также Лоритценом и соавт. ([1360], 1990), — приведенный в данной книге критерий представляет собой простой частный случай этих критериев. Рина Дехтер ([580], 1999) показала, что идея устранения переменной по сути идентична ▶**непоследовательному динамическому программированию** (Бертеле и Бриоши [196], 1972).

В этом подходе алгоритмы байесовских сетей применяются в сочетании с методами решения задач УО и дается прямая оценка меры сложности точного вывода в терминах ширины дерева сети. Предотвращение экспоненциального роста размера факторов в методе устранения переменной может быть реализовано посредством сбрасывания переменных из крупных факторов (Дехтер и Риш [584], 2003); также возможно связать введенную этим ошибку (Векслер и Мик [2328], 2009). В качестве альтернативы факторы могут быть сжаты за счет представления их с использованием алгебраических диаграмм принятия решений вместо таблиц (Гогат и Домингос [877], 2011).

Точные методы, основанные на рекурсивном перечислении (см. рис. 13.11) в сочетании с кешированием, включают алгоритм рекурсивного преобразования (Дарвич [524], 2001), алгоритм с отклонением значений (Бахус и др. [102], 2003) и AND–OR поиск (Дехтер и Матисс [588], 2007). Метод взвешенной модели подсчета (Санг и др. [1975], 2005; Чавира и Дарвич [403], 2008) обычно строится по принципу решателей SAT в стиле DPLL (рис. 7.17, раздел 7.6.1). Аналогично им в нем также выполняется рекурсивное перечисление переменных с кешированием, поэтому данный подход на самом деле очень похож на них. Все три эти алгоритма позволяют реализовать полный диапазон компромиссов затрат пространства/времени.

Поскольку в них рассматриваются присваивания переменных, эти алгоритмы легко могут использовать преимущества детерминизма и специфической для контекста независимости в модели. Они могут также быть модифицированы для использования эффективных алгоритмов с линейной временной зависимостью всякий раз, когда частичное присваивание превращает оставшуюся часть сети в полидерево. (Это версия **метода разрыва цикла**, который был описан для задач УО в главе 6.) Для точного вероятностного вывода в больших моделях, в которых пространственные требования методов кластеризации и устранения переменной становятся чрезмерными, эти рекурсивные алгоритмы часто являются наиболее практичным выбором.

Помимо вычисления предельных вероятностей, в байесовских сетях есть и другие важные задачи вероятностного вывода. ►**Наиболее вероятное объяснение** или MPE — это наиболее вероятное присваивание для переменных, отличных от переменных свидетельства. (MPE является частным случаем MAP — *maximum a posteriori* (максимальный апостериорный) — вероятностный вывод, запрашивающий наиболее вероятное присваивание *подмножеству* переменных, отличных от переменных свидетельства, с учетом свидетельства.) Для таких задач было разработано много различных алгоритмов, некоторые из которых связаны с алгоритмами кратчайшего пути или поиска AND–OR (краткое изложение приведено в работе Маринеску и Дехтера [1493] (2009)).

Первый результат по вопросу сложности вероятностного вывода в байесовских сетях принадлежит Куперу ([474], 1990), который показал, что общая проблема вероятностного вывода в байесовских сетях без ограничений является NP-трудной. Как было отмечено в данной главе, этот результат может быть усилен до #P-трудности через сокращение от подсчета удовлетворяющих присваиваний (Рот, 1996). Это также подразумевает NP-сложность приблизительного вывода (Дагум и Луби [515], 1993). Однако для случая, когда вероятности могут быть отделены от 0 и 1, форма алгоритма взвешивания по правдоподобию сходится за (рандомизированное) полиномиальное время (Дагум и Луби [516], 1997). Шимони ([2051], 1994) показал, что поиск наиболее вероятного объяснения является NP-полной, т.е. неразрешимой задачей, но несколько более простой, чем вычисления без ограничений. В то же время Парк и Дарвич ([1732], 2004) предоставили тщательный анализ сложности расчета MAP и показали, что эта задача попадает в класс NP^{PP}-полных задач, т.е. она несколько сложнее, чем вычисления без ограничений.

Разработка быстрых аппроксимирующих алгоритмов для вероятностного вывода в байесовских сетях представляет собой очень активную научную область, испытывающую положительное влияние со стороны статистики, компьютерных наук и физики. Способ формирования выборок с отклонением представляет собой общий метод, давно известный статистикам, — его история восходит по крайней мере к задаче Бюффона об игле ([342], 1777). Впервые он был применен к байесовским сетям Максом Хенрионом ([1011], 1988), который назвал этот метод **логическим формированием выборок**. Метод взвешивания по правдоподобию изначально был предложен для использования в области физики (Кан [1169–1170],

1950). К байесовским сетям его впервые применили Фунг и Чанг ([802], 1989) (называвшие этот алгоритм "взвешиванием свидетельства"), а также Шахтер и Пеот ([2032], 1989).

В статистике **адаптивная выборка** применяется ко всем видам алгоритмов Монте-Карло с целью ускорения сходимости. Основная идея состоит в том, чтобы адаптировать распределение, из которого формируются выборки, на основе результатов предыдущих выборок. Гилкс и Вайлд ([857], 1992) разработали адаптивную выборку с отклонением, в то время как адаптивная выборка по значимости, по-видимому, появилась независимо в физике (Лепаж, [1387] 1978), гражданском строительстве (Карамчандани и др. [1187], 1989), статистике (О и Бергер [1707], 1992), а также в компьютерной графике (Вич и Гуибас [2267], 1995). Ченг и Друздзел ([415], 2000) описывают адаптивный вариант выборки по значимости, применяемый к вероятностному выводу в байесовских сетях. Относительно недавно Ли и соавт. ([1368], 2017) продемонстрировали использование систем глубокого обучения для создания пропозиционального распределения, ускоряющего работу алгоритма выборки по значимости на несколько порядков.

Развитие алгоритмов Монте-Карло на основе цепи Маркова (MCMC) началось с алгоритма Метрополиса, описанного в статье [1560], 1953), которая стала также первой публикацией об алгоритме эмуляции отжига (см. главу 4). Позднее Гастингс ([982], 1970) ввел в исходный алгоритм этап "принять/отклонить", который теперь является неотъемлемой частью того, что ныне называют алгоритмом Метрополиса–Гастингса. Метод выборки Гиббса был предложен Геманом и Геманом ([833], 1984) для вероятностного вывода в неориентированных сетях Маркова. Применение выборки Гиббса к байесовским сетям было предложено Перлом ([1754], 1987). Статьи, собранные Гилксом и соавт. ([854], 1996), охватывают как теорию, так и применение алгоритмов семейства MCMC.

К середине 1990-х годов алгоритмы MCMC уже стали рабочей лошадкой байесовской статистики и различных статистических вычислений во многих других дисциплинах, включая физику и биологию. Справочник *Handbook of Markov Chain Monte Carlo* (Брукс и др. [317], 2011) охватывает многие аспекты по этой литературе. Пакет BUGS (Гилкс и др. [855], 1994) был ранней и влиятельной системой для моделирования байесовской сети и выполнения вероятностного вывода с использованием выборки Гиббса. Система STAN (названная в честь Станислава Улама, создателя методов Монте-Карло в физике) более новая, использующая метод вывода Монте-Карло Гамильтона (Карпентер и др. [375], 2017).

Есть два очень важных семейства методов аппроксимации, которые не были рассмотрены в этой главе. Первым из них является семейство методов **вариационной аппроксимации**, которые могут использоваться для упрощения сложных вычислений любых типов. Основная идея состоит в том, что необходимо предложить сокращенную версию первоначальной задачи, с которой будет легче работать, но которая напоминает первоначальную задачу настолько близко, насколько это возможно. Сокращенная задача описывается с помощью некоторых **вариационных**

параметров λ, которые корректируются с целью минимизации функции расстояния D между оригинальной и сокращенной задачами, часто путем решения системы уравнений $\partial D/\partial\lambda = 0$. Во многих случаях могут быть получены строгие верхние и нижние границы. Вариационные методы уже давно использовались в статистике (Рустаги [1950], 1976). В статистической физике метод **поля осредненных величин** (*mean field*) представляет собой особую вариационную аппроксимацию, в которой предполагается, что отдельные переменные, входящие в состав модели, являются полностью независимыми.

Эта идея была применена для поиска решений в крупных неориентированных сетях Маркова (Петерсон и Андерсон [1781], 1987; Паризи [1730], 1988). Саул и др. ([1982], 1996) представили результаты разработки математических основ применения вариационных методов к байесовским сетям и получения точных аппроксимаций нижней границы для сигмоидальных сетей с использованием методов поля осредненных величин. После этих первых работ вариационные методы были применены для многих конкретных семейств моделей. В замечательной статье Уайнрайта и Джордана ([2285], 2008) предоставляется общий теоретический анализ литературы по вариационным методам.

Второе важное семейство алгоритмов аппроксимации основано на алгоритме Перла передачи сообщений в полидереве ([1749], 1982). Как указал Перл ([1755], 1988), этот алгоритм может применяться к "петельчатым" сетям общего типа. Иногда результаты могут оказаться неверными или алгоритм не может нормально завершить работу, но во многих случаях полученные значения близки к истинным. Так называемый подход с ►**циклическим распространением оценок уверенности** привлекал мало внимания до тех пор, пока Макэлис и др. ([1547], 1998) не установили, что эти расчеты полностью аналогичны вычислениям, выполняемым в алгоритме ►**турбодекодирования** (*turbo decoding*) (Берроу и др. [194], 1993), который стал крупным научным прорывом в области разработки эффективных кодов с коррекцией ошибок.

Из этого следует вывод, что способ циклического распространения быстро и точно работает в очень крупных и тесно связанных сетях, используемых для декодирования, и поэтому может найти более широкое применение. Теоретическая поддержка этих результатов, включая доказательства сходимости для некоторых особых случаев, была предоставлена Вейссом ([2306], 2000), Вейссом и Фрименом ([2307], 2001), а также Йедидой и др. ([2402], 2005) и опирается на связь с идеями статистической физики.

Теории причинно-следственных связей, выходящие за рамки рандомизированных контрольных экспериментов, были предложены Рубином ([1926], 1974) и Робинсом ([1898], 1986), но эти идеи оставались неясными и противоречивыми, пока Джуда Перл не разработал и не представил полностью сформулированную теорию причинности, основанную на причинно-следственных сетях (Перл [1756], 2000). Петерс и соавт. ([1779], 2017) предоставили дальнейшее развитие этой теории с

акцентом на обучение. Более поздняя работа, *The Book of Why* (Перл и Маккензи [1757], 2018), предлагает менее математическое, но более читаемое и широкое введение в эту область.

Неопределенные рассуждения в ИИ не всегда основывались на теории вероятностей. Как отмечалось в главе 12, интерес к ранним вероятностным системам значительно снизился в начале 1970-х годов, а образовавшийся вакуум был частично заполнен альтернативными методами. К ним относятся основанные на правилах экспертные системы, теория Демпстера–Шейфера и (в некоторой степени) нечеткая логика.[9]

Основанные на правилах подходы к неопределенности надеялись развить успех логических систем, основанных на правилах, но добавляли своего рода "фактор выдумки" — более вежливо называемый **фактором уверенности** — к каждому правилу, чтобы включить в него некоторую неопределенность. Первой такой системой была MYCIN (Шортлайф [2056], 1976), медицинская экспертная система по бактериальным инфекциям. Коллекция *Rule-Based Expert Systems* (Бучнан и Шортлайф [338], 1984) предоставляет полный обзор системы MYCIN и ее потомков (см. также Штефик [2125], 1995).

Дэвид Хекерман ([998], 1986) показал, что в одних случаях слегка модифицированная версия на основе вычислений с фактором определенности позволяет получить правильные вероятностные результаты, но в других случаях приводит к серьезной переоценке важности свидетельств. Если набор правил увеличивается, нежелательные взаимодействия между правилами получают большее распространение, и практикующий обнаруживает, что определенные факторы многих других правил требуется "подправить", когда в систему добавляются новые правила. Основные математические свойства, которые допускают *цепочки* логических рассуждений, просто не подходят для вероятности.

Теория Демпстера–Шефера впервые была представлена в статье Артура Демпстера ([599], 1968), предложившего обобщение способа представления вероятностей в виде интервальных значений и обосновавшего правила комбинирования для их использования. Такой подход позволяет уменьшить сложность точного определения вероятностей. После опубликования более поздней статьи Гленна Шефера ([2033], 1976) теория Демпстера–Шефера стала рассматриваться как подход, способный конкурировать с вероятностным подходом. Перл ([1755], 1988) и Руспини и соавт. ([1935], 1992) проанализировали связь между теорией Демпстера–Шефера и стандартной теорией вероятностей. Во многих случаях теория вероятностей не требует, чтобы вероятности были определены точно: мы можем выразить неопределенность относительно значений вероятности в виде вероятностного распределения (второго порядка), как это объясняется в главе 20.

[9] В четвертом подходе, **рассуждения по умолчанию**, выводы рассматриваются не как "верим в определенной степени", а как "верим, пока не появится лучшая причина верить во что-то другое". Этот подход обсуждался в главе 10.

Нечеткие множества были разработаны Лотфи Задэ ([2418], 1965) в целях устранения общепризнанных сложностей предоставления точных входных данных для интеллектуальных систем. Нечеткое множество является таким множеством, членство в котором — это вопрос степени. **Нечеткая логика** — это метод рассуждения с логическими выражениями, описывающими членство в нечетких множествах. **Нечеткий контроль** является методологией построения управляющих систем, в которых отображение между реальными значениями входных и выходных параметров представлено нечеткими правилами. Нечеткий контроль уже проявил себя очень успешно во многих коммерческих продуктах, таких как автоматические коробки передач, видеокамеры или электрические бритвы. Учебник Циммермана ([2447], 2001) представляет собой исчерпывающее введение в теорию нечетких множеств, а статьи по приложениям нечетких множеств собраны в работе Циммермана [2446] (1999).

Нечеткую логику часто трактуют неправильно, усматривая в ней прямого конкурента теории вероятностей, тогда как в этой теории фактически рассматриваются совсем другие вопросы: вместо учета неопределенности в отношении к истинности четко определенных высказываний нечеткая логика работает с **неопределенностью** в отображении термов символической теории на реальный мир. Подобная неопределенность является реальной проблемой при любом применении логики, вероятностей и даже стандартных математических моделей к реальности. Даже такая, казалось бы, безупречная величина, как масса Земли, при проверке оказывается изменяющейся во времени за счет непрерывного падения метеоритов и диссипации молекул атмосферы в космос. Кстати, последнее утверждение также неточно, — масса нашей планеты включает в себя ее атмосферу? Если да, то до какой высоты? В некоторых случаях дальнейшая разработка модели может снизить подобную неопределенность, но нечеткая логика принимает такие неясности как данность и развивает вокруг этого свою теорию.

Для вычислений с учетом неопределенности в нечетких системах была предложена ▶**теория возможностей** (*possibility theory*) (Задэ [2419], 1978). Она имеет много общего с теорией вероятностей (Дюбуа и Праде [658], 1994).

Многие исследователи ИИ в 1970-х годах отвергали вероятность, поскольку численные расчеты, которые, как считалось, требовались в теории вероятностей, не были достаточно очевидны для самоанализа и предполагали нереалистичный уровень точности в наших неопределенных знаниях. Разработка **качественных вероятностных сетей** (Веллман [2316], 1990), предоставила чисто качественную абстракцию байесовских сетей, используя понятие о положительных и отрицательных влияниях между переменными. Веллман показал, что во многих случаях такой информации будет вполне достаточно для принятия оптимального решения без необходимости точного указания вероятностных значений. Гольдшмидт и Перл ([883], 1996) использовали аналогичный подход. В работе Дарвича и Гинзберга [525] (1992) извлекаются основные свойства обусловленности и комбинации

свидетельств из теории вероятностей и показано, что они также могут быть применены в логических рассуждениях и рассуждениях по умолчанию.

Несколько отличных учебников (Йенсен [1135], 2007; Дарвич [526], 2009; Коллер и Фридман [1266], 2009; Корб и Никольсон [1282], 2010; Дехтер [586], 2019) предоставляют подробное рассмотрение всех тем, которые обсуждались в этой главе. Новости об исследованиях в области вероятностных рассуждений публикуются как в основных журналах по ИИ, таких как *Artificial Intelligence* и *Journal of AI Research*, так и в более специализированных журналах, таких как *International Journal of Approximate Reasoning*. Многие статьи о графических моделях, которые включают байесовские сети, появляются в статистических журналах. Материалы конференций *Uncertainty in Artificial Intelligence* (UAI), *Neural Information Processing Systems* (NeurIPS) и *Artificial Intelligence and Statistics* (AISTATS) также являются хорошими источниками информации о текущих исследованиях.

Упражнения

13.1. Имеется мешок с тремя поддельными монетами, a, b и c, с вероятностью выпадения орла 20%, 60% и 80% соответственно. Одна из монет случайным образом достается из мешка (с равной вероятностью выбора каждой из трех монет), а затем подбрасывается три раза с получением результатов X_1, X_2 и X_3.

а) Нарисуйте байесовскую сеть, соответствующую этим условиям, и определите необходимые таблицы условной вероятности (CPT).

б) Подсчитайте, какая монета, вероятнее всего, была извлечена из пакета, если в результате бросков два раза выпал орел и один раз — решка.

13.2. Имеется мешок с тремя поддельными монетами, a, b и c, с вероятностью выпадения орла 30%, 60% и 75% соответственно. Одна из монет случайным образом достается из мешка (с равной вероятностью выбора каждой из трех монет), а затем подбрасывается три раза с получением результатов X_1, X_2 и X_3.

а) Нарисуйте байесовскую сеть, соответствующую этим условиям, и определите необходимые таблицы условной вероятности (CPT).

б) Подсчитайте, какая монета, вероятнее всего, была извлечена из пакета, если в результате бросков два раза выпал орел и один раз — решка.

13.3. Уравнение (13.1) в разделе 13.2 определяет совместное распределение, представленное байесовской сетью, в терминах параметров $\theta(X_i \mid Parents(X_i))$. Из данного определения выведите эквивалентность между этими параметрами и условными вероятностями $\mathbf{P}(X_i \mid Parents(X_i))$.

а) Рассмотрите простую сеть $X \rightarrow Y \rightarrow Z$ с тремя булевыми переменными. Используйте уравнения (12.3) в разделе 12.2.1 и (12.7) в разделе 12.3, чтобы выразить условную вероятность $P(z \mid y)$ в виде отношения двух сумм, в каждой из которых суммируются элементы совместного распределения $\mathbf{P}(X, Y, Z)$.

б) Теперь используйте уравнение (13.1), чтобы записать это выражение в терминах параметров сети $\theta(X)$, $\theta(Y \mid X)$ и $\theta(Z \mid Y)$.

в) Далее расширьте суммирование в выражении, полученном в п. б, явно записав термы для значений *true* и *false* каждой суммируемой переменной. Предполагая, что все параметры сети удовлетворяют ограничению $\sum_{x_i} \theta(x_i \mid Parents(X_i)) = 1$, покажите, что полученное выражение сводится к $\theta(x \mid y)$.

7) Обобщите этот вывод, чтобы показать, что $\theta(X_i \mid Parents(X_i)) = \mathbf{P}(X_i \mid Parents(X_i))$ для любой байесовской сети.

13.4. Операция **инверсии ребра** в байесовской сети позволяет изменять направление ребра $X \rightarrow Y$, сохраняя совместное распределение вероятностей, представляемое этой сетью. Для инверсии ребра может потребоваться введение новых ребер: все родители X также становятся родителями Y, а все родители Y также становятся родителями X.

а) Предположим, что X и Y исходно имеют m и n родителей соответственно и что все переменные имеют k значений. Рассчитав изменение размера таблиц CPT для X и Y, покажите, что общее количество параметров в сети не может уменьшиться в процессе инверсии ребра. (*Подсказка.* Родители X и Y не должны быть непересекающимися.)

б) При каких обстоятельствах общее число параметров может оставаться постоянным?

в) Пусть родителями X будут $U \cup V$, а родителями Y будут $V \cup W$, где множества U и W не пересекаются. Формулы для новых таблиц CPT после инверсии ребра будут следующими:

$$\mathbf{P}(Y \mid \mathbf{U}, \mathbf{V}, \mathbf{W}) = \sum_x \mathbf{P}(Y \mid \mathbf{V}, \mathbf{W}, x)\mathbf{P}(x \mid \mathbf{U}, \mathbf{V}),$$

$$\mathbf{P}(X \mid \mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{Y}) = \mathbf{P}(Y \mid \mathbf{X}, \mathbf{V}, \mathbf{W})\mathbf{P}(X \mid \mathbf{U}, \mathbf{V})/\mathbf{P}(Y \mid \mathbf{U}, \mathbf{V}, \mathbf{W}).$$

Докажите, что новая сеть выражает то же самое совместное распределение по всем переменным, что и исходная сеть.

13.5. Обратимся к байесовской сети, представленной на рис. 13.2.

а) Если не наблюдается никаких свидетельств, будут ли независимыми переменные *Burglary* и *Earthquake*? Обоснуйте свой ответ, исходя из числовой семантики и топологической семантики.

б) Если наблюдаемое значение переменной *Alarm* = *true*, будут ли независимыми переменные *Burglary* и *Earthquake*? Обоснуйте свой ответ, рассчитав, удовлетворяют ли значения соответствующих вероятностей определению условной независимости.

13.6. Предположим, что в байесовской сети, содержащей ненаблюдаемую переменную Y, наблюдаются все переменные ее марковского покрытия $MB(Y)$.

а) Докажите, что удаление вершины Y из сети не повлияет на апостериорное распределение для любой другой ненаблюдаемой переменной в сети.

б) Поясните, можно ли удалить вершину Y, если планируется использовать алгоритмы: *а*) выборки с отклонением; *б*) взвешивания по правдоподобию.

13.7. Пусть H_x — случайная величина, обозначающая характеристику отдельного x, с возможными значениями l (левша) или r (правша). Распространена гипотеза, что наследование характеристики левши или правши определяется простым механизмом, т.е., возможно, существует ген G_x, также со значениями l или r, и, возможно, фактическая характеристика индивида в основном будет такой же (с некоторой вероятностью s), что и значение гена, которым обладает данный индивид. Кроме того, возможно, что этот ген с равной вероятностью наследуется от любого из родителей индивида с небольшой ненулевой вероятностью m случайной мутации, переворачивающей его значение.

а) Какая из трех сетей на рис. 13.24 утверждает, что

$$\mathbf{P}(G_{father}, G_{mother}, G_{child}) = \mathbf{P}(G_{father})\mathbf{P}(G_{mother})\mathbf{P}(G_{child})?$$

б) Какие из трех сетей независимо друг о друга утверждают, что они совместимы с упомянутой выше гипотезой о наследовании характеристики "правша/левша"?

в) Какая из трех сетей является наилучшим описанием предложенной выше гипотезы?

г) Запишите таблицу CPT для узла G_{child} в сети **а)** в терминах s и m.

д) Предположим, что $P(G_{father} \mid l) = P(G_{mother} \mid l) = q$. В сети **а)** выведите выражение для $P(G_{child} \mid l)$ в терминах только вероятностей m и q с учетом обусловливания от его родительских узлов.

е) В условиях генетического равновесия можно ожидать, что распределение генов в разных поколениях будет одинаковым. Используйте это положение для вычисления значения q, а затем, с учетом того, что известно о фактической распространенности правшей и левшей среди людей, объясните, почему гипотеза, предложенная в условиях этого упражнения, должна быть неверной.

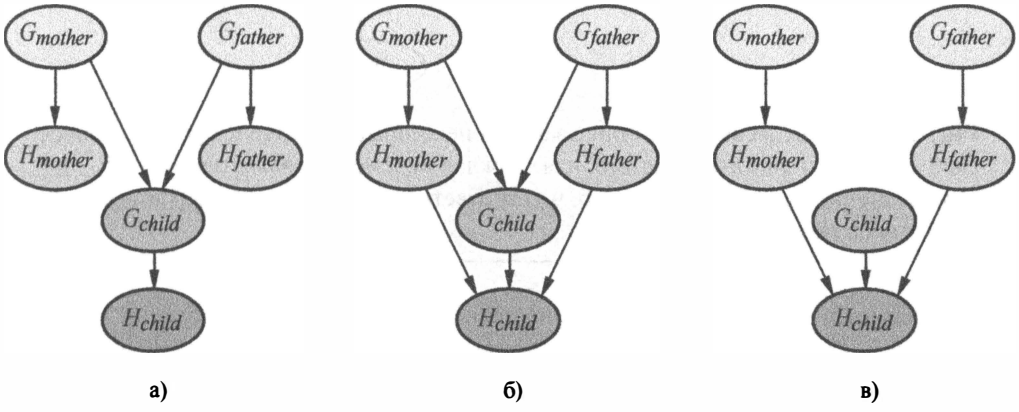


Рис. 13.24. Три возможные структуры байесовской сети, описывающей принцип генетического наследования характеристики "правша/левша"

13.8. Определение марковского покрытия переменной дано в разделе 14.2.1. Докажите, что переменная не зависит от всех других переменных в сети, если дано ее марковское покрытие, и выведите уравнение (13.10), приведенное в разделе 13.4.2.

13.9. Рассмотрите сеть для диагностики автомобиля, представленную на рис. 13.25.

а) Дополните сеть булевыми переменными *IcyWeather* (морозная погода) и *StarterMotor* (стартер).

б) Приведите приемлемые таблицы условных вероятностей для всех вершин.

в) Сколько независимых значений содержится в совместном распределении вероятностей для восьми булевых вершин, если исходить из предположения, что нет известных связывающих их отношений условной независимости?

г) Сколько независимых значений вероятности содержится в таблицах вашей сети?

д) Условное распределение для вершины *Starts* (запускается) может быть описано как распределение зашумленного AND. Дайте общее определение этого семейства распределений и покажите его связь с распределениями зашумленного OR.

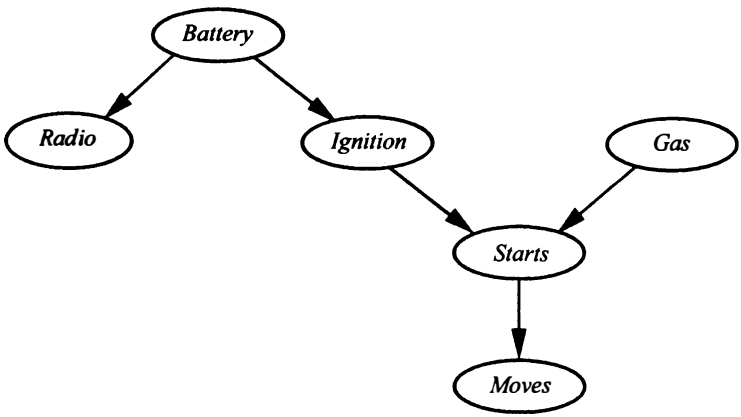


Рис. 13.25. Байесовская сеть, описывающая некоторые характеристики электрической системы и двигателя автомобиля. Каждая переменная является булевой, при этом значение *true* указывает на то, что соответствующая подсистема автомобиля находится в рабочем состоянии

13.10. Рассмотрим простую байесовскую сеть с корневыми переменными *Cold* (простуда), *Flu* (грипп) и *Malaria* (малярия) и дочерней переменной *Fever* (жар) с условным распределением зашумленного OR для переменной *Fever*, как описано в разделе 13.2.2. Добавив соответствующие вспомогательные переменные для событий понижения и повышения температуры, создайте эквивалентную байесовскую сеть, таблицы CPT которой (за исключением корневых

переменных) будут детерминированными. Определите эти CPT и докажите эквивалентность.

13.11. Рассмотрим семейство линейных гауссовых сетей, которое было определено в разделе 13.2.3.

а) Допустим, что в сети с двумя переменными переменная X_1 является родительской по отношению к переменной X_2, переменная X_1 имеет гауссово распределение априорных вероятностей, а $P(X_2 \mid X_1)$ — линейное гауссово распределение. Покажите, что совместное распределение $P(X_1, X_2)$ представляет собой многомерное гауссово распределение, и рассчитайте его матрицу ковариации.

б) Докажите методом индукции, что совместное распределение для линейной гауссовой сети общего вида по переменным $X_1, \ldots, X_n$ также является многомерным гауссовым распределением.

13.12. Пробит-распределение, описанное в разделе 13.2.3, описывает распределение вероятностей для булевой дочерней переменной, если задана одна непрерывная родительская переменная.

а) Как можно расширить это определение, чтобы оно охватывало несколько непрерывных родительских переменных?

б) Как оно может быть расширено на случай многозначной дочерней переменной? Рассмотрите как те случаи, в которых значения дочерней переменной упорядочены (например, в случае выбора при управлении автомобилем передачи в зависимости от скорости, крутизны подъема, требуемого ускорения и т.д.), так и те случаи, в которых они не упорядочены (например, при выборе автобуса, трамвая или автомобиля для поездки на работу). (*Подсказка*. Рассмотрите способы разделения возможных значений на два множества для имитации булевой переменной.)

13.13. На атомной электростанции предусмотрена тревожная сигнализация, срабатывающая, если показания датчика температуры превышают некоторое пороговое значение. Датчик измеряет температуру в реакторе. Рассмотрите булевы переменные A (звучит сигнал тревоги), FA (тревожная сигнализация неисправна) и FG (неисправен датчик), а также многозначные вершины G (показания датчика) и T (фактическая температура в реакторе).

а) Нарисуйте байесовскую сеть для этой проблемной области с учетом того, что вероятность отказа датчика повышается, если температура в реакторе становится слишком высокой.

б) Является ли ваша сеть полидеревом? Почему да или почему нет?

в) Предположим, что есть только два значения фактической и измеряемой температур — нормальная и высокая. Вероятность того, что датчик сообщает правильную температуру, равна x, если он работает, и равна y, если он неисправен. Составьте таблицу условных вероятностей, связанную с вершиной G.

г) Предположим, что тревожная сигнализация работает правильно, при условии, что она не вышла из строя; в последнем случае сигнал тревоги никогда не зазвучит. Составьте таблицу условных вероятностей, связанную с вершиной A.

д) Предположим, что тревожная сигнализация и датчик исправны и что сигнал тревоги звучит. Вычислите выражение для вероятности того, что температура в реакторе слишком высока, в терминах различных условных вероятностей в этой сети.

13.14. Два астронома в различных частях Земли с помощью своих телескопов провели подсчеты, M_1 и M_2, количества звезд N на некотором небольшом участке неба. При подсчетах обычно имеет место небольшая вероятность ошибки e на одну звезду в большую или меньшую сторону. Каждый телескоп также может (с гораздо меньшей вероятностью f) оказаться сильно расфокусированным (события F_1 и F_2), и в этом случае ученый не досчитается трех или более звезд (или, если N меньше 3, вообще не обнаружит ни одной звезды). Рассмотрите три байесовские сети, приведенные на рис. 13.26.

а) Какая из этих трех байесовских сетей является правильным (но не обязательно эффективным) представлением приведенной выше информации?

б) Какая из этих сетей лучше? Объясните, почему.

в) Запишите распределение условных вероятностей $P(M_1 \mid N)$ для случая, где $N\{1, 2, 3\}$ и $M_1\{0, 1, 2, 3, 4\}$. Каждая запись в распределении условных вероятностей должна быть представлена как функция от параметров e и/или f.

г) Предположим, что $M_1 = 1$ и $M_2 = 3$. Каково возможное количество звезд, если предполагается, что на значения N не налагаются априорные ограничения?

д) Каково наиболее вероятное количество звезд, если даны эти наблюдения? Объясните, как рассчитать эту вероятность, или, если ее невозможно рассчитать, объясните, какая требуется дополнительная информация и как она может повлиять на результат.

13.15. Рассмотрим сеть, показанную на рис. 13.26, *б*. Предположим, что два телескопа дают идентичные результаты, $N\{1, 2, 3\}$ и M_1, $M_2\{0, 1, 2, 3, 4\}$, а также что используются те же символические таблицы CPT, которые описаны в упражнении 13.14. Используя алгоритм перебора, рассчитайте распределение вероятностей $\mathbf{P}(N \mid M_1 = 2, M_2 = 2)$.

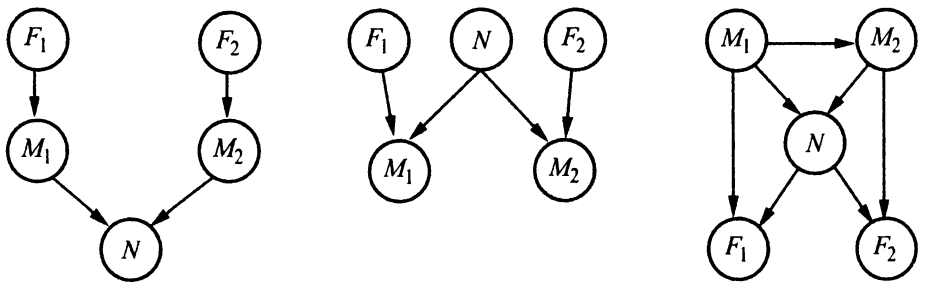


Рис. 13.26. Три возможных варианта сети для задачи об астрономах и телескопах

13.16. Рассмотрим байесовскую сеть, показанную на рис. 13.27.

а) Какие из следующих утверждений подтверждаются *структурой* сети?
   1. $\mathbf{P}(B, I, M) = \mathbf{P}(B)\mathbf{P}(I)\mathbf{P}(M)$
   2. $\mathbf{P}(J \mid G) = \mathbf{P}(J \mid G, I)$
   3. $\mathbf{P}(M \mid G, B, I) = \mathbf{P}(M \mid G, B, I, J)$

б) Рассчитайте значение $P(b, i, \neg m, g, j)$.

в) Рассчитайте вероятность того, что кто-то попадет в тюрьму, если он нарушил закон, ему предъявлено обвинение и он столкнулся с политически мотивированным прокурором.

г) Контекстно специфическая независимость (см. раздел 13.2.2) позволяет переменной быть независимой от некоторых ее родительских переменных при определенных значениях других. В дополнение к обычным условным независимостям, определяемым структурой графа, какие контекстно специфические независимости присутствуют в байесовской сети, представленной на рис. 13.27?

д) Предположим, в эту сеть требуется добавить переменную *PPresidentialPardon* (президентское помилование). Нарисуйте новую сеть и кратко объясните присутствие любых ребер, которые были добавлены.

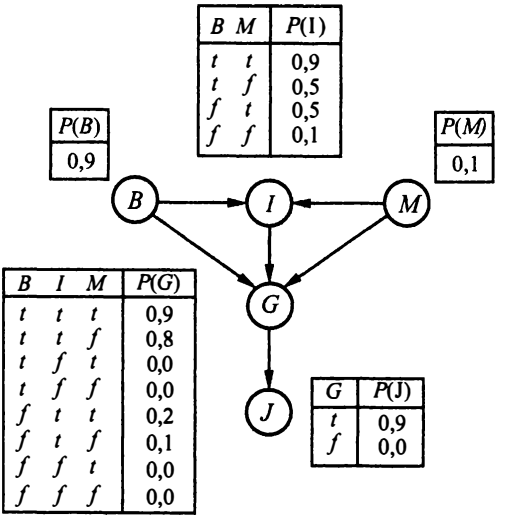


Рис. 13.27. Простая байесовская сеть с булевыми переменными $B = \{BrokeElectionLaw\}$ (нарушен избирательный закон), $I = \{Indicted\}$ (предъявлено обвинение), M = {*PoliticallyMotivationProvisor*} (политически мотивированный прокурор), G = {*FoundGuilty*} (признан виновным), J = {*Jailed*} (заключен в тюрьму)

13.17. Рассмотрим байесовскую сеть, представленную на рис. 13.27.

а) Что из приведенного ниже, если таковое имеется, подтверждается структурой сети (пока CPT игнорируются)?

1. $\mathbf{P}(B, I, M) = \mathbf{P}(B)\mathbf{P}(I)\mathbf{P}(M)$
2. $\mathbf{P}(J \mid G) = \mathbf{P}(J \mid G, I)$
3. $\mathbf{P}(M \mid G, B, I) = \mathbf{P}(M \mid G, B, I, J)$

б) Рассчитайте значение $P(b, i, m, \neg g, j)$.

в) Рассчитайте вероятность того, что кто-то попадет в тюрьму, если он нарушил закон, ему предъявлено обвинение и он столкнулся с политически мотивированным прокурором.

г) Контекстно специфическая независимость (см. раздел 13.2.2) позволяет переменной быть независимой от некоторых ее родительских переменных при определенных значениях других. В дополнение к обычным условным независимостям, определяемым структурой графа, какие контекстно специфические независимости присутствуют в байесовской сети, представленной на рис. 13.27?

д) Предположим, в эту сеть требуется добавить переменную *PPresidentialPardon* (президентское помилование). Нарисуйте новую сеть и кратко объясните присутствие любых ребер, которые были добавлены.

13.18. Рассмотрим алгоритм устранения переменной, представленный на рис. 13.13 (раздел 13.3.2).

а) В разделе 13.3 алгоритм устранения переменной применялся к следующему запросу:

$$\mathbf{P}(Burglary \mid JohnCalls = true, MaryCalls = true).$$

Проведите указанные вычисления и проверьте правильность ответа.

б) Подсчитайте количество выполненных арифметических операций и сравните его с количеством операций, выполняемых в алгоритме перебора.

в) Предположим, что сеть имеет форму *цепи* — последовательности булевых переменных $X_1, \ldots, X_n$, где $Parents(X_i) = \{X_{i-1}\}$ для $i = 2, \ldots, n$. Какова сложность вычисления выражения $\mathbf{P}(X_1 \mid X_n = true)$ с использованием алгоритма перебора? С использованием алгоритма устранения переменной?

г) Докажите, что сложность применения алгоритма устранения переменной в сети, имеющей форму полидерева, линейно зависит от размера дерева при любом упорядочении переменных, согласованном со структурой сети.

13.19. Проведите исследование сложности точного вывода в байесовских сетях общего вида.

а) Докажите, что любую задачу 3-SAT можно свести к задаче точного вывода в байесовской сети, построенной для представления данной конкретной задачи, и поэтому такой точный вывод является NP-трудным. (*Подсказка.* Рассмотрите сеть с одной переменной для каждого пропозиционального символа, с одной — для каждого выражения и с одной — для конъюнкции выражений.)

б) Проблема подсчета количества выполняющих присваиваний для задачи 3-SAT является #P-полной. Покажите, что задача точного вывода является по меньшей мере такой же трудной, как эта.

13.20. Рассмотрим задачу формирования случайной выборки из заданного распределения по одной переменной. Предположим, что в вашем распоряжении имеется генератор случайных чисел, возвращающий случайное число с равномерным распределением в интервале от 0 до 1.

а) Пусть X — дискретная переменная с $P(X = x_i) = p_i$ для $i = \{1,\ldots, k\}$. **Кумулятивное распределение** (*cumulative distribution*) X задает вероятность того, что $X = \{x_1,\ldots, x_j\}$ для каждого возможного j. Объясните, как рассчитать это кумулятивное распределение за время $O(k)$ и как получить из него одну выборку X. Может ли последняя операция быть выполнена за время меньше $O(k)$?

б) Теперь предположим, что необходимо сформировать N выборок X, где $N >> k$. Объясните, как выполнить эту операцию с ожидаемым временем выполнения в расчете на выборку, которое является *постоянным* (т.е. независимым от k).

в) После этого рассмотрим непрерывную переменную с параметризованным распределением (например, с гауссовым). Как можно формировать выборки с помощью такого распределения?

г) Предположим, что необходимо сформулировать запрос, касающийся непрерывной переменной, и что для вероятностного вывода используется такой алгоритм, как LIKELIHOODWEIGHTING. Как вы модифицировали бы процесс поиска ответов на такие запросы?

13.21. Рассмотрим запрос $\mathbf{P}(Rain \mid Sprinkler = true, WetGrass = true)$ в байесовской сети, приведенной на рис. 13.15, *а*, и то, как можно получить на него ответ с помощью алгоритма MCMC.

а) Какое количество состояний имеет эта цепь Маркова?

б) Рассчитайте **матрицу переходов** Q, содержащую значение $q(\mathbf{y} \rightarrow \mathbf{y}')$ для всех $\mathbf{y}, \mathbf{y}'$.

в) Что представляет собой Q^2, квадрат матрицы переходов?

г) А что можно сказать о выражении Q^n, где $n \rightarrow \infty$?

д) Объясните, как следует выполнять вероятностный вывод в байесовских сетях при условии, что доступно выражение Q^n. Является ли такой способ вероятностного вывода практически применимым?

13.22. В этом упражнении исследуется стационарное распределение для методов выборки Гиббса.

а) Выпуклая композиция $[\alpha, q_1; 1 - \alpha, q_2]$ для распределений q_1 и q_2 представляет собой распределение вероятностей перехода, которое сначала выбирает что-то одно из распределений q_1 и q_2 с вероятностями α и $1 - \alpha$ соответственно, а затем применяет то, что выбрано. Докажите, что если распределения q_1 и q_2 находятся в детализированном равновесии с π, то их выпуклая композиция также находится в детализированном равновесии с π. (*Примечание.* Этот результат оправдывает вариант алгоритма GIBBS-ASK, в котором переменные выбираются случайным образом, а не в фиксированной последовательности.)

б) Докажите, что если распределения q_1 и q_2 имеют π в качестве стационарного распределения, то последовательная композиция для распределений q_1 и q_2 также имеет π в качестве стационарного распределения.

13.23. Алгоритм Метрополиса–Гастингса является членом семейства алгоритмов МСМС. Как таковой он предназначен для генерации выборок x (в конечном итоге) в соответствии с целевыми вероятностями $\pi(x)$. (Обычно нас интересует выборка из $\pi(\mathbf{x})P(\mathbf{x} \mid \mathbf{e})$.) Как и алгоритм имитации отжига, алгоритм Метрополиса–Гастингса работает в два этапа. На первом формируется новое состояние $\mathbf{x}'$ из **вспомогательного распределения** $q(\mathbf{x}' \mid \mathbf{x})$ при заданном текущем состоянии $\mathbf{x}$. На втором состояние $\mathbf{x}'$ вероятностно принимается или отклоняется в соответствии с **вероятностью успешного приема**

$$a(\mathbf{x}' \mid \mathbf{x}) = \min\left(1, \frac{\pi(\mathbf{x}')q(\mathbf{x} \mid \mathbf{x}')}{\pi(\mathbf{x})q(\mathbf{x}' \mid \mathbf{x})}\right).$$

Если предложение $\mathbf{x}'$ отклонено, сохраняется состояние $\mathbf{x}$.

а) Рассмотрите случай, когда на первом этапе выполняется обычная выборка Гиббса для конкретной переменной X_i. Покажите, что результат выполнения этого этапа, рассматриваемый как предложение, гарантированно будет принят в алгоритме Метрополиса–Гастингса. (Следовательно, выборка Гиббса является частным случаем алгоритма Метрополиса–Гастингса.)

б) Покажите, что описанный выше двухэтапный процесс, рассматриваемый как распределение вероятностей перехода, находится в детализированном равновесии с π.

13.24. Три футбольные команды, A, B и C, играют друг с другом по одному разу. Каждый матч проводится между двумя командами и может закончиться для команды выигрышем, ничьей или проигрышем. Каждой команде присвоена постоянная, неизвестная оценка ее класса (целое число в диапазоне от 0 до 3), и результат матча вероятностно зависит от разницы в классе между двумя участвующими командами.

а) Сформируйте реляционную вероятностную модель для описания этой проблемной области и предложите реальные числовые значения для всех необходимых распределений вероятностей.

б) Постройте эквивалентную байесовскую сеть для трех матчей.

в) Предположим, что в первых двух матчах команда A побеждает команду B и заканчивает игру вничью с командой C. Используя алгоритм точного вывода по своему выбору, вычислите распределение апостериорных вероятностей для результатов третьего матча.

г) Предположим, что в этой футбольной лиге n команд и у нас есть результаты всех матчей, кроме последнего. Как сложность предсказания последней игры зависит от n?

д) Рассмотрите возможность применения алгоритма МСМС для решения этой задачи. Насколько быстро этот алгоритм сходится на практике и насколько хорошо он масштабируется?

ГЛАВА 14

Вероятностные рассуждения во времени

В данной главе предпринимаются попытки интерпретировать настоящее, понимать прошлое и, возможно, предсказывать будущее, даже когда очень немногое полностью ясно.

Агенты, действующие в частично наблюдаемых средах, должны быть способны отслеживать текущее состояние среды в тех пределах, которые обеспечивают их датчики. В разделе 4.4 была представлена методология, позволяющая сделать это: агент поддерживает **доверительное состояние**, представляющее, какие состояния мира возможны на текущий момент. На основании доверительного состояния и **модели перехода** агент может предсказать, как может измениться мир на следующем временном этапе. Исходя из наблюдаемых восприятий и модели сенсоров или **модели восприятия**, агент может обновлять свое доверительное состояние. Это универсальная, всеохватывающая идея: в главе 4 доверительные состояния были представлены явно перечисленными множествами состояний, тогда как в главах 7 и 11 они были представлены логическими формулами. Однако во всех этих подходах доверительные состояния определялись лишь с точки зрения того, какие состояния мира были *возможны*, но они ничего не могли сказать о том, насколько эти состояния были *вероятны* или *маловероятны*. В этой главе теория вероятностей будет использована для получения количественной оценки степени доверия к элементам доверительного состояния.

В разделе 14.1 поясняется выбранный базовый подход: время само по себе будет представлено так же, как в главе 7: изменяющийся мир моделируется с использованием переменных, представляющих каждый аспект состояния мира *в каждый момент времени*. Модель переходов и модель восприятия могут быть неопределенными: модель переходов описывает распределение вероятностей переменных в момент времени t при известном состоянии мира в прошедшие моменты времени, тогда как модель восприятия описывает вероятность каждого восприятия в момент времени t с учетом текущего состояния мира. В разделе 14.2 определяются основные задачи вероятностного вывода и описана общая структура алгоритмов вероятностного вывода для временных моделей. Затем рассматриваются три конкретных

типа моделей: **скрытые марковские модели**, **фильтры Калмана** и **динамические байесовские сети** (которые включают скрытые марковские модели и фильтры Калмана в качестве частных случаев).

14.1. Время и неопределенность

В предыдущих главах методы вероятностных рассуждений рассматривались в контексте *статических* миров, в которых каждая случайная переменная имела одно фиксированное значение. Например, в задаче о ремонте автомобиля предполагалось, что любой неисправный узел останется неисправным на протяжении всего процесса диагностики, и наше задание заключалось лишь в вероятностном выводе информации о состоянии автомобиля, исходя из наблюдаемых свидетельств, которые также оставались фиксированными.

Теперь рассмотрим несколько иную задачу: лечение больного диабетом. Как и в случае ремонта автомобиля, в нашем распоряжении имеются свидетельства, такие как последние дозы приема инсулина, рацион питания, результаты измерения количества сахара в крови и другие физические симптомы. Задание состоит в том, чтобы оценить текущее состояние пациента, включая фактический уровень сахара в крови и уровень инсулина. При наличии такой информации можно будет принять решение о рационе питания больного и необходимой дозе инсулина. Но, в отличие от случая с ремонтом автомобиля, здесь становятся важными *динамические* аспекты задачи. Значения уровня сахара в крови и результаты его измерения могут быстро изменяться во времени в зависимости от последнего приема пищи больным и доз инсулина, от интенсивности обмена веществ в организме, времени суток и т.д. Чтобы оценить текущее состояние больного по хронологии накопленных фактов и предсказать результаты терапевтических действий, необходимо моделировать эти изменения.

Точно такие же соображения возникают во многих других контекстах, подобных отслеживанию местоположения робота, наблюдению за экономической активностью некоторого государства или анализу смысла некоторой последовательности произнесенных или записанных слов. Как же можно смоделировать подобные динамические ситуации?

14.1.1. Состояния и наблюдения

В этой главе рассматриваются модели с ▶**дискретным представлением времени**, в которых мир рассматривается как серия снимков или ▶**временных срезов**.[1] Мы просто будем нумеровать временные срезы (0, 1, 2 и т.д.) вместо того, чтобы присваивать им некоторое конкретное значение времени. Обычно интервал

[1] Неопределенность в отношении непрерывного времени может быть смоделирована с помощью стохастических дифференциальных уравнений (*Stochastic Differential Equations* — SDE). Модели, анализируемые в этой главе, можно рассматривать как приближения к SDE в дискретном времени.

времени Δ между срезами предполагается одинаковым для каждого интервала. Для любого конкретного применения необходимо выбрать конкретное значение Δ. Иногда оно может задаваться особенностями датчика, например видеокамера может предоставлять изображения с интервалом 1/30 секунды. В других случаях интервал диктуется типичной скоростью изменения соответствующих переменных, например в случае мониторинга содержания глюкозы в крови ситуация может резко измениться в течение десяти минут, поэтому вполне обоснованным будет выбор интервала в одну минуту. С другой стороны, при моделировании дрейфа континентов во временных масштабах геологических эпох хорошим выбором может быть интервал в один миллион лет.

Каждый временной срез в вероятностной модели с дискретным представлением времени содержит множество случайных переменных, одни из которых являются наблюдаемыми, а другие — нет. Для простоты мы будем предполагать, что одно и то же подмножество переменных можно наблюдать в каждом временном срезе (хотя такое требование не является строго необходимым в отношении всего последующего изложения). Будем использовать $\mathbf{X}_t$ для обозначения множества переменных состояния во время t, которые предполагаются ненаблюдаемыми, и $\mathbf{E}_t$ — для обозначения множества наблюдаемых переменных свидетельства. Результатами наблюдения во время t является $\mathbf{E}_t = \mathbf{e}_t$, т.е. некоторое множество значений $\mathbf{e}_t$.

Рассмотрим следующий простой пример: предположим, что на некоем секретном подземном объекте имеется охранник, который на весь период вахты никогда его не покидает. Если он захочет узнать, идет ли сегодня дождь, то единственную доступную информацию о состоянии внешнего мира он сможет получить только утром, встречая директора, пришедшего с зонтиком или без него. Таким образом, для каждого дня t множество $\mathbf{E}_t$ включает единственную переменную свидетельства $Umbrella_t$ (зонтик) или U_t (определяющую, был ли у директора зонтик), а множество $\mathbf{X}_t$ содержит единственную переменную состояния $Rain_t$ (дождь) или R_t (указывающую, идет ли дождь). Другие задачи могут включать более крупные множества переменных. В частности, в примере с диабетом могут использоваться такие переменные свидетельства, как результаты измерения уровня сахара в крови $MeasuredBloodSugar_t$ и частоты пульса $PulseRate_t$, а также переменные состояния, такие как фактический уровень сахара в крови $BloodSugar_t$ и содержимое желудка $StomachContents_t$. (Обратите внимание, что $BloodSugar_t$ и $MeasuredBloodSugar_t$ — это не одна и та же переменная, именно так принято поступать с зашумленными результатами измерений реальных величин.)

Будем считать, что в задачах последовательность состояний начинается в момент времени $t = 0$, а информация свидетельства начинает поступать с момента времени $t = 1$. Таким образом, мир задачи с зонтиком может быть представлен переменными состояния $R_0, R_1, R_2, \ldots$ и переменными свидетельства $U_1, U_2, \ldots$ Для обозначения последовательности целых чисел от a до b (включительно) будем использовать запись $a{:}b$ и запись $\mathbf{X}_{a:b}$ — для обозначения соответствующего ряда переменных от $\mathbf{X}_a$

до $\mathbf{X}_b$. Например, $U_{1:3}$ соответствует переменным U_1, U_2, U_3. (Обратите внимание, что этот подход отличается от нотации, используемой в языках программирования, таких как Python или Go, где запись `U[1:3]` *не включает* `U[3]`.)

14.1.2. Модель перехода и модель восприятия

После выбора для рассматриваемой задачи множества переменных состояния и переменных свидетельства на следующем этапе следует указать, как этот мир развивается (модель перехода) и как переменные свидетельства получают свои значения (модель восприятия).

Модель перехода определяет распределение вероятностей для переменных последнего состояния при заданных их предыдущих значениях, т.е. $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{0:t-1})$. И здесь возникает проблема: множество $\mathbf{X}_{0:t-1}$ неограниченно возрастает в размерах по мере увеличения t. Эта проблема решается принятием ▶**марковского предположения** о том, что текущее состояние зависит только от *конечного фиксированного числа* предыдущих состояний. Процессы, удовлетворяющие этому предположению, впервые были глубоко исследованы статистиком Андреем Марковым (1856–1922) и называются ▶**марковскими процессами** или **марковскими цепями**. Существует несколько разновидностей таких процессов, и простейшим из них является ▶**марковский процесс первого порядка**, в котором текущее состояние зависит только от *предыдущего* состояния и не зависит от каких-либо более ранних состояний. Другими словами, каждое состояние предоставляет достаточно информации, чтобы сделать будущее условно независимым от прошлого:

$$\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1}). \tag{14.1}$$

Таким образом, в марковском процессе первого порядка модель перехода представлена условным распределением $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$. Моделью перехода для марковского процесса второго порядка является условное распределение $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-2}, \mathbf{X}_{t-1})$. На рис. 14.1 приведены структуры байесовских сетей, соответствующие марковским процессам первого и второго порядка.

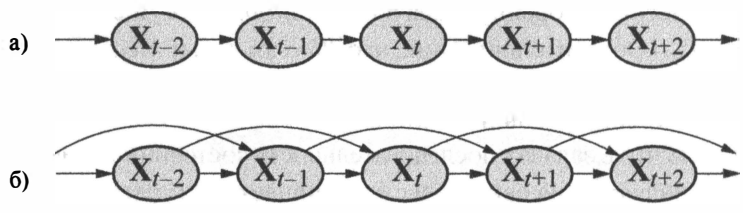


Рис. 14.1. а) Структура байесовской сети, соответствующая марковскому процессу первого порядка с состояниями, определяемыми переменными $\mathbf{X}_t$. **б)** Марковский процесс второго порядка

Но даже при принятии марковского предположения все еще существует другая проблема: имеется бесконечно много возможных значений t. Необходимо определять иное распределение для каждого временного этапа? Можно избежать этой проблемы, если предположить, что изменения в состоянии мира обусловлены ►**стационарным** (*time-homogeneous*) процессом, т.е. процессом изменения, подчиняющимся законам, не изменяющимся с течением времени. Тогда в мире задачи с зонтиком условная вероятность дождя, $\mathbf{P}(R_t \mid R_{t-1})$, будет одной и той же для всех t и достаточно будет определить только одну таблицу условной вероятности.

Теперь обратимся к модели восприятия. Переменные свидетельства $\mathbf{E}_t$ *могут* зависеть как от предыдущих, так и от текущих значений переменных состояния, но любого заслуживающего внимания состояния должно быть достаточно для генерации текущего значения датчика. Следовательно, ►**марковское допущение для датчиков** можно сформулировать следующим образом:

$$\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_{0:t}, \mathbf{E}_{1:t-1}) = \mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t). \tag{14.2}$$

Таким образом, модель восприятия (иногда ее называют **моделью наблюдения**) может быть определена как $\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$. На рис. 14.2 показаны модель перехода и модель восприятия для примера задачи с зонтиком. Обратите внимание на направление зависимости между состоянием и датчиками: стрелки направлены от фактического состояния мира к значениям датчиков, поскольку именно состояние мира *вынуждает* датчики принимать определенные значения: дождь *вызывает* появление зонтика. (Процесс вероятностного вывода, конечно же, идет в другом направлении, — различие между направлением смоделированных зависимостей и направлением вывода является одним из главных преимуществ байесовских сетей.)

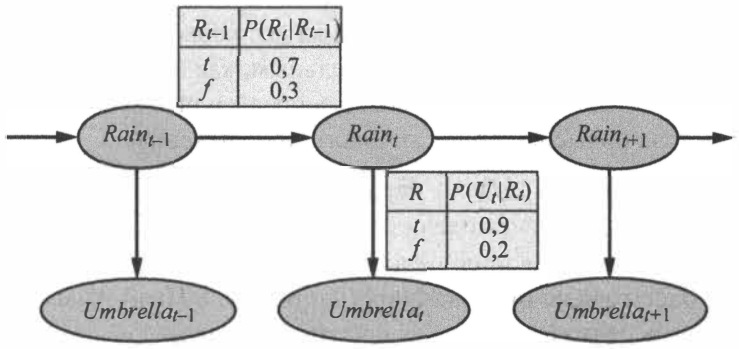


Рис. 14.2. Структура байесовской сети и распределения условных вероятностей, описывающие мир задачи с зонтиком. Моделью перехода является $\mathbf{P}(Rain_t \mid Rain_{t-1})$, а моделью восприятия является $\mathbf{P}(Umbrella_t \mid Rain_t)$

В дополнение к модели перехода и модели восприятия также необходимо определить, как все начиналось, т.е. задать распределение априорных вероятностей в момент времени 0, $\mathbf{P}(\mathbf{X}_0)$. Этих трех распределений достаточно, чтобы получить спецификацию полного совместного распределения по всем переменным, используя уравнение (13.2). Для любого этапа времени t получаем:

$$\mathbf{P}(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = \mathbf{P}(\mathbf{X}_0) \prod_{i=1}^{t} \mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1}) \mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i). \qquad (14.3)$$

Три элемента в правой части уравнения являются начальным состоянием модели $\mathbf{P}(\mathbf{X}_0)$, моделью перехода $\mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1})$ и моделью восприятия $\mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i)$. Это уравнение определяет семантику семейства временны*х* моделей, представленных тремя данными элементами. Обратите внимание, что стандартные байесовские сети не могут представлять такие модели, поскольку они требуют, чтобы множество переменных было конечным. Способность работать с бесконечными множествами переменных появляется из-за двух особенностей: во-первых, из-за определения бесконечного множества с использованием целочисленных индексов и, во-вторых, из-за использования неявного универсального квантификатора (см. раздел 8.2) в определении модели восприятия и модели перехода для каждого временно*го* этапа.

Структура, приведенная на рис. 14.2, представляет марковский процесс первого порядка — предполагается, что вероятность дождя зависит только от того, был ли дождь в предыдущий день. Разумно ли такое предположение, зависит от самой проблемной области. Марковское предположение первого порядка говорит о том, что переменные состояния содержат *всю* информацию, необходимую для описания распределения вероятностей для следующего временного среза. Иногда такое предположение полностью соответствует истине; например, если некоторая частица совершает случайное блуждание вдоль оси x, изменяя свою позицию на величину ± 1 в каждый интервал времени, то применение в качестве состояния координаты x позволяет определить марковский процесс первого порядка. Иногда такое предположение является лишь приблизительным, как в случае предсказания дождя лишь на основании того, был ли дождь в предыдущие сутки. Существуют два возможных метода повысить точность такого приближения.

1. Повышение порядка модели марковского процесса. Например, можно перейти к использованию модели второго порядка, введя переменную $Rain_{t-2}$ в качестве родительской по отношению к $Rain_t$, что позволит получить немного более точные предсказания (например, в Пало-Альто, Калифорния, дождь очень редко идет больше двух дней подряд).
2. Расширение множества переменных состояния. Например, можно ввести переменную $Season_t$, представляющую время года, что позволит включить в рассмотрение хронологические данные о дождливых временах года,

или ввести переменные $Temperature_t$ (температура), $Humidity_t$ (влажность) и $Pressure_t$ (атмосферное давление), чтобы появилась возможность использования физической модели условий, способствующих возникновению дождя.

В упражнении 14.1 предлагается показать, что первое решение (увеличение порядка модели) всегда можно переформулировать как расширение множества переменных состояния с сохранением исходного порядка. Обратите внимание, что введение дополнительных переменных состояния позволяет повысить предсказательные возможности системы, но при этом повышает *требования* к прогнозированию, поскольку теперь потребуется также прогнозировать значения новых переменных. Следовательно, необходимо найти "самодостаточное" множество переменных, что фактически означает требование понять "физику" моделируемого процесса. Очевидно, что требования к точному моделированию процесса можно ослабить, если есть возможность ввести новые результаты восприятия (например, результаты измерения температуры и атмосферного давления), которые непосредственно предоставляют информацию о новых переменных состояния.

Рассмотрим, например, задачу слежения за роботом, случайным образом блуждающим на плоскости X–Y. Можно предположить, что достаточно воспользоваться лишь таким множеством переменных состояния, как положение и скорость, — ведь для вычисления нового положения можно просто использовать законы Ньютона, а скорость может изменяться непредсказуемо. Но если робот работает на электрической энергии от аккумулятора, то постепенный разряд аккумулятора будет оказывать систематическое влияние на изменение скорости. А поскольку само это влияние зависит от того, сколько электроэнергии было израсходовано при всех предыдущих маневрах, то свойство *марковости* (т.е. принадлежности марковской цепи к модели определенного порядка) будет нарушаться.

Марковость цепи можно восстановить, включив переменную $Battery_t$, представляющую уровень заряда аккумулятора, в состав переменных состояния, входящих в множество $\mathbf{X}_t$. Это позволит лучше предсказывать движения робота, но, в свою очередь, потребует создания модели для предсказания значения $Battery_t$ на основании значения $Battery_{t-1}$ и скорости. В некоторых случаях такое моделирование может быть выполнено достаточно надежно, в большинстве случаев вскоре будет обнаружено, что ошибка предсказания накапливается со временем. В этом случае повышения точности можно будет добиться путем *ввода нового датчика* для измерения уровня заряда аккумулятора. Мы еще вернемся к примеру с аккумулятором в разделе 14.5.

14.2. Вероятностный вывод во временных моделях

Теперь, после определения структуры общей временной модели, можно сформулировать основные задачи вероятностного вывода, которые должны быть решены с помощью этой модели.

- ►**Фильтрация**[2] или ►**оценка состояния**. Это задача вычисления ►**доверительного состояния** $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ — распределения апостериорных вероятностей, относящихся к текущему состоянию, с учетом всех полученных к данному моменту свидетельств. В примере задачи с зонтиком это может означать вычисление вероятности дождя сегодня, если даны все результаты наблюдений о наличии зонтика, полученные до сих пор. Фильтрацией называют те действия, которые рациональный агент должен выполнять для отслеживания текущего состояния таким образом, чтобы получить возможность принимать рациональные решения. Как оказалось, почти идентичные расчеты выполняются при определении правдоподобия последовательности свидетельств, $P(\mathbf{e}_{1:t})$.
- ►**Предсказание**. Это задача вычисления распределения апостериорных вероятностей для *будущих* состояний с учетом всех свидетельств, полученных к данному моменту. Это означает, что может потребоваться вычислить $\mathbf{P}(\mathbf{X}_{t+k} \mid \mathbf{e}_{1:t})$ для некоторого $k > 0$. В примере задачи с зонтиком такие вычисления могут выполняться с целью определения вероятности дождя через три дня от нынешнего с учетом всех результатов наблюдений о наличии зонтика, полученных до сих пор. Предсказание полезно для оценки возможных действий, исходя из их ожидаемых результатов.
- ►**Сглаживание**. Это задача вычисления распределения апостериорных вероятностей для *прошлых* состояний с учетом всех свидетельств вплоть до нынешнего состояния. И это означает, что требуется вычислить значение $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ для некоторого k, такого, что $0 \le k < t$. В примере задачи с зонтиком это может означать вычисление вероятности того, что дождь шел в прошлую среду, с учетом всех результатов наблюдений о наличии зонтика, полученных до сих пор. Сглаживание обеспечивает лучшую оценку состояния в момент времени k, чем та, которая была доступна в тот момент, поскольку она включает больше свидетельств.[3]
- **Наиболее вероятное объяснение**. При наличии результатов последовательности наблюдений может потребоваться найти такую последовательность

[2] Термин "фильтрация" относится к истокам этой задачи, впервые рассматривавшейся в ранних работах по обработке сигналов, где проблема состояла в том, чтобы отфильтровать шум от сигнала посредством оценки его базовых свойств.

[3] В частности, при отслеживании движущегося объекта посредством неточных наблюдений за положением сглаживание дает более гладкую оценочную траекторию, чем фильтрация, — отсюда и название.

состояний, которая с наибольшей вероятностью стала причиной получения этих результатов. Это означает, что необходимо вычислить значение $\text{argmax}_{x_{1:t}} P(x_{1:t} | e_{1:t})$. Например, если наличие зонтика было отмечено в каждый из первых трех дней, а на четвертый день зонтика у директора не было, то наиболее вероятным объяснением становится наличие дождя в первые три дня и его отсутствие на четвертый. Алгоритмы решения такой задачи оказались полезными во многих приложениях, включая распознавание речи — здесь целью является поиск наиболее вероятной последовательности слов при наличии серии звуков — и реконструкцию цепочек битов, передаваемых по зашумленному каналу.

В дополнение к этим задачам вероятностного вывода можно упомянуть еще одну.

- **Обучение**. Модель перехода и модель восприятия, если они еще не известны, можно изучить, исходя из наблюдений. Как и в случае статических байесовских сетей, обучение динамических байесовских сетей может быть выполнено как побочный продукт вероятностного логического вывода. Вероятностный вывод дает оценку того, какие переходы между состояниями действительно произошли и какие состояния определили результаты восприятия, — эти оценки могут быть использованы для обучения моделей. Процесс обучения может работать через итеративный алгоритм обновления, получивший название "ожидание-максимизация" (*expectation–maximization* — EM), либо стать результатом байесовского обновления параметров моделей, исходя из имеющихся свидетельств. Подробнее об этом читайте в главе 20.

В оставшейся части этого раздела описываются общие алгоритмы для приведенных выше четырех задач вероятностного вывода, не зависящие от конкретного вида используемой модели. Возможные усовершенствования этих алгоритмов, специфические для каждой модели, рассматриваются в последующих разделах.

14.2.1. Фильтрация и предсказание

Как указывалось в разделе 7.7.3, полезный алгоритм фильтрации должен поддерживать оценку текущего состояния и обновлять ее, а не возвращаться назад по всей истории восприятий при каждом обновлении. (В противном случае стоимость каждого обновления будет увеличиваться с течением времени.) Другими словами, при заданном результате фильтрации на момент времени t агенту необходимо вычислить результат фильтрации для момента $t+1$, исходя из нового свидетельства $\mathbf{e}_{t+1}$. Итак, для некоторой функции f имеет место следующее:

$$\mathbf{P}(\mathbf{X}_{t+1} | \mathbf{e}_{1:t+1}) = f(\mathbf{e}_{t+1}, \mathbf{P}(\mathbf{X}_t | \mathbf{e}_{1:t})).$$

Этот процесс называют **рекурсивной оценкой**. (См. также разделы 4.4 и 7.7.3.) Эти вычисления можно рассматривать как состоящие из двух частей: сначала текущее распределение вероятностей проектируется вперед от t к $t+1$, а затем оно обновляется с использованием нового свидетельства $\mathbf{e}_{t+1}$. Такое двухэтапное протекание процесса выразить формально можно просто за счет перестановки.

$$\begin{aligned}
&\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}, \mathbf{e}_{t+1}) = && \text{(разделение свидетельства)} \\
&= \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) = && \text{(применение правила Байеса при заданном } \mathbf{e}_{1:t}) \\
&= \alpha \underbrace{\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})}_{\text{обновление}} \underbrace{\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})}_{\text{предсказание}} && \text{(по свойству марковости свидетельства)}
\end{aligned} \tag{14.4}$$

Здесь и далее в этой главе α — это нормализующая константа, используемая для того, чтобы вероятности в сумме составляли 1. Теперь добавим выражение для одношагового предсказания $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$, полученное обусловливанием вероятностей текущего состояния $\mathbf{X}_t$. Результирующее уравнение оценки нового состояния является центральным результатом в этой главе.

$$\begin{aligned}
&\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t, \mathbf{e}_{1:t}) P(\mathbf{x}_t \mid \mathbf{e}_{1:t}) = \\
&= \alpha \underbrace{\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})}_{\text{модель восприятия}} \sum_{\mathbf{x}_t} \underbrace{\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)}_{\text{модель перехода}} \underbrace{P(\mathbf{x}_t \mid \mathbf{e}_{1:t})}_{\text{рекурсия}} && \text{(по свойству марковости)}
\end{aligned} \tag{14.5}$$

Отфильтрованная оценка $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ может рассматриваться как "сообщение" $\mathbf{f}_{1:t}$, которое распространяется в прямом направлении вдоль последовательности состояний, модифицируется при каждом переходе и обновляется при получении каждого нового результата наблюдения. Этот процесс можно представить следующим образом:

$$\mathbf{f}_{1:t+1} = \text{FORWARD}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1}),$$

где функция FORWARD реализует обновление, описанное в уравнении (14.5), и процесс начинается с $\mathbf{f}_{1:0} = \mathbf{P}(\mathbf{X}_0)$. Если все переменные состояния являются дискретными, то затраты времени на каждое обновление остаются постоянными (т.е. не зависят от t) и потребность в пространстве также остается постоянной. (Эти постоянные показатели временной и пространственной сложности, безусловно, зависят от размера самого пространства состояний и конкретного типа используемой временной модели.) ➡ *Требования к затратам времени и пространства для обновления должны быть постоянными, если агенту с ограниченной памятью необходимо следить за распределением вероятностей текущего состояния на протяжении неограниченно долгой последовательности наблюдений.*

Проиллюстрируем процесс фильтрации, состоящий из двух этапов, на простом примере задачи с зонтиком (см. рис. 14.2). Сначала вычислим $\mathbf{P}(R_2 \mid u_{1:2})$.

- В день 0 пока еще нет ни одного наблюдения, однако у охранника могут быть некоторые собственные убеждения о вероятности дождя в этот день. Предположим, что их можно представить как $\mathbf{P}(R_0) = \langle 0{,}5;\ 0{,}5 \rangle$.
- В день 1 директор пришел с зонтиком, поэтому $U_1 = \mathit{true}$. Предсказание для перехода от $t = 0$ к $t = 1$ будет следующим:

$$\mathbf{P}(R_1) = \sum_{r_0} \mathbf{P}(R_1 \mid r_0) P(r_0) =$$
$$= \langle 0{,}7;\ 0{,}3 \rangle \times 0{,}5 + \langle 0{,}3;\ 0{,}7 \rangle \times 0{,}5 = \langle 0{,}5;\ 0{,}5 \rangle.$$

Затем на этапе обновления этот результат просто умножается на вероятности свидетельства для $t = 1$ и результат нормализуется, как это было представлено в уравнении (14.4).

$$\mathbf{P}(R_1 \mid u_1) = \alpha\, \mathbf{P}(u_1 \mid R_1) \mathbf{P}(R_1) = \alpha\, \langle 0{,}9;\ 0{,}2 \rangle \langle 0{,}5;\ 0{,}5 \rangle =$$
$$= \alpha\, \langle 0{,}45;\ 0{,}1 \rangle \approx \langle 0{,}818;\ 0{,}182 \rangle$$

- В день 2 также было отмечено наличие зонтика, поэтому $U_2 = \mathit{true}$. Предсказание для перехода от $t = 1$ к $t = 2$ будет следующим:

$$\mathbf{P}(R_2 \mid u_1) = \sum_{r_1} \mathbf{P}(R_2 \mid r_1) P(r_1 \mid u_1) =$$
$$= \langle 0{,}7;\ 0{,}3 \rangle \times 0{,}818 + \langle 0{,}3;\ 0{,}7 \rangle \times 0{,}182 \approx \langle 0{,}627;\ 0{,}373 \rangle,$$

а после их обновления с помощью свидетельства для $t = 2$ получаем

$$\mathbf{P}(R_2 \mid u_1, u_2) = \alpha\, \mathbf{P}(u_2 \mid R_2) \mathbf{P}(R_2 \mid u_1) = \alpha\, \langle 0{,}9;\ 0{,}2 \rangle \langle 0{,}627;\ 0{,}373 \rangle =$$
$$= \alpha\, \langle 0{,}565;\ 0{,}075 \rangle \approx \langle 0{,}883;\ 0{,}117 \rangle.$$

Интуитивно понятно, что вероятность дождя от дня 1 ко дню 2 повышается, поскольку дождь, видимо, продолжается. В упражнении 14.2, *а* предлагается продолжить исследование этой тенденции.

Задача **предсказания** может рассматриваться просто как фильтрация без добавления новых свидетельств. В действительности процесс фильтрации уже включает одношаговое предсказание, и поэтому можно легко вывести следующую формулу рекурсивного вычисления для предсказания состояния в момент времени $t + k + 1$ на основании предсказания для $t + k$:

$$\mathbf{P}(\mathbf{X}_{t+k+1} \mid \mathbf{e}_{1:t}) = \sum_{\mathbf{x}_{t+k}} \underbrace{\mathbf{P}(\mathbf{X}_{t+k+1} \mid \mathbf{x}_{t+k})}_{\text{модель перехода}} \underbrace{P(\mathbf{x}_{t+k} \mid \mathbf{e}_{1:t})}_{\text{рекурсия}}. \tag{14.6}$$

Естественно, что в это вычисление включена только модель перехода, но не модель восприятия.

Интересно рассмотреть, что произойдет при попытке предсказывать все дальше и дальше в будущее. Как показано в упражнении 14.2, *б*, прогнозируемое распределение для дождя сходится к фиксированной точке $\langle 0{,}5;\ 0{,}5\rangle$, после чего уже не изменяется.[4] Это так называемое **стационарное распределение** для марковского процесса, определяемого с помощью модели перехода (см. также раздел 13.4.2). О свойствах таких распределений и о ►**времени смешивания** (грубо говоря, о затратах времени, необходимых для достижения фиксированной точки) известно очень многое. С точки зрения практики эти знания сводятся к печальному выводу, что любая попытка предсказать *фактическое* состояние для количества этапов, превышающего лишь небольшую часть времени смешивания, обречена на неудачу, если только стационарное распределение само по себе не является острым пиком в небольшой области пространства состояний. Чем более неопределенной является модель перехода, тем короче будет время смешивания и тем более туманным становится будущее.

Помимо фильтрации и предсказания, рекурсия в прямом направлении может использоваться и для вычисления **правдоподобия** последовательности свидетельств, $P(\mathbf{e}_{1:t})$. Эта величина может оказаться полезной, если потребуется сравнить различные временны*е* модели, способные вырабатывать одну и ту же последовательность свидетельств (например, две различные модели для продолжительной дождливой погоды). Для такой рекурсии используется сообщение о правдоподобии $\ell_{1:t}(\mathbf{X}_t) = \mathbf{P}(\mathbf{X}_t, \mathbf{e}_{1:t})$. Несложно показать (см. упражнение 14.5), что справедливо следующее соотношение:

$$\ell_{1:t+1} = \text{FORWARD}(\ell_{1:t}, \mathbf{e}_{t+1}).$$

После вычисления $\ell_{1:t}$ получим фактическое значение правдоподобия, исключив суммированием значение $\mathbf{X}_t$:

$$L_{1:t} = \mathbf{P}(\mathbf{e}_{1:t}) = \sum_{\mathbf{x}_t} \ell_{1:t}(\mathbf{x}_t). \tag{14.7}$$

Обратите внимание, что с течением времени правдоподобие сообщения представляет вероятности все более и более длинных последовательностей свидетельств, и поэтому численно становится все меньше и меньше, что приводит к проблеме потери значимости, свойственной арифметике с плавающей точкой. На практике это важная проблема, но здесь мы не будем вдаваться в методы ее решения.

---

[4] Если в качестве момента $t = 0$ выбирается некий произвольный день, то имеет смысл выбрать априорное распределение $\mathbf{P}(Rain_0)$ соответствующим стационарному распределению, и именно поэтому распределение $\langle 0{,}5,\ 0{,}5\rangle$ было выбрано нами как априорное. Если бы в этом качестве было выбрано другое распределение, стационарное распределение все равно осталось бы тем же самым, $\langle 0{,}5,\ 0{,}5\rangle$.

14.2.2. Сглаживание

Как было указано выше, сглаживание — это процесс вычисления распределения вероятностей значений переменных в прошлых состояниях при наличии свидетельств вплоть до нынешнего состояния; иными словами, $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ для $0 \le k < t$ (рис. 14.3.) В предвидении еще одного рекурсивного подхода к передаче сообщений разделим необходимые вычисления на два этапа: применительно к свидетельствам вплоть до момента времени k и к свидетельствам от $k+1$ до t:

$$\begin{aligned}\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) = \\ &= \alpha\, \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\, \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{e}_{1:k}) = && \text{(по правилу Байеса с учетом } \mathbf{e}_{1:k}) \\ &= \alpha\, \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\, \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) = && \text{(по правилу условной независимости)} \\ &= \alpha\, \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}.\end{aligned} \tag{14.8}$$

Здесь символ "×" представляет операцию точечного умножения векторов. В этом уравнении было определено "обратное" сообщение $\mathbf{b}_{k+1:t} = \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k)$ по аналогии с прямым сообщением $\mathbf{f}_{1:k}$. Прямое сообщение $\mathbf{f}_{1:k}$ может быть вычислено фильтрацией в прямом направлении от 1 до k в соответствии с уравнением (14.5). Как оказалось, обратное сообщение $\mathbf{b}_{k+1:t}$ может быть вычислено с помощью рекурсивного процесса, осуществляемого в *обратном направлении* от t.

$$\begin{aligned}\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) &= \\ &= \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) = && \text{(обусловливание по } \mathbf{X}_{k+1}) \\ &= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t} \mid \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) = && \text{(правило условной независимости)} \\ &= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1}, \mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) = \\ &= \sum_{\mathbf{x}_{k+1}} \underbrace{P(\mathbf{e}_{k+1} \mid \mathbf{x}_{k+1})}_{\text{модель восприятия}} \underbrace{P(\mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1})}_{\text{рекурсия}} \underbrace{\mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k)}_{\text{модель перехода}}\end{aligned} \tag{14.9}$$

Здесь последний этап вычислений следует из свойства условной независимости $\mathbf{e}_{k+1}$ и $\mathbf{e}_{k+2:t}$ при заданном $\mathbf{x}_{k+1}$. Все три множителя в этой операции суммирования можно получить непосредственно с помощью модели либо из предыдущего обратного сообщения. Следовательно, это уравнение является искомой рекурсивной формулировкой. С использованием обозначения для сообщений получим следующее:

$$\mathbf{b}_{k+1:t} = \text{BACKWARD}(\mathbf{b}_{k+2:t}, \mathbf{e}_{k+1}),$$

где функция BACKWARD осуществляет обновление, описанное в уравнении (14.9). Как и при рекурсии в прямом направлении, затраты времени и пространства на каждое обновление являются постоянными и не зависят от t.

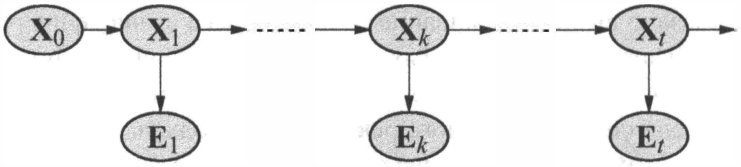


Рис. 14.3. Процесс сглаживания, в котором вычисляется $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ — распределение апостериорных вероятностей значений переменных в состоянии, наблюдавшемся в какой-то прошлый момент времени k с учетом полной последовательности наблюдений от 1 до t

Теперь становится очевидным, что оба терма, приведенные в уравнении (14.8), можно вычислить с помощью рекурсий по времени, одна из которых проходит в прямом направлении, от 1 до k, и в ней используется уравнение фильтрации (14.5), а другая проходит в обратном направлении, от t до $k+1$, и в ней используется уравнение (14.9).

Для инициализации обратной фазы можно воспользоваться выражением $\mathbf{b}_{t+1:t} = \mathbf{P}(\mathbf{e}_{t+1:t} \mid \mathbf{X}_t) = \mathbf{P}(\mid \mathbf{X}_t) = \mathbf{1}$, где $\mathbf{1}$ — это вектор из единиц, поскольку $\mathbf{e}_{t+1:t}$ — пустая последовательность, вероятность наблюдения которой равна 1.

Теперь применим этот алгоритм к примеру с зонтиком и вычислим сглаженную оценку вероятности дождя в момент времени $k = 1$ с учетом наблюдений о наличии зонтика в дни 1 и 2. Согласно уравнению (14.8), это значение определяется следующим образом:

$$\mathbf{P}(R_1 \mid u_1, u_2) = \alpha\, \mathbf{P}(R_1 \mid u_1)\, \mathbf{P}(u_2 \mid R_1). \tag{14.10}$$

Как нам уже известно, первый терм равен $\langle 0{,}818;\ 0{,}182 \rangle$, — по результатам применения процесса прямой фильтрации, описанного выше. Второй терм можно вычислить, применив обратную рекурсию по уравнению (14.9):

$$\begin{aligned}\mathbf{P}(u_2 \mid R_1) &= \sum_{r_2} P(u_2 \mid r_2) P(\mid r_2) \mathbf{P}(r_2 \mid R_1) = \\ &= (0{,}9 \times 1 \times \langle 0{,}7;\ 0{,}3 \rangle) + (0{,}2 \times 1 \times \langle 0{,}3;\ 0{,}7 \rangle) = \langle 0{,}69;\ 0{,}41 \rangle.\end{aligned}$$

Подставляя эти значения в уравнение (14.10), можно найти, что сглаженная оценка вероятности дождя в день 1 такова:

$$\mathbf{P}(R_1 \mid u_1, u_2) = \alpha \langle 0{,}818;\ 0{,}182 \rangle \times \langle 0{,}69;\ 0{,}41 \rangle \approx \langle 0{,}883;\ 0{,}117 \rangle.$$

Таким образом, в данном случае сглаженная оценка *выше*, чем отфильтрованная оценка (0,818). Это связано с тем, что появление директора с зонтиком в день 2 повышает вероятность того, что в день 2 шел дождь, что, в свою очередь, поскольку дожди часто являются затяжными, приводит к повышению вероятности дождя и в день 1.

И для прямой, и для обратной рекурсии требуется постоянное количество времени на каждый этап; поэтому временная сложность сглаживания по отношению к свидетельству $\mathbf{e}_{1:t}$ составляет $O(t)$. Это сложность сглаживания для определенного момента времени k. А если потребуется провести сглаживание всей последовательности, то один из очевидных методов состоит в выполнении всего процесса сглаживания по одному разу для каждого интервала времени, для которого необходимо выполнить сглаживание. Такой подход приводит к получению временной сложности $O(t^2)$.

Лучшим подходом можно считать использование очень простого приложения динамического программирования, позволяющего свести сложность расчетов к величине $O(t)$. Одну из подсказок, как это сделать, можно найти в приведенном выше анализе примера с зонтиком, в котором была показана возможность повторного использования результатов прямой фильтрации. Ключом к созданию алгоритма с линейными затратами времени является *регистрация результатов* прямой фильтрации по всей последовательности. В таком случае можно выполнить обратную рекурсию от t вплоть до 1, вычисляя сглаженную оценку для каждого этапа k из вычисленного обратного сообщения $\mathbf{b}_{k+1:t}$ и сохраненного прямого сообщения $\mathbf{f}_{1:k}$. Этот алгоритм, обоснованно называемый ▶ **прямым-обратным алгоритмом** (*forward-backward algorithm*), приведен на рис. 14.4.

---

function FORWARD-BACKWARD(**ev**, *prior*) **returns** вектор распределений вероятностей
  inputs: **ev**, вектор значений свидетельств для этапов 1, ..., t
    prior, распределение априорных вероятностей в начальном состоянии, $\mathbf{P}(\mathbf{X}_0)$
  local variables: **fv**, вектор прямых сообщений для этапов 0, ..., t
    b, представление обратного сообщения, первоначально состоящее из одних единиц
    sv, вектор сглаженных оценок для этапов 1, ..., t

  fv[0] ← *prior*
  for i = 1 **to** t **do**
    fv[i] ← FORWARD(**fv**[i − 1], **ev**[i])
  for i = t **down to** 1 **do**
    sv[i] ← NORMALIZE(**fv**[i] × **b**)
    b ← BACKWARD(**b**, **ev**[i])
  return sv

Рис. 14.4. Прямой-обратный алгоритм сглаживания: вычисление апостериорных вероятностей последовательности состояний при заданной последовательности наблюдений. Операторы FORWARD и BACKWARD определены в уравнениях (14.5) и (14.9) соответственно

Внимательный читатель должен был заметить, что структура байесовской сети, приведенной на рис. 14.3, представляет собой *полидерево*, как оно было определено в разделе 13.3.3. Это означает, что непосредственное применение алгоритма кластеризации также приводит к созданию алгоритма с линейными затратами времени, который вычисляет сглаженные оценки для всей последовательности. Поэтому вполне понятно, что прямой-обратный алгоритм фактически представляет собой частный случай алгоритма распространения в полидереве, используемого в сочетании с методами кластеризации (хотя эти два алгоритма были разработаны независимо).

Прямой-обратный алгоритм образует вычислительную основу для многих приложений, работающих с последовательностями зашумленных результатов наблюдений. Как было указано выше, он имеет два недостатка. Первым из них является пространственная сложность, которая может оказаться слишком высокой применительно к приложениям, в которых пространство состояний велико, а последовательности имеют большую длину. Его потребности в пространстве определяются как $O(|\mathbf{f}|t)$, где $|\mathbf{f}|$ — размер представления прямого сообщения. Потребность в пространстве можно уменьшить до $O(|\mathbf{f}|\log t)$ — за счет соответствующего увеличения временной сложности на коэффициент $\log t$, как это показано в упражнении 14.3. В некоторых случаях (см. раздел 14.3) может использоваться алгоритм с постоянными потребностями в пространстве.

Вторым недостатком этого базового алгоритма является то, что он требует модификации для использования в приложениях, работающих в режиме *реального времени*, когда сглаженные оценки должны вычисляться для более ранних временных срезов по мере того, как к концу последовательности непрерывно добавляются новые наблюдения. В таких случаях чаще всего используется ►**сглаживание с постоянным запаздыванием**, при котором требуется вычислять сглаженную оценку $\mathbf{P}(\mathbf{X}_{t-d} \mid \mathbf{e}_{1:t})$ для фиксированного значения d. Иначе говоря, в этом случае сглаживание выполняется для временного среза, отстоящего на d этапов от текущего времени t; и по мере возрастания t сглаживание не должно отставать. Безусловно, что прямой-обратный алгоритм можно вызывать на выполнение применительно к данным d-этапного "окна" по мере добавления результатов каждого нового наблюдения, но такой подход, по-видимому, будет неэффективным. В разделе 14.3 будет показано, что сглаживание с постоянным запаздыванием в некоторых случаях может выполняться за постоянное время в расчете на каждое обновление, независимо от запаздывания d.

14.2.3. Поиск наиболее вероятной последовательности

Предположим, что [*true*, *true*, *false*, *true*, *true*] — последовательность наблюдений за наличием зонтика, проведенных охранником в первые пять дней своей вахты. Какая последовательность состояний погоды может стать наиболее вероятным объяснением для этих данных? Означает ли отсутствие зонтика в день 3, что

дождя не было или что директор просто забыл его взять? А если в день 3 дождя не было, то, возможно, дождя не было и в день 4 (поскольку погода обычно является устойчивой), однако директор захватил с собой зонтик просто на всякий случай. В целом существует 2^5 возможных последовательностей состояний погоды, которые могут быть приняты в качестве объяснения. Но есть ли способ найти наиболее вероятную из этих последовательностей, не перебирая их все?

Один из возможных подходов состоит в использовании следующей процедуры с линейными затратами времени: методом сглаживания найти распределения апостериорных вероятностей погоды на каждом временном интервале, а затем составить искомую последовательность, на каждом этапе выбрав наиболее вероятное состояние погоды в соответствии с полученными апостериорными вероятностями. Но такой подход должен вызвать у читателя сомнения, поскольку апостериорные вероятности, вычисленные методом сглаживания, представляют собой распределения вероятностей для *отдельных* временных интервалов, тогда как для нахождения наиболее вероятной *последовательности* необходимо рассматривать *совместные* вероятности по всем временным интервалам. Полученные результаты в этих двух случаях могут оказаться существенно различающимися (см. упражнение 14.4).

Алгоритм поиска наиболее вероятной последовательности с линейными затратами времени все же *существует*, но это требует дополнительного осмысления задачи. Он должен быть основан на том же свойстве марковости, которое обеспечило построение эффективных алгоритмов фильтрации и сглаживания. Идея состоит в том, чтобы рассматривать каждую последовательность как *путь* в графе, узлами которого являются возможные *состояния* на каждом временном этапе. Подобный граф для мира задачи с зонтиком представлен на рис. 14.5, *а*. Рассмотрим задачу поиска наиболее вероятного пути через этот граф, в котором значение правдоподобия любого пути представляет собой произведение вероятностей перехода вдоль этого пути и вероятностей фактических наблюдений в каждом состоянии.

Давайте сосредоточимся, в частности, на путях, которые достигают состояния $Rain_5 = true$. Исходя из свойства марковости, можно полагать, что наиболее вероятный путь к состоянию $Rain_5 = true$ состоит из наиболее вероятного пути к *некоторому* состоянию, достигнутому на этапе 4, за которым следует переход в состояние $Rain_5 = true$, а состоянием в момент времени 4, которое станет частью пути к состоянию $Rain_5 = true$, будет то, которое максимизирует значение правдоподобия этого пути. Иными словами, ➡ *существует рекурсивная связь между наиболее вероятными путями в каждое состояние* $\mathbf{x}_{t+1}$ *и наиболее вероятными путями в каждое состояние* $\mathbf{x}_t$.

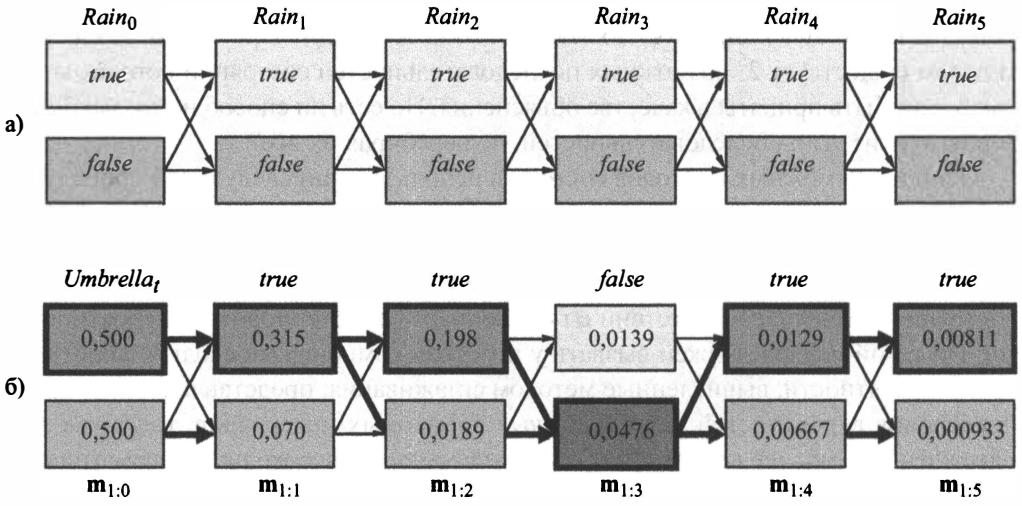


Рис. 14.5. а) Возможные последовательности состояний для переменной $Rain_t$ могут рассматриваться как пути через граф возможных состояний на каждом временном этапе (состояния представлены прямоугольниками, чтобы избежать путаницы с узлами в байесовской сети). **б)** Применение алгоритма Витерби к последовательности наблюдений за наличием зонтика, [*true*, *true*, *false*, *true*, *true*], где данные свидетельства начинают появляться с момента времени 1. Для каждого значения t показано значение сообщения $\mathbf{m}_{1:t}$, представляющее вероятность наилучшей последовательности, достигающей каждого состояния во время t. Кроме того, для каждого состояния ведущая к нему жирная стрелка задает его наилучшего предшественника, оцениваемого по произведению вероятности предшествующей последовательности и вероятности перехода. Наиболее вероятная последовательность определяется прохождением по жирным стрелкам в обратном направлении от наиболее вероятного состояния в $\mathbf{m}_{1:5}$ к исходному состоянию, — на рисунке состояния этой последовательности выделены жирным контуром и более темной заливкой

Это свойство можно использовать непосредственно для построения рекурсивного алгоритма вычисления наиболее вероятного пути при заданных переменных свидетельства. Будем использовать рекурсивно вычисленное сообщение $m_{1:t}$ как прямое сообщение $\mathbf{f}_{1:t}$ в алгоритме фильтрации. Это прямое сообщение определяется следующим образом:[5]

[5] Обратите внимание, что это не совсем те вероятности наиболее вероятных путей достижения состояний $\mathbf{X}_t$ с учетом свидетельства, которые являлись бы условными вероятностями $\max_{\mathbf{x}_{1:t-1}} \mathbf{P}(\mathbf{x}_{1:t-1}, \mathbf{X}_t \mid \mathbf{e}_{1:t})$, а два вектора, связанных постоянным множителем $P(\mathbf{e}_{1:t})$. Однако это различие несущественно, поскольку результат операции max не зависит от постоянных множителей, — определив $\mathbf{m}_{1:t}$ подобным образом, мы получили немного более простую рекурсию.

$$\mathbf{m}_{1:t} = \max_{\mathbf{x}_{1:t-1}} \mathbf{P}(\mathbf{x}_{1:t-1}, \mathbf{X}_t, \mathbf{e}_{1:t}).$$

Чтобы получить рекурсивное соотношение между $\mathbf{m}_{1:t+1}$ и $\mathbf{m}_{1:t}$, можно повторить приблизительно те же этапы, которые использовались в уравнении (14.5):

$$\begin{aligned}
\mathbf{m}_{1:t+1} &= \max_{\mathbf{x}_{1:t}} \mathbf{P}(\mathbf{x}_{1:t}, \mathbf{X}_{t+1}, \mathbf{e}_{1:t+1}) = \max_{\mathbf{x}_{1:t}} \mathbf{P}(\mathbf{x}_{1:t}, \mathbf{X}_{t+1}, \mathbf{e}_{1:t}, e_{t+1}) = \\
&= \max_{\mathbf{x}_{1:t}} \mathbf{P}(e_{t+1} \mid \mathbf{x}_{1:t}, \mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \mathbf{P}(\mathbf{x}_{1:t}, \mathbf{X}_{t+1}, \mathbf{e}_{1:t}) = \\
&= \mathbf{P}(e_{t+1} \mid \mathbf{X}_{t+1}) \max_{\mathbf{x}_{1:t}} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t) P(\mathbf{x}_{1:t}, \mathbf{e}_{1:t}) = \\
&= \mathbf{P}(e_{t+1} \mid \mathbf{X}_{t+1}) \max_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t) \max_{\mathbf{x}_{1:t-1}} P(\mathbf{x}_{1:t-1}, \mathbf{x}_t, \mathbf{e}_{1:t}),
\end{aligned} \tag{14.11}$$

где последний член $\max_{\mathbf{x}_{1:t-1}} P(\mathbf{x}_{1:t-1}, \mathbf{x}_t, \mathbf{e}_{1:t})$ — это в точности элемент конкретного состояния $\mathbf{x}_t$ в векторе сообщения $\mathbf{m}_{1:t}$. По существу уравнение (14.11) идентично уравнению фильтрации (14.5) за исключением того, что суммирование по $\mathbf{x}_t$ в уравнении (14.5) здесь заменено максимизацией по $\mathbf{x}_t$ и в уравнении (14.11) отсутствует константа нормализации α. Таким образом, алгоритм для вычисления наиболее вероятной последовательности подобен алгоритму фильтрации: он начинает работу в момент времени 0 с априорным распределением $\mathbf{m}_{1:0} = \mathbf{P}(\mathbf{X}_0)$, а затем проходит в прямом направлении вдоль последовательности и вычисляет сообщение $\mathbf{m}$ на каждом временном этапе, используя уравнение (14.11). Ход этих вычислений представлен на рис. 14.5, *б*.

В конце последовательности наблюдений сообщение $\mathbf{m}_{1:t}$ будет содержать вероятность наиболее вероятной последовательности, достигающей *каждого* из конечных состояний. Следовательно, теперь можно будет легко выбрать конечное состояние наиболее вероятной последовательности в целом (на рис. 14.5, *б* оно выделено утолщенной рамкой на этапе 5). Чтобы иметь возможность фактически определить оптимальную последовательность, а не просто вычислить ее вероятность, в этом алгоритме необходимо поддерживать указатели от каждого состояния обратно к наилучшему состоянию, приведшему к нему, — на рис. 14.5, *б* эти указатели показаны жирными стрелками. Тогда фактическую оптимальную последовательность можно будет определить, следуя по этим указателям в обратном направлении, начиная от наилучшего конечного состояния.

Только что описанный алгоритм называется ►**алгоритмом Витерби** в честь его создателя Эндрю Витерби. Временная сложность этого алгоритма, как и алгоритма фильтрации, линейно зависит от t, т.е. от длины последовательности. Однако, в отличие от алгоритма фильтрации, использующего постоянный объем пространства, потребность алгоритма Витерби в пространстве также линейно зависит от t. Это связано с тем, что в алгоритме Витерби необходимо следить за указателями, определяющими наилучшую последовательность, ведущую к каждому состоянию.

И еще один, последний, практический момент: для алгоритма Витерби существенной проблемой является возможность потери значимости. На рис. 14.5, *б* вероятности на каждом этапе становятся все меньше и меньше, а ведь это лишь игрушечный пример. Реальные приложения для анализа ДНК или декодирования сообщений могут работать с последовательностями в тысячи и миллионы этапов. Одним возможным решением является простая нормализация сообщения $\mathbf{m}$ на каждом этапе, — подобное изменение масштаба никак не влияет на правильность расчетов, поскольку $\max(cx, cy) = c \cdot \max(x, y)$. Второе решение заключается в повсеместном использования логарифмов вероятностей и замене операций умножения сложением. И в этом случае правильность расчетов остается неизменной, поскольку логарифмическая функция монотонна; поэтому $\max(\log x, \log y) = \log \max(x, y)$.

14.3. Скрытые марковские модели

В предыдущем разделе были разработаны алгоритмы формирования временных вероятностных рассуждений с использованием общей инфраструктуры, независимой от конкретной формы моделей перехода и моделей восприятия, а также независимой от природы состояний и переменных свидетельства. В этом и следующих двух разделах будут обсуждаться более конкретные модели и приложения, иллюстрирующие мощь этих простых алгоритмов, а в некоторых случаях допускающие и дополнительные усовершенствования.

Начнем с обсуждения ▶**скрытой марковской модели**, или сокращенно **HMM** (*Hidden Markov Model*). Любая модель HMM — это временная вероятностная модель, в которой состояние процесса описано с помощью *единственной дискретной* случайной переменной. Возможными значениями этой переменной являются возможные состояния мира. Пример задачи с зонтиком, обсуждавшийся в предыдущем разделе, представляет собой одну из моделей HMM, поскольку в нем применяется лишь единственная переменная состояния, $Rain_t$. А как поступить, если в модели имеется две или более переменных состояния? В этом случае выхода за пределы инфраструктуры HMM можно избежать за счет комбинирования этих переменных в единственную "мегапеременную", значениями которой являются все возможные кортежи значений отдельных переменных состояния. Далее будет показано, что благодаря ограниченной структуре моделей HMM появляется возможность создавать простые и элегантные матричные реализации всех основных алгоритмов.[6]

Хотя в скрытой марковской модели требуется наличие единственной дискретной переменной *состояния*, в ней нет подобных ограничений в отношении

[6] Читателю, не знакомому с основными операциями над векторами и матрицами, может потребоваться обратиться к приложению А, прежде чем продолжить чтение данного раздела.

переменных *свидетельства*. Причина в том, что переменные свидетельства всегда наблюдаемы, а это означает, что нет необходимости отслеживать какое-либо распределение по их значениям. (Если переменная не наблюдается, ее можно просто исключить из модели для данного временного этапа.) В модели может присутствовать много переменных свидетельства, как дискретных, так и непрерывных.

14.3.1. Упрощенные матричные алгоритмы

При наличии лишь единственной дискретной переменной состояния X_t можно определить более сжатую форму представлений модели перехода, модели восприятия, а также прямых и обратных сообщений. Предположим, что переменная состояния X_t имеет значения, обозначенные целыми числами 1,... , S, где S — количество возможных состояний. В таком случае модель перехода $\mathbf{P}(X_t \mid X_{t-1})$ преобразуется в матрицу $\mathbf{T}$ размером $S \times S$, где

$$\mathbf{T}_{ij} = P(X_t = j \mid X_{t-1} = i).$$

Таким образом, матрица перехода $\mathbf{T}_{ij}$ содержит вероятности перехода из состояния i в состояние j. Например, если в задаче с зонтиком обозначить состояния $Rain = true$ и $Rain = false$ как 1 и 2 соответственно, то матрица перехода для этого мира, как он определен на рис. 14.5, будет следующей:

$$\mathbf{T} = \mathbf{P}(X_t \mid X_{t-1}) = \begin{pmatrix} 0,7 & 0,3 \\ 0,3 & 0,7 \end{pmatrix}.$$

Теперь переведем в матричную форму модель восприятия. В этом случае, поскольку значение переменной свидетельства E_t в момент времени t известно (обозначим его как e_t), для каждого состояния необходимо определить лишь то, насколько вероятно, что данное состояние вызовет появление значения e_t, т.е. нас интересует $P(e_t \mid X_t = i)$ для каждого состояния i. Для удобства с точки зрения математики поместим эти значения в диагональную ▶ **матрицу наблюдений** $\mathbf{O}t$ размером $S \times S$ для каждого временного интервала t. Иначе говоря, составим матрицу $\mathbf{O}_t$, в которой i-е диагональные элементы представлены значениями $P(e_t \mid X_t = i)$, а все остальные элементы равны 0. Например, для задачи с зонтиком в день 1, как показано на рис. 14.5, было получено значение $U_1 = true$, а в день 3 — $U_3 = false$, поэтому имеем следующее:

$$\mathbf{O}_1 = \begin{pmatrix} 0,9 & 0 \\ 0 & 0,2 \end{pmatrix}; \qquad \mathbf{O}_3 \begin{pmatrix} 0,1 & 0 \\ 0 & 0,8 \end{pmatrix}.$$

Теперь, если для представления прямых и обратных сообщений использовать векторы столбцов, то все вычисления преобразуются в простые матрично-векторные операции. Прямое уравнение (14.5) принимает вид

$$\mathbf{f}_{1:t+1} = \alpha \mathbf{O}_{t+1}\mathbf{T}^{\top}\mathbf{f}_{1:t}, \tag{14.12}$$

а обратное уравнение (14.9) становится следующим:

$$\mathbf{b}_{k+1:t} = \mathbf{T}\mathbf{O}_{k+1}\mathbf{b}_{k+2:t}. \tag{14.13}$$

Из этих уравнений видно, что временная сложность прямого-обратного алгоритма (см. рис. 14.4), применяемого к последовательности длиной t, равна $O(S^2t)$, поскольку на каждом этапе требуется умножать вектор с S элементами на матрицу размером $S \times S$. Потребность в пространстве характеризуется величиной $O(St)$, так как при проходе в прямом направлении необходимо сохранить в памяти t векторов размером S.

Помимо предоставления элегантного способа описания алгоритмов фильтрации и сглаживания для моделей HMM, такая матричная формулировка открывает возможности для создания улучшенных алгоритмов. Первым из них является простой вариант прямого-обратного алгоритма, позволяющий выполнить сглаживание с использованием *постоянного* пространства, независимо от длины последовательности. Идея этого алгоритма состоит в том, что для выполнения сглаживания в любом конкретном временном срезе k требуется одновременное присутствие в памяти и прямого, и обратного сообщений, $\mathbf{f}_{1:k}$ и $\mathbf{b}_{k+1:t}$, согласно уравнению (14.8). В прямом-обратном алгоритме это достигается за счет сохранения значений $\mathbf{f}$, вычисленных во время прохода в прямом направлении, чтобы они были доступны и во время прохода в обратном направлении. Другой способ достижения этой цели заключается в использовании одного прохода, в котором и значение $\mathbf{f}$, и значение $\mathbf{b}$ распространяются в одном и том же направлении. Например, можно добиться распространения "прямого" сообщения $\mathbf{f}$ в обратном направлении, преобразовав уравнение (14.12) таким образом, чтобы оно работало в другом направлении:

$$\mathbf{f}_{1:t} = \alpha'(\mathbf{T}^{\top})^{-1}\mathbf{O}_{t+1}^{-1}\mathbf{f}_{1:t+1}.$$

Этот модифицированный алгоритм сглаживания действует так, что в нем вначале осуществляется стандартный проход в прямом направлении для вычисления значения $\mathbf{f}_{t:t}$ (при этом все промежуточные результаты уничтожаются), после чего выполняется проход в обратном направлении для совместного вычисления $\mathbf{b}$ и $\mathbf{f}$, а затем эти значения используются для вычисления сглаженной оценки для каждого интервала. Поскольку требуется только одна копия каждого сообщения, потребности в памяти являются постоянными (т.е. независимыми от длины последовательности t). Тем не менее этот алгоритм имеет одно существенное ограничение — в нем требуется, чтобы матрица перехода была обратимой, а модель восприятия не имела нулей, иными словами, чтобы каждое наблюдение было возможным в любом состоянии.

Второе направление, в котором матричная формулировка предоставляет возможности для усовершенствования, — это оперативное сглаживание с постоянным

запаздыванием. Тот факт, что сглаживание может быть выполнено при фиксированных затратах пространства, наводит на мысль, что может существовать эффективный рекурсивный алгоритм для оперативного сглаживания, т.е. такой алгоритм, временная сложность которого будет независимой от величины запаздывания. Предположим, что запаздывание равно d; это означает, что сглаживание проводится во временном срезе $t-d$ и что текущее время равно t. Согласно уравнению (14.8), для среза $t-d$ потребуется рассчитать значение следующего выражения:

$$\alpha \mathbf{f}_{1:t-d} \times \mathbf{b}_{t-d+1:t}.$$

Затем, после поступления новых результатов наблюдения, необходимо будет вычислить следующее выражение для среза $t-d+1$:

$$\alpha \mathbf{f}_{1:t-d+1} \times \mathbf{b}_{t-d+2:t+1}.$$

Как можно выполнить эту операцию инкрементно? Прежде всего, можно вычислить $\mathbf{f}_{1:t-d+1}$ из $\mathbf{f}_{1:t-d}$, используя стандартный процесс фильтрации, в соответствии с уравнением (14.5).

Инкрементное вычисление обратного сообщения является более сложной задачей, поскольку не существует простого соотношения между старым обратным сообщением $\mathbf{b}_{t-d+1:t}$ и новым обратным сообщением $\mathbf{b}_{t-d+2:t+1}$. Вместо этого рассмотрим соотношение между старым обратным сообщением $\mathbf{b}_{t-d+1:t}$ и обратным сообщением в начале последовательности, $\mathbf{b}_{t+1:t}$. Для этого d раз применим уравнение (14.13), чтобы получить следующее уравнение:

$$\mathbf{b}_{t-d+1:t} = \left(\prod_{i=t-d+1}^{t} \mathbf{TO}_i \right) \mathbf{b}_{t+1:t} = \mathbf{B}_{t-d+1:t} \mathbf{1}. \tag{14.14}$$

Здесь матрица $\mathbf{B}_{t-d+1:t}$ является произведением последовательности матриц $\mathbf{T}$ и $\mathbf{O}$, а $\mathbf{1}$ — вектор из единиц. Матрицу $\mathbf{B}$ можно рассматривать как "оператор преобразования", который преобразует более позднее обратное сообщение в более раннее. Аналогичное уравнение остается справедливым для новых обратных сообщений, сформированных после поступления результатов следующего наблюдения:

$$\mathbf{b}_{t-d+2:t+1} = \left(\prod_{i=t-d+2}^{t+1} \mathbf{TO}_i \right) \mathbf{b}_{t+2:t+1} = \mathbf{B}_{t-d+2:t+1} \mathbf{1}. \tag{14.15}$$

Изучив выражения в произведениях в уравнениях (14.14) и (14.15), можно увидеть, что их связывает между собой простое соотношение: чтобы получить второе произведение, достаточно "разделить" первое произведение на первый элемент $\mathbf{TO}_{t-d+1}$ и умножить на новый последний элемент $\mathbf{TO}_{t+1}$. Поэтому на языке алгебры матриц можно записать следующее простое соотношение между старой и новой матрицами $\mathbf{B}$:

$$\mathbf{B}_{t-d+2:t+1} = \mathbf{O}_{t-d+1}^{-1} \mathbf{T}^{-1} \mathbf{B}_{t-d+1:t} \mathbf{T} \mathbf{O}_{t+1}. \tag{14.16}$$

Это уравнение предоставляет способ вычисления инкрементного обновления для матрицы **B**, которая, в свою очередь (в силу уравнения 14.15), позволяет вычислить новое обратное сообщение $\mathbf{b}_{t-d+2:t+1}$. Полный алгоритм, в котором предусматривается хранение и обновление значений **f** и **B**, представлен на рис. 14.6.

---

function FIXED-LAG-SMOOTHING(e_t, *hmm*, d) **returns** распределение по $\mathbf{X}_{t-d}$
  inputs: e_t, текущее свидетельство для временного интервала t
    hmm, скрытая марковская модель с матрицей переходов **T** размером $S \times S$
    d, величина запаздывания при сглаживании
  persistent: t, текущее время, исходно равно 1
    f, прямое сообщение $\mathbf{P}(X_t \mid e_{1:t})$, исходно *hmm*.PRIOR
    B, матрица d-этапного обратного преобразования, исходно единичная матрица
    $e_{t-d:t}$, двухсторонний список свидетельств от $t-d$ до t, исходно пустой
  local variables: $\mathbf{O}_{t-d}$, $\mathbf{O}_t$, диагональные матрицы, содержащие информацию модели восприятия

  добавить e_t в конец списка $e_{t-d:t}$
  $\mathbf{O}_t \leftarrow$ диагональная матрица, содержащая $\mathbf{P}(e_t | X_t)$
  if $t > d$ **then**
    $\mathbf{f} \leftarrow$ FORWARD($\mathbf{f}$, e_{t-d})
    удалить e_{t-d-1} из начала списка $e_{t-d:t}$
    $\mathbf{O}_{t-d} \leftarrow$ диагональная матрица, содержащая $\mathbf{P}(e_{t-d} | X_{t-d})$
    $\mathbf{B} \leftarrow \mathbf{O}_{t-d}^{-1}\mathbf{T}^{-1}\mathbf{B}\mathbf{T}\mathbf{O}_t$
  else $\mathbf{B} \leftarrow \mathbf{B}\mathbf{T}\mathbf{O}_t$
  $t \leftarrow t+1$
  if $t > \mathrm{d}+1$ **then return** NORMALIZE($\mathbf{f} \times \mathbf{B1}$) **else return** пустое значение

Рис. 14.6. Алгоритм сглаживания с постоянным временным запаздыванием на d этапов, реализованный как алгоритм реального времени, который вычисляет новую сглаженную оценку после получения данных наблюдения, относящихся к новому временному интервалу. Обратите внимание, что конечный результат NORMALIZE($\mathbf{f} \times \mathbf{B1}$) является просто произведением $\alpha\mathbf{f} \times \mathbf{b}$, согласно уравнению (14.14)

14.3.2. Пример скрытой марковской модели: определение местоположения

В разделе 4.4.4 была представлена простая форма задачи **локализации** для мира пылесоса: робот должен был определить свое текущее местоположение при наличии у него карты мира и заданной последовательности восприятий и действий. В той версии задачи робот обладал единственным недетерминированным действием *Move* и его датчики достоверно сообщали ему о наличии препятствий в

направлении на север, юг, восток и запад. Доверительное состояние робота представляло собой множество возможных мест, где он мог бы находиться.

В данном случае задача будет несколько более реалистичной за счет учета возможного шума в датчиках и формализации идеи о том, что робот перемещается случайным образом, — с равной вероятностью он может перейти в любой из соседних пустых квадратов. Переменная состояния X_t представляет местоположение робота в дискретном клеточном мире, — областью определения этой переменной является множество пустых квадратов, которые мы будем обозначать целыми числами $\{1,\ldots, S\}$. Далее, пусть NEIGHBORS(i) будет множеством пустых квадратов, смежных с квадратом i, и пусть $N(i)$ будет определять размер этого множества. Тогда модель перехода для действия *Move* должна говорить нам о том, что после его выполнения робот с равной вероятностью может оказаться в любой соседней клетке:

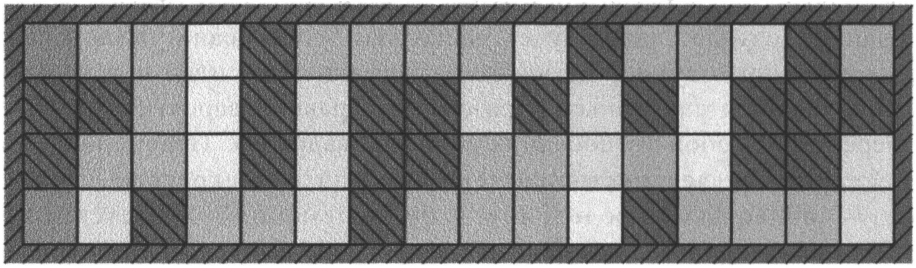
$$P(X_{t+1} = j \mid X_t = i) = \mathbf{T}_{ij} = \begin{cases} 1/N(i) & \text{если j} \in \text{NEIGHBORS}(i) \\ 0 & \text{в противном случае.} \end{cases}$$

Мы не знаем, откуда робот начинает движение, поэтому примем равномерное распределение по всем квадратам, т.е. $P(X_0 = i) = 1/S$. Для конкретного варианта этого мира, представленного на рис. 4.7, $S = 42$, а матрица перехода $\mathbf{T}$ имеет $42 \times 42 = 1764$ элемента.

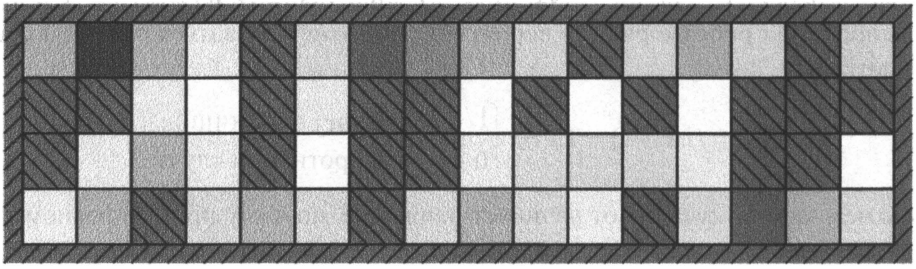
Переменная наблюдаемых значений датчика E_t может принимать 16 возможных значений длиной четыре бита, каждый из которых определяет наличие или отсутствие препятствия в направлении одной из сторон света в порядке N (*север*) – E (*восток*) – S (*юг*) – W (*запад*). Например, значение 1010 означает, что датчик сообщает о наличии препятствий в направлениях на север и юг, тогда как в направлении на восток или запад их нет. Предположим, что в каждом из направлений датчик может давать неверный результат с частотой ϵ и что ошибки по каждому из направлений происходят независимо друг от друга. В этом случае вероятность получения правильных значений для всех четырех битов будет равна $(1-\epsilon)^4$, а вероятность получения для всех четырех ошибочных значений — ϵ^4 соответственно. Более того, если через d_{it} обозначить расхождение — количество битов, которые не совпадают в истинном значении для квадрата i и фактически полученном восприятии e_t, то вероятность того, что робот в квадрате i получит от датчика восприятие e_t будет равна

$$P(E_t = e_t \mid X_t = i) = (\mathbf{O}_t)_{ii} = (1-\epsilon)^{4-d_{it}} \epsilon^{d_{it}}.$$

Например, вероятность того, что для квадрата с препятствиями в направлении на север и юг датчик выдаст показание 1110, будет равна $(1-\epsilon)^3\epsilon^1$.



а) Условное распределение для местоположения робота после получения восприятия E_1=1011



б) Условное распределение для местоположения робота после получения восприятия $E_1 = 1011, E_2 = 1010$

Рис. 14.7. Условное распределение для местоположения робота. **а)** После получения первого восприятия $E_1 = 1011$ (т.е. препятствия имеются в направлении на север, юг и запад). **б)** После перемещения в случайно выбранное смежное местоположение и получения второго восприятия $E_2 = 1010$ (т.е. препятствия имеются в направлении на север и юг). Цвет заливки каждой из клеток (без штриховки) соответствует вероятности того, что робот находится в ней, — чем она темнее, тем вероятность выше. Частота ошибок датчика для каждого бита $\epsilon = 0{,}2$

При заданных матрицах **T** и **O** робот может воспользоваться уравнением (14.12) для вычисления апостериорного распределения по всем возможным местоположениям, чтобы попытаться определить, где он находится. На рис. 14.7 приведены два таких распределения — $\mathbf{P}(X_1 \mid E_1 = 1011)$ и $\mathbf{P}(X_2 \mid E_1 = 1011, E_2 = 1010)$. Это тот же самый лабиринт, который был приведен на рис. 4.18 (раздел 4.4.4), но тогда мы использовали логическую фильтрацию для отыскания местоположений, которые были *возможны* при условии правильной работы датчика. В данном случае те же самые местоположения все также наиболее *вероятны*, но уже при зашумленном восприятии, из-за которого теперь у *каждого* местоположения есть некоторая ненулевая вероятность, поскольку в любом местоположении можно получить от датчика любые значения.

В дополнение к фильтрации для оценки своего текущего местоположения робот может использовать и сглаживание (уравнение (14.13)), что позволит ему установить, где он был в любой заданный момент времени в прошлом, например когда он начинал движение в момент времени 0. Также он может использовать алгоритм Витерби для определения наиболее вероятного пути, по которому он добрался туда, где сейчас находится. На рис. 14.8 представлены графики зависимости ошибок локализации и ошибок алгоритма Витерби в определении пути от числа наблюдений для различных значений частоты ошибок датчика на каждый бит ϵ. Даже когда ϵ равно 0,20 — а это означает, что в целом показания датчика являются ошибочными в 59% случаев, — робот, как правило, оказывается в состоянии определить свое местоположение в пределах двух квадратов уже после 20 наблюдений. Этот результат обеспечивается способностью алгоритма интегрировать свидетельства во времени и принимать во внимание вероятностные ограничения, налагаемые на последовательность местоположений моделью перехода. При $\epsilon = 0{,}10$ или менее робот нуждается лишь в нескольких наблюдениях, чтобы определить, где он находится, и точно установить свой путь. Однако при ϵ, равном 0,40, и ошибка локализации, и ошибка определения пути по алгоритму Витерби долгое время остаются значительными и почти неизменными, а это означает, что фактически робот потерялся. Это происходит потому, что датчик с частотой ошибки 0,40 дает роботу слишком мало информации, чтобы противодействовать потере информации о его местоположении, происходящей от непредсказуемости случайного движения.

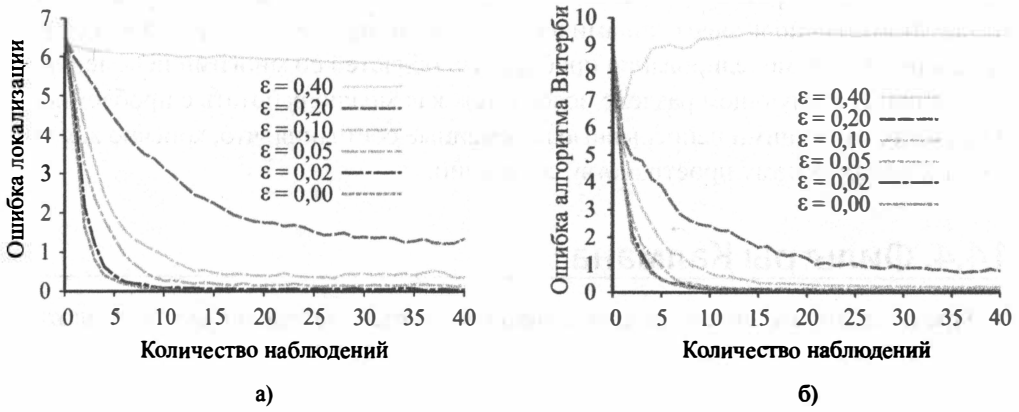


Рис. 14.8. Производительность HMM-локализации как функция от длины последовательности наблюдений для различных значений частоты ошибки датчика ϵ (данные усреднены по 400 прогонам). **а)** Ошибка локализации, определяемая как манхеттенское расстояние от истинного местоположения. **б)** Ошибка в определении пути по алгоритму Витерби, определяемая как среднее манхеттенское расстояние от состояний на пути, определенном алгоритмом Витерби, до соответствующих состояний на истинном пути

В примере, рассматриваемом в этом разделе, переменной состояния является физическое местоположение в мире. В других задачах, конечно же, могут анализироваться другие аспекты мира. В упражнении 14.8 предлагается рассмотреть иную версию мира пылесоса, в которой для робота установлено ограничение — двигаться в одном направлении до тех пор, пока это будет возможно. Выбрать новое направление он может, только натолкнувшись на препятствие. Для моделирования этого робота каждое состояние в модели должно состоять уже из пары переменных — *location* (местоположение) и *heading* (направление). Для среды, показанной на рис. 14.7 и включающей 42 пустых квадрата, это приводит к 168 состояниям и переходной матрице размером $168^2 = 28\,224$ элемента — все еще вполне приемлемое количество.

Если добавить возможность присутствия мусора в каждом из 42 квадратов, число состояний возрастет до 2^{42}, а матрица переходов будет включать уже более 10^{29} элементов, что превышает любые мыслимые пределы. В общем случае, если состояние включает n дискретных переменных с не более чем d значениями для каждой, соответствующая HMM-матрица переходов будет иметь размер $O(d^{2n})$ и время вычисления каждого обновления также будет пропорционально $O(d^{2n})$.

По этой причине, хотя скрытые марковские модели находят множество применений в самых различных областях — от распознавания речи до молекулярной биологии, — они принципиально ограничены в способности представлять сложные процессы. Согласно терминологии, предложенной в главе 2, в моделях HMM используется *атомарное* представление: состояния мира не имеют внутренней структуры и просто помечены целыми числами. В разделе 14.5 будет показано, как можно использовать динамические байесовские сети — *развернутое* представление — для моделирования проблемных областей со многими переменными состояния. В следующем разделе поясняется, как можно работать с проблемными областями, имеющими непрерывные переменные состояния, что, конечно же, приводит к бесконечному пространству состояний.

14.4. Фильтры Калмана

Представьте, что вы следите за маленькой птицей, летящей в сумраке плотной листвы джунглей: вы замечаете лишь краткие отрывочные моменты ее движения и каждый раз пытаетесь угадать, где сейчас находится птица и где она появится в следующий момент, чтобы ее не потерять. Или вообразите себя оператором радара во время Второй мировой войны, напряженно следящим за крошечной блуждающей вспышкой, появляющийся на экране через каждые 10 секунд. А если вернуться в прошлое еще дальше, вообразите, что вы, как Кеплер, пытаетесь реконструировать орбиты движения планет на основании совокупности крайне неточных угловых измерений, полученных через нерегулярные и неточно измеренные интервалы времени.

Во всех этих случаях вы выполняете то, что называют фильтрацией: оценку значений переменных состояния (здесь это положение и скорость движущегося объекта) по зашумленным результатам наблюдений во времени. Если бы эти переменные были дискретными, можно было бы описать систему, воспользовавшись скрытой марковской моделью. Но в приведенных примерах переменные являются непрерывными, поэтому в этом разделе рассматриваются методы обработки непрерывных переменных с использованием алгоритма, называемого ▶**фильтрацией Калмана**, по имени одного из его изобретателей — Рудольфа Калмана.

Например, траектория полета птицы может быть задана шестью непрерывными переменными в каждый момент времени: тремя — для положения (X_t, Y_t, Z_t) и тремя — для скорости ($\dot{X}_t$, $\dot{Y}_t$, $\dot{Z}_t$). Необходимо также иметь соответствующие плотности условных вероятностей для представления модели перехода и модели восприятия. Как и в главе 13, здесь будут использоваться **линейные гауссовы распределения**. Это означает, что следующее состояние $\mathbf{X}_{t+1}$ должно представлять собой линейную функцию от текущего состояния $\mathbf{X}_t$ с добавлением некоторого количества гауссова шума, — такое условие, как оказалось, является весьма оправданным на практике. Рассмотрим, например, координату X птицы, игнорируя на данный момент все другие координаты. Допустим, что интервал между наблюдениями равен Δ, и предположим, что в пределах этого интервала птица летит с постоянной скоростью. Тогда через каждый интервал ее новое положение можно будет определить с помощью уравнения $X_{t+\mathrm{D}} = X_t + \dot{X}\,\Delta$. После введения в него гауссова шума получим линейную гауссову модель перехода:

$$P(X_{t+\Delta} = x_{t+\Delta} \mid X_t = x_t, \dot{X}_t = \dot{x}_t) = \mathcal{N}(x_{t+\Delta}; x_t + \dot{x}_t\Delta, \sigma^2).$$

Структура байесовской сети для системы с векторами положения $\mathbf{X}_t$ и скорости $\dot{\mathbf{X}}_t$ показана на рис. 14.9. Обратите внимание на то, что это — весьма специфичная форма линейной гауссовой модели. Общая форма этой модели будет описана ниже в этом разделе, — она охватывает самый широкий спектр приложений, далеко выходящий за рамки простейшего примера моделирования движения, приведенного в первом абзаце этого раздела. Читателю может потребоваться обратиться к приложению А для ознакомления с некоторыми математическими свойствами гауссовых распределений. Для наших непосредственных целей наиболее важным из них является то, что **многомерное гауссово** распределение для d переменных задается d-элементным средним μ и матрицей ковариации Σ размером $d \times d$.

14.4.1. Обновление гауссовых распределений

В главе 13 (раздел 13.2.3) было описано ключевое свойство семейства линейных гауссовых распределений: при операциях обновления в байесовской сети оно остается замкнутым. (То есть при любом заданном свидетельстве апостериорное распределение остается по-прежнему относящимся к семейству линейных

гауссовых распределений.) В этом разделе данное утверждение будет уточнено в контексте фильтрации в рамках временной вероятностной модели. Ниже перечислены требуемые свойства, соответствующие процессу двухэтапного вычисления результатов фильтрации с помощью уравнения (14.5).

1. Если текущее распределение $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ является гауссовым, а модель перехода $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)$ — линейной гауссовой, то прогнозируемое на один этап вперед распределение, заданное уравнением

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) = \int_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t) P(\mathbf{x}_t \mid \mathbf{e}_{1:t}) d\mathbf{x}_t , \tag{14.17}$$

также является гауссовым распределением.

2. Если прогнозируемое распределение $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ является гауссовым и модель восприятия $\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})$ является линейной гауссовой, то после обусловливания на основании нового свидетельства следующее обновленное распределение, определяемое как

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = \alpha \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}), \tag{14.18}$$

также является гауссовым распределением.

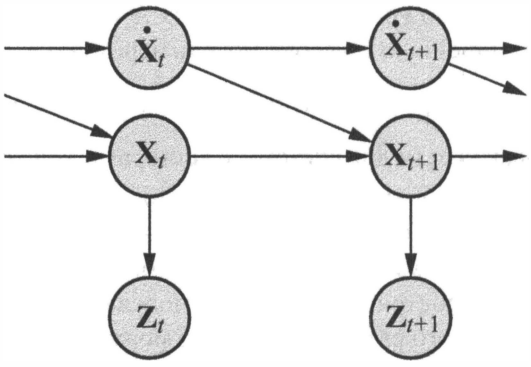


Рис. 14.9. Структура байесовской сети для линейной динамической системы с переменными, определяющими положение $\mathbf{X}_t$, скорость $\dot{\mathbf{X}}_t$ и результаты измерения позиции $\mathbf{Z}_t$

Таким образом, оператор FORWARD для калмановской фильтрации принимает на входе гауссово прямое сообщение $\mathbf{f}_{1:t}$, заданное с помощью среднего μ_t и матрицы ковариации Σ_t, и вырабатывает новое многомерное гауссово прямое сообщение $\mathbf{f}_{1:t+1}$, заданное с помощью среднего μ_{t+1} и матрицы ковариации Σ_{t+1}. Поэтому, если начать с гауссова априорного сообщения $\mathbf{f}_{1:0} = \mathbf{P}(\mathbf{X}_0) = \mathcal{N}(\mu_0, \Sigma_0)$ и провести

фильтрацию с помощью линейной гауссовой модели, можно получить гауссово распределение вероятностей состояний для любых временных срезов.

Очевидно, что это привлекательный и элегантный результат, но почему он имеет такое большое значение? Причина состоит в том, что за исключением нескольких частных случаев, подобных рассматриваемому, ➡ *в процессе фильтрации с использованием непрерывных или гибридных (дискретных и непрерывных) сетей вырабатываются распределения вероятностей состояний, размеры представления которых растут во времени без ограничения.* Это утверждение нелегко доказать, но в упражнении 14.12 показано, что в простых примерах так и происходит.

14.4.2. Простой одномерный пример

Выше уже было сказано, что оператор FORWARD для фильтра Калмана отображает исходное гауссово распределение на новое гауссово распределение. Применение этого оператора сводится к вычислению новых значений среднего и матрицы ковариации из предыдущих значений среднего и матрицы ковариации. Для вывода правила обновления в общем (многомерном) случае требуется большой объем выкладок в линейной алгебре, поэтому здесь мы пока остановимся на очень простом одномерном случае, а позже будут представлены результаты для общего случая. Но даже в одномерном случае вычисления являются довольно трудоемкими, однако авторы считают, что с ними следует ознакомиться, поскольку применимость фильтра Калмана слишком тесно связана с математическими свойствами гауссовых распределений.

Во временной модели, которая будет здесь рассматриваться, представлено **случайное блуждание** единственной непрерывной переменной состояния X_t, отслеживаемое посредством зашумленных результатов наблюдения Z_t. Одним из подходящих реальных примеров может служить показатель "доверия потребителя", который может быть промоделирован как переменная, каждый месяц подвергающаяся случайному изменению с вероятностью, представленной с помощью гауссова распределения, и измеряемая с помощью опроса случайно выбранных потребителей, в котором также вносится гауссов шум формирования выборки. Предполагается, что распределение априорных вероятностей является гауссовым с дисперсией σ_0^2:

$$P(x_0) = \alpha e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)}.$$

(Для упрощения в этом разделе мы будем использовать один и тот же символ α для обозначения всех констант нормализации.) В модели перехода просто добавляется гауссово возмущение постоянной дисперсии σ_x^2 к текущему состоянию:

$$P(x_{t+1} \mid x_t) = \alpha e^{-\frac{1}{2}\left(\frac{(x_{t+1}-x_t)^2}{\sigma_x^2}\right)}.$$

Модель восприятия предполагает наличие гауссова шума с дисперсией σ_z^2:

$$P(z_t \mid x_t) = \alpha e^{-\frac{1}{2}\left(\frac{(z_t - x_t)^2}{\sigma_z^2}\right)}.$$

Теперь, с учетом распределения априорных вероятностей $P(X_0)$, прогнозируемое на один этап распределение можно получить из уравнения (14.17):

$$P(x_1) = \int_{-\infty}^{\infty} P(x_1 \mid x_0) P(x_0) dx_0 = \alpha \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{(x_1 - x_0)^2}{\sigma_x^2}\right)} e^{-\frac{1}{2}\left(\frac{(x_0 - \mu_0)^2}{\sigma_0^2}\right)} dx_0 =$$

$$= \alpha \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{\sigma_0^2 (x_1 - x_0)^2 + \sigma_x^2 (x_0 - \mu_0)^2}{\sigma_0^2 \sigma_x^2}\right)} d_{x_0}.$$

Этот интеграл выглядит довольно сложно. Ключом к его упрощению может стать замечание, что экспонента представляет собой сумму двух выражений, которые *квадратично* зависят от x_0, и поэтому сама экспонента квадратично зависит от x_0. Простой прием, известный как ▶**дополнение до полного квадрата**, позволяет переписать любое квадратное уравнение $ax^2 + bx + c$ как сумму возведенного в квадрат терма $a(x - \frac{-b}{2a})^2$ и остаточного терма $c - \frac{b^2}{4a}$, который не зависит от x. В данном случае у нас $a = (\sigma_0^2 + \sigma_x^2) / (\sigma_0^2 \sigma_x^2)$, $b = -2(\sigma_0^2 x_1 + \sigma_x^2 \mu_0) / (\sigma_0^2 \sigma_x^2)$ и $c = (\sigma_0^2 x_1^2 + \sigma_x^2 \mu_0^2 / (\sigma_0^2 \sigma_x^2)$. Соответствующий остаточный терм может быть вынесен за пределы интеграла, что дает нам следующее уравнение:

$$P(x_1) = \alpha e^{-\frac{1}{2}\left(c - \frac{b^2}{4a}\right)} \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(a\left(x_0 - \frac{-b}{2a}\right)^2\right)} dx_0.$$

Теперь рассматриваемый интеграл представляет собой обычный интеграл гауссова распределения по всей области его определения, который равен 1. Таким образом, от квадратного уравнения сохраняется лишь его остаточный терм. Подставив обратно выражения для a, b и c, а затем упростив, получаем следующее:

$$P(x_1) = \alpha e^{\frac{1}{2}\left(\frac{(x_1 - \mu_0)^2}{\sigma_0^2 + \sigma_x^2}\right)}.$$

Таким образом, распределение, прогнозируемое на один этап, представляет собой гауссово распределение с тем же средним μ_0 и дисперсией, равной сумме первоначальной дисперсии σ_0^2 и дисперсии перехода σ_x^2.

Для завершения этапа обновления необходимо обусловить вероятность результатами наблюдения на первом временном этапе, а именно — z_1. Согласно уравнению (14.18), эта операция определяется следующим уравнением:

$$P(x_1|z_1) = \alpha P(z_1|x_1)P(x_1) =$$
$$= \alpha e^{-\frac{1}{2}\left(\frac{(z_1-x_1)^2}{\sigma_z^2}\right)} e^{-\frac{1}{2}\left(\frac{(x_1-\mu_0)^2}{\sigma_0^2+\sigma_x^2}\right)}.$$

Снова объединим экспоненты и дополним квадрат (упражнение 14.13), получив следующее:

$$P(x_1|z_1) = \alpha e^{-\frac{1}{2}\frac{\left(x_1 - \frac{(\sigma_0^2+\sigma_x^2)z_1 + \sigma_z^2\mu_0}{\sigma_0^2+\sigma_x^2+\sigma_z^2}\right)^2}{(\sigma_0^2+\sigma_x^2)\sigma_z^2/(\sigma_0^2+\sigma_x^2+\sigma_z^2)}}. \qquad (14.19)$$

Следовательно, после одного цикла обновления будет получено новое гауссово распределение для переменной состояния.

Из гауссовой формулы, приведенной в уравнении (14.19), можно видеть, что новые значения среднего и среднеквадратичного отклонений можно вычислить на основе старых значений среднего и среднеквадратичного отклонений следующим образом:

$$\mu_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2)z_{t+1} + \sigma_z^2\mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2} \text{ и } \sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2)\sigma_z^2}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}. \qquad (14.20)$$

На рис. 14.10 показан один цикл обновления фильтра Калмана в одномерном случае для конкретных значений модели перехода и модели восприятия.

Пара уравнений (14.20) здесь играет точно такую же роль, как и общее уравнение фильтрации (14.5) или уравнение фильтрации для HMM (14.12). Но из-за особого характера гауссовых распределений эти уравнения обладают некоторыми интересными дополнительными свойствами.

Во-первых, вычисление нового значения среднего μ_{t+1} можно интерпретировать как вычисление *взвешенного среднего* от новых результатов наблюдения z_{t+1} и прежнего значения среднего μ_t. Если результаты наблюдения ненадежны, то значение σ_z^2 увеличивается и больший вес придается старому значению среднего. Если же ненадежно старое значение среднего (значение σ_t^2 велико) или процесс крайне непредсказуем (велико значение σ_x^2), то больший вес придается результатам наблюдения.

Во-вторых, обратите внимание, что обновление для дисперсии σ_{t+1}^2 является *независимым от результатов наблюдения*, поэтому с помощью вычислений можно заранее определить, какой должна быть последовательность значений дисперсии. В-третьих, последовательность значений дисперсии быстро сходится к постоянному значению, которое зависит только от σ_x^2 и σ_z^2, что существенно упрощает дальнейшие вычисления (см. упражнение 14.14).

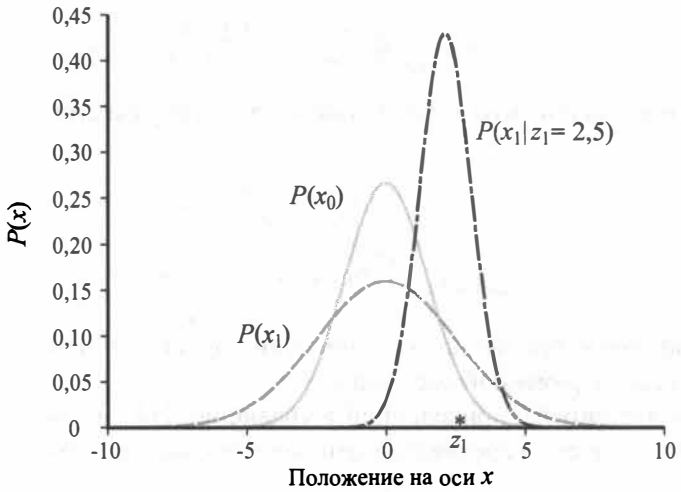


Рис. 14.10. Этапы цикла обновления фильтра Калмана для случайного блуждания с априорной вероятностью, заданной параметрами $\mu_0 = 0{,}0$ и $\sigma_0 = 1{,}5$, шумом перехода, заданным как $\sigma_x = 2{,}0$, шумом восприятия, заданным как $\sigma_z = 1{,}0$, и первым результатом наблюдения $z_1 = 2{,}5$ (это значение представлено звездочкой на оси x). Обратите внимание, как предсказание $P(x_1)$ сглаживается относительно $P(x_0)$ под влиянием шума перехода. Также обратите внимание, что среднее апостериорной вероятности $P(x_1 \mid z_1)$ находится немного левее относительно результата наблюдения z_1, поскольку это среднее представляет собой взвешенное среднее от предсказания и наблюдения

14.4.3. Общий случай

Приведенные выше выводы иллюстрируют ключевое свойство гауссовых распределений, которое обеспечивает функционирование методов фильтрации Калмана: экспонента находится в квадратичной форме. Это свойство относится не только к рассмотренному одномерному случаю; полное многомерное гауссово распределение имеет следующую форму:

$$\mathcal{N}(\mathbf{x}; \mu, \Sigma) = \alpha e^{-\frac{1}{2}\left((\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)\right)}.$$

Произведение термов в экспоненте указывает, что экспонента также является квадратичной функцией от случайных переменных x_i в векторе $\mathbf{x}$. Следовательно, фильтрация сохраняет гауссов характер распределения вероятностей состояний.

Прежде всего определим общую временную модель, применяемую в калмановской фильтрации. И модель перехода, и модель восприятия требуют применения

линейного преобразования с дополнительным гауссовым шумом. Таким образом, получаем следующее:

$$\begin{aligned} P(\mathbf{x}_{t+1} \mid \mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t+1}; \mathbf{F}\mathbf{x}_t, \Sigma_x) \\ P(\mathbf{z}_t \mid \mathbf{x}_t) &= \mathcal{N}(\mathbf{z}_t; \mathbf{H}\mathbf{x}_t, \Sigma_z), \end{aligned} \tag{14.21}$$

где $\mathbf{F}$ и Σ_x — матрицы, описывающие линейную модель перехода и ковариацию шума перехода, а $\mathbf{H}$ и Σ_z — соответствующие матрицы для модели восприятия. Теперь уравнения обновления для среднего и ковариации в их полном, ужасающе сложном виде приобретают следующий вид:

$$\begin{aligned} \mu_{t+1} &= \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t) \\ \Sigma_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x), \end{aligned} \tag{14.22}$$

где $\mathbf{K}_{t+1} = (\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top + \Sigma_z)^{-1}$ называется ▶**калмановской матрицей усиления**. Хотите — верьте, хотите — нет, но эти уравнения имеют определенный интуитивный смысл. Например, рассмотрим обновление для оценки значения среднего μ для некоторого состояния. Терм $\mathbf{F}\mu_t$ представляет *прогнозируемое* состояние в момент времени $t+1$, поэтому $\mathbf{H}\mathbf{F}\mu_t$ является *прогнозируемым* результатом наблюдения. Следовательно, терм $\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t$ представляет ошибку в прогнозируемых результатах наблюдений. Это значение умножается на $\mathbf{K}_{t+1}$ для корректировки прогнозируемого состояния, значит, $\mathbf{K}_{t+1}$ представляет собой меру того, *насколько важными следует считать новые результаты наблюдения* применительно к предсказанию. Как и в уравнениях (14.20), здесь соблюдается то же свойство: обновление дисперсии не зависит от результатов наблюдений. Поэтому последовательность значений Σ_t и $\mathbf{K}_t$ можно вычислить в автономном режиме и фактический объем вычислений, требуемых во время оперативного слежения, становится достаточно скромным.

Для иллюстрации этих уравнений в действии применим их к задаче слежения за объектом, движущимся на плоскости X–Y. Переменными состояния являются $\mathbf{X} = (X, Y, \dot{X}, \dot{Y})^\top$, поэтому $\mathbf{F}$, Σ_x, $\mathbf{H}$ и Σ_z представляют собой матрицы размером 4×4. На рис. 14.11, *а* показаны истинная траектория, ряд зашумленных результатов наблюдения и траектория, оцениваемая с помощью калмановской фильтрации, вместе с ковариациями, указанными с помощью контуров единичного среднеквадратичного отклонения. Процесс фильтрации позволяет весьма успешно следить за фактическим перемещением, к тому же, как и предполагалось, дисперсия быстро достигает фиксированной точки.

С помощью линейных гауссовых моделей можно вывести не только уравнения фильтрации, но и уравнения *сглаживания*. Результаты сглаживания показаны на рис. 14.11, *б*. Обратите внимание, как резко сокращается дисперсия в оценке позиции, за исключением концов траектории (объясните, почему), и насколько более гладкой становится оцениваемая траектория.

Рис. 14.11. а) Результаты калмановской фильтрации для объекта, движущегося по плоскости *X*–*Y*: истинная траектория (слева направо), ряд зашумленных наблюдений и траектория, оцениваемая с помощью фильтра Калмана. Дисперсия в оценке позиции показана с помощью овалов. **б)** Результаты калмановского сглаживания для той же последовательности результатов наблюдения

14.4.4. Применение калмановской фильтрации

Фильтр Калмана и его модификации применяются во множестве различных приложений. Одним из "классических" приложений является приложение для слежения за самолетами и ракетами с помощью радаров. К такому же типу относятся приложения, в которых осуществляется акустическое слежение за подводными лодками и наземными транспортными средствами, а также визуальное слежение за транспортными средствами и людьми. К немного более узким областям применения относится использование фильтров Калмана для реконструкции траектории частиц по фотографиям, сделанным в пузырьковой камере, и океанских течений по данным измерений, выполненных на поверхности океана со спутников. Но спектр возможных приложений выходит далеко за пределы простого отслеживания движений — к ним относится любая система, характеризующаяся непрерывными переменными состояния и зашумленными результатами измерений. К их числу относятся целлюлозные фабрики, химические установки, ядерные реакторы, экосистемы растений и национальные экономики.

Тот факт, что калмановскую фильтрацию можно применить к какой-то системе, еще не означает, что результаты этого применения будут действительными или полезными. Используемые в этом случае допущения (что модель перехода и модель

восприятия относятся к типу линейных гауссовых) на самом деле являются очень строгими. В ►**расширенном фильтре Калмана** (*Extended Kalman Filter* — ►**EKF**) предпринимается попытка преодолеть нелинейности моделируемой системы. Система является ►**нелинейной**, если ее модель перехода нельзя описать с помощью матричного умножения векторов состояния, как в уравнении (14.21). Фильтр EKF действует посредством моделирования системы как *локально* линейной в области $\mathbf{x}_t$, т.е. в такой области, где $\mathbf{x}_t = \mu_t$, среднему текущему распределению вероятностей состояния. Такой подход хорошо действует применительно к гладким системам с устойчивым поведением и позволяет программе слежения сопровождать и обновлять такое гауссово распределение вероятностей состояния, которое будет приемлемой аппроксимацией истинной апостериорной вероятности. Подробный пример использования этого подхода приводится в главе 26.

А что понимается под системой, которая является "не гладкой" или поведение которой "неустойчиво"? Формально под этим подразумевается, что отклик системы в области, "близкой" (согласно ковариации Σ_t) к текущему среднему μ_t, проявляет существенную нелинейность. Чтобы понять суть этого описания неформально, рассмотрим пример слежения за птицей, которая летит через джунгли. Иногда создается впечатление, что птица на высокой скорости направляется прямо на ствол дерева. Фильтр Калмана (обычный или расширенный) позволяет получить только гауссово предсказание местонахождения птицы, при том что среднее соответствующего гауссова распределения будет находиться напротив центра ствола, как показано на рис. 14.12, *а*. С другой стороны, более разумная модель полета птицы должна предсказывать ее действия по уклонению от удара о ствол за счет поворота в ту или иную сторону, как показано на рис. 14.8, *б*. Такая модель является существенно нелинейной, поскольку птица принимает решение об уклонении от удара внезапно, в зависимости от того, где именно она находится по отношению к стволу.

Очевидно, что для работы с примерами, подобными этому, требуется более выразительный язык представления поведения моделируемой системы. В сообществе специалистов по теории управления, для которых в таких задачах, как маневры самолета по предотвращению столкновения, возникают аналогичные сложности, стандартное решение заключается в использовании ►**переключательных фильтров Калмана**. При таком подходе предусмотрена одновременная эксплуатация нескольких фильтров Калмана, в каждом из которых используются разные модели систем, например в одном из них моделируется прямой полет, в другом — резкий поворот налево, а в третьем — резкий поворот направо. При этом используется взвешенная сумма предсказаний, где вес зависит от того, насколько точно данные каждого фильтра совпадают с текущими данными. Как показано в следующем разделе, такой подход представляет собой частный случай общей модели динамической байесовской сети, созданной путем введения дискретной переменной состояния "маневра" в сеть, показанную на рис. 14.9. Переключательные фильтры Калмана рассматриваются дополнительно в упражнении 14.12.

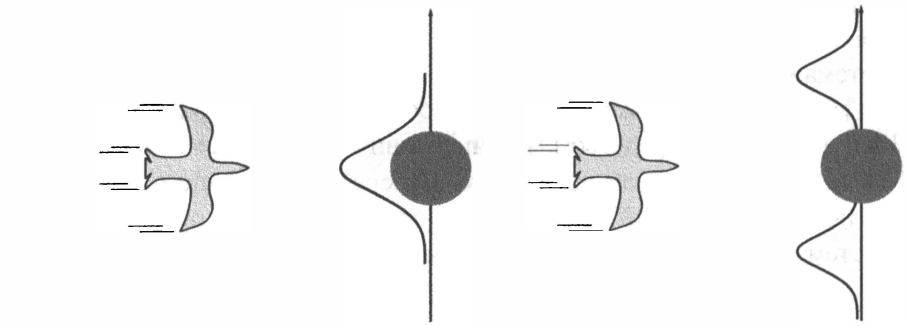


Рис. 14.12. Птица, летящая прямо на ствол дерева (вид сверху). **а)** Фильтр Калмана предсказывает местонахождение птицы с использованием единственного гауссова распределения, центр которого находится напротив препятствия. **б)** Более реалистичная модель допускает выполнение птицей действий во избежание столкновения, предсказывая, что птица облетит препятствие с той или иной стороны

14.5. Динамические байесовские сети

В ►**динамических байесовских сетях**, или **DBN** (*Dynamic Bayesian Network*), стандартная семантика байесовских сетей расширяется так, чтобы обеспечить обработку временных вероятностных моделей такого типа, как описано в разделе 14.1. Выше уже рассматривались примеры DBN: сеть в задаче с зонтиком (см. рис. 14.2) и сеть фильтра Калмана (см. рис. 14.9). Вообще говоря, каждый временной срез динамической байесовской сети может иметь любое количество переменных состояния $\mathbf{X}_t$ и переменных свидетельства $\mathbf{E}_t$. Для упрощения мы будем предполагать, что эти переменные, связи между ними и их условные распределения точно копируются от среза к срезу и что сеть DBN представляет марковский процесс первого порядка, так что каждая переменная может иметь родительские переменные только в собственном временном срезе или в непосредственно предшествующем временном срезе. При таком подходе сеть DBN соответствует байесовской сети с неограниченным, бесконечным количеством переменных.

Должно быть понятно, что любая скрытая марковская модель может быть представлена в виде сети DBN с единственной переменной состояния и с единственной переменной свидетельства. Справедливо также утверждение, что каждая сеть DBN с дискретными переменными может быть представлена в виде модели HMM, как это пояснялось в разделе 14.3: можно скомбинировать все переменные состояния в сети DBN в одну переменную состояния, значениями которой являются все возможные кортежи значений отдельных переменных состояния. Тогда, если каждая модель HMM представляет собой сеть DBN, а каждая сеть DBN может быть

преобразована в модель HMM, то в чем состоит различие между ними? Это различие заключается в том, что ➔ *благодаря декомпозиции состояния сложной системы на составляющие его переменные сеть DBN позволяет воспользоваться преимуществами* разреженности *временной вероятностной модели.*

Чтобы пояснить, что это означает на практике, напомним, что в разделе 14.3 было показано, что представлению HMM для временного процесса с n дискретными переменными, каждая из которых может иметь до d значений, необходима матрица перехода размером $O(d^{2n})$. С другой стороны, представление в виде сети DBN имеет размер $O(nd^k)$, если k — максимальное число родителей любой переменной. Другими словами, представление в виде сети DBN требует объема памяти, линейно, а не экспоненциально пропорционального количеству переменных. Для робота из мира пылесоса с 42 возможными местоположениями, в которых может присутствовать мусор, количество требуемых вероятностей сокращается с 5×10^{29} до нескольких тысяч.

Выше уже объяснялось, что каждая модель с фильтром Калмана может быть представлена в виде сети DBN с непрерывными переменными и линейными гауссовыми распределениями условных вероятностей (см. рис. 14.9). Из обсуждений в конце предыдущего раздела должно быть очевидно, что *не каждая* сеть DBN может быть представлена с помощью модели с фильтром Калмана. В фильтре Калмана текущее распределение вероятностей состояния всегда представляет собой единственное многомерное гауссово распределение, т.е. распределение с единственным "максимумом", расположенным в определенном месте, тогда как сети DBN позволяют моделировать произвольные распределения.

Для многих реальных приложений такая гибкость является существенно важным аспектом. Рассмотрим, например, текущее местонахождение связки ключей некоторого лица. Она может находиться в его кармане, на ночном столике или на полке в прихожей, торчать в замочной скважине входной двери или быть запертой в автомобиле. Единственный максимум гауссова распределения, охватывающего распределения вероятностей нахождения связки ключей во всех упомянутых местах, присвоил бы значительную вероятность тому предположению, что ключи находятся где-то в промежуточной позиции, например висят прямо в воздухе в прихожей. Таким образом, аспекты реального мира, — такие, как целенаправленные агенты, препятствия и тупики — приводят к появлению "нелинейности" и по этой причине требуют использования сочетаний дискретных и непрерывных переменных с целью создания приемлемой модели.

14.5.1. Создание сетей DBN

Для построения сети DBN необходимо определить три вида информации: распределение априорных вероятностей по переменным состояния $\mathbf{P}(\mathbf{X}_0)$, модель перехода $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{X}_t)$ и модель восприятия $\mathbf{P}(\mathbf{E}t \mid \mathbf{X}t)$. Чтобы задать модель перехода и модель восприятия, дополнительно необходимо определить топологию связей

между последовательными срезами, а также между переменными состояния и свидетельства. Поскольку предполагается, что модели перехода и восприятия являются стационарными (одинаковыми для всех t), удобнее всего задать их для первого среза. Например, полная спецификация сети DBN для мира задачи с зонтиком может быть задана с помощью сети с тремя узлами, показанной на рис. 14.13, *а*. На основании этой спецификации при необходимости можно будет создать полную сеть DBN с неограниченным количеством временных срезов, полученных посредством копирования первого среза.

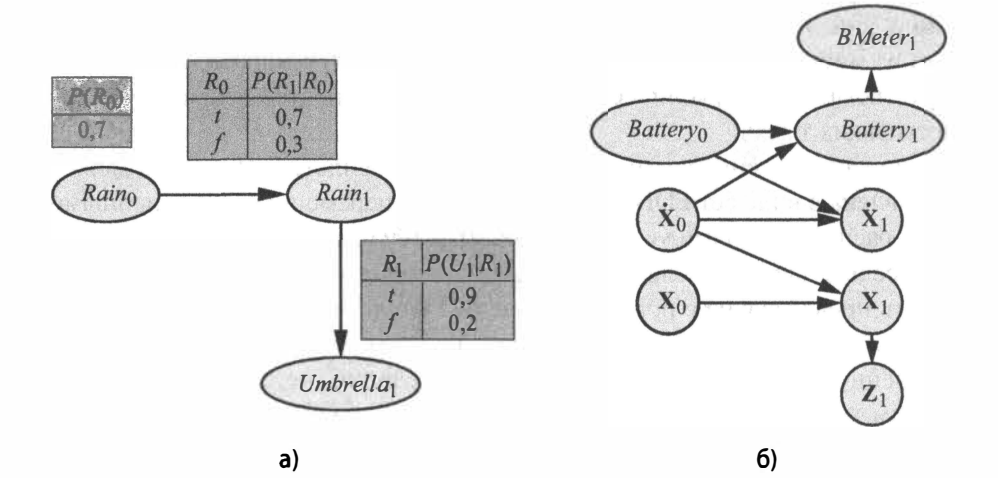


Рис. 14.13. а) Спецификация распределения априорных вероятностей модели перехода и модели восприятия для сети DBN задачи с зонтиком. Все последующие срезы являются копиями среза 1. **б)** Простая сеть DBN для моделирования движения робота на плоскости X–Y

Теперь давайте рассмотрим более интересный пример: наблюдение за роботом с питанием от аккумулятора, который движется на плоскости X–Y, — впервые речь о нем шла в разделе 14.1. Начнем с определения переменных состояния: это переменная положения на плоскости $\mathbf{X}_t = (X_t, Y_t)$ и переменная скорости $\dot{\mathbf{X}}_t = (\dot{X}_t, \dot{Y}_t)$. Предполагается, что для измерения координат положения используется определенный метод (возможно, фиксированная видеокамера или бортовая GPS-система), позволяющий получить результаты измерений $\mathbf{Z}_t$. Положение робота в следующем временном интервале зависит от текущего положения и скорости, как и при использовании стандартной модели с фильтром Калмана. Скорость в следующем временном интервале зависит от текущей скорости и уровня заряда аккумулятора. Введем переменную $Battery_t$ представляющую фактический уровень заряда аккумулятора, родительскими переменными которой являются предыдущий уровень

заряда аккумулятора и скорость, а также введем переменную $BMeter_t$, которая будет представлять показания измерения уровня заряда аккумулятора. В результате будет получена исходная модель, показанная на рис. 14.9, *б*.

Более глубокого анализа заслуживает характер модели восприятия для переменной $BMeter_t$. Для простоты предположим, что переменные $Battery_t$ и $BMeter_t$ могут принимать дискретные значения от 0 до 5. (В упражнении 14.19 предлагается связать эту дискретную модель с соответствующей непрерывной моделью.) Если этот измеритель всегда дает точные показания, то таблица условных вероятностей $\mathbf{P}(BMeter_t \mid Battery_t)$ должна содержать вероятности 1,0 в элементах, расположенных "вдоль диагонали", и вероятности 0,0 — во всех других элементах. Но в действительности в результаты измерения всегда проникает шум. Для непрерывных измерений может использоваться гауссово распределение с небольшой дисперсией.[7] Применительно к дискретным переменным, рассматриваемым в данном примере, гауссово распределение можно аппроксимировать с помощью распределения, в котором снижение вероятности ошибки соответствует реальной ситуации, поэтому вероятность крупной ошибки весьма мала. Далее мы будем использовать термин ▶ **гауссова модель ошибки** применительно и к непрерывной, и к дискретной версиям.

Любой специалист, имеющий практический опыт работы в области робототехники, компьютеризированного управления процессами или в другой области применения различных форм автоматического сбора информации, охотно подтвердит тот факт, что небольшие количества измерительного шума часто не представляют серьезной проблемы. Однако реальные датчики могут *выходить из строя*, и когда это происходит, они далеко не всегда посылают такой сигнал: "Кстати, теперь данные, которые я буду вам отправлять, будут взяты с потолка". Вместо этого они просто отправляют бессмыслицу. Отказом простейшего типа является ▶ **временный отказ**, при котором датчик время от времени передает бессмысленные данные. Например, может оказаться, что датчик уровня заряда аккумулятора имеет печальное свойство давать нулевое показание каждый раз, когда робот ударяется о препятствие, даже если аккумулятор полностью заряжен.

Рассмотрим, что произойдет при возникновении временного отказа, если используется гауссова модель ошибок, не приспособленная к таким отказам. Например, предположим, что робот спокойно стоит и наблюдает 20 последовательных показаний датчика заряда аккумулятора, равных 5, а затем этот датчик допускает временный сбой и передает показание $BMeter_{21} = 0$. К какому выводу относительно значения переменной $Battery_{21}$ приведет нас простая гауссова модель ошибки? Согласно правилу Байеса, ответ на этот вопрос зависит и от модели восприятия

[7] Строго говоря, гауссово распределение не совсем подходит, поскольку в нем ненулевая вероятность присваивается большим отрицательным уровням зарядки аккумулятора. Иногда для переменных, область определения которых ограничена, лучше подходит **бета-распределение**.

$\mathbf{P}(BMeter_{21} = 0 \mid Battery_{21})$, и от предсказания $\mathbf{P}(Battery_{21} \mid BMeter_{1:20})$. Если вероятность большой ошибки датчика является значительно менее правдоподобной, чем вероятность перехода в состояние $Battery_{21} = 0$, даже если последнее весьма неправдоподобно, то в распределении апостериорных вероятностей будет присвоена высокая вероятность ситуации, что аккумулятор полностью разряжен.

Если же в момент времени $t = 22$ будет получено еще одно показание о нулевом заряде, то такой вывод станет почти полностью безоговорочным. А после того как этот временный отказ исчезнет и показания вернутся к 5, начиная с момента $t = 23$ и далее во все последующие моменты, то оценка уровня заряда аккумулятора, как по волшебству, быстро вернется к 5. (Это не означает, что алгоритм полагает, что аккумулятор был магически перезаряжен; это физически просто невозможно. Вместо этого алгоритм теперь полагает, что аккумулятор никогда не разряжался, и крайне маловероятную гипотезу о том, что у измерителя уровня заряда аккумулятора произошли две последовательные огромные ошибки, теперь следует считать правильным объяснением.) Такой ход событий проиллюстрирован на верхней кривой, приведенной на рис. 14.14, *а*, которая представляет изменение во времени математического ожидания M (см. приложение А) значения переменной $Battery_t$ при использовании дискретной гауссовой модели ошибки.

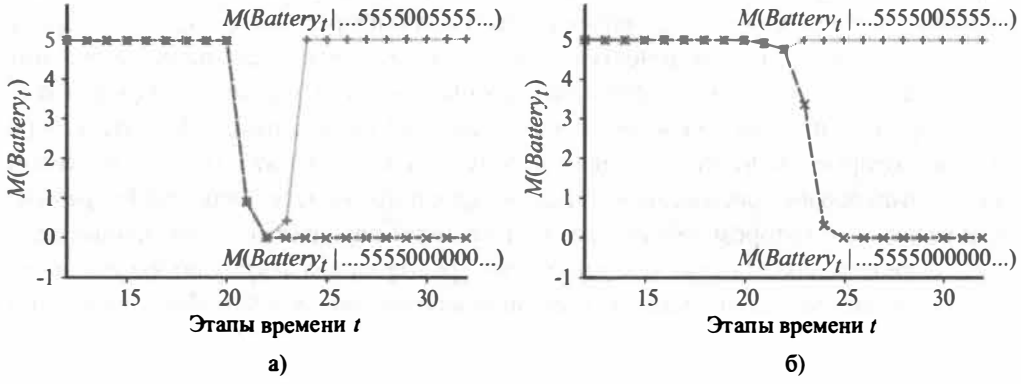


Рис. 14.14. а) Верхняя кривая представляет траекторию ожидаемого значения переменной $Battery_t$ для последовательности наблюдений, состоящей из значений 5 во всех случаях, кроме равных нулю показаний в моменты времени $t = 21$ и $t = 22$, когда используется простая гауссова модель ошибки. Нижняя кривая — это траектория, при которой результаты наблюдения остаются на уровне 0, начиная с момента времени $t = 21$. **б)** Тот же эксперимент, но выполненный с использованием модели временного отказа. Обратите внимание, что временный отказ преодолевается успешно, а постоянный приводит к излишне пессимистической оценке уровня заряда аккумулятора

Несмотря на последующее восстановление правильных показаний, есть такой момент времени ($t = 22$), в котором робот принял сообщение о полном разряде аккумулятора, а в такой ситуации он, в принципе, должен выдать сигнал тревоги и отключиться. Таким образом, чрезмерно упрощенная модель восприятия, к сожалению, завела робота в тупик. Мораль этой истории состоит в следующем: ➨ *для того чтобы система правильно обрабатывала отказы датчика, модель восприятия должна допускать вероятность его отказа.*

В модели отказа простейшего вида для датчика допускается определенная вероятность того, что он может выдать полностью неправильное значение, независимо от истинного состояния мира. Например, если измеритель заряда аккумулятора отказывает, выдавая значение 0, то можно принять, что

$$P(BMeter_t = 0 \mid Battery_t = 5) = 0{,}03,$$

что, очевидно, значительно больше, чем вероятность, присваиваемая при использовании простой гауссовой модели ошибки. Назовем соответствующую модель ►**моделью временного отказа**. Как это может помочь, если придется столкнуться с показанием датчика, равным 0? При условии, что *прогнозируемая* вероятность полного разряда аккумулятора, согласно полученным до текущего момента времени показаниям, гораздо меньше 0,03, наилучшим объяснением причины наблюдения $BMeter_{21} = 0$ будет то, что произошел временный отказ датчика. Интуитивно понятно, что такой подход позволяет рассматривать уверенность в истинности данных об уровне заряда аккумулятора как имеющую определенную долю "инерции", которая обеспечит преодоление временных сбоев в показаниях датчика. Верхняя кривая на рис. 14.10, *б* показывает, что модель временного отказа позволяет преодолевать временные отказы без катастрофического изменения в представлениях об истинности данных.

На этом и закончим обсуждение временных отказов. А что будет, если отказ датчика окажется постоянным? К сожалению, отказы такого рода встречаются слишком часто. Если датчик возвратит 20 показаний со значением 5, за которыми последует 20 показаний со значением 0, то применение модели временного отказа датчика, описанной в предыдущем абзаце, приведет к тому, что робот постепенно все же придет к выводу, что его аккумулятор разряжен, тогда как в действительности мог произойти отказ датчика. Нижняя кривая, приведенная на рис. 14.10, *б*, показывает "траекторию" изменения уверенности в истинности показаний датчика для этого случая. Ко времени $t = 25$ (после получения пяти нулевых показаний датчика) робот все же приходит к выводу, что его аккумулятор разряжен. Безусловно, было бы предпочтительнее, чтобы робот приобрел уверенность в том, что неисправен измеритель уровня заряда его аккумулятора, — если это действительно более вероятное событие.

Неудивительно, что для учета постоянных отказов требуется ►**модель постоянного отказа**, которая описывает, как датчик ведет себя при нормальных условиях и после отказа. Для этого необходимо дополнить скрытое состояние системы дополнительной переменной, скажем, *BMBroken*, которая описывает состояние

измерителя уровня заряда аккумулятора. Постоянство отказа может быть промоделировано дугой, связывающей переменные $BMBroken_0$ и $BMBroken_1$. Такая ▶**дуга постоянства** имеет таблицу условных вероятностей, которая задает на каждом временном интервале малую вероятность отказа, допустим 0,001, но определяет, что после выхода из строя датчик остается неисправным. Когда датчик исправен, то модель восприятия для переменной *BMeter* идентична модели временного отказа, а после того как датчик выходит из строя, эта модель указывает, что значение *BMeter* всегда будет равно 0, независимо от фактического уровня заряда аккумулятора.

Модель постоянного отказа для датчика уровня заряда аккумулятора показана на рис. 14.15, *а*. Показатели ее работы при двух последовательностях данных (временный сбой и постоянный отказ) приведены на рис. 14.15, *б*. В отношении этих кривых необходимо сделать несколько замечаний. Во-первых, в случае временного сбоя вероятность того, что датчик вышел из строя, существенно повышается после второго показания со значением 0, но немедленно падает вновь до нуля после получения нового результата наблюдения 5. Во-вторых, в случае постоянного отказа вероятность того, что датчик неисправен, быстро повышается почти до 1 и остается на этом уровне. И наконец, как только становится известно, что датчик уровня заряда аккумулятора вышел из строя, робот в дальнейшем может руководствоваться лишь предположением, что его аккумулятор разряжается с "обычной" скоростью, — на рисунке это представлено постепенно снижающимися ожидаемыми значениями $M(Battery_t...)$.

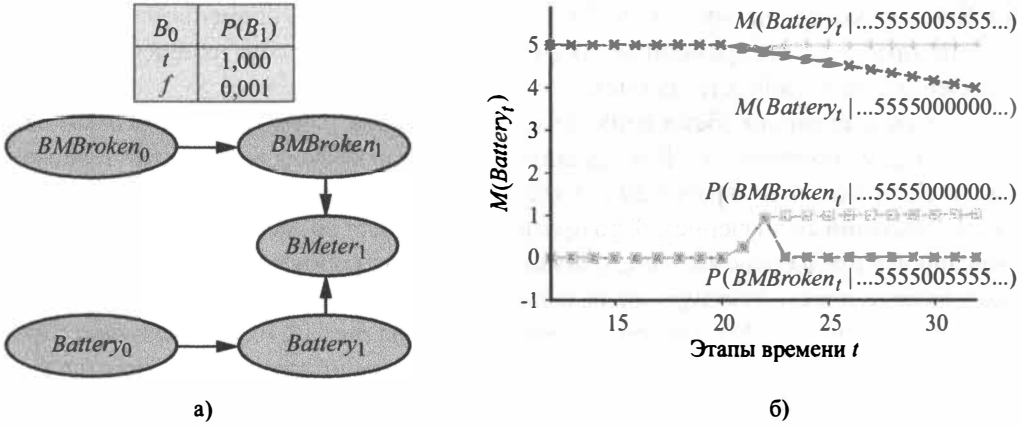


Рис. 14.15. а) Фрагмент сети DBN, представляющий переменные состояния датчика уровня заряда аккумулятора, необходимые для моделирования ситуации постоянного отказа этого датчика. **б)** Верхние кривые — траектории ожидаемого значения переменной $Battery_t$ для последовательностей наблюдений, характерных для "временного отказа" и "постоянного отказа". Нижние кривые — траектории вероятностей для переменной *BMBroken* для двух указанных последовательностей наблюдений

В приведенном описании мы лишь слегка коснулись поверхности проблемы представления сложных процессов. Применяемое на практике разнообразие моделей перехода буквально огромно и охватывает такие разные направления, как моделирование эндокринной системы человека и моделирование потока множества автомобилей, движущихся по скоростному шоссе. Создание моделей восприятия также является обширной самостоятельной областью. Например, динамические байесовские сети позволяют моделировать даже такие тонкие явления, как дрейф показаний датчика, внезапная раскалибровка или влияние на показания прибора внешних условий (таких, как погода).

14.5.2. Точный вероятностный вывод в сетях DBN

Кратко рассмотрев некоторые идеи, касающиеся представления сложных процессов в виде сетей DBN, перейдем к вопросу вероятностного вывода. В определенном смысле на этот вопрос уже был получен ответ: динамические байесовские сети прежде всего *являются* байесовскими сетями, и нам уже известны алгоритмы выполнения вероятностного вывода в байесовских сетях. При наличии последовательности наблюдений можно построить представление сети DBN в виде полной байесовской сети путем повторения временных срезов до тех пор, пока сеть не станет достаточно большой, чтобы в ней можно было учесть все наблюдения, как показано на рис. 14.16. Такой метод называется **развертыванием**. (С формальной точки зрения сеть DBN эквивалентна полубесконечной сети, полученной путем развертывания в одну сторону до бесконечности. Но временные срезы, вводимые за пределами последнего наблюдения, не оказывают влияния на вероятностные выводы в пределах периода наблюдения и поэтому могут быть исключены.) После того как сеть DBN развернута, в ней может использоваться любой из алгоритмов вероятностного вывода (алгоритм с устранением переменной, методы кластеризации и т.д.), описанных в главе 13.

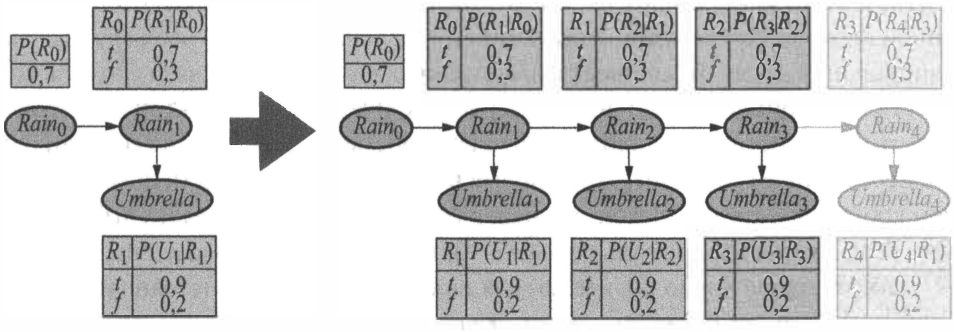


Рис. 14.16. Развертывание динамической байесовской сети: для размещения результатов последовательности наблюдений $Umbrella_{1:3}$ временные срезы дублируются. Последующие срезы не влияют на вероятностные выводы в пределах периода наблюдения

К сожалению, простое, непродуманное применение развертывания не всегда будет достаточно эффективным. Так, если требуется выполнить фильтрацию или сглаживание с использованием длинной последовательности наблюдений $\mathbf{e}_{1:t}$, то для представления развернутой сети потребуется пространство $O(t)$, и оно будет неограниченно возрастать по мере добавления новых результатов наблюдений. Более того, если просто заново запускать алгоритм вероятностного вывода после каждого добавления новых результатов наблюдения, то затраты времени на вероятностный вывод при каждом обновлении также будут расти пропорционально $O(t)$.

Еще раз обратившись к разделу 14.2.1, можно заметить, что при фильтрации достичь постоянных затрат времени и пространства в расчете на каждое обновление можно, если выполнять вычисления в рекурсивной форме. По сути, обновление результатов фильтрации в уравнении (14.5) осуществляется по принципу *исключения путем суммирования* переменных состояния, относящихся к предыдущему временному этапу, что позволяет получить распределение для нового временного этапа. Исключение переменных путем суммирования — это именно то, что выполняет алгоритм **устранения переменной** (см. рис. 13.13), и, как оказалось, применение процедуры устранения переменной к переменным во временном порядке точно моделирует функционирование рекурсивного обновления результатов фильтрации в уравнении (14.5). В модифицированном алгоритме предусмотрено одновременное хранение в памяти не более двух временных срезов: начиная со среза 0, добавляем срез 1, затем исключаем путем суммирования срез 0, после этого добавляем срез 2, на следующем этапе исключаем путем суммирования срез 1 и т.д. Такая организация вычислений позволяет добиться постоянных затрат пространства и времени в расчете на каждое обновление результатов фильтрации. (Такой же производительности можно достичь путем введения соответствующих модификаций в алгоритм кластеризации.) В упражнении 14.20 предлагается проверить это утверждение на примере сети для задачи с зонтиком.

До сих пор речь шла только о преимуществах рекурсивного подхода, но он имеет и недостатки: как оказалось, "постоянные" значения временной и пространственной сложности для каждой операции обновления почти во всех случаях экспоненциально зависят от количества переменных состояния. В связи с этим в ходе осуществления процесса устранения переменной количество факторов возрастает так, что в их состав начинают входить все переменные состояния (или, точнее, все те переменные состояния, которые имеют родительские переменные в предыдущем временном срезе). Максимальный размер фактора составляет $O(d^{n+k})$, а стоимость обновления измеряется как $O(nd^{n+k})$, где d — размер области переменных, а k — максимальное число родителей для любой переменной состояния.

Безусловно, такие значения намного меньше по сравнению со стоимостью обновления для скрытой марковской модели, пропорциональной $O(d^{2n})$, но они все еще неприемлемы при наличии большого количества переменных. Этот обескураживающий факт означает, что ➔ *даже несмотря на то, что сети DBN могут*

использоваться для представления очень сложных временных процессов с многочисленными переменными с разрозненными связями между ними, мы не можем эффективно и точно рассуждать об этих процессах. Сама модель DBN, которая представляет априорное совместное распределение по всем переменным, может быть разложена на составляющие ее таблицы условных вероятностей, но обусловленное последовательностью наблюдений апостериорное совместное распределение (т.е. прямое сообщение), как правило, *не поддается* разбиению на факторы. В общем случае проблема неразрешима, поэтому мы вынуждены обращаться к приближенным методам.

14.5.3. Приближенный вероятностный вывод в сетях DBN

В разделе 13.4 были описаны два алгоритма аппроксимации — взвешивание по правдоподобию (см. рис. 13.18) и метод Монте-Карло на основе цепи Маркова (алгоритм MCMC; см. рис. 13.20). Из этих двух алгоритмов проще всего к контексту DBN адаптируется первый алгоритм. (Алгоритм фильтрации на базе MCMC кратко описан в разделе "Библиографические и исторические заметки" в конце этой главы.) Однако, как будет показано ниже, чтобы получить практически применимый метод, в стандартный алгоритм взвешивания по правдоподобию необходимо внести несколько усовершенствований.

Напомним, что алгоритм взвешивания по правдоподобию работает по принципу осуществления в топологическом порядке выборок в узлах сети, не являющихся узлами свидетельства, и взвешивания каждой выборки с учетом правдоподобия того, что она соответствует наблюдаемым переменным свидетельства. Как и в случае точных алгоритмов, алгоритм взвешивания по правдоподобию можно применить непосредственно к развернутой сети DBN, однако по мере увеличения длины последовательностей наблюдений это приведет к возникновению тех же сложностей, связанных с увеличением требований ко времени и пространству в расчете на каждое обновление. Проблема состоит в том, что в стандартном алгоритме каждая выборка обрабатывается последовательно, по всей сети.

Вместо этого можно просто пропустить через сеть DBN все N выборок вместе, проходя каждый раз через один временной срез. Этот модифицированный алгоритм имеет такую же общую форму, как и другие алгоритмы фильтрации, но в нем в качестве прямого сообщения используется множество из N выборок. Поэтому первое ключевое усовершенствование состоит в ➡ *использовании самих выборок в качестве приближенного представления распределения вероятностей текущего состояния*. Такой подход соответствует требованию обеспечения "постоянных" затрат времени в расчете на каждое обновление, хотя само это постоянное значение зависит от количества выборок, необходимых для достижения приемлемой аппроксимации. Кроме того, нет необходимости развертывать сеть DBN, поскольку в памяти требуется держать только текущий временной срез и следующий временной срез. Такой подход называют ▶ **последовательной выборкой по значимости**, или **SIS** (*Sequential Importance Sampling*).

В описании метода взвешивания по правдоподобию, приведенному в главе 13, было указано, что точность алгоритма снижается, если переменные свидетельства занимают "последние места" в упорядочении переменных, по которым осуществляется выборка, поскольку в таком случае выборки формируются, не испытывая какого-либо влияния со стороны свидетельства.

Взглянув на типичную структуру сети DBN — скажем, сети DBN для задачи с зонтиком, представленную на рис. 14.16, — можно убедиться, что в действительности выборка более ранних переменных состояния будет выполняться без учета полученных в дальнейшем свидетельств. На самом деле тщательный анализ показывает, что у *любой* из переменных состояния среди ее предков нет *ни одной* переменной свидетельства! Поэтому, хотя вес каждой выборки зависит от свидетельства, фактически сформированное множество выборок будет *полностью независимым* от него. Например, даже если директор всю неделю каждый день приходит с зонтиком, процесс формирования выборки по-прежнему может полагать, что солнечные дни не кончаются.

С точки зрения практики это означает, что доля выборок, остающихся достаточно близкими к фактическому ряду событий (и, следовательно, имеющих достаточно значимый вес), падает экспоненциально с увеличением значения t, т.е. длины последовательности наблюдений. Иными словами, чтобы поддерживать заданный уровень точности, необходимо увеличивать количество выборок экспоненциально в зависимости от t. Учитывая то, что алгоритм фильтрации, работающий в режиме реального времени, может использовать лишь ограниченное количество выборок, на практике после небольшого количества этапов обновления ошибка становится весьма значительной. На рис. 14.19 в конце этого раздела наглядно демонстрируется этот эффект для метода SIS при его применении к задаче локализации в клеточном мире, обсуждавшейся в разделе 14.3: даже при 100 000 выборок аппроксимация по методу SIS терпит полную неудачу после примерно 20 этапов.

Очевидно, что требуется найти лучшее решение. Второе важное нововведение состоит в том, что ➨ *множество выборок следует формировать преимущественно в областях пространства состояний, характеризующихся высокой вероятностью*. Такой подход можно реализовать, отбрасывая все выборки, которые, согласно наблюдениям, имеют очень малый вес, и увеличивая количество выборок, имеющих большой вес. В результате популяция выборок будет оставаться достаточно близкой к реальности. Если выборки рассматривать как информационные ресурсы для моделирования распределения апостериорных вероятностей, то имеет смысл формировать больше выборок в тех областях пространства состояний, где апостериорная вероятность выше.

Для решения именно этой задачи предназначено семейство алгоритмов, называемых алгоритмами ►**фильтрации частиц**. (Другим, более ранним, было название **последовательная выборка по важности с перевыборкой**, но по некоторым причинам оно не прижилось.) Метод фильтрации частиц действует следующим образом: сначала формируется популяция из N выборок, сформированных на

основании распределения априорных вероятностей $\mathbf{P}(\mathbf{X}_0)$, а затем для каждого временного этапа повторяется цикл обновления, как описано ниже.

1. Каждая выборка распространяется в прямом направлении путем формирования выборки значения переменной следующего состояния $\mathbf{x}_{t+1}$, при этом в качестве выборки берется текущее значение $\mathbf{x}_t$ и используется модель перехода $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)$.
2. Каждая выборка взвешивается по правдоподобию, назначенному новому свидетельству, $P(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1})$.
3. Эта популяция выборок подвергается *перевыборке* для формирования новой популяции из N выборок. Каждая новая выборка берется из текущей популяции; вероятность того, что будет выбрана конкретная выборка, пропорциональна ее весу. Новые выборки рассматриваются как не имеющие веса.

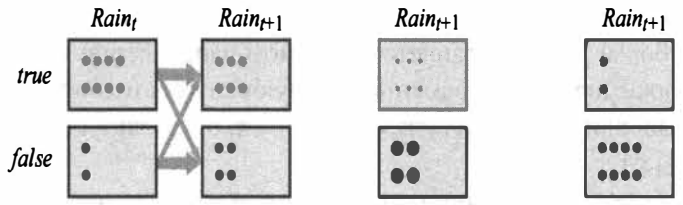
Этот алгоритм во всех деталях представлен на рис. 14.17, а результаты его применения к сети DBN для задачи с зонтиком показаны на рис. 14.18.

---

function PARTICLE-FILTERING(**e**, N, *dbn*) **returns** множество выборок для следующего временного этапа
  inputs: **e**, новое полученное свидетельство
    N, количество выборок, которые должен сформировать алгоритм
    dbn, сеть DBN, заданная распределением априорных вероятностей $\mathbf{P}(\mathbf{X}_0)$, моделью перехода $\mathbf{P}(\mathbf{X}_1 \mid \mathbf{X}_0)$ и моделью восприятия $\mathbf{P}(\mathbf{E}_1 \mid \mathbf{X}_1)$
  persistent: S, вектор выборок размера N, первоначально формируемый из $\mathbf{P}(\mathbf{X}0)$
  local variables: W, вектор весов размера N

  for i = 1 to N **do**
    $S[i] \leftarrow$ выборка из $\mathbf{P}(\mathbf{X}_1 \mid \mathbf{X}_0 = S[i])$ // *этап 1*
    $W[i] \leftarrow \mathbf{P}(\mathbf{e} \mid \mathbf{X}_1 = S[i])$ // *этап 2*
  $S \leftarrow$ WEIGHTED-SAMPLE-WITH-REPLACEMENT(N, S, W) // *этап 3*
  return S

Рис. 14.17. Алгоритм фильтрации частиц, реализованный как рекурсивная операция обновления данных о состоянии (множество выборок). Каждая операция формирования выборок включает формирование выборки значений соответствующих переменных временного среза в топологическом порядке, выполняемое во многом так же, как и в процедуре PRIOR-SAMPLE. Операция WEIGHTED-SAMPLE-WITH-REPLACEMENT может быть реализована так, чтобы она выполнялась за ожидаемое время $O(N)$. На рисунке номера этапов соответствуют их описанию в тексте



а) Распространение б) Взвешивание в) Повторная выборка

Рис. 14.18. Цикл обновления алгоритма фильтрации частиц применительно к сети DBN для задачи с зонтиком при $N = 10$; показаны популяции выборок в каждом состоянии. **а)** В момент времени t 8 выборок указывают *rain*, а 2 выборки — $\neg$*rain*. Каждая из них распространяется в прямом направлении путем формирования выборок в следующем состоянии через модель перехода. В момент времени $t + 1$ выясняется, что 6 выборок указывают *rain*, а 4 выборки — $\neg$*rain*. **б)** В момент времени $t + 1$ наблюдается $\neg$*umbrella*. Каждая выборка взвешивается с учетом ее правдоподобия применительно к этому наблюдению, что на рисунке указывается соответствующим размером кружков. **в)** Формируется новое множество из 10 выборок путем случайного выбора со взвешиванием из текущего множества. В результате получено 2 выборки, указывающие *rain*, и 8 выборок, указывающих $\neg$*rain*

Рассмотрев операции во время одного цикла обновления, можно показать, что этот алгоритм является согласованным — позволяет получить правильные значения вероятностей, если N стремится к бесконечности. Предполагается, что формирование популяции выборок начинается с использования правильного представления прямого сообщения, т.е. $\mathbf{f}_{1:t} = \mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ во время t. Поэтому, записав выражение $N(\mathbf{x}_t \mid \mathbf{e}_{1:t})$ для количества выборок, входящих в состояние $\mathbf{x}_t$ после обработки наблюдений $\mathbf{e}_{1:t}$, получаем следующее соотношение для больших значений N:

$$N(\mathbf{x}_t \mid \mathbf{e}_{1:t}) / N = P(\mathbf{x}_t \mid \mathbf{e}_{1:t}). \tag{14.23}$$

Теперь распространим каждую выборку в прямом направлении, осуществляя формирование выборок значений переменных состояния во время $t + 1$ с учетом для каждой выборки значений во время t. Количество выборок, достигающих состояния $\mathbf{x}_{t+1}$ из каждого состояния $\mathbf{x}_t$, является вероятностью перехода, умноженной на величину популяции $\mathbf{x}_t$, поэтому общее количество выборок, достигающих $\mathbf{x}_{t+1}$, будет равно

$$N(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t}) = \sum_{\mathbf{x}_t} P(\mathbf{x}_{t+1} \mid \mathbf{x}_t) N(\mathbf{x}_t \mid \mathbf{e}_{1:t}).$$

Далее выполним взвешивание каждой выборки по ее правдоподобию применительно к свидетельству во время $t + 1$. Любая выборка в состоянии $\mathbf{x}_{t+1}$ получает

вес $P(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1})$, следовательно, суммарный вес выборок, достигших состояния $\mathbf{x}_{t+1}$, будет

$$W(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t+1}) = P(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1}) N(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t}).$$

Теперь выполняется этап повторного формирования выборки. Поскольку каждая выборка тиражируется с вероятностью, пропорциональной ее весу, количество выборок в состоянии $\mathbf{x}_{t+1}$ после повторного формирования выборки пропорционально суммарному общему весу в состоянии $\mathbf{x}_{t+1}$ перед повторным формированием, будет таким:

$$\begin{aligned}
N(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t+1}) / N &= \alpha W(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t+1}) = \\
&= \alpha P(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1}) N(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t}) = \\
&= \alpha P(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{x}_{t+1} \mid \mathbf{x}_t) N(\mathbf{x}_t \mid \mathbf{e}_{1:t}) = \\
&= \alpha N P(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{x}_{t+1} \mid \mathbf{x}_t) P(\mathbf{x}_t \mid \mathbf{e}_{1:t}) = \qquad \text{(согласно 14.23)} \\
&= \alpha' P(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{x}_{t+1} \mid \mathbf{x}_t) P(\mathbf{x}_t \mid \mathbf{e}_{1:t}) = \\
&= P(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t+1}) . \qquad \text{(согласно 14.5)}
\end{aligned}$$

Поэтому популяция выборок после одного цикла обновления правильно представляет прямое сообщение во время $t = 1$.

Следовательно, алгоритм фильтрации частиц является *согласованным*, но является ли он *эффективным*? Для многих практических примеров ответ на этот вопрос будет, по-видимому, положительным: фильтрация частиц позволяет поддерживать хорошую аппроксимацию истинных апостериорных вероятностей с использованием постоянного количества выборок. На рис. 14.19 показано, что фильтрация частиц хорошо работает для задачи локализации в клеточном мире уже лишь при тысяче выборок. Она работает и в отношении реальных задач: этот алгоритм лежит в основе тысяч приложений, используемых в науке и технике. (Некоторые ссылки даны в конце главы, в разделе "Библиографические и исторические заметки".) Он справляется с обработкой комбинации дискретных и непрерывных переменных, а также нелинейных и негауссовых моделей для непрерывных переменных. При некоторых допущениях — в частности, о том, что вероятности в модели перехода и модели восприятия не имеют значений 0 и 1 — также становится возможным доказать, что аппроксимация с высокой вероятностью обеспечивает ограниченную ошибку, как это показано на рисунке.

Однако у алгоритма фильтрации частиц есть и недостатки. Посмотрим, как он будет выполняться в случае задачи о локализации в мире пылесоса при добавлении информации о мусоре. Из раздела 14.3.2 вспомним, что это увеличивает размер пространства состояний задачи до 2^{42}, а это делает точный вероятностный

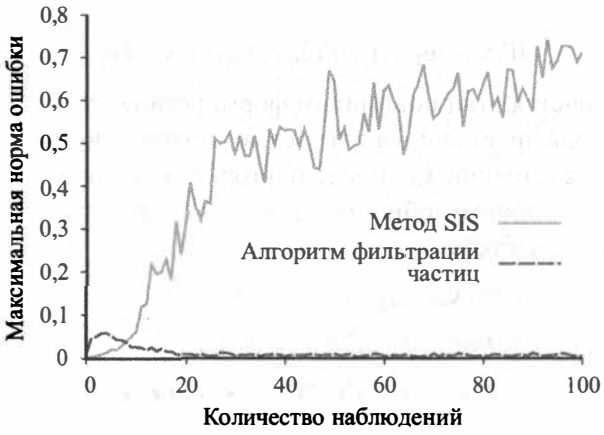


Рис. 14.19. Оценка максимальной нормы ошибки для задачи локализации в клеточном мире (в сравнении с точным вероятностным выводом) для алгоритма взвешивания по правдоподобию (метод последовательной выборки по значимости — SIS) при 100 000 выборок и для алгоритма фильтрации частиц при 1000 выборок. Данные усреднены по 50 прогонам

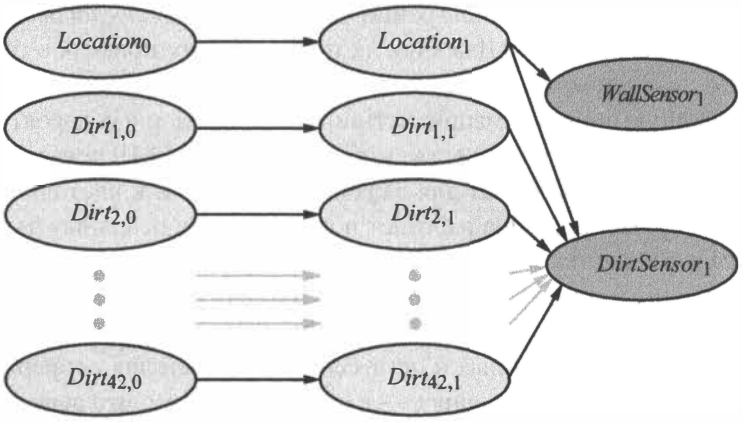


Рис. 14.20. Динамическая байесовская сеть для задачи локализации и одновременного создания карты стохастического клеточного мира пылесоса с наличием мусора. Мусор в квадратах сохраняется с вероятностью p, а в чистых квадратах может появиться с вероятностью $1-p$. Локальный датчик наличия мусора с вероятностью 0,9 дает правильные показания для того квадрата, в котором робот находится в данный момент

HMM-вывод неосуществимым. В этой задаче робот должен бродить по миру и создать карту мира с указанием, где находится мусор. (Это простой пример **задачи одновременной локализации и отображения**, или **SLAM** (*simultaneous localization and mapping*), которая будет подробно обсуждаться в главе 26.) Пусть переменная $Dirt_{i,t}$ определяет наличие мусора в квадрате i в момент времени t и пусть переменная $DirtSensor_t$ будет иметь значение *true* тогда и только тогда, когда робот обнаруживает мусор в момент времени t. Будем полагать, что любой заданный квадрат остается замусоренным с вероятностью p, тогда как чистый квадрат становится замусоренным с вероятностью $1 - p$ (а это означает, что каждый квадрат содержит мусор в среднем половину времени). Робот имеет датчик наличия мусора в его текущем местоположении, дающий правильные показания с вероятностью 0,9. На рис. 14.20 представлена соответствующая сеть DBN.

Для простоты начнем с допущения, что робот имеет не зашумленный датчик препятствия, а правильно работающий датчик местоположения. Показатели работы алгоритма показаны на рис. 14.21, *a*: его оценки наличия мусора даны в сравнении с результатами точного вывода. (Скоро будет показано, как точный вывод становится возможным.) Для малых значений вероятности сохранности мусора p ошибка остается небольшой, но это нельзя считать значительным достижением, поскольку для каждого квадрата истинные апостериорные вероятности о наличия мусора будут близки к 0,5, если робот не посещал этот квадрат в недавнем времени. Для более высоких значений p мусор сохраняется в целом дольше, так что посещение квадрата дает роботу более полезную информацию, сохраняющую свою актуальность на более продолжительный период времени. Возможно, может показаться удивительным тот факт, что при более высоких значениях p алгоритм фильтрация частиц работает *хуже*. И он полностью не способен работать при $p = 1$, даже несмотря на то, что этот случай кажется самым простым: мусор появляется в квадратах в момент времени 0 и остается в них навсегда, так что после нескольких обходов мира робот должен уже иметь карту местонахождения мусора, близкую к абсолютно точной. Почему алгоритм фильтрации частиц не работает в этом случае?

Как оказалось, теоретическое условие, требующее, чтобы "вероятности в модели перехода и модели восприятия были строго больше 0 и строго меньше 1", является чем-то большим, чем просто математической педантичностью. Вот что происходит: сначала каждая частица исходно содержит 42 предположения из $\mathbf{P}(\mathbf{X}_0)$ о том, в каких квадратах есть мусор, а в каких — нет. Затем состояние для каждой частицы проектируется в прямом направлении во времени в соответствии с моделью перехода. К сожалению, модель перехода для детерминированного расположения мусора является *детерминированной*: мусор остается именно там, где он был. Таким образом, начальные догадки в каждой частице никогда не обновляются с учетом свидетельств.

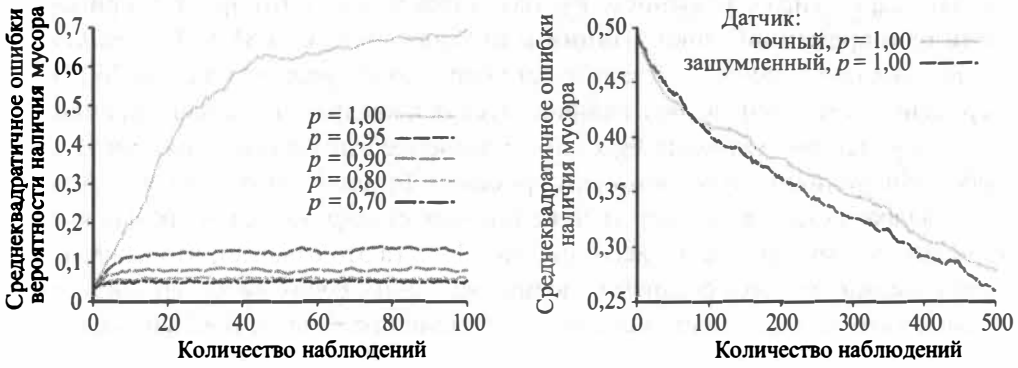


Рис. 14.21. а) Эффективность работы стандартного алгоритма фильтрации частиц при 1000 частиц, представленная среднеквадратической ошибкой в расчетной предельной вероятности наличия мусора в сравнении с результатами точного вероятностного вывода для различных значений вероятности устойчивости загрязнений p. **б)** Эффективность алгоритма фильтрации частиц Рао-Блэквелла (100 частиц) в сравнении с истинной ситуацией как для случая точного определения местоположения, так и для случая зашумленного обнаружения стен, но при детерминированном распределении мусора. Данные усреднены для 20 прогонов

Вероятность того, что исходные догадки были верны, равна 2^{-42} или около 2×10^{-13}, так что исчезающе маловероятно, что среди тысячи (или даже миллиона) частиц будет хотя бы одна с правильной картой местонахождения мусора. В типичном случае лучшая частица из тысячи будет содержать примерно 32 правильных и 10 неправильных догадок, и, как правило, будет только одна такая частица или, возможно, несколько. Одна из этих лучших частиц станет доминировать в общей вероятности с ходом времени и разнообразие в популяции частиц начнет сокращаться. Тогда, поскольку все частицы согласятся с одной, неправильной картой, алгоритм придет к убеждению, что именно эта карта является правильной и никогда не изменит этого мнения.

К счастью, задача одновременной локализации и отображения имеет особую структуру: обусловленный последовательностью местоположений робота, статус наличия мусора для отдельных квадратов является независимым. Более конкретно

$$\mathbf{P}(Dirt_{1,0:t},\ldots,Dirt_{42,0:t} \mid DirtSensor_{1:t}, WallSensor_{1:t}, Location_{1:t}) = \\ = \prod_i \mathbf{P}(Dirt_{i,0:t} \mid DirtSenson_{1:t}, Location_{1:t}). \qquad (14.24)$$

Это означает, что здесь будет полезно применить статистический прием, получивший название ►**Рао-Блэквеллизация** и основанный на простой идее, что

точный вывод всегда будет более точным, чем выборки, даже если это только некоторое подмножество из всех переменных. Для задачи SLAM запустим частицы фильтрации в местонахождении робота, а затем для каждой частицы выполним точный вероятностный HMM-вывод независимо для каждого квадрата с мусором, обусловленный последовательностью расположений в этой частице. Таким образом, каждая частица содержит выборочное местоположение плюс 42 точных предельных апостериорных распределения для 42 квадратов, — точных при допущении, что предполагаемая траектория местоположений, которой следует эта частица, является правильной. Этот подход, называемый ▶**фильтром частиц Рао-Блэквелла**, без затруднений справляется с ситуацией детерминированного размещения мусора и постепенно строит точную карту местонахождения мусора с использованием как правильно работающего датчика местоположения, так и зашумленного датчика препятствия, что и показано на рис. 14.21, *б*.

Во всех случаях, которые не удовлетворяют тому типу структуры условной независимости, который проиллюстрирован уравнением (14.24), метод Рао-Блэквелла не применим. В разделе "Библиографические и исторические заметки" в конце главы упоминается ряд алгоритмов, которые в свое время были предложены для решения общей задачи фильтрации со статическими переменными. Ни один из них не обладает элегантностью и широтой применимости, характерными для алгоритма фильтра частиц, но на практике некоторые из них весьма эффективны для определенных классов задач.

Резюме

В этой главе рассматривалась общая проблема представления и формирования рассуждений о вероятностных временных процессах. Основные идеи, изложенные в этой главе, следующие.

- Изменение состояния мира можно учесть, используя множество случайных переменных для представления этого состояния в каждый момент времени.
- Эти представления могут быть спроектированы (приблизительно) таким образом, чтобы они удовлетворяли **свойству марковости**, согласно которому будущее не зависит от прошлого, если дано настоящее. В сочетании с предположением о том, что рассматриваемый процесс является **стационарным** (т.е. таким, что его законы не изменяются со временем), это позволяет намного упростить представление.
- Временная вероятностная модель может рассматриваться как содержащая **модель перехода**, описывающую процесс развития, и **модель восприятия**, описывающую процесс наблюдения.
- Основными задачами вероятностного вывода во временных моделях являются **фильтрация (оценка состояния), предсказание, сглаживание** и определение с помощью вычислений **наиболее вероятного объяснения**.

Каждая из этих задач может быть решена с помощью простых, рекурсивных алгоритмов, время выполнения которых линейно зависит от длины рассматриваемой последовательности.

- Более подробно рассматривались три семейства временных моделей: **скрытые марковские модели, фильтры Калмана** и **динамические байесовские сети** (последняя модель включает две первые в качестве частных случаев).
- Если не приняты особые предположения, как при использовании фильтров Калмана, точный вероятностный вывод при наличии многих переменных состояния становится неосуществимым. На практике алгоритм **фильтрации частиц** и его производные являются достаточно эффективным семейством алгоритмов.

Библиографические и исторические заметки

Многие важные идеи, касающиеся оценки состояния динамических систем, были высказаны математиком К.Ф. Гауссом ([821], 1809), сформулировавшим детерминированный алгоритм наименьших квадратов для решения задачи прогнозирования орбит небесных тел на основании астрономических наблюдений. Российский математик А.А. Марков ([1494], 1913) в своих трудах, посвященных анализу стохастических процессов, изложил подход, получивший в дальнейшем название **марковское предположение**; он провел оценку свойств марковской цепи первого порядка, состоящей из букв текста поэмы "Евгений Онегин". Общая теория марковских цепей и время их смешивания подробно обсуждаются в работе Левина и соавт. [1393] (2008).

Важная классификационная работа по фильтрации была выполнена во время Второй мировой войны Винером ([2338], 1942) для непрерывных временных процессов и Колмогоровым ([1270], 1941) для дискретных временных процессов. Хотя эта работа привела к важным технологическим усовершенствованиям, достигнутым в течение следующих 20 лет, в ней использовалось представление данных из области определения частот, поэтому многие вычисления оказались весьма громоздкими. Как было показано Питером Сверлингом ([2164], 1959) и Рудольфом Калманом ([1175], 1960), непосредственное моделирование стохастических процессов с помощью пространства состояний оказалось намного проще. В последней статье описывается то, что теперь принято называть фильтром Калмана для прямого вероятностного вывода в линейных системах с гауссовым шумом. Однако результаты Калмана ранее уже были получены датским астрономом Торвольдом Тилем ([2201], 1880) и русским физиком Русланом Стратоновичем ([2143], 1959). При посещении в 1960 году Исследовательского центра Эймса, принадлежащего НАСА, Калман увидел возможность применения этого метода для отслеживания траекторий ракет, в результате чего его фильтр нашел применение в миссии "Аполлон".

Важнейшие результаты в области сглаживания были получены Раухом и соавт. ([1862], 1965), а предложенный ими метод, получивший выразительное название "метод сглаживания Рауха–Тунга–Стрибеля", все также широко применяется и в наши дни. Многие ранние результаты исследований были собраны Гелбом ([827], 1974). Бар-Шалом и Фортманн ([129], 1988) предоставили их более современную трактовку в байесовском стиле, а также многочисленные ссылки на обширную литературу по этой теме. Четфилд ([402], 1989), а также Бокс и соавторы ([276], 2016) предложили подход в стиле теории управления к анализу временных рядов.

Скрытая марковская модель и связанные с ней алгоритмы вероятностного вывода и обучения, включая прямой-обратный алгоритм, были разработаны Баумом и Петри ([143], 1966). Алгоритм Витерби впервые был предложен его автором в [2278] (1967). Аналогичные идеи были также независимо высказаны в сообществе специалистов по калмановской фильтрации (Раух и др. [1862], 1965).

Прямой-обратный алгоритм был одним из основных предшественников более общей формулировки алгоритма EM (Демпстер и др. [600], 1977; см. также главу 20). Описание процедуры сглаживания в постоянном пространстве впервые появилось в работе Биндера и соавт. [217] (1997), так же как и алгоритм, действующий по принципу "разделяй и властвуй", который предлагается разработать в упражнении 14.3. Сглаживание с постоянным временем и фиксированным отставанием для скрытых марковских моделей впервые было предложено Расселом и Норвигом в [1944] (2003).

Скрытые марковские модели (HMM) уже нашли широкое применение в обработке естественного языка (Чарняк [393], 1993), распознавании речи (Рабинер и Цзуанг [1840], 1993), машинном переводе (Оч и Ней [1703], 2003), вычислительной биологии (Крог и др. [1314], 1994; Балди и др. [117], 1994), финансах и экономике (Бхар и Хамори [209], 2004) и других областях. Было предложено несколько расширений основной модели HMM, например в иерархической HMM (Файн и др. [740], 1998) и многослойной HMM (Оливер и др. [1711], 2004) структура вновь вводится в модель, заменяя единственную переменную состояния классической HMM.

Динамические байесовские сети (*Dynamic Bayesian network* — DBN) могут рассматриваться как способ разреженного кодирования марковского процесса; впервые они были применены в области искусственного интеллекта Дином и Канадзава ([571], 1989), Николсоном и Бреди ([1679], 1992), а также Кьерульфом ([1236], 1992). Последняя работа включает описание расширения системы на основе байесовских сетей Hugin с целью поддержки динамических байесовских сетей. Книга Дина и Веллмана [572] (1991) способствовала популяризации DBN и применения вероятностного подхода к планированию и контролю в рамках ИИ. Мерфи в [1637] (2002) предоставил глубокий анализ динамических байесовских сетей.

Динамические байесовские сети стали популярным выбором для моделирования различных сложных процессов движения в системах машинного зрения (Хуанг и др. [1084], 1994; Интилл и Бобик [1114], 1999). Как и скрытая марковская модель, сети DBN нашли применение в системах распознавания речи (Цвейг и

Рассел [2455], 1998; Ливеску и др. [1435], 2003), локализации роботов (Теохарос и др. [2200], 2004) и исследования геномов (Мерфи и Миан [1639], 1999; Ли и др. [1407], 2011). Другие области применения включают анализ жестов (Сук и др. [2147], 2010), выявление усталости водителя (Янг и др. [2396], 2010), а также моделирование городского трафика (Хофлейтнер и др. [1047], 2012).

В работе Смита и соавт. [2105] (1997) явно показана связь между моделями HMM и сетями DBN, а также между прямым-обратным алгоритмом и алгоритмом распространения в байесовской сети. Результаты дальнейшего обобщения фильтров Калмана (и других статистических моделей) представлены Ровейсом и Гахрамани в [1923] (1999). Существуют процедуры для обучения параметров (Биндер и др. [216], 1997; Гахрамани [844], 1998) и структуры (Фридман и др. [792], 1998) сетей DBN. **Байесовские сети с непрерывным временем** (Ноделман и др. [1696], 2002) представляют собой дискретный аналог сети DBN с непрерывным временем, что исключает необходимость выбора определенной фиксированной длительности временных этапов.

Первые алгоритмы формирования выборки для фильтрации (также называемые последовательными методами Монте-Карло) были разработаны в сообществе теории управления Хеншином и Мейном ([597], 1969), а идея повторной выборки, являющаяся центральным элементом метода фильтрации частиц, впервые была упомянута в российском журнале по теории управления (Зарицкий и др. [2420], 1975). Позднее этот подход был заново изобретен в статистике под названием **последовательная выборка по важности с перевыборкой** (*Sequential Importance sampling with Resampling* — **SIR**) (Рубин [1927], 1988; Лю и Чен [1426], 1998), в теории управления как метод фильтрации частиц (Гордон и др. [907], 1993; Гордон [908], 1994), в области ИИ как **выживание приспособленных** (Каназава и др. [1181], 1995) и в области компьютерного зрения как **конденсация** (Изард и Блейк [1117], 1996).

Статья Каназавы и соавт. [1181] (1995) содержит предложение улучшения под названием ▶ **разворот свидетельства**, согласно которому выборка для состояния в момент времени $t + 1$ обусловливается как состоянием в момент времени t, так и свидетельством в момент времени $t + 1$. Это позволяет свидетельству оказывать непосредственное влияние на формирование выборки и способствует уменьшению ошибки аппроксимации, что было доказано Дусе в [640] (1997) и Лю и Ченом в [1426] (1998).

Метод фильтрации частиц нашел применение во многих областях, в том числе в отслеживании сложных закономерностей движений в видео (Изард и Блейк [1117], 1996), в прогнозировании на фондовом рынке (Де-Фрейтас и др. [555], 2000) и в диагностике неисправностей у планетоходов (Верма и др. [2271], 2004). С момента его изобретения по применению и вариантам этого алгоритма были опубликованы десятки тысяч работ. Сейчас большое значение придается масштабируемым реализациям систем на базе параллельно работающих аппаратных средств. Хотя может показаться, что нет ничего сложного в том, чтобы распределить N частиц среди N вычислительных потоков параллельно работающих процессоров, основной

алгоритм требует строго синхронизированного взаимодействия между этими потоками на этапе перевыборки (Хендеби и др. [1010], 2010). **Алгоритм каскада частиц** (Пайге и др. [1723], 2015) исключает необходимость подобной синхронизации, что приводит к значительному ускорению параллельных вычислений.

Фильтр частиц Рао-Блэквелла был предложен Дусе и соавт. ([642], 2000) и Мерфи и Расселом ([1640], 2001). Его применение на практике для решения задач локализации и отображения в области робототехники описывается в главе 26. Для решения более общих задач фильтрации со статическими или почти статическими переменными были предложены и многие другие алгоритмы, включая алгоритм повторной выборки со смещением (Гилкс и Берцуини [856], 2001), алгоритм Лю-Веста (Лю и Вест [1427], 2001), фильтр Сторвика (Сторвик [2141], 2002), расширенный фильтр параметров (Эрол и др. [696], 2013) и фильтр предполагаемых параметров (Эрол и др. [697], 2017). Последний представляет собой гибрид алгоритма фильтрации частиц с гораздо более старой идеей, называемой ►**фильтром предполагаемой плотности**. В методе фильтра предполагаемой плотности принимается допущение, что апостериорное распределение по состояниям в момент времени t принадлежит определенному конечно параметризованному семейству. Если этапы проецирования и обновления выводят его за пределы этого семейства, распределение проецируется обратно, чтобы дать наилучшее приближение в пределах семейства. Для сетей DBN в алгоритме Бойена-Коллера (Бойен и др. [280], 1999) и алгоритме ►**факторизованной границы** (Мерфи и Вейсс [1641], 2001) предполагается, что апостериорное распределение может быть хорошо аппроксимировано произведением малых факторов.

К задаче фильтрации могут быть применены методы MCMC (см. раздел 13.4.2), например выборка Гиббса может быть применена непосредственно к развернутой сети DBN. Семейство алгоритмов ►**частиц MCMC** (Андрю и др. [55], 2010; Линдстен и др. [1418], 2014) сочетает в себе алгоритм фильтрации частиц и методы MCMC, применяемые к развернутой временной модели для генерации вспомогательных распределений MCMC. Хотя в общем случае эти алгоритмы сходятся к правильному апостериорному распределению (т.е. как со статическими, так и с динамическими переменными), это автономные алгоритмы. Чтобы избежать проблем увеличения времени обновления по мере роста развернутой сети, в фильтре ►**затухающего MCMC** (Марти и др. [1504], 2002) отдается предпочтение формированию выборок из относительно недавних переменных состояния с вероятностью, уменьшающейся для переменных из более далекого прошлого.

В книге Дусе и соавт. [641] (2001) собрано много важных работ по ►**последовательным алгоритмам Монте-Карло** (SMC), среди которых алгоритм фильтрации частиц является наиболее важным. Есть полезные учебные пособия, выпущенные Арулампаламом и соавт. ([79], 2002), а также Дусе и Йохансеном ([643], 2011). Существует также несколько теоретических работ, касающихся условий, при которых методы SMC сохраняют ограниченную ошибку по отношению к истинной апостериорной вероятности (Крисе и Дусе [497], 2002; Дел Морал [596], 2004; Дел Морал и соавт. [595], 2006).

Упражнения

14.1. Покажите, что любой марковский процесс второго порядка может быть переоформлен в виде марковского процесса первого порядка с дополненным множеством переменных состояния. Может ли такое преобразование всегда быть выполнено *экономно*, т.е. без увеличения количества параметров, необходимых для определения модели перехода?

14.2. В этом упражнении рассматривается, что происходит с вероятностями в мире задачи с зонтиком по мере приближения к пределу в длинных временных последовательностях.

а) Предположим, что наблюдается нескончаемая последовательность дней, в которых директор появляется на работе с зонтиком или без зонтика. Покажите, что по мере того, как эти дни проходят, вероятность дождя в текущий день возрастает монотонно в направлении к фиксированной точке. Рассчитайте эту фиксированную точку.

б) Теперь рассмотрим задачу прогнозирования все дальше и дальше в будущее по данным только первых двух наблюдений о наличии зонтика. Вначале рассчитайте вероятность $P(r_{2+k} \mid u_1, u_2)$ для $k = 1 \ldots 20$ и нанесите результаты на график. Вы обнаружите, что эта вероятность сходится в фиксированной точке. Рассчитайте точное значение для этой фиксированной точки.

14.3. В этом упражнении разрабатывается вариант прямого-обратного алгоритма, приведенного на рис. 14.4 (раздел 14.2.2). Требуется вычислить значение $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ для $k = 1, \ldots, t$. Такую задачу можно решить с помощью подхода по принципу "разделяй и властвуй".

а) Для упрощения примем предположение, что значение t является нечетным, и допустим, что промежуточная точка определяется выражением $h = (t+1)/2$. Покажите, что значение $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ можно вычислить для $k = 1, \ldots, h$, если даны лишь первоначальное прямое сообщение $\mathbf{f}_{1:0}$, обратное сообщение $\mathbf{b}_{h+1:t}$ и свидетельство $\mathbf{e}_{1:h}$.

б) Предоставьте аналогичный результат для второй половины последовательности.

в) Имея результаты выполнения пп. а и б, можно сформировать рекурсивный алгоритм "разделяй и властвуй", вначале выполнив прогон вдоль последовательности в прямом направлении, а затем — в обратном направлении, начав от ее конца и сохранив лишь необходимые сообщения в середине и на концах. Затем алгоритм вызывается на каждой половине последовательности. Составьте подробный листинг этого алгоритма.

г) Определите временную и пространственную сложность алгоритма как функцию от t, длины последовательности. Как изменятся эти результаты, если входные данные будут разделены более чем на две части?

14.4. В разделе 14.2.3 была кратко описана некорректная процедура определения наиболее вероятной последовательности состояний, в которой используется последовательность наблюдений. В этой процедуре предусматривается поиск в

каждом временном интервале наиболее вероятного состояния, применение операции сглаживания и возврат последовательности, в которой собраны эти состояния. Покажите, что при использовании некоторых временных вероятностных моделей и последовательностей наблюдений эта процедура возвращает невозможную последовательность состояний (т.е. такую последовательность, что ее апостериорная вероятность равна нулю).

14.5. Уравнение (14.12) описывает процесс фильтрации для матричной формулировки HMM. Приведите аналогичное уравнение для расчета правдоподобия, которое в общем случае было описано в уравнении (14.7).

14.6. Рассмотрим миры пылесоса, представленные на рис. 4.18 (идеальное восприятие) и 14.7 (зашумленное восприятие). Предположим, что робот получает такую последовательность наблюдений, что при идеальном восприятии существует только одно возможное местоположение, в котором он может находиться. Обязательно ли это местоположение будет наиболее вероятным местоположением и при зашумленном восприятии с достаточно малой вероятностью шума ϵ? Докажите свое утверждение или предоставьте контрпример.

14.7. В разделе 14.3.2 в задаче мира пылесоса априорное распределение по местоположениям является равномерным, и модель перехода предполагает равную вероятность перехода к любому соседнему квадрату. Но что, если эти предположения неверны? Предположим, что начальное местоположение фактически выбирается с равной вероятностью в северо-западном квадранте комнаты, а действие на самом деле имеет тенденцию к переходу в юго-восточном направлении. Сохраняя модель HMM неизменной, изучите влияние на точность локализации и точность определения пути при увеличении тенденции перемещения на юго-восток для различных значений ϵ.

14.8. Рассмотрим версию мира пылесоса (раздел 14.3.2), в которой для робота установлено ограничение двигаться в одном направлении до тех пор, пока это будет возможно. Выбрать новое направление (случайным образом) он может, только натолкнувшись на препятствие. Для моделирования поведения этого робота каждое состояние в модели должно состоять из пары переменных — *location* (местоположение) и *heading* (направление). Реализуйте эту модель и посмотрите, насколько хорошо теперь алгоритм Витерби позволяет роботу отслеживать свое местоположение. Наложенные на робота ограничения более жесткие, чем в случае, когда ему позволено случайное блуждание. Значит ли это, что в данном случае прогноз его наиболее вероятного пути будет более точным?

14.9. Выше для задачи мира пылесоса (см. рис. 14.7) были предложены три варианта ограничений, налагаемых на возможности перемещения робота-пылесоса: 1) случайное блуждание с равной вероятностью выбора направления; 2) тенденция к перемещению на юго-восток, как описано в упражнении 14.7; 3) ограничения, описанные в упражнении 14.8. Предположим, что сторонний наблюдатель получает всю последовательность восприятия от робота-пылесоса, но не уверен в том, какой тип ограничений из трех возможных на него наложен. Какой подход должен использовать наблюдатель, чтобы найти наиболее вероятный путь робота, с учетом поступающей последовательности его восприятия?

Реализуйте этот подход и протестируйте его. Насколько ухудшилась точность локализации по сравнению со случаем, когда наблюдатель точно знает, какой именно вариант ограничений наложен на робота?

14.10. Это упражнение связано с фильтрацией в среде без ориентиров. Рассмотрим вариант задачи мира пылесоса, в котором робот находится в пустой комнате, представленной прямоугольной сеткой размером $n \times m$. Местоположение робота скрыто, и единственное свидетельство, доступное наблюдателю, поступает от зашумленного датчика местоположения, приблизительно определяющего местоположение робота. Если робот находится в местоположении (x, y), то с вероятностью 0,1 датчик выдаст это правильное местоположение; с вероятностью 0,05 для каждого он может указать на один из 8 квадратов, непосредственно окружающих квадрат (x, y); либо с вероятностью 0,025 для каждого укажет на один из 16 квадратов, окружающих предыдущие 8, а с оставшейся вероятностью 0,1 может сообщить "нет данных". Наложенные на робота ограничения требуют от него выбрать направление и следовать ему с вероятностью 0,8 на каждом этапе, а с оставшейся вероятностью 0,2 робот может переключиться на случайно выбранный новый курс (или с вероятностью 1,0, если наталкивается на стену). Реализуйте эту задачу как модель HMM и выполните фильтрацию для отслеживания пути робота. Насколько точно можно отслеживать путь робота при заданных условиях?

14.11. Это упражнение связано с фильтрацией в среде без ориентиров. Рассмотрим вариант задачи мира пылесоса, в котором робот находится в пустой комнате, представленной прямоугольной сеткой размером $n \times m$. Местоположение робота скрыто, и единственное свидетельство, доступное наблюдателю, поступает от зашумленного датчика местоположения, приблизительно определяющего местоположение робота. Если робот находится в местоположении (x, y), то с вероятностью 0,1 датчик выдаст это правильное местоположение; с вероятностью 0,05 для каждого он может указать на один из 8 квадратов, непосредственно окружающих квадрат (x, y); либо с вероятностью 0,025 для каждого укажет на один из 16 квадратов, окружающих предыдущие 8, а с оставшейся вероятностью 0,1 может сообщить "нет данных". Наложенные на робота ограничения требуют от него выбрать направление и следовать ему с вероятностью 0,7 на каждом этапе, а с оставшейся вероятностью 0,3 робот может переключиться на случайно выбранный новый курс (или с вероятностью 1,0, если наталкивается на стену). Реализуйте эту задачу как модель HMM и выполните фильтрацию для отслеживания пути робота. Насколько точно можно отслеживать путь робота при заданных условиях?

14.12. Часто возникает необходимость осуществлять текущий контроль за системой с непрерывным состоянием, поведение которой переключается непредсказуемым образом с одного режима на другой в множестве из k различных режимов. Например, самолет, пытающийся избежать поражения ракетой, может выполнить ряд различных маневров, которые попытается отследить система управления ракетой. Представление такой модели **переключательного фильтра Калмана** в виде байесовской сети показано на рис. 14.22.

а) Допустим, что дискретное состояние S_t имеет k возможных значений и что априорная непрерывная оценка состояния $\mathbf{P}(\mathbf{X}_0)$ представляет собой многомерное гауссово распределение. Покажите, что предсказание $\mathbf{P}(\mathbf{X}_1)$ представляет собой **сочетание гауссовых распределений**, т.е. такую взвешенную сумму гауссовых распределений, что веса в сумме составляют 1.

б) Покажите, что если текущая оценка непрерывного состояния $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ представляет собой сочетание m гауссовых распределений, то в общем случае обновленная оценка состояния $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1})$ будет представлять собой сочетание km гауссовых распределений.

в) Какой аспект временного процесса представляют веса в сочетании гауссовых распределений?

Результаты выполнения пп. а и б, вместе взятые, показывают, что объем этого представления апостериорных вероятностей беспредельно возрастает даже при использовании переключательных фильтров Калмана, которые являются простейшими гибридными динамическими моделями.

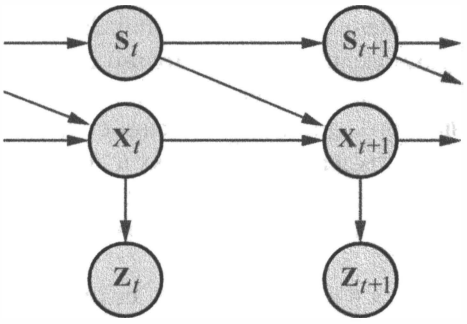


Рис. 14.22. Представление переключательного фильтра Калмана в виде байесовской сети. Переключательная переменная S_t представляет собой дискретную переменную состояния, значение которой определяет модель перехода для непрерывных переменных состояния $\mathbf{X}_t$. Для любого дискретного состояния i модель перехода $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{X}_t, S_{t=i})$ представляет собой линейную гауссову модель, так же как и в обычном фильтре Калмана. Модель перехода для дискретного состояния, $\mathbf{P}(S_{t+1} \mid S_t)$, может рассматриваться как матрица по аналогии со скрытой марковской моделью

14.13. Дополните недостающий этап вывода уравнения (14.19) (раздел 14.4.2) — первый этап обновления для одномерного фильтра Калмана.

14.14. Рассмотрим ход выполнения операции обновления дисперсии в уравнении (14.20) (раздел 14.4.2).

а) Нанесите на график значения выражения σ_t^2 как функции от t при наличии различных значений для σ_x^2 и σ_z^2.

б) Покажите, что эта операция обновления имеет фиксированную точку σ^2, такую, что $\sigma_t^2 \to \sigma^2$, когда $t \to \infty$, и рассчитайте значение σ^2.

в) Дайте качественное объяснение того, что происходит по мере того, как $\sigma_x^2 \to 0$ и $\sigma_z^2 \to 0$.

14.15. Профессор хочет выяснить, достаточно ли спят студенты. Каждый день профессор наблюдает, спят ли студенты на занятиях и красные ли у них глаза. Он разработал следующее теоретическое описание для данной предметной области.

– При отсутствии наблюдений априорная вероятность того, что студент выспался, составляет 0,7.

– Вероятность выспаться в ночь t равна 0,8, если известно, что студент достаточно выспался предыдущей ночью, и равна 0,3, если это не так.

– Вероятность появления красных глаз составляет 0,2, если студент выспался, и 0,7, если он не выспался.

– Вероятность уснуть на занятиях составляет 0,1, если студент выспался, и 0,3, если этот не так.

Исходя из этой информации, постройте динамическую байесовскую сеть, которую профессор смог бы использовать для фильтрации или прогнозирования на основании серии выполненных им наблюдений. Затем переформулируйте ее как скрытую марковскую модель, имеющую только одну переменную наблюдения. Дайте полные таблицы вероятностей для этой модели.

14.16. Профессор хочет выяснить, достаточно ли спят студенты. Каждый день профессор наблюдает, спят ли студенты на занятиях и красные ли у них глаза. Он разработал следующее теоретическое описание для данной предметной области.

– При отсутствии наблюдений априорная вероятность того, что студент выспался, составляет 0,6.

– Вероятность выспаться в ночь t равна 0,8, если известно, что студент достаточно выспался предыдущей ночью, и равна 0,2, если это не так.

– Вероятность появления красных глаз составляет 0,2, если студент выспался, и 0,7, если он не выспался.

– Вероятность уснуть на занятиях составляет 0,1, если студент выспался, и 0,3, если этот не так.

Исходя из этой информации, постройте динамическую байесовскую сеть, которую профессор смог бы использовать для фильтрации или прогнозирования на основании серии выполненных им наблюдений. Затем переформулируйте ее как скрытую марковскую модель, имеющую только одну переменную наблюдения. Составьте полные таблицы вероятностей для этой модели.

14.17. Для сети DBN, определенной в упражнении 14.15, и при значениях свидетельств

$\mathbf{e}_1$ = нет красных глаз, нет сна на занятиях

$\mathbf{e}_2$ = красные глаза, нет сна на занятиях

$\mathbf{e}_3$ = красные глаза, уснул на занятиях

выполните следующие вычисления (EnoughSleep — выспался).

а) **Оценка состояния**. Вычислить $P(EnoughSleep_t | \mathbf{e}_{1:t})$ для каждого $t = 1, 2, 3$.
б) **Сглаживание**. Вычислить $P(EnoughSleep_t | \mathbf{e}_{1:3})$ для каждого $t = 1, 2, 3$.
в) Сравнить отфильтрованные и сглаженные вероятности для $t = 1$ и $t = 2$.

14.18. Предположим, что какой-то конкретный студент появляется с красными глазами и каждый день спит на занятиях. Учитывая модель, описанную в упражнении 14.15, объясните, почему вероятность того, что ученик выспался предыдущей ночью, сходится к фиксированной точке, а не продолжает снижаться по мере того, как собирается все больше свидетельств за прошедшие дни. Что такое фиксированная точка? Дайте ответ на этот вопрос как численно (с помощью вычислений), так и аналитически.

14.19. В этом упражнении более подробно анализируется устойчивая к отказам модель для датчика уровня заряда аккумулятора, показанная на рис. 14.15, *а* (раздел 14.5.1).

а) График на рис. 14.15, *б* обрывается при $t = 32$. Дайте качественное описание того, что должно произойти по мере стремления t к бесконечности, $t \to \infty$, если датчик продолжает выдавать показания 0.
б) Предположим, что температура окружающей среды влияет на датчик уровня заряда аккумулятора таким образом, что по мере возрастания температуры временные отказы становятся все более вероятными. Покажите, как с учетом этого дополнить структуру сети DBN, приведенную на рис. 14.15, *а*, и объясните, какие изменения потребуется внести в таблицы условных вероятностей.
в) После определения новой структуры сети сможет ли робот использовать показания датчика уровня заряда аккумулятора для вероятностного вывода данных о текущей температуре?

14.20. Рассмотрите задачу применения алгоритма устранения переменной к сети DBN для задачи с зонтиком, развернутую на три временных среза, в которой используется запрос $\mathbf{P}(R_3 | u_1, u_2, u_3)$. Покажите, что сложность этого алгоритма (размер наибольшего фактора) является одинаковой независимо от того, устраняются ли переменные, касающиеся дождя, в прямом или обратном порядке.

ГЛАВА 15

Вероятностное программирование

В этой главе обсуждается идея универсальных языков для вероятностного представления знаний и вероятностного вывода в проблемных областях, характеризующихся наличием неопределенности.

Возможный спектр вариантов представления данных — атомарное, развернутое и структурное — является постоянной темой в области ИИ. В случае детерминированных моделей алгоритмы поиска допускают только атомарное представление, методы решения задач удовлетворения ограничений и логика высказываний предусматривают развернутое представление, а логика первого порядка и системы планирования используют преимущества структурного представления. Выразительная сила, обеспечиваемая структурным представлением, позволяет создавать модели, несравненно более краткие, чем эквивалентные им структурные или атомарные описания.

В случае вероятностных моделей байесовские сети, как указывалось в главах 13 и 14, являются развернутым представлением задачи: множество случайных переменных является фиксированным и конечным, причем для каждой из них определен фиксированный диапазон возможных значений. Этот факт ограничивает применимость байесовских сетей, поскольку представление достаточно сложной проблемной области в виде байесовской сети просто оказывается слишком большим. Это делает невозможным как создание таких представлений вручную, так и их обучение на базе любого разумного количества данных.

Проблема создания выразительного формального языка для представления вероятностной информации в свое время изучалась некоторыми из величайших умов в истории человечества, в том числе Готфридом Лейбницем (независимо создавшим математический анализ), Якобом Бернулли (открывшим число e, создавшим вариационное исчисление и открывшим закон больших чисел), Огастесом де Морганом, Джорджем Булем, Чарльзом Сандерсом Пирсом (одним из ведущих логиков XIX века), Джоном Мейнардом Кейнсом (ведущим экономистом XX века) и Рудольфом Карнапом (выдающимся философом-аналитиком XX века). Несмотря

на усилия этих и многих других исследователей решить эту задачу не удавалось вплоть до 1990-х годов.

Отчасти благодаря разработке байесовских сетей в настоящее время уже имеются математически элегантные и в высшей степени практичные формальные языки, позволяющие создавать вероятностные модели для очень сложных проблемных областей. Эти языки являются *универсальными* в том же смысле, что и машина Тьюринга: они способны представить любую вычислимую вероятностную модель точно так, как машина Тьюринга способна представить любую вычислимую функцию. Кроме того, эти языки включают алгоритмы вероятностного вывода общего назначения, аналогичные непротиворечивым и полным алгоритмам логического вывода, подобным правилу резолюций.

Есть два способа введения необходимой выразительной силы в теорию вероятностей. Первый — с помощью логики: разработать язык, в котором вероятности определяются по возможным мирам логики первого порядка, а не по возможным мирам логики высказываний байесовских сетей. Этот способ обсуждается в разделах 15.1 и 15.2, а в разделе 15.3 рассматривается конкретный пример рассуждений во времени. Второй способ — обратиться к традиционным языкам программирования: в них вводятся стохастические элементы — например, случайный выбор, — а программы рассматриваются как определенные вероятностные распределения по их собственным путям выполнения. Этот подход обсуждается в разделе 15.4.

Оба способа приводят к созданию ▶**языка вероятностного программирования (PPL)**. В первом случае это будут декларативные языки PPL, имеющие примерно такое же родство с общими языками PPL, как логическое программирование (глава 9) с общими языками программирования.

15.1. Реляционные вероятностные модели

Вспомним из главы 12, что вероятностная модель определяет множество Ω возможных миров с вероятностью $P(\omega)$ для каждого мира ω. Для байесовской сети возможные миры представляют собой присваивания значений переменным, в частности для случая булевых переменных возможные миры идентичны мирам логики высказываний.

Тогда можно полагать, что для вероятностной модели первого порядка потребуются возможные миры, являющиеся таковыми в логике первого порядка, т.е. множество объектов с отношениями между ними и интерпретацией, отображающей символы констант на объекты, символы предикатов на отношения и символы функций на функции, определенные на этих объектах (см. раздел 8.2.) Модель также должна определять вероятность для каждого из таких возможных миров — так же, как байесовская сеть определяет вероятность для каждого присваивания значений переменных.

На минуту предположим, что нам уже известно, как это можно сделать. Тогда, как обычно (см. раздел 12.2.1), мы можем получить вероятность любого логического высказывания первого порядка ϕ (*фи*) в виде суммы по возможным мирам, где оно является истинным:

$$P(\phi) = \sum_{\omega:\phi \text{ является истинным в } \omega} P(\omega) \qquad (15.1)$$

Условные вероятности $P(\phi \mid \mathbf{e})$ могут быть получены аналогичным образом, поэтому мы можем, в принципе, задать модели любой вопрос — и получить ответ. Пока все очень хорошо.

Однако здесь есть одна проблема: множество моделей первого порядка *бесконечно*. Это вполне очевидно было продемонстрировано на рис. 8.4 в разделе 8.2.2, — для удобства он еще раз приведен на рис. 15.1, *а*. А это означает, что, во-первых, суммирование в уравнении (15.1) может оказаться неосуществимым, а во-вторых, задача определения полного, согласованного распределения на бесконечном множестве миров может оказаться очень трудной.

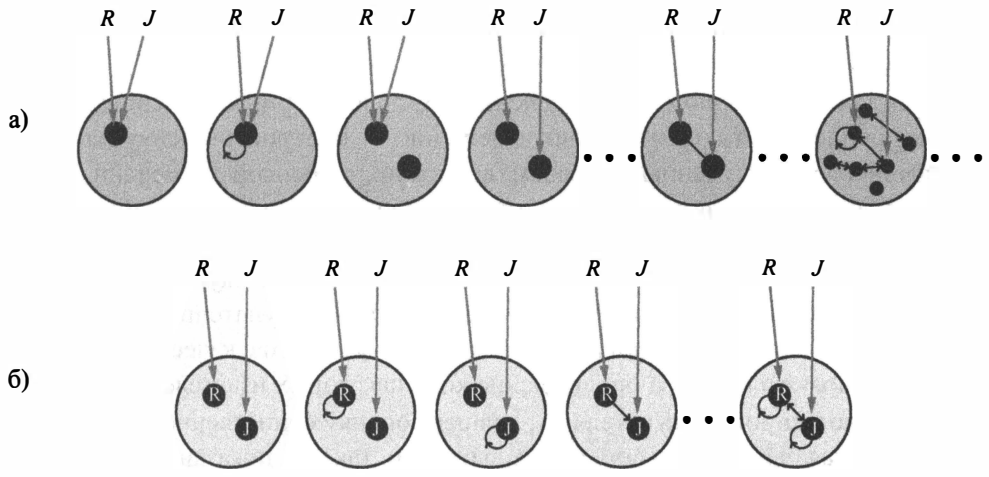


Рис. 15.1. а) Некоторые члены множества всех возможных миров для языка с двумя символами констант, *R* и *J*, и одним бинарным символом отношения, определенные в соответствии со стандартной семантикой логики первого порядка. **б)** Возможные миры в семантике базы данных для тех же исходных условий. Интерпретация символов констант фиксирована, и существует отдельный объект для каждого из этих символов

В этом разделе мы уклонимся от решения данной проблемы, обратившись к **семантике базы данных**, определенной в разделе 8.2.8. В семантике базы данных

принимается допущение **об уникальности имен**, — здесь мы принимаем его для постоянных символов. Также в этой семантике действует правило **замыкания проблемной области** — в ней не могут существовать никакие другие объекты, кроме именованных. В этом случае можно гарантировать конечность множества возможных миров, обеспечив точное соответствие множества объектов в каждом мире множеству используемых символов констант, как показано на рис. 5.1, *б*, где нет никакой неопределенности в отношении отображения символов на объекты или в отношении существующих объектов.

Модели, определенные подобным образом, мы будем называть ▶ **реляционными вероятностными моделями**, или RPM (*Relational Probability Model*).[1] Наиболее существенное различие между семантикой RPM и семантикой базы данных, введенной в разделе 8.2.8, заключается в том, что в семантике RPM *не делается* допущения о замкнутости мира, — в вероятностной системе рассуждений мы не можем вот так просто предположить, что каждый неизвестный факт обязательно является ложным.

15.1.1. Синтаксис и семантика

Начнем с простого примера: предположим, что владелец книжного интернет-магазина хотел бы предоставить на своем сайте общие оценки по отдельным книгам на основании отзывов, полученных от его покупателей. Оценка будет иметь вид апостериорного распределения по качеству книги на основании имеющихся свидетельств. Самое простое решение — построить оценку на основании среднего по полученным отзывам, возможно, с оценкой дисперсии, определяемой с учетом их количества, но в этом случае не принимается во внимание тот факт, что некоторые покупатели добрее остальных, а некоторые менее честны, чем остальные. Добросердечные покупатели склонны давать высокую оценку даже довольно посредственным книгам, тогда как нечестные покупатели дают очень высокие или очень низкие оценки по причинам, не связанным с качеством книги, например им могут платить за продвижение книг некоторых издателей.[2]

Для единственного покупателя C_1, давшего оценку единственной книге B_1, соответствующая байесовская сеть может выглядеть так, как показано на рис. 15.2, *а*. (Как и в разделе 9.1, выражения с круглыми скобками, такие как $Honest(C_1)$, являются просто произвольными обозначениями, в данном случае — произвольно выбранными именами для случайных переменных.) При наличии двух покупателей и двух книг соответствующая байесовская сеть будет выглядеть так, как показано

[1] Название *реляционная вероятностная модель* было введено Пфеффером в статье [1786] (2000) в несколько ином представлении, но положенные в основу идеи те же самые.

[2] Теоретик, специализирующийся на теории игр, мог бы посоветовать нечестным покупателям иногда рекомендовать и хорошую книгу от конкурента, просто чтобы избежать раскрытия. Подробнее об этом читайте в главе 18.

на рис. 15.2, *б*. Для большего количества книг и покупателей будет весьма непрактично определять подобную байесовскую сеть вручную.

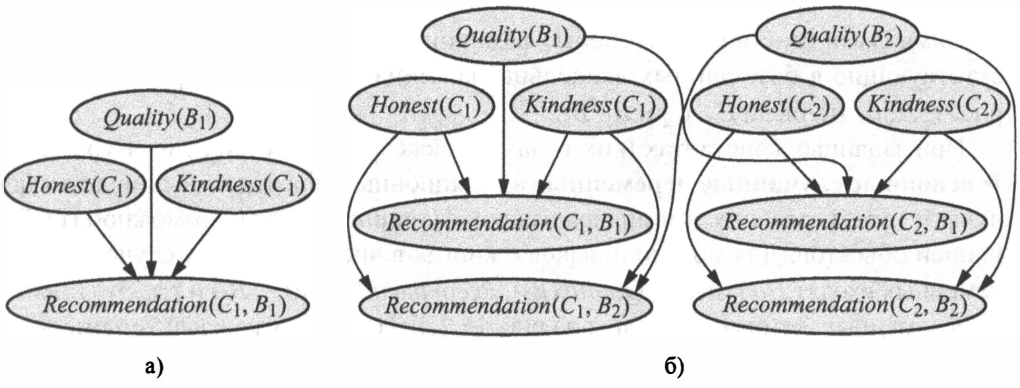


Рис. 15.2. а) Байесовская сеть для одного покупателя C_1, давшего отзыв на одну книгу B_1. Переменная *Honest*(C_1) является булевой, а все остальные переменные имеют целочисленные значения в диапазоне от 1 до 5. **б)** Байесовская сеть для двух покупателей и двух книг

К счастью, в этой сети имеется множество повторяющихся структур. Каждая переменная *Recommendation*(*c*, *b*) (отзыв) в качестве родительских имеет переменные *Honest*(*c*) (честный), *Kindness*(*c*) (добрый) и *Quality*(*b*) (качество). Более того, таблицы условных вероятностей (CPT) для всех переменных *Recommendation*(*c*, *b*) идентичны, как и таблицы CPT для всех переменных *Honest*(*c*), и т.д. Ситуация кажется специально "заточенной" под язык первого порядка. Можно было бы сказать что-то вроде

$$Recommendation(c, b) \sim RecCPT(Honest(c), Kindness(c), Quality(b)),$$

что означает, что отзыв покупателя о книге вероятностно зависит от честности и доброты покупателя и качества книги в соответствии с фиксированной таблицей условных вероятностей.

Как и логика первого порядка, модели RPM включают символы констант, функций и предикатов. Также для каждой функции мы будем использовать ▶**сигнатуру типа**, т.е. спецификацию типа каждого ее аргумента и возвращаемого функцией значения. (Если тип каждого объекта известен, с помощью этого механизма будут устранены многие *паразитные* возможные миры, например нам не нужно будет беспокоиться о доброте каждой книги, оценке покупателя книгой и т.д.) Для проблемной области оценки книг типами являются *Customer* (покупатель) и *Book* (книга), а сигнатуры типа для функций и предикатов будут следующими.

$$Honest : Customer \rightarrow \{true, false\}$$
$$Kindness : Customer \rightarrow \{1, 2, 3, 4, 5\}$$
$$Quality : Book \rightarrow \{1, 2, 3, 4, 5\}$$
$$Recommendation : Customer \times Book \rightarrow \{1, 2, 3, 4, 5\}$$

Символами констант будут любые фамилии покупателей и названия книг, присутствующие в базе данных владельца магазина. В примере, приведенном на рис. 15.2, *б*, это были C_1, C_2 и B_1, B_2.

При заданных константах и их типах, а также функциях и их сигнатурах типа ►**основные случайные переменные** в реляционных вероятностных моделях получают путем создания экземпляра каждой функции с каждой возможной комбинацией объектов. Для модели отзывов о книгах в число основных случайных переменных входят $Honest(C_1)$, $Quality(B_2)$, $Recommendation(C_1, B_2)$ и т.д. Это именно те переменные, которые показаны на рис. 15.2, *б*. Поскольку каждый тип имеет конечное число экземпляров (благодаря правилу замыкания проблемной области), общее количество основных случайных переменных также конечно.

Для завершения построения реляционной вероятностной модели осталось описать зависимости, обусловливающие эти случайные переменные. Для каждой функции существует одна формулировка зависимости, в которой каждый аргумент функции является логической переменной (т.е. переменной, которая пробегает по объектам, как в логике первого порядка). Например, следующая зависимость утверждает, что для каждого покупателя c априорным распределением вероятностей будет 0,99 для значения *true* и 0,01 — для значения *false*:

$$Honest(c) \sim \langle 0{,}99;\ 0{,}01 \rangle.$$

Аналогичным образом можно указать априорные вероятности для доброты каждого покупателя и качества каждой книги, оцениваемых по шкале от 1 до 5:

$$Kindness(c) \sim \langle 0{,}1;\ 0{,}1;\ 0{,}2;\ 0{,}3;\ 0{,}3 \rangle$$
$$Quality(b) \sim \langle 0{,}05;\ 0{,}2;\ 0{,}4;\ 0{,}2;\ 0{,}15 \rangle.$$

И наконец, осталось определить зависимость для рекомендаций: для любого покупателя c и книги b оценка зависит от честности и доброты клиента и качества книги:

$$Recommendation(c, b) \sim RecCPT(Honest(c), Kindness(c), Quality(b)),$$

где *RecCPT* — это отдельно определяемая таблица условных вероятностей, включающая $2 \times 5 \times 5 = 50$ строк, в каждой из которых по 5 элементов. В качестве иллюстрации будем полагать, что честный отзыв о книге качества q от покупателя с добротой k будет получен с вероятностью, равномерно распределенной в диапазоне $\left[\left\lfloor \frac{q+k}{2} \right\rfloor, \left\lceil \frac{q+k}{2} \right\rceil\right]$.

Семантика реляционной вероятностной модели может быть получена путем создания экземпляров этих зависимостей для всех известных констант при заданной байесовской сети (как на рис. 15.2, *б*), определяющей совместное распределение для случайных переменных модели.[3]

Множество возможных миров представляет собой декартово произведение диапазонов всех основных случайных переменных, и, как и для байесовской сети, вероятность каждого возможного мира является произведением соответствующих условных вероятностей из модели. При наличии C покупателей и B книг в модели будет C переменных *Honest*, C переменных *Kindness*, B переменных *Quality* и BC переменных *Recommendation*, что приводит к $2^C 5^{C+B+BC}$ возможным мирам. При десяти миллионах книг и миллиарде клиентов это будет примерно $10^{7\times10^{15}}$ возможных миров. Благодаря выразительной мощности RPM, полная вероятностная модель по-прежнему будет иметь лишь менее 300 параметров — большинство из них будет в таблице RecCPT.

Модель можно улучшить утверждением о наличии **контекстно специфической независимости** (см. раздел 13.2.2), отражающей тот факт, что нечестные клиенты игнорируют качество книги, давая ее оценку. Более того, в их решениях доброта также не играет никакой роли. Следовательно, переменная *Recommendation*(*c*, *b*) не зависит от переменных *Kindness*(*c*) и *Quality*(*b*), если переменная *Honest*(*c*) =*false*:

$$\begin{aligned} \mathit{Recommendation}(c, b) \sim & \textbf{if } \mathit{Honest}(c) \textbf{ then} \\ & \quad \mathit{HonestRecCPT}(\mathit{Kindness}(c), \mathit{Quality}(b)) \\ & \textbf{else } \langle 0{,}4;\ 0{,}1;\ 0{,}0;\ 0{,}1;\ 0{,}4 \rangle . \end{aligned}$$

Данный вариант описания зависимости очень похож на обычный оператор *if–then–else* в языках программирования, но между ними есть ключевое различие: механизму вероятностного вывода *необязательно будет известно точное значение результата проверки условия*, поскольку *Honest*(*c*) является случайной переменной.

Эту модель, желая сделать ее более реалистичной, можно улучшать бесконечным количеством способов. Например, можно предположить, что честный покупатель, являющийся поклонником (*Fan*) автора книги (*Author*), всегда даст любой его книге оценку ровно (*Exactly*) 5, независимо от ее качества:

$$\begin{aligned} \mathit{Recommendation}(c, b) \sim & \textbf{if } \mathit{Honest}(c) \textbf{ then} \\ & \quad \textbf{if } \mathit{Fan}(c, \mathit{Author}(b)) \textbf{ then } \mathit{Exactly}(5) \\ & \quad \textbf{else } \mathit{HonestRecCPT}(\mathit{Kindness}(c), \mathit{Quality}(b)) \\ & \textbf{else } \langle 0{,}4;\ 0{,}1;\ 0{,}0;\ 0{,}1;\ 0{,}4 \rangle . \end{aligned}$$

[3] Для правильного определения распределения вероятностей RPM должна отвечать некоторым техническим условиям. Во-первых, ее зависимости должны быть *ациклическими*, иначе полученная байесовская сеть будет включать циклы. Во-вторых, зависимости должны быть (обычно) *хорошо обоснованными*: не должно существовать бесконечных цепочек предков, которые могут возникнуть из-за рекурсивных зависимостей.

И вновь, точный результат проверки условия $Fan(c, Author(b))$ будет неизвестен, но если читатель выставляет книгам конкретного автора только 5 и не проявляет особой доброты в остальных случаях, то апостериорная вероятность того, что данный читатель является поклонником этого автора, будет весьма высока. Более того, апостериорное распределение будет проявлять тенденцию сбрасывать со счетов выставленные этим читателем 5 при оценке качества книг данного автора.

В этом примере неявно предполагалось, что значение переменной $Author(b)$ известно для каждой книги b, но это может быть и не так. Как система может рассуждать, скажем, о том, что покупатель C_1 является поклонником $Author(B_2)$, когда $Author(B_2)$ неизвестен? Ответ состоит в том, что системе, возможно, придется рассуждать обо *всех возможных авторах*. Предположим (чтобы упростить ситуацию), что существует только два автора, A_1 и A_2. Тогда $Author(B_2)$ — это случайная переменная с двумя возможными значениями, A_1 и A_2, которая будет родительской для переменной $Recommendation(C_1, B_2)$. Переменные $Fan(C_1, A_1)$ и $Fan(C_1, A_2)$ также будут ее родителями. Тогда условное распределение для переменной $Recommendation(C_1, B_2)$ по существу будет представлять собой ▶ **мультиплексор** (переключатель), в котором родительская переменная $Author(B_2)$ будет действовать как управляющий элемент, определяющий, какая из переменных, $Fan(C_1, A_1)$ или $Fan(C_1, A_2)$, действительно окажет влияние на оценку книги. Фрагмент эквивалентной байесовской сети показан на рис. 15.3. Неопределенность в значении переменной $Author(B_2)$, влияющая на зависимую от нее структуру в сети, является примером ▶ **реляционной неопределенности**.

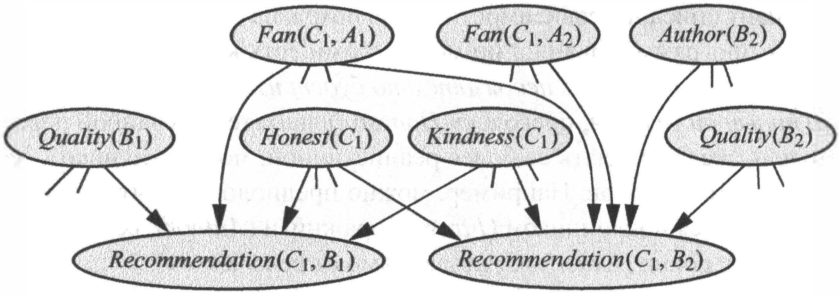


Рис. 15.3. Фрагмент эквивалентной байесовской сети для RPM задачи об оценке книг, когда автор книги B_2 неизвестен (переменная $Author(B_2)$)

В случае, если вам интересно, как система сможет установить, кто является автором книги B_2, рассмотрим ситуацию, когда три других покупателя являются поклонниками автора A_1 (и не имеют никаких других общих любимых авторов) и все они дали книге B_2 оценку 5, несмотря даже на то, что большинство других покупателей находят ее довольно мрачной. В этом случае весьма вероятно, что именно A_1

является автором книги B_2. Проведение достаточно сложных рассуждений, подобных этому, простейшей моделью RPM всего из нескольких строк — интригующий пример того, как в этой модели вероятностное влияние распространяется по сети взаимосвязей между объектами. По мере того как добавляется больше зависимостей и больше объектов, общая картина, выражаемая апостериорным распределением, часто становится все яснее и отчетливее.

15.1.2. Пример: рейтинг уровня мастерства игрока

Для многих игр, по которым проводятся соревнования, разработана числовая шкала оценки уровня мастерства игроков, которую иногда называют ▶**рейтингом**. Возможно, самым известным из них является рейтинг Эло для игроков в шахматы, согласно которому уровень типичного начинающего оценивается примерно в 800 баллов, а рейтинг чемпиона мира, как правило, чуть выше 2800. Хотя рейтинги Эло строятся на статистической основе, они обладают и некоторыми специальными элементами. Байесовскую схему рейтинга можно разработать следующим образом: каждый игрок i имеет основной уровень мастерства $Skill(i)$ и в каждой игре g он проявляет фактическую результативность $Performance(i, g)$, которая может отличаться от его основного уровня. Победителем в игре g является игрок, чья результативность в этой игре была лучше. Представленная в виде RPM, соответствующая модель будет выглядеть следующим образом.

$$Skill(i) \sim \mathcal{N}(\mu, \sigma^2)$$
$$Performance(i, g) \sim \mathcal{N}(Skill(i), \beta^2)$$
$$Win(i, j, g) = \textbf{If } Game(g, i, j) \textbf{ then } (Performance(i, g) > Performance(j, g))$$

Здесь β^2 является дисперсией фактической результативности игрока в любой конкретной игре относительно его основного уровня мастерства. При заданном множестве игроков и игр, а также результатов некоторых игр механизм вероятностного вывода RPM может вычислить распределение условных вероятностей для уровня мастерства каждого игрока и вероятного исхода любой новой игры, которая может быть проведена.

Для командных игр в первом приближении можно предположить, что общая результативность команды t в игре g представляет собой сумму индивидуальных показателей производительности игроков команды t:

$$TeamPerformance(t, g) = \sum_{i \in t} Performance(i, g).$$

Даже несмотря на то, что индивидуальные показатели результативности игроков не видны механизму определения рейтинга, уровни мастерства игроков все же можно оценить по результатам нескольких игр, если состав команды меняется от игры к игре. Система оценки Microsoft TrueSkill™ использует эту модель вместе с эффективным алгоритмом приблизительного вероятностного вывода для обслуживания сотен миллионов пользователей каждый день.

Подобную модель можно разработать различными способами. Например, можно предположить, что слабые игроки имеют более высокую дисперсию в своей результативности, можно включить в модель показатели роли игрока в команде или даже рассмотреть различные конкретные типы результативности и навыков — например, в защите и нападении, — чтобы получить возможность улучшать состав команды и повысить точность прогнозирования.

15.1.3. Вероятностный вывод в реляционных вероятностных моделях

Наиболее прямолинейный подход к выполнению вероятностного вывода в моделях RPM — это просто построить эквивалентную байесовскую сеть при заданных известных символах констант, принадлежащих к каждому типу. При наличии B книг и C покупателей определенная выше базовая модель может быть построена с помощью простых циклов, как показано ниже.[4]

for $b = 1$ **to** B *do*
  добавить узел $Quality_b$, не имеющий родителей, с априорными вероятностями $\langle 0{,}05; 0{,}2; 0{,}4; 0{,}2; 0{,}15\rangle$
for $c = 1$ **to** C *do*
  добавить узел $Honest_c$, не имеющий родителей, с априорными вероятностями $\langle 0{,}99; 0{,}01\rangle$
  добавить узел $Kindness_c$, не имеющий родителей, с априорными вероятностями $\langle 0{,}1; 0{,}1; 0{,}2; 0{,}3; 0{,}3\rangle$
  for $b = 1$ **to** B *do*
    добавить узел $Recommendation_{c,b}$ с родителями $Honest_c$, $Kindness_c$, $Quality_b$ и условным распределением $RecCPT(Honest_c, Kindness_c, Quality_b)$

Этот метод называется ►**обоснованием** или ►**развертыванием** и является точным аналогом метода **пропозиционализации** в логике первого порядка (раздел 9.1.1). Очевидным недостатком в этом случае является то, что результирующая байесовская сеть может оказаться очень большой. Кроме того, если имеется много кандидатов объектов с неизвестным отношением или функцией — например, неизвестный автор B_2, — то некоторые переменные в сети могут иметь много родителей.

К счастью, часто можно избежать генерации всей неявной байесовской сети. Как было показано при обсуждении алгоритма устранения переменной (раздел 13.3.2), каждая переменная, которая не является предком какой-либо переменной запроса или переменной свидетельства, не имеет отношения к запросу. Кроме того, если запрос является условно независимым от некоторой переменной с

[4] В нескольких статистических пакетах этот код рассматривается как *определяющий* RPM, а не просто как построение байесовской сети для выполнения вывода в RPM. Однако в этом представлении упускается важная роль синтаксиса в RPM: без синтаксиса с четкой семантикой невозможно узнать структуру модели из данных.

учетом свидетельства, то эта переменная также не имеет отношения к запросу. Таким образом, последовательно проходя по модели, начиная с переменных запроса и свидетельства, можно определить множество только тех переменных, которые имеют отношение к запросу. И только эти переменные необходимо конкретизировать для создания потенциально небольшого фрагмента неявной байесовской сети. Вероятностный вывод в этом фрагменте даст тот же самый ответ, что и вывод по всей неявной байесовской сети.

Другой подход к повышению эффективности вероятностного вывода строится на наличии в развернутой байесовской сети повторяющихся подструктур. Имеется в виду, что многие из тех факторов, которые были построены в процессе устранения переменной (и аналогичных видов таблиц, построенных алгоритмами кластеризации) будут идентичны, и эффективные схемы их кеширования позволяют достичь ускорения работы систем до трех порядков для крупных сетей.

Далее, алгоритмы вероятностного вывода MCMC обладают некоторыми интересными свойствами при их применении к моделям RPM с реляционной неопределенностью. Алгоритм MCMC работает путем построения выборок полных возможных миров, поэтому в каждом состоянии реляционная структура будет полностью известна. В приведенном выше примере в каждом состоянии, сформированном алгоритмом MCMC, значение переменной $Author(B_2)$ будет определено, и поэтому другие потенциальные авторы уже не будут родительскими узлами для книги B_2. Значит, для алгоритма MCMC реляционная неопределенность не приводит к увеличению сложности сети, — вместо этого процесс MCMC включает в себя переходы, изменяющие реляционную структуру, а следовательно, и структуру зависимостей развернутой сети.

И наконец, в некоторых случаях может оказаться возможным избежать обоснования всей модели в целом. Системы доказательства теорем резолюции и логического программирования избегают пропозиционализации посредством конкретизации логических переменных только тогда, когда это необходимо, чтобы позволить логическому выводу дойти до конца. Иначе говоря, они *поднимают* процесс логического вывода выше уровня обоснования пропозициональных высказываний, что позволяет каждому поднятому этапу выполнить работу многих этапов обоснования.

Эта же идея может быть применена и при вероятностном выводе. Например, в алгоритме устранении переменной поднятый фактор может представлять все множество факторов обоснования, присваивающих вероятности случайным переменным в модели RPM, где эти случайные переменные различаются только символами констант, использованных для их построения. Детальное обсуждение этого метода выходит за рамки этой книги, но соответствующие ссылки приведены в конце главы, в разделе “Библиографические и исторические заметки”.

15.2. Вероятностные модели с открытой вселенной

Выше утверждалось, что семантика базы данных является подходящей для ситуаций, в которых нам точно известно множество существующих релевантных объектов и мы можем идентифицировать их однозначно. (В частности, все наблюдения в отношении объекта правильно ассоциируются с символом константы, который его именует.) Однако во многих ситуациях реального мира эти предположения оказываются несостоятельными. Например, в розничной книготорговле код ISBN (*International Standard Book Number*) может использоваться как символ константы для именования каждой из книг, даже при тех условиях, что данная "логическая" книга (например, "Унесенные ветром" Маргарет Митчелл) может иметь несколько кодов ISBN, соответствующих изданиям в твердом переплете, в мягкой обложке, с крупным шрифтом, переизданиям и т.д. Имеет смысл объединить отзывы по всем таким кодам ISBN для одной и той же книги, но розничный книготорговец не может знать наверняка, какие именно коды ISBN относятся к разным вариантам ее издания. (Обратите внимание, что здесь речь не идет об *отдельных экземплярах* книги, что может потребоваться в случае продажи подержанных книг, продажи автомобилей и т.д.) Еще хуже то, что каждый покупатель идентифицируется по его логину — идентификатору входа в систему, но нечестный покупатель может завести себе тысячу подобных логинов! В области компьютерной безопасности эти множественные идентификаторы называются ▶**сивиллами** (*sybils*), а их использование с целью запутать систему репутации называется ▶**атакой Сивиллы**.[5] В результате даже простое приложение в виде относительно четко определенной интерактивной проблемной области включает как ▶**неопределенность существования** (чем являются реальные книги и покупатели, выступающие как источник наблюдаемых данных), так и ▶**неопределенности идентичности** (какие логические термы действительно ссылаются на один и тот же объект).

Феномены неопределенности существования и неопределенности идентичности выходят далеко за рамки проблемной области интернет-книготорговли. На самом деле они почти всеохватывающи.

- Любая видеосистема не знает, что находится (и вообще что-то есть) за ближайшим углом, и не может знать, является ли объект, который она видит в настоящее время, тем же самым, который она видела несколько минут назад.
- Система распознавания текста не знает заранее о тех сущностях, которые будут представлены в тексте, и должна выполнять рассуждения о том, являются ли такие слова, как "Мэри", "д-р Смит", "она", "его кардиолог" и "его мать", относящимися к одному и тому же объекту.
- Аналитик разведки, охотящийся на шпионов, никогда не знает, сколько на самом деле есть шпионов, и может только догадываться, относятся ли

[5] Название "Сивилла" происходит от известного описанного в литературе случая множественного расстройства личности.

разные псевдонимы, номера телефонов и визуальные наблюдения к одному и тому же человеку.

И действительно, большая часть человеческого познания, кажется, построена на том, что объекты существуют и есть возможность связать наблюдения — которые почти никогда не приходят с прикрепленными к ним уникальными идентификаторами — с гипотетическими объектами в мире.

Таким образом, мы должны иметь возможность определить ▶**вероятностную модель с открытой вселенной** (*Open Universe Probability Model* — ▶**OUPM**) на основе стандартной семантики логики первого порядка, как показано на рис. 15.1, *а*. Язык для моделей OUPM предоставляет возможность легко описывать такие модели, гарантируя при этом уникальное, совместимое распределение вероятностей по бесконечному пространству возможных миров.

15.2.1. Синтаксис и семантика

Основная идея состоит в том, чтобы понять, как обычную байесовскую сеть и модель RPM реализовать так, чтобы они определяли уникальную вероятностную модель, а затем передать это представление в окружение первого порядка. По сути, байесовская сеть *генерирует* каждый возможный мир, событие за событием, в топологическом порядке, определяемом структурой сети, где каждое событие представляет собой присваивание значения переменной. Модель RPM расширяет эту схему до целых множеств событий, определяемых возможными конкретизациями логических переменных в заданном предикате или функции. Модели OUPM идут еще дальше, допуская этапы порождения, на которых к возможному создаваемому миру *добавляются объекты*, количество и тип которых могут зависеть от объектов, которые уже находятся в этом мире, а также от их свойств и отношений. Иначе говоря, генерируемым событием является не присвоение значения переменной, но само *существование* объектов.

Одним из способов сделать это в модели OUPM является использование специального ▶**оператора #** (*number statement*), определяющего условное распределение по некоторому числу объектов различного типа. Например, в проблемной области отзывов о книгах может потребоваться отличать *покупателей* (реальные люди) от их *логинов* — идентификаторов для входа в систему. (На самом деле отзывы в системе связаны именно с логинами, а не с покупателями!) Предположим (для упрощения), что число покупателей представлено случайной величиной от 1 до 3, а количество книг — случайной величиной от 2 до 4 (обе эти величины с равномерным распределением):

$$\begin{aligned} {}^{\#}Customer &\sim UniformInt(1, 3) \\ {}^{\#}Book &\sim UniformInt(2, 4). \end{aligned} \tag{15.2}$$

Также ожидается, что у честного покупателя будет только один личный идентификатор для входа в систему, а у нечестных покупателей идентификаторов может быть от 2 до 5:

$$^{\#}LoginID(Owner = c) \sim \textbf{if } Honest(c) \textit{ then } Exactly(1) \textbf{ else } UniformInt(2, 5). \tag{15.3}$$

Этот оператор # определяет распределение по количеству идентификаторов входа в систему, для которых покупатель *c* является владельцем (*Owner*). Функция *Owner* называется ►**функцией источника**, поскольку она указывает, откуда пришел каждый объект, сгенерированный данным оператором #.

В примере из предыдущего абзаца используется равномерное распределение по целым числам от 2 до 5 с целью указать, сколько может быть логинов у нечестного покупателя. Это конкретное распределение ограничено, но в общем случае априорного ограничения на количество объектов может и не быть. В качестве распределения по неотрицательным целым числам чаще всего используется ►**распределение Пуассона**. Это распределение имеет один параметр λ, указывающий ожидаемое количество объектов, а случайная переменная X, подчиняющаяся распределению Пуассона с параметром λ, будет иметь следующее распределение:

$$P(X = k) = \lambda^k e^{-\lambda}/k!.$$

Дисперсия для распределения Пуассона также равна λ, так что стандартное отклонение $\sigma = \sqrt{\lambda}$. Это означает, что для больших значений λ данное распределение является узким по сравнению со значением среднего μ; например, если количество муравьев в муравейнике моделируется по распределению Пуассона со средним значением один миллион, то стандартное отклонение составит всего тысячу или 0,1%. Для больших чисел часто имеет больше смысла использовать ►**дискретное логнормальное распределение**, которое является наиболее подходящим, когда логарифм количества объектов имеет нормальное распределение. Особенно интуитивно понятна форма, которую называют ►**распределением по порядку величины**, — в ней используются десятичные логарифмы, поэтому распределение OM(3, 1) будет иметь среднее значение 10^3 и стандартное отклонение в один порядок по размерности, т.е. основная часть всей массы вероятностей попадает между 10^2 и 10^4.

Формальные семантики моделей OUPM начинаются с определения объектов, населяющих возможные миры. В стандартной семантике типизированной логики первого порядка объекты являются просто нумерованными токенами с типом. В моделях OUPM каждый объект представлен историей его генерации; например, объект может быть "четвертым идентификатором входа седьмого покупателя". (Причина такого слегка причудливого построения вскоре станет ясна.) Для типов без функций происхождения — например, типов *Customer* и *Book*

в уравнении (15.2) — созданные объекты будут иметь пустое происхождение. Например, ⟨*Customer*, , 2⟩ относится ко второму покупателю, сгенерированному по соответствующему оператору #. Для операторов # с функциями происхождения — например, как в уравнении (15.3) — в каждый созданный объект записывается его происхождение; так, объект ⟨*LoginID*, ⟨*Owner*, ⟨*Customer*, , 2⟩⟩, 3⟩ представляет собой третий идентификатор входа, принадлежащий второму покупателю.

В моделях OUPM ▶ **переменные #** определяют, сколько существует объектов каждого типа с каждым возможным происхождением в каждом возможном мире. Следовательно, запись $\#LoginID_{\langle Owner, \langle Customer, , 2 \rangle \rangle}(\omega) = 4$ означает, что в мире ω покупатель 2 имеет 4 идентификатора входа. Как и в реляционных вероятностных моделях, **основные случайные переменные** определяют значения предикатов и функций для всех кортежей объектов, поэтому $Honest_{\langle Customer, , 2 \rangle}(\omega) = true$ означает, что в мире ω клиент 2 честен. Каждый возможный мир определяется значениями всех переменных # и основных случайных переменных. Мир может быть сгенерирован на основании модели посредством выборки в топологическом порядке. На рис. 15.4 показан соответствующий пример. Вероятность мира, построенного таким образом, является произведением вероятностей для всех выбранных значений, в данном случае — $1{,}2672 \times 10^{-11}$. Теперь становится понятно, почему каждый объект содержит историю своего происхождения: эта особенность обеспечивает, что каждый мир можно будет построить в точности по одной последовательности генерации. Если бы это было не так, вероятность мира представляла бы собой громоздкую комбинаторную сумму по всем возможным последовательностям генерации, по которым его создавали.

Вероятностные модели с открытой вселенной могут иметь бесконечно много случайных переменных, поэтому в полную теорию включены нетривиальные теоретико-мерные соображения. Например, операторы # с распределением Пуассона или с распределением по порядку величины допускают неограниченное количество объектов, что ведет к неограниченному количеству случайных переменных для свойств и отношений между этими объектами. Более того, модели OUPM могут иметь рекурсивные зависимости и бесконечные типы (целые числа, строки и т.д.). Наконец, требование хорошей сформированности модели запрещает циклические зависимости и бесконечно удлиняющиеся цепочки предшественников, поскольку в общем случае наличие этих условий делает задачу неразрешимой, но некоторые синтаксически достаточные условия, тем не менее, могут быть легко проверены.

Переменная	Значение	Вероятность
$\#Customer$	2	0,3333
$\#Book$	3	0,3333
$Honest_{\langle Customer,,1\rangle}$	*true*	0,99
$Honest_{\langle Customer,,2\rangle}$	*false*	0,01
$Kindness_{\langle Customer,,1\rangle}$	4	0,3
$Kindness_{\langle Customer,,2\rangle}$	1	0,1
$Quality_{\langle Book,,1\rangle}$	1	0,05
$Quality_{\langle Book,,2\rangle}$	3	0,4
$Quality_{\langle Book,,3\rangle}$	5	0,15
$\#LoginID_{\langle \text{Owner},\langle Customer,,1\rangle\rangle}$	1	1,0
$\#LoginID_{\langle \text{Owner},\langle Customer,,2\rangle\rangle}$	2	0,25
$Recommendation_{\langle \text{LoginID},\langle \text{Owner},\langle Customer,,1\rangle\rangle,1\rangle,\langle Book,,1\rangle\rangle}$	2	0,5
$Recommendation_{\langle \text{LoginID},\langle \text{Owner},\langle Customer,,1\rangle\rangle,1\rangle,\langle Book,,2\rangle\rangle}$	4	0,5
$Recommendation_{\langle \text{LoginID},\langle \text{Owner},\langle Customer,,1\rangle\rangle,1\rangle,\langle Book,,3\rangle\rangle}$	5	0,5
$Recommendation_{\langle \text{LoginID},\langle \text{Owner},\langle Customer,,2\rangle\rangle,1\rangle,\langle Book,,1\rangle\rangle}$	5	0,4
$Recommendation_{\langle \text{LoginID},\langle \text{Owner},\langle Customer,,2\rangle\rangle,1\rangle,\langle Book,,2\rangle\rangle}$	5	0,4
$Recommendation_{\langle \text{LoginID},\langle \text{Owner},\langle Customer,,2\rangle\rangle,1\rangle,\langle Book,,3\rangle\rangle}$	1	0,4
$Recommendation_{\langle \text{LoginID},\langle \text{Owner},\langle Customer,,2\rangle\rangle,2\rangle,\langle Book,,1\rangle\rangle}$	5	0,4
$Recommendation_{\langle \text{LoginID},\langle \text{Owner},\langle Customer,,2\rangle\rangle,2\rangle,\langle Book,,2\rangle\rangle}$	5	0,4
$Recommendation_{\langle \text{LoginID},\langle \text{Owner},\langle Customer,,2\rangle\rangle,1\rangle,\langle Book,,3\rangle\rangle}$	1	0,4

Рис. 15.4. Один конкретный мир в модели OUPM задачи об отзывах о книгах. Переменные # и основные случайные переменные приведены в топологическом порядке вместе с выбранными для них значениями и вероятностями для этих значений

15.2.2. Вероятностный вывод в моделях с открытой вселенной

Из-за потенциально огромного, а иногда и неограниченного размера неявной байесовской сети, соответствующей типичной модели OUPM, полное ее развертывание с последующим выполнением точного вывода будет весьма непрактичным. Вместо этого следует рассмотреть приближенные алгоритмы вывода, такие как MCMC (см. раздел 13.4.2).

Грубо говоря, для моделей OUPM алгоритм MCMC будет исследовать пространство возможных миров, определенных с помощью множеств объектов и отношений между ними, как показано на рис. 15.1, *а*. В этом пространстве переход между соседними состояниями не может лишь изменять отношения и функции, но также должен добавлять или удалять объекты и изменять интерпретацию символов констант. Даже если каждый возможный мир может быть сколь угодно огромным, вероятностные вычисления, необходимые на каждом этапе — будь то выборка Гиббса или алгоритм Метрополиса–Гастингса, — будут полностью локальными

и в большинстве случаев потребуют лишь постоянного времени. Так происходит потому, что вероятностные соотношения между соседними мирами зависят от подграфа постоянного размера, охватывающего только те переменные, значения которых изменяются. Более того, логический запрос в каждом посещаемом мире может оцениваться *инкрементно*, обычно за постоянное время для каждого мира, вместо того чтобы пересчитываться с нуля.

Некоторые особые соображения следует привести в отношении того факта, что типичная модель OUPM может иметь возможные миры бесконечного размера. В качестве примера рассмотрим модель многоцелевого слежения за самолетами (ее код приведен ниже, на рис. 15.9: функция $X(a, t)$, обозначающая состояние самолета a в момент времени t, соответствует бесконечной последовательности переменных для неограниченного числа воздушных судов на каждом этапе. По этой причине алгоритм MCMC для формирования выборок моделей OUPM определяет возможные миры не полностью, а вместо этого использует *частичные* миры, каждый из которых соответствует непересекающемуся множеству полных миров. Частичный мир является *минимальной независимой конкретизацией*[6] подмножества *релевантных* переменных, т.е. включает предков переменных свидетельства и запроса. Например, переменные $X(a, t)$ для значений t, больших, чем время последнего наблюдения (или время запроса, в зависимости от того, что больше), будут несущественны, поэтому алгоритм может рассматривать только конечный участок бесконечной последовательности.

15.2.3. Примеры

Стандартный "вариант использования" моделей OUPM включает три элемента: собственно *модель*, *свидетельство* (известные факты в рассматриваемом сценарии) и *запрос*, который может быть представлен любым выражением, возможно, со свободными логическими переменными. Ответом является совместное условное распределение для каждого возможного множества подстановок для свободных переменных при заданном свидетельстве согласно модели.[7] Каждая модель включает в себя объявления типов, сигнатуры типов для предикатов и функций, один или более операторов # для каждого типа и один оператор зависимости для каждого предиката и функции. (В приведенных ниже примерах объявления и сигнатуры будут опущены там, где их смысл будет очевиден.) Как и в моделях RPM, в операторах зависимости используется синтаксис `if-then-else` для обработки зависимостей, зависящих от контекста.

---

[6] Независимая (*self-supporting*) конкретизация множества переменных — это такая конкретизация, в которой родители каждой переменной в множестве также находятся в этом множестве.

[7] Как и в случае языка Prolog, может существовать бесконечно много множеств подстановок неограниченного размера. Разработка исследовательских интерфейсов для представления таких ответов — интересная задача из области визуализации.

Соответствие ссылок на литературу

В Интернете можно найти миллионы технических отчетов и статей о научных исследованиях, представленных в виде файлов PDF. В конце таких документов, как правило, присутствует раздел под названием "Литература" (*References*) или "Библиография" (*Bibliography*), содержащий некоторое количество ссылок — символьных строк определенного формата, — предоставляющих читателю информацию о работах, имеющих отношение к данной теме. Эти строки могут быть найдены и извлечены из PDF-файлов с целью создания некоторого представления, своего рода базы данных, предназначенного для установления связей между статьями и исследователями по авторству и предоставляемым ссылкам. Подобное представление своим пользователям предоставляют, например, системы CiteSeer и Google Scholar, — их внутренние алгоритмы обеспечивают поиск документов, извлечение из них ссылок и идентификацию тех реальных документов, на которые эти ссылки указывают. Все это является достаточно трудной задачей, потому что ссылки не содержат каких-либо идентификаторов объектов и могут содержать любые ошибки — синтаксические и орфографические, в отношении пунктуации и содержания. Чтобы наглядно проиллюстрировать это утверждение, ниже приведен относительно простой пример: два реальных варианта ссылки на один и тот же документ.

1. [Lashkari et al 94] Collaborative Interface Agents, Yezdi Lashkari, Max Metral, and Pattie Maes, Proceedings of the Twelfth National Conference on Articial Intelligence, MIT Press, Cambridge, MA, 1994.
   [Лашкари и др. 94] Агенты с совместным интерфейсом, Йезди Лашкари, Макс Метрал и Патти Мэйс, Материалы двенадцатой Национальной конференции по искусственному интеллекту, MIT Press, Кембридж, Массачусетс, 1994.
2. Metral M. Lashkari, Y. and P. Maes. Collaborative interface agents. In Conference of the American Association for Artificial Intelligence, Seattle, WA, August 1994.
   Метрал М., Лашкари Й. и П. Мэйс. Агенты с совместным интерфейсом. На конференции Американской ассоциации искусственного интеллекта, Сиэтл, штат Вашингтон, август 1994.

Ключевой вопрос касается идентичности: это ссылки на одну и ту же или на разные статьи? Отвечая на этот вопрос, даже эксперты расходятся во мнениях или вообще отказываются принимать решение, утверждая, что рассуждения в условиях полной неопределенности неизбежно будут существенной частью решения этой проблемы.[8] Специальные подходы — такие, как методы, основанные на измерении текстуального сходства — часто с треском проваливаются в подобных ситуациях. Например, в 2002 году система CiteSeer сообщала о более чем 120 различных книгах, совместно написанных Расселом и Норвигом.

[8] Ответ на этот вопрос — да, это один и тот же документ. "Национальная конференция по искусственному интеллекту" — это еще одно название конференции AAAI. Двенадцатая конференция проходила в Сиэтле, тогда как ее материалы были изданы в Кембридже.

Чтобы решить эту задачу с использованием вероятностного подхода, для данной проблемной области нужно определить порождающую модель. Иначе говоря, нас интересует, как такие строки ссылок появляются в мире. Процесс их появления начинается с исследователей, которые имеют собственные имена. (Нам не нужно беспокоиться о том, как появились сами исследователи; нам достаточно выразить свою неуверенность в том, сколько их существует.) Эти исследователи пишут некоторые статьи, которые имеют названия; далее другие люди ссылаются на эти статьи, объединяя имена авторов и названия статей (с ошибками) в текст ссылки в соответствии с некоторой грамматикой. Основные элементы этой модели показаны на рис. 15.5, она охватывает случай, когда у статьи есть только один автор.[9]

При заданных только строках ссылок как свидетельстве вероятностный вывод в этой модели с целью получения наиболее вероятного объяснения данных приводит к частоте ошибок, которая в 2–3 раза ниже, чем на сайте CiteSeer (Пасула и др. [1741], 2003). Процесс вывода также проявляет форму коллективного, основанного на знаниях устранения двусмысленности: чем больше ссылок на определенную статью, тем точнее анализируется каждая из них, поскольку в ходе анализа приходится согласовывать факты о статье.

---

type Researcher, Paper, Citation
random String Name(Researcher)
random String Title(Paper)
random Paper PubCited(Citation)
random String Text(Citation)
random Boolean Professor(Researcher)
origin Researcher Author(Paper)

#Researcher ~ *OM*(3, 1)
Name(*r*) ~ *NamePrior*()
Professor(*r*) ~ *Boolean*(0,2)
#Paper(*Author* = *r*) ~ **if** *Professor*(*r*) **then** *OM*(1,5; 0,5) **else** *OM*(1; 0,5)
Title(*p*) ~ *PaperTitlePrior*()
CitedPaper(*c*) ~ *UniformChoice*({*Paper p*})
Text(*c*) ~ *HMMGrammar*(*Name*(*Author*(*CitedPaper*(*c*))), *Title*(*CitedPaper*(*c*)))

Рис. 15.5. Модель OUPM для извлечения информации о ссылках на литературные источники. Для упрощения в модели предполагается, что у каждой статьи только один автор, и опускаются детали моделей грамматики и обработки ошибок

---

[9] Случай с несколькими авторами будет иметь такую же общую структуру, но немного сложнее. Части модели, которые не показаны — *NamePrior*, *rTitlePrior* и *HMMGrammar*, — являются традиционными вероятностными моделями. Например, *NamePrior* представляет собой комбинацию из категориального распределения по действительным именам и модели буквенных триграмм (см. раздел 23.1) для обработки имен, которые ранее не встречались; обе были обучены на базе данных переписи населения США.

Мониторинг испытаний ядерного оружия

Проверка соблюдения договора о полном запрете испытаний ядерного оружия требует выявления всех происходящих на нашей планете сейсмических событий с магнитудой, которая выше установленного минимума. Соответствующая комиссия ООН развернула сеть датчиков — Международную систему мониторинга (*International Monitoring System* — IMS), при этом программное обеспечение для автоматической обработки поступающих данных, разработанное на основании результатов 100-летних исследований в области сейсмологии, характеризуется уровнем ошибок обнаружения около 30%. Система NET-VISA (Арора и др. [75], 2013), построенная на базе модели OUPM, позволяет значительно сократить количество ошибок.

В модели NET-VISA (рис. 15.6) соответствующая геофизическая модель выражена напрямую. Она описывает распределение по числу событий в заданный интервал времени (большинство из которых имеют естественное происхождение), также как и по их продолжительности, магнитуде, глубине и местоположению. Местоположения естественных событий характеризуются распределением в соответствии с пространственной априорной вероятностью, которая была обучена (как и другие элементы модели) на основании исторических данных. Техногенные события по правилам договора предполагаются равномерно распределенными по поверхности Земли. На каждой станции s каждая фаза p (тип сейсмической волны) в событии e дает либо 0, либо 1 обнаружений (сигналы выше порога). Вероятность обнаружения зависит от магнитуды и глубины события, а также от его расстояния от станции. Обнаружение "ложной тревоги" также происходит в соответствии с параметром скорости, установленным для конкретной станции. Измеренное время прибытия, амплитуда и другие свойства обнаружения d для реального события зависят от свойств инициирующего события и его удаления от станции.

После обучения модель работает непрерывно. Свидетельства состоят из обнаружений (90% из которых являются ложными тревогами), извлеченных из необработанных данных IMS о форме сигнала, и запрос обычно запрашивает наиболее вероятную историю событий, или *бюллетень*, с учетом имеющихся данных. Результаты пока обнадеживают; например, в 2009 году в автоматическом бюллетене SEL3 ООН было пропущено 27,4% из 27 294 событий в диапазоне магнитуд 3–4, тогда как система NET-VISA пропустила только 11,1%. Более того, сравнение с плотными региональными сетями показывает, что система NET-VISA обнаруживает на 50% больше реальных событий, чем приводится в заключительных бюллетенях, подготовленных экспертами ООН по сейсмоанализу. Система NET-VISA также имеет тенденцию связывать больше обнаружений с заданным событием, что приводит к более точным оценкам местоположения (рис. 15.7). С 1 января 2018 года система NET-VISA была развернута как часть системы мониторинга за соблюдением договора о запрете ядерных испытаний ООН.

$$
\begin{aligned}
&\#SeismicEvents \sim Poisson(T * \lambda_e)\\
&Time(e) \sim UniformReal(0,\ T)\\
&EarthQuake(e) \sim Boolean(0{,}999)\\
&Location(e) \sim \textbf{if } Earthquake(e) \textbf{ then } SpatialPrior() \textbf{ else } UniformEarth()\\
&Depth(e) \sim \textbf{if } Earthquake(e) \textbf{ then } UniformReal(0, 700) \textbf{ else } Exactly(0)\\
&Magnitude(e) \sim Exponential(log(10))\\
&Detected(e, p, s) \sim Logistic(weights(s, p), Magnitude(e), Depth(e), Dist(e, s))\\
&\#Detections(site = s) \sim Poisson(\mathrm{T} * \lambda_f(s))\\
&\#Detections(event = e, phase = p, station = s) = \textbf{if } Detected(e, p, s) \textbf{ then } 1 \textbf{ else } 0\\
&OnsetTime(a, s) \textbf{ if } (event(a) = null) \textbf{ then} \sim UniformReal(0, T)\\
&\quad \textbf{else} = Time(event(a)) + GeoTT(Dist(event, \mathrm{s}), Depth(event(a)), phase(a))\\
&\qquad\qquad + Laplace(\mu_t(s), \sigma_t(s))\\
&Amplitude(a, s) \textbf{ if } (event(a) = null) \textbf{ then} \sim NoiseAmpModel(s)\\
&\quad \textbf{else} = AmpModel(Magnitude(event(a)), Dist(event(a), \mathrm{s}), Depth(event(a)),\\
&\qquad\qquad phase(a))\\
&Azimuth(a, s) \textbf{ if } (event(a) = null) \textbf{ then} \sim UniformReal(0, 360)\\
&\quad \textbf{else} = GeoAzimuth(Location(event(a)), Depth(event(a)), phase(a), Site(\mathrm{s}))\\
&\qquad\qquad + Laplace(0, \sigma_a(s))\\
&Slowness(a, s) \textbf{ if } (event(a) = null) \textbf{ then} \sim UniformReal(0, 20)\\
&\quad \textbf{else} = GeoSlowness(Location(event(a)), Depth(event(a)), phase(a), Site(\mathrm{s}))\\
&\qquad\qquad + Laplace(0, \sigma_s(s))\\
&ObservedPhase(a, s) \sim CategoricalPhaseModel(phase(a))
\end{aligned}
$$

Рис. 15.6. Упрощенная версия модели NET-VISA

Несмотря на поверхностные различия, два приведенных выше примера структурно схожи: есть неизвестные объекты (статьи, землетрясения), генерирующие восприятия в соответствии с некоторыми физическими процессами (предоставление ссылок, сейсмическое распространение). Эти восприятия являются неоднозначными в отношении их происхождения, но когда для нескольких восприятий предполагается, что их происхождение связано с одним и тем же неизвестным объектом, свойства этого объекта могут быть установлены более точно.

Та же структура и шаблоны рассуждений подходят и для таких областей, как удаление дубликатов в базах данных и понимание естественного языка. В одних случаях получение вывода о существовании объекта предполагает группировку восприятий в нечто более общее — процесс, напоминающий задачу кластеризации в машинном обучении. В других случаях объект может вообще не генерировать восприятий и все же обнаруживать свое существование в результатах вероятностного вывода, — как произошло, например, когда тщательные наблюдения за движением планеты Уран привели к открытию планеты Нептун. Существование ненаблюдаемых объектов следует из тех эффектов, которые они оказывают на поведение и свойства наблюдаемых объектов.

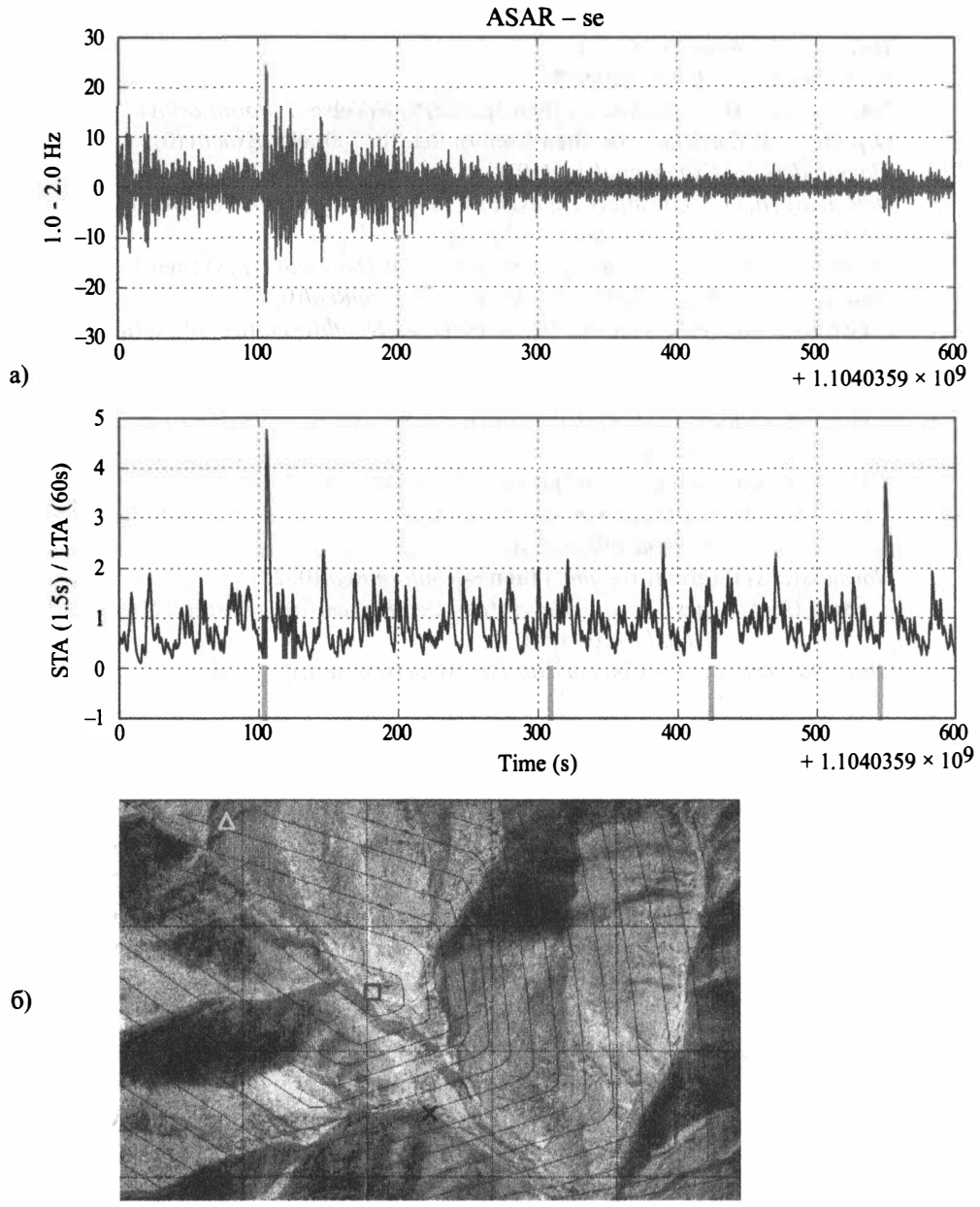


Рис. 15.7. а) Вверху: пример сейсмической волны, зарегистрированной в Алис-Спрингс, Австралия. Внизу: формы волны после обработки с целью определения времени прибытия сейсмических волн. Темные штрихи (выше) — автоматически обнаруженные моменты прибытия, светлые штрихи (ниже) — моменты истинного прибытия. **б)** Оценки местоположения ядерного испытания в КНДР от 12 февраля

2013 года: бюллетень ООН (светлый треугольник вверху слева); результаты системы NET-VISA (серый квадрат в центре). Вход в подземный испытательный полигон (черный символ "x" ниже по центру) находится на расстоянии 0,75 км от оценки системы NET-VISA. Тонкие контуры показывают условное распределение местоположения в модели NET-VISA

15.3. Отслеживание состояния сложного мира

В главе 14 проблема отслеживания состояния мира уже рассматривалась, но обсуждались только случаи атомарных представлений (модель HMM) и развернутых представлений (сети DBN и фильтры Калмана). Такой подход имеет смысл для миров с единственным наблюдаемым объектом, например с одним пациентом в палате интенсивной терапии или с одной птицей, летящей в лесу. В этом разделе мы рассмотрим, что происходит, когда наблюдения генерируют два или более объектов. То, что, собственно, отличает этот случай от обычной прежней оценки состояния, состоит в том, что теперь существует вероятность *неопределенности* в отношении того, какой из объектов какое наблюдение генерирует. Это проблема **неопределенности идентичности**, впервые упомянутая в разделе 15.2, и в этом случае она рассматривается уже во временном контексте. В литературе по теории управления ее называют проблемой ▶**ассоциации данных**, т.е. задачей связывания данных наблюдений с объектами, ответственными за их генерацию. Хотя мы можем рассматривать это как еще один пример вероятностного моделирования с открытой вселенной, на практике эта проблема достаточно важна, чтобы выделить для ее обсуждения собственный раздел.

15.3.1. Пример: многоцелевое отслеживание

Проблема ассоциации данных первоначально изучалась в контексте радиолокационного слежения за несколькими целями, когда отраженные импульсы обнаруживались через фиксированные интервалы времени с помощью вращающейся антенны радара. На каждом временном этапе на экране могли появляться сразу несколько вспышек, но прямого наблюдения за тем, какие вспышки в момент времени t соответствуют каким вспышкам в момент времени $t-1$, не было. На рис. 15.8, *а* приведен простой пример с двумя вспышками на каждом из пяти временных этапов. Каждая вспышка отмечена номером этапа, на котором он появился, но никакой дополнительной идентифицирующей информации нет.

Предположим, что на данный момент точно известно, что вспышки генерируют два самолета, A_1 и A_2. В терминологии моделей OUPM A_1 и A_2 являются ▶**гарантированными объектами** и это означает, что они гарантированно существуют и являются различными объектами. Более того, в нашем случае никаких других объектов нет. (Другими словами, в отношении самолетов этот сценарий

соответствует семантике базы данных, которая предполагается в моделях RPM.) Пусть их истинными позициями будут $X(A_1, t)$ и $X(A_2, t)$, где t является неотрицательным целым числом, индексирующим моменты обновления показаний датчика. Будем предполагать, что первое наблюдение поступает в момент времени $t = 1$, а в момент времени 0 априорным распределением для местоположения каждого самолета будет *InitX*(). Исключительно для упрощения также предположим, что каждый самолет движется независимо в соответствии с известной моделью перехода, например с линейной гауссовой, которая используется в фильтре Калмана (см. раздел 14.4).

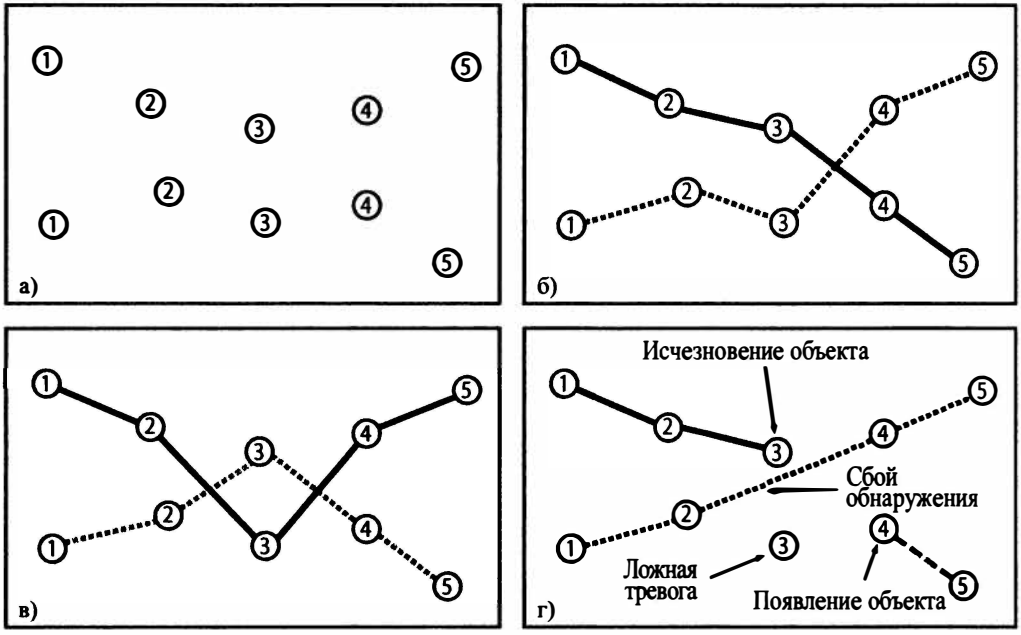


Рис. 15.8. а) Наблюдения местоположения объектов в 2D-пространстве, сделанные на протяжении пяти временных этапов. Каждое наблюдение представлено вспышкой с отметкой о временном этапе, когда оно имело место, но объект, который стал причиной его появления, никак не отмечен. **б) и с)** Возможные гипотезы о движении наблюдаемых объектов. **д)** Гипотезы для случая, в котором могут иметь место ложные тревоги, сбои обнаружения и появление/исчезновение объекта

Последним элементом является модель восприятия; опять же, примем линейную гауссову модель, когда самолет в положении **x** производит на экране радара вспышку b, причем наблюдаемая позиция этой вспышки на экране $Z(b)$ является линейной функцией от x с добавлением гауссова шума. Каждый самолет на каждом временном этапе генерирует ровно одну вспышку, поэтому вспышка в

качестве родителей имеет самолет и этап времени. Таким образом, опустив пока априорные вероятности, нашу модель можно определить следующим образом.

$$\textbf{guaranteed}\ Aircraft\ A_1, A_2$$
$$X(a, t) \sim \textbf{if}\ t = 0\ \textbf{then}\ InitX()\ \textbf{else}\ \mathcal{N}(\mathbf{F}\, X(a, t-1), \Sigma_x)$$
$$\#Blip(Source = a, Time = t) = 1$$
$$Z(b) \sim \mathcal{N}(\mathbf{H}\, X(Source(b), Time(b)), \Sigma_z)$$

Здесь $\mathbf{F}$ и Σ_x являются матрицами, описывающими линейную модель перехода и ковариацию шума перехода, а $\mathbf{H}$ и Σ_z являются соответствующими матрицами для модели восприятия (см. раздел 14.4.3).

Ключевым различием между этой моделью и стандартным фильтром Калмана является то, что здесь имеется *два* объекта, влияющие на показания датчика (вспышки). Это означает, что на любом заданном временном этапе существует *неопределенность* в отношении того, какой объект производит какое показание датчика. Каждый возможный мир в этой модели включает в себя ассоциацию — определяемую значениями всех переменных $Source(b)$ для всех временных этапов — между самолетами и вспышками. Две возможные гипотезы ассоциации показаны на рис. 15.8, *б* и *в*. В общем случае для n объектов и T временных этапов существует $(n!)^T$ способов ассоциации вспышек и самолетов — невероятно большое число.

Описываемый до этого момента сценарий включал n известных объектов, генерирующих n наблюдений на каждом временном этапе. Реальные приложения ассоциации данных, как правило, намного сложнее. Часто поступающие наблюдения включают в себя ▶ **ложные тревоги** (также известные как ▶ **помехи**), которые не были вызваны реальными объектами. Также может произойти ▶ **сбой обнаружения**, означающий, что наблюдение от реального объекта не поступило. Наконец, могут появляться новые объекты, а прежние исчезать. Все эти явления, создающие еще больше возможных миров, о которых следует побеспокоиться, проиллюстрированы на рис. 15.8, *д*. Соответствующая модель OUPM приведена на рис. 15.9.

По причине ее практической важности как для гражданских, так и для военных приложений по проблеме многоцелевого отслеживания и ассоциации данных были написаны десятки тысяч статей. Авторы многих из них просто пытались разработать сложные математические детали вероятностных расчетов для модели, приведенной на рис. 15.9 или для ее более простых версий. В каком-то смысле это не нужно, если выразить данную модель на языке вероятностного программирования, поскольку механизм вероятностного вывода общего назначения корректно выполнит любые математические расчеты для любой модели, включая и эту. Более того, уточнения сценария (формирование полета; объекты, движущиеся в неизвестном направлении; объекты, совершающие взлет или посадку, и т.д.) могут быть реализованы небольшими изменениями в модели без новых математических выводов и сложного программирования.

$\#Aircraft(EntryTime = t) \sim Poisson(\lambda_a)$
$Exits(a, t) \sim$ **if** $InFlight(a, t)$ **then** $Boolean(\alpha_e)$
$InFlight(a, t) = (t = EntryTime(a)) \vee (InFlight(a, t-1) \wedge \neg Exits(a, t-1))$
$X(a, t) \sim$ **if** $t = EntryTime(a)$ **then** $InitX()$
else if $InFlight(a, t)$ **then** $\mathcal{N}(\mathbf{F}\, X(a, t-1), \Sigma_x)$
$\#Blip(Source = a, Time = t) \sim$ **if** $InFlight(a, t)$ **then** $Bernoulli(DetectionProb(X(a, t)))$
$\#Blip(Time = t) \sim Poisson(\lambda_f)$
$Z(b) \sim$ **if** $Source(b) = null$ **then** $UniformZ(R)$ **else** $\mathcal{N}(\mathbf{H}\, X(Source(b), Time(b)), \Sigma_z)$

Рис. 15.9. Модель OUPM для радиолокационного слежения за несколькими целями с обработкой ложных тревог (помех), сбоев обнаружения, а также появления и исчезновения самолетов. Частота, с которой новые самолеты появляются в сцене, есть λ_a, тогда как вероятность на этап времени, что самолет покинет сцену, есть α_e. Ложные тревоги (т.е. вспышки, не связанные с самолетами) появляются в пространстве с равномерным распределением с нормой λ_f на этап времени. Вероятность того, что самолет будет обнаружен (т.е. произведет вспышку), зависит от его текущего местоположения

С практической точки зрения основная проблема таких моделей — сложность вероятностного вывода. Как и для всех вероятностных моделей, вероятностный логический вывод означает устранение суммированием всех переменных, отличных от переменных запроса и свидетельства. При фильтрации в моделях HMM и сетях DBN есть возможность устранить суммированием переменные состояния от 1 до $t-1$ с помощью простого приема динамического программирования. Для фильтров Калмана также есть возможность воспользоваться особыми свойствами гауссианов. Однако в случае ассоциации данных ситуация сложнее. Здесь нет (известных) эффективных точных алгоритмов, — по той же причине, по которой их нет для переключающих фильтров Калмана (см. раздел 14.4.4): фильтрация распределения, описывающего совместное распределение по номерам и расположениям самолетов на экране на каждом временном этапе, заканчивается как смесь экспоненциально большого множества распределений, по одному для каждого варианта выбора последовательности наблюдений, назначаемого каждому самолету.

В ответ на сложность точного вывода уже использовалось несколько приближенных методов. Самый простой подход состоит в том, чтобы на каждом временном этапе выбрать одно "лучшее" присваивание при известных прогнозируемых положениях объектов на текущий момент. Это присваивание связывает наблюдения с объектами и позволяет обновить путь каждого объекта и сделать предсказание на следующий временной этап. Для выбора "лучшего" присваивания чаще всего используется метод так называемого ▶**фильтра ближайшего соседа** (*nearest-neighbor filter*), в котором раз за разом выбирается ближайшая пара из

предсказанного положения и наблюдения, которая и добавляется в присваивание. Фильтр ближайшего соседа успешно работает, когда объекты достаточно хорошо разделены в пространстве состояний, а неопределенность прогноза и ошибка наблюдения невелики, — другими словами, когда нет возможности возникновения путаницы.

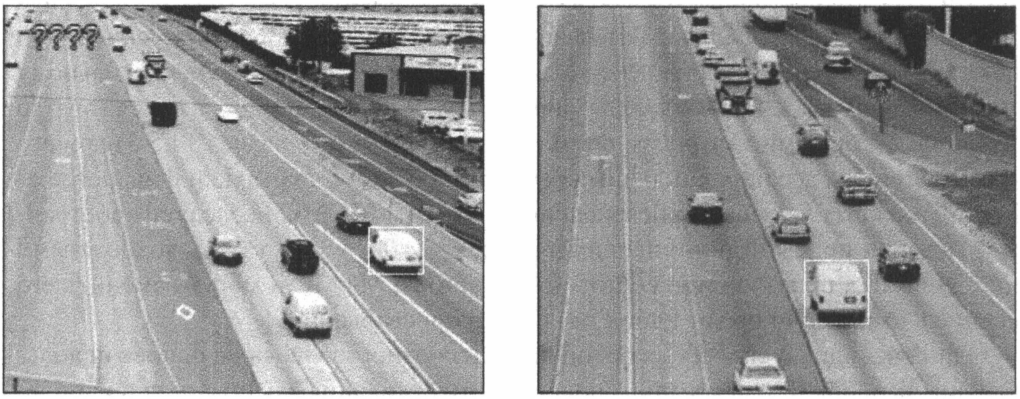
Когда имеет место значительная неопределенность в отношении правильного присваивания, лучшим подходом будет выбор того присваивания, которое максимизирует совместную вероятность текущих наблюдений при заданных предсказанных позициях. Эту задачу можно эффективно решить с помощью ▶**венгерского алгоритма** (Кун [1320], 1955), даже несмотря на то, что на каждом новом временном этапе потребуется делать выбор из $n!$ возможных присваиваний.

Любой метод, фиксирующий одно наилучшее назначение на каждом временном этапе, с треском проваливается в более сложных условиях. В частности, если алгоритм фиксирует неправильное назначение, прогноз на следующий временной этап может оказаться существенно неверным, что приведет к еще более неправильному присваиванию, и т.д. Подходы с формированием выборок могут оказаться гораздо более эффективными. Алгоритм **фильтрации частиц** (см. раздел 14.5.3) при его применении для ассоциации данных работает способом поддержки большого набора возможных текущих присваиваний. Алгоритм **MCMC** исследует пространство истории присваиваний; например, рис. 15.8, *б* и *в* могут быть возможными состояниями в пространстве состояний алгоритма MCMC и могут изменить его мнение в отношении предыдущих решений о присваивании.

Один очевидный способ ускорения вероятностного вывода на основе использования выборок в задачах многоцелевого отслеживания заключается в использовании приема **Рао-Блэквеллизации**, обсуждавшегося в разделе 14.5.3. При заданной конкретной гипотезе ассоциации для всех объектов расчеты фильтрации для каждого объекта, как правило, можно выполнить точнее и эффективнее в сравнении с формированием выборок для многих возможных последовательностей состояний для объектов. Например, для модели, представленной на рис. 15.9, вычисление фильтрации означает просто запуск фильтра Калмана для последовательности наблюдений, присвоенной заданному предполагаемому объекту. Более того, при переходе от одной гипотезы ассоциации к другой, заново провести расчеты потребуется только для тех объектов, у которых изменились ассоциированные с ними наблюдения. Современные методы ассоциации данных по алгоритму MCMC способны обрабатывать многие сотни объектов в режиме реального времени, давая при этом хорошее приближение к истинным апостериорным распределениям.

15.3.2. Пример: мониторинг дорожного движения

На рис. 15.10 приведены два изображения, полученные с двух камер видеонаблюдения на автостраде в Калифорнии, расположенных далеко друг от друга. В этом приложении нас интересуют два вопроса: оценка времени, необходимого в текущих условиях дорожного движения для перемещения из одного места в другое по системе автострад, и измерение *спроса*, т.е. как много транспортных средств проезжает между любыми двумя точками в системе в определенное время суток и в определенные дни недели. Обе цели требуют решения задачи ассоциации данных на обширной территории с большим количеством видеокамер и десятками тысяч транспортных средств, наблюдаемых за один час.



а) б)

Рис. 15.10. Изображения с камер видеонаблюдения на въезде (**а**) и выезде (**б**) с определенного участка длиной около двух миль на шоссе 99 в Сакраменто, штат Калифорния. Автомобиль, взятый в рамку, был идентифицирован на изображениях с обеих камер

При визуальном наблюдении ложные тревоги (помехи) могут быть вызваны движущимися тенями, автомобилями с прицепом, отражением в лужах и так далее, а сбои обнаружения могут возникнуть из-за тумана, темноты, отсутствия визуального контраста или в тех случаях, когда один автомобиль заслоняет другой. Транспортные средства постоянно въезжают на автостраду и покидают ее в различных точках, которые могут не контролироваться. Кроме того, внешний вид любого конкретного транспортного средства может резко различаться для разных камер (в зависимости от условий освещения и ракурса этого транспортного средства на изображении), а модель перехода может изменяться при возникновении пробки или ее ликвидации. Наконец, в плотном транспортном потоке при достаточно далеко расположенных камерах наблюдения ошибка предсказания

в модели перехода для автомобиля, движущегося от одной камеры к другой, будет гораздо больше, чем при обычной дистанции между транспортными средствами. Несмотря на эти проблемы, современные алгоритмы ассоциации данных достигли значительных успехов в оценке параметров дорожного движения в реальных условиях.

Корректная ассоциация данных является важнейшим условием для отслеживания состояния мира, поскольку нет никакого иного способа объединения многих наблюдений любого заданного объекта. Когда объекты в мире взаимодействуют друг с другом в сложных видах деятельности, понимание мира требует объединения механизмов ассоциации данных с реляционными вероятностными моделями и вероятностными моделями с открытой вселенной, обсуждавшимися в разделе 15.2. В настоящее время это одна из наиболее активных областей исследований в ИИ.

15.4. Программы как вероятностные модели

Многие вероятностные языки программирования были построены на понимании того факта, что вероятностные модели могут быть определены с помощью выполняемого кода на любом языке программирования, в который встроен источник случайности. Для таких моделей возможные миры являются трассами выполнения, а вероятность любой такой трассы — это вероятность случайного выбора, необходимого для реализации этой трассы. Языки PPL, созданные подобным образом, наследуют всю выразительную силу языка, положенного в их основу, включая сложные структуры данных, рекурсию и, в некоторых случаях, функции более высокого порядка. Многие языки PPL фактически универсальны в вычислительном отношении: они могут представлять любое распределение вероятностей, которое может быть взято из вероятностной машины Тьюринга, которая останавливается.

15.4.1. Пример: чтение текста

Этот подход к вероятностному моделированию и вероятностному выводу мы проиллюстрируем с помощью задачи написания программы, способной считывать размытый текст. Этот тип моделей может использоваться для прочтения текста, который был смазан, запачкан, размыт водой или сильно поврежден из-за старения бумаги, на которой напечатан. Их также можно создавать для взлома некоторых видов защиты типа CAPTCHA.

На рис. 15.11 приведена генерирующая программа, содержащая два компонента: а) код генерации последовательности букв; б) подпрограмму генерации искаженного, размытого образа этих букв с использованием готовой графической библиотеки. На рис. 15.12, *а* представлены примеры изображений, полученных при двенадцати обращениях к функции GENERATE-IMAGE.

```
function GENERATE-IMAGE() returns изображение с несколькими буквами
  letters ← GENERATE-LETTERS(10)
  return RENDER-NOISY-IMAGE(letters, 32, 128)

function GENERATE-LETTERS(λ) returns вектор букв
  n ~ Poisson(λ)
  letters ← []
  for i = 1 to n do
      letters[i] ~ UniformChoice({a, b, c, ...})
  return letters

function RENDER-NOISY-IMAGE(letters, width, height) returns зашумленное
                изображение последовательности букв
  clean_image ← RENDER(letters, width, height, text_top = 10, text_left = 10)
  noisy_image ← []
  noise_variance ~ UniformReal(0,1;  1)
  for row = 1 to width do
      for col = 1 to height do
          noisy_image[row, col] ~ N(clean_image[row, col],  noise_variance)
  return noisy_image
```

Рис. 15.11. Генерирующая программа для вероятностной модели с открытой вселенной, выполняющей распознавание графических символов. Эта генерирующая программа создает размытые изображения, содержащие некоторую последовательность букв, посредством генерации самой последовательности, ее рендеринга в 2D-изображения и внесения дополнительного шума для каждого пикселя

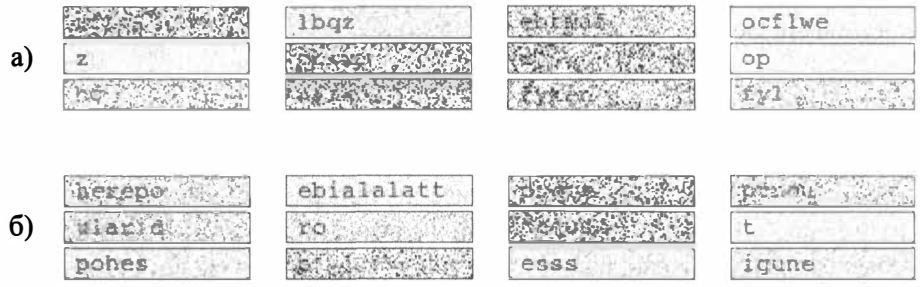


Рис. 15.12. а) Двенадцать искаженных изображений, полученных при выполнении генерирующей программы, показанной на рис. 15.11. Количество букв, их выбор, количество дополнительного шума и специфический пиксельный шум — все это элементы проблемной области вероятностной модели. **б)** Двенадцать искаженных изображений, полученных при выполнении генерирующей программы, показанной на рис. 15.15. Использование марковской модели для генерации букв обычно дает последовательности букв, которые легче произносить

15.4.2. Синтаксис и семантика

Понятие ▶**генерирующая программа** (*generative program*) относится к выполняемой программе, в которой каждый случайный выбор определяет случайную переменную в связанной с ней вероятностной модели. Давайте мысленно, шаг за шагом, проследим ход выполнения программы, осуществляющей случайный выбор. Пусть X_i — случайная переменная, соответствующая i-му случайному выбору, сделанному программой, а x_i, как обычно, будет обозначать возможное значение переменной X_i. Назовем множество $\omega = \{x_i\}$ ▶**трассой выполнения** (*execution trace*) генерирующей программы; фактически это последовательность возможных значений для случайных выборов. Однократный запуск программы приводит к генерированию одной такой трассы, отсюда и термин "генерирующая программа".

Пространство всех возможных трасс выполнения Ω можно рассматривать как выборочное пространство вероятностной модели, определенное генерирующей программой. Распределение вероятностей по трассам можно определить как произведение вероятностей каждого отдельного случайного выбора: $P(\omega) = \prod_i P(x_i \mid x_1, \ldots x_{i-1})$. Это аналогично распределению по мирам в модели OUPM.

Концептуально просто преобразовать любую модель OUPM в соответствующую генерирующую программу. Эта генерирующая программа выполняет случайный выбор для каждого оператора # и значения каждой базовой случайной переменной, существование которой подразумевается в соответствии с операторами #. Основная дополнительная работа, которую генерирующая программа также должна выполнить, заключается в создании структур данных, представляющих объекты, функции и отношения в возможных мирах модели OUPM. Эти структуры данных будут созданы автоматически механизмом вероятностного вывода модели OUPM, поскольку в модели OUPM предполагается, что каждый возможный мир является моделью структуры первого порядка, тогда как в типичном языке PPL таких предположений не делается.

Изображения на рис. 15.12 можно использовать для расширения интуитивного понимания распределения вероятностей $P(\Omega)$: на них видны различные уровни шума, а на менее зашумленных изображениях также заметны последовательности букв различной длины. Пусть ω_1 — это трасса, соответствующая изображению в верхнем правом углу на рис. 15.12, *a*, содержащему буквы `ocflwe`. Если развернуть трассу ω_1 в байесовскую сеть, она будет иметь 4104 узла: 1 узел — для переменной n, 6 узлов — для переменных *letters*[i], 1 узел — для переменной дисперсии шума *noise_variance* и 4096 узлов — для переменных, представляющих пиксели в изображении с шумом *noisy_image*. Таким образом, мы видим, что эта генерирующая программа определяет вероятностную модель с открытой вселенной: число случайных выборов, которые она делает, исходно не ограничено, в действительности оно зависит от значения случайной величины n.

15.4.3. Результаты вероятностного вывода

Давайте применим эту модель для интерпретации изображений букв, которые были ухудшены добавлением шума. На рис. 15.13 показано анализируемое изображение с добавленным шумом вместе с результатами трех независимых прогонов алгоритма MCMC. Для каждого прогона приведен рендеринг букв, содержащихся в трассе после остановки цепи Маркова. Во всех трех случаях результатом является последовательность букв `uncertainty`, что ясно указывает на то, что апостериорное распределение сильно сконцентрировано на правильной интерпретации.

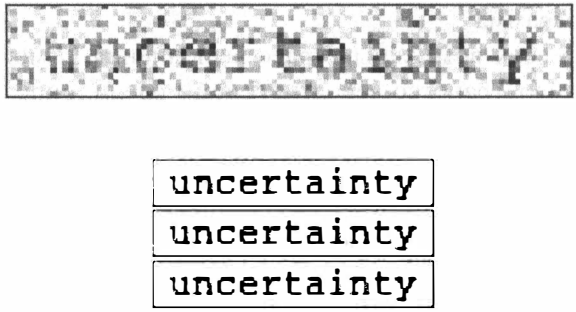


Рис. 15.13. а) Зашумленное входное изображение. **б)** Результаты вероятностного вывода, полученные с помощью трех трасс, каждая по 25 итераций алгоритма MCMC — тех же, что и в случае рис. 15.11. Обратите внимание, что процесс вероятностного вывода правильно определяет последовательность букв

Теперь давайте еще больше ухудшим текст, размыв его настолько, что людям будет очень трудно его прочитать, — это хорошо видно на рис. 15.14, *а*. На рис. 15.14, *б* показаны результаты вероятностного вывода для этого более сложного входного изображения. На этот раз, хотя вероятностный вывод в алгоритме MCMC, по-видимому, не сомневается в правильном количестве букв (как и должно быть), первая буква ошибочно идентифицируется как `q`, а также имеет место неопределенность в отношении пяти из десяти остальных букв.

На данный момент у нас есть много возможных способов интерпретации результатов. Вполне возможно, что вероятностный вывод в алгоритме MCMC хорошо смешался и полученные результаты верно отражают истинное апостериорное распределение при заданных модели и изображении. В этом случае неопределенность в некоторых буквах и ошибка в первой букве неизбежны. Чтобы получить лучшие результаты, нам может потребоваться улучшить текстовую модель или снизить уровень шума. Также может быть, что вероятностный вывод в алгоритме MCMC не смешался должным образом: если запустить 300 цепочек для 25 тысяч

или 25 млн итераций, можно будет найти совсем другое распределение результатов, возможно, указывающее, что первой буквой с большей вероятностью является u, а не q.

а)

б)

qnucitainty
qncurtbinty
qncuftaintv

в)

uncertainty
uncsitainty
unchftsinty

Рис. 15.14. а) Очень зашумленное входное изображение. **б)** Результаты трех трасс вероятностного вывода по 25 итераций алгоритма MCMC — тех же, что и в случае рис. 15.11. **в)** Результаты трех трасс вероятностного вывода улучшенной модели, использующей буквенные биграммы (см. раздел 15.4.4). Обе модели демонстрируют неоднозначность в результатах, но результаты последней модели отражают свойственное ей априорное знание о вероятных последовательностях букв

Выполнение большего вероятностного вывода может быть связано с увеличением затрат и увеличением времени ожидания. Более того, не существует надежного теста на сходимость вероятностного вывода по методу Монте-Карло. Можно было бы попытаться улучшить алгоритм вероятностного вывода, возможно, разработав лучшее вспомогательное распределение для алгоритма MCMC или используя текстовые подсказки на изображении, чтобы предложить лучшие начальные гипотезы. Подобные улучшения требуют дополнительного обдумывания, реализации и отладки. Третий вариант — улучшить модель. Например, можно было бы включить в нее определенные знания об особенностях английских слов, — скажем, таких, как вероятность появления определенных пар букв. В следующем разделе мы рассмотрим именно этот вариант.

15.4.4. Улучшение генерирующей программы путем включения в нее марковской модели

Вероятностные языки программирования являются модульными, что существенно упрощает поиск возможностей улучшения базовой модели. На рис. 15.15 показана генерирующая программа для улучшенной модели, которая генерирует буквы как единую последовательность, а не независимо друг от друга. Эта генерирующая программа использует марковскую модель, которая выбирает каждую

очередную букву с учетом предыдущей буквы с переходными вероятностями, установленными на основании опорного списка английских слов.

```
function GENERATE-MARKOV-LETTERS(λ) returns вектор букв
  n ~ Poisson(λ)
  letters ← []
  letter_probs ← MARKOV-INITIAL()
  for i = 1 to n do
      letters[i] ~ Categorical(letter_probs)
      letter_probs ← MARKOV-TRANSITION(letters[i])
  return letters
```

Рис. 15.15. Генерирующая программа для улучшенного оптического распознавания символов, генерирующая последовательность букв с использованием модели буквенных биграмм, в которой частоты появления пар букв оцениваются на основании опорного списка английских слов

На рис. 15.12, *б* представлены двенадцать образцов изображений, полученных с помощью этой генерирующей программы. Обратите внимание, что буквенные последовательности значительно больше похожи на английские слова в сравнении с теми, которые представлены на рис. 15.12, *а* и были сгенерированы программой, приведенной на рис. 15.11. На рис. 15.14, *в* показаны результаты вероятностного вывода этой модели Маркова, примененной к изображению с высоким уровнем шума. Интерпретации более точно соответствуют генерирующей трассе, хотя все еще существует некоторая неопределенность.

15.4.5. Вероятностный вывод в генерирующих программах

Как и в случае моделей OUPM, точный вывод в генерирующих программах обычно слишком затратен или просто невозможен. С другой стороны, легко увидеть, как можно выполнить выборку с отклонением: запустить программу, сохранить только те трассы, которые согласуются со свидетельством, и подсчитать различные ответы на запрос, найденные на этих трассах. Взвешивание по правдоподобию также реализуется очевидным образом: для каждой генерируемой трассы следим за ее весом путем умножения вероятностей всех значений, наблюдаемых вдоль этой трассы.

Метод взвешивания по правдоподобию работает хорошо только тогда, когда данные достаточно вероятны в соответствии с моделью. В более сложных случаях хорошим вариантом для выбора обычно является алгоритм MCMC. При применении к вероятностным программам алгоритм MCMC включает выборку и изменение трасс выполнения. Многие из соображений, сделанных в отношении моделей OУПМ, применимы и в этом случае. Кроме того, при выполнении алгоритма

следует проявлять осторожность в отношении изменения трассы выполнения; например, если изменить результат выполнения оператора **if**, вся оставшаяся часть этой трассы может стать недействительной.

Дальнейшие улучшения в вероятностном выводе возможны по нескольким направлениям работы. Некоторые улучшения могут привести к фундаментальным сдвигам в классе проблем, которые разрешимы для заданного языка PPL, даже в принципе. Такой эффект может дать, например, поднятый вероятностный вывод, описанный ранее для моделей RPM. Во многих случаях универсальный алгоритм MCMC слишком медленный, и необходимы специальные вспомогательные распределения, позволяющие быстро смешать процесс вероятностного вывода.

В последних работах в области языков PPL в центре внимания было стремление сделать проще для пользователей определение и использование таких вспомогательных распределений, благодаря которым эффективность PPL-вывода могла бы соответствовать тем пользовательским алгоритмам вывода, которые были разработаны для конкретных моделей.

Многие перспективные подходы направлены на снижение накладных расходов на вероятностный вывод. Идея компиляции, которая описывалась для байесовских сетей в разделе 13.4.3, может применяться и к вероятностному выводу в моделях OUPM и PPL, — как правило, в результате достигается ускорение расчетов на два-три порядка. Также были сделаны предложения по разработке *специализированного оборудования* для некоторых типов алгоритмов, таких как передача сообщений и MCMC. Например, в аппаратном обеспечении для методов Монте-Карло используются низкая точность вероятностных представлений и массовое распараллеливание коротких процессов, что позволяет получить выигрыш в 100–10 000 раз в скорости выполнения расчетов и энергосбережении.

Методы, основанные на обучении, также могут дать существенные улучшения в скорости. Например, ►**адаптивные вспомогательные распределения** могут постепенно научиться генерировать такие вспомогательные распределения алгоритма MCMC, которые будут приняты с достаточной степенью вероятности и будут достаточно эффективны для изучения вероятностного ландшафта модели с целью обеспечения быстрого смешивания. Также возможно обучать модели глубокого обучения (см. главу 21) для представления вспомогательных распределений при выборке по значимости, используя синтетические данные, полученные из базовой модели.

В целом можно ожидать, что любое формальное математическое представление, выстроенное поверх языков программирования общего назначения, будет способствовать преодолению барьера невычислимости, и это справедливо для языков PPL. Однако если учесть, что выполнение базовых программ неизбежно будет тормозить необходимость ввода данных наблюдений и выполнения всех случайных выборов, не сделают ли эти дополнительные требования задачу вероятностного вывода неразрешимой? Как оказалось, ответ на этот вопрос — "да", но только для вычислительных моделей с бесконечной точностью непрерывных

случайных переменных. При этих условиях оказывается возможным написание вычислимой вероятностной модели, в которой вероятностный вывод сталкивается с проблемой останова. С другой стороны, в случае чисел конечной точности и гладких вероятностных распределений, обычно используемых в реальных приложениях, вероятностный вывод остается разрешимой задачей.

Резюме

В этой главе рассматривались выразительные средства представления для вероятностных моделей, основанные как на логике, так и на компьютерных программах.

- В **реляционных вероятностных моделях** (RPM) вероятностные модели определяются на мирах, полученных с использованием **семантики базы данных** для языков первого порядка. Эти модели подходят для случая, когда все объекты и их идентичности известны с уверенностью.
- При заданной модели RPM объекты в каждом возможном мире соответствуют символам констант в модели, а базовыми случайными переменными являются все возможные экземпляры символов предикатов с объектами, заменяющими каждый аргумент. Следовательно, множество возможных миров конечно.
- Модели RPM представляют собой очень сжатые модели для миров с большим количеством объектов и позволяют справиться с неопределенностью в отношениях.
- **Вероятностные модели с открытой вселенной** (OUPM) строятся на основе полной семантики логики первого порядка, что допускает присутствие в них новых видов неопределенности, таких как неопределенность идентичности и неопределенность существования.
- **Генерирующие программы** — это представления вероятностных моделей, включая и модели OUPM, в виде выполняемых программ на **вероятностном языке программирования**, или языке **PPL**. Генерирующая программа формирует распределение по **трассам выполнения** программы. Языки PPL обычно обеспечивают *универсальную* выразительную силу для вероятностных моделей.

Библиографические и исторические заметки

В своих работах Гальперин ([945], 1984) и Хаусон ([1078], 2003) пересказывают долгую историю попыток соединения вероятности и логики, начиная с книги Лейбница "Nouveaux Essais", вышедшей в 1704 году. В этих попытках обычно предполагалось присоединение вероятностей непосредственно к логическим высказываниям. Первым строгим подходом была пропозициональная ▶**вероятностная логика** Гайфмана (Гайфман [805], 1964). Идея заключалась в том, что вероятностное утверждение $P(\phi) \geq p$ является ограничением на распределение по возможным мирам — так же, как обычное логическое высказывание является ограничением на сами возможные миры. Любое распределение P, удовлетворяющее ограничению, является моделью — в стандартном логическом смысле — вероятностного утверждения, и одно вероятностное утверждение влечет за собой другое всякий раз, когда модели первого утверждения являются подмножеством моделей второго.

В рамках такой логики можно доказать, например, что $P(\alpha \wedge \beta) \leq P(\alpha \Rightarrow \beta)$. Удовлетворяемость множеств вероятностных утверждений можно определить в пропозициональном случае с помощью линейного программирования (Гальперин [945], 1984; Нильссон [1689], 1986). Таким образом, мы имеем "логику вероятности" в том же смысле, что и "временную логику": как логическую систему, специализированную для вероятностных рассуждений.

Чтобы применить вероятностную логику к таким задачам, как доказательство интересных теорем в теории вероятности, требовался более выразительный язык. Гайфман в [804] (1964) предложил вероятностную логику *первого порядка* с возможными мирами, являвшимися модельными структурами первого порядка, и с вероятностями, прикрепленными к высказываниям (без функций) логики первого порядка. Скотт и Краусс ([2014], 1966) расширили результаты Гайфмана, разрешив бесконечную вложенность кванторов и бесконечные множества высказываний.

В области ИИ прямым следствием этих идей стало появление **программ вероятностной логики** (Лукашевич [1464], 1998), в которых диапазон вероятностей прикреплялся к каждому хорновскому выражению первого порядка, а логический вывод выполнялся путем решения задачи линейного программирования, как это было предложено Гальпериным. Ранее Гальперин ([951], 1990) и Бахус ([96], 1990) также заинтересовались подходом Гайфмана, исследуя некоторые из базовых проблем представления знаний, но уже с точки зрения ИИ, а не теории вероятностей или математической логики.

В подобласти ▶**вероятностных баз данных** также используются логические высказывания, помеченные вероятностями (Далви и др. [519], 2009), но в этом случае вероятности прикреплены непосредственно к кортежам данных в базе. (В ИИ и статистике вероятность прикрепляется к общим отношениям, тогда как наблюдения рассматриваются как неопровержимые свидетельства.) Хотя вероятностные базы данных позволяют моделировать сложные зависимости, на практике в таких

системах чаще всего можно обнаружить использование предположений о глобальной независимости кортежей.

Прикрепление вероятностей к высказываниям очень затрудняет определение полных и непротиворечивых вероятностных моделей. Каждое неравенство *ограничивает* лежащую в основе вероятностную модель, заключая ее в полупространство в многомерном пространстве вероятностных моделей. Сочетание утверждений соответствует пересечению ограничений. Обеспечить, чтобы пересечение давало единственную точку, совсем непросто. В действительности основной результат Гайфмана ([804], 1964) является конструкцией единственной вероятностной модели, требующей: *а*) вероятности для каждого возможного базового высказывания; *б*) вероятностных ограничений для бесконечно большого числа высказываний, стоящих под квантором существования.

Одно решение этой проблемы состоит в разработке частичной теории и ее последующем "завершении" путем выбора одной канонической модели из допустимого множества. Нильссон в [1689] (1986) предложил выбирать модель *максимальной энтропии*, совместимой с заданными ограничениями. Паскин ([1740], 2002) разработал "вероятностную логику с максимальной энтропией" с ограничениями, представленными в виде весовых коэффициентов (относительных вероятностей), прикрепленных к выражениям первого порядка. Такие модели часто называют **марковским логическими сетями**, или MLN (Ричардсон и Домингос [1878], 2006), и они уже стали популярным выбором для приложений, связанных с реляционными данными. Подходы с максимальной энтропией, включая MLN, в некоторых случаях могут давать неинтуитивные результаты (Милх [1571], 2006; Джайн и др. [1126], 2007; [1125], 2010).

В начале 1990-х годов исследователи, работавшие над сложными приложениями, отметили ограничения в выразительности байесовских сетей и разработали различные языки для написания "шаблонов" с логическими переменными, из которых затем можно было автоматически строить крупные сети для каждого экземпляра задачи (Бриз [296], 1992; Веллман и др. [2318], 1992). Наиболее важным таким языком был BUGS (*Bayesian inference Using Gibbs Sampling*, — байесовский вывод с использованием выборки Гиббса), объединявший байесовские сети с нотацией ►**индексированных случайных переменных**, принятой и в статистике (Гилкс и др. [855], 1994; Лунн и др. [1466], 2013). (В языке BUGS индексированная случайная переменная выглядит как $X[i]$, где i имеет определенный целочисленный диапазон.)

Эти языки для замкнутых миров унаследовали ключевое свойство байесовских сетей: каждая правильно сформированная база знаний определяют уникальную, непротиворечивую вероятностную модель. Другие языки для замкнутых миров опирались на репрезентативные средства и возможности логического вывода логического программирования (Пул [1808], 1993; Сато и Камея [1981], 1997; Керстинг и др. [1218], 2000) и семантических сетей (Коллер и Пфеффер [1265], 1998; Пфеффер [1786], 2000).

Исследования в области вероятностных моделей с открытой вселенной имеют несколько истоков. В статистике проблема ►**связывания записей** возникает в тех случаях, когда записи данных не содержат стандартных уникальных идентификаторов; например, в различных ссылках на эту книгу ее первого автора могут упоминать как "Стюарт Дж. Рассел" или "С. Рассел", или даже "Стюарт Рассел", но при этом есть и другие авторы с именем "С. Рассел".

Сотни компаний существуют исключительно для решения проблем с записями в финансовых, медицинских, переписных и других данных. Вероятностный анализ восходит к работе Данна ([663], 1946). Модель Фелледжи-Сунтера ([721], 1969), которая по существу является наивным байесовским классификатором, применяемым для согласования, по-прежнему доминирует в текущей практике. Неопределенность идентичности также рассматривается в задаче многоцелевого отслеживания (Ситтлер [2080], 1964), суть которой схематически представлена в разделе 15.3.1.

В области ИИ до начала 1990-х годов полагалось, что датчики могут предоставлять только логические высказывания с уникальными идентификаторами для объектов, поскольку именно так было в случае робота Shakey. Изменения пришли из области понимания естественного языка, когда Чарняк и Голдмен ([396], 1992) предложили вероятностный анализ *кореферентности* для случая, когда два лингвистических выражения (например, "Обама" и "президент") могут относиться к одной и той же сущности. Хуанг и Расселл ([1085], 1998), а также Пасула и соавт. ([1742], 1999) разработали метод байесовского анализа неопределенности идентичности для наблюдения за дорожным движением. Позднее Пасула и соавт. ([1741], 2003) разработали комплексную порождающую модель для авторов, статей и строк ссылок, включая неопределенность как ссылочную, так и идентичности, и продемонстрировали высокую точность по извлечению сведений о цитировании.

Первым формальным языком для вероятностных моделей с открытой вселенной был BLOG (Милх и др. [1572], 2005; Милх [1571], 2006), который поставлялся с (очень медленным) движком MCMC-алгоритма вероятностного вывода общего назначения. Ласки ([1357], 2008) описывает другой язык моделирования с открытой вселенной, называемый ►**байесовскими сетями с многими сущностями** (*Multi-Entity Bayesian Networks* — ►**MEBN**). Глобальная сейсмическая система мониторинга NET-VISA была описана в этой главе благодаря работе Ароры и соавт. [75] (2013). Рейтинговая система Эло была разработана в 1959 году Арпадом Эло ([685], 1978), но по сути это то же самое, что и модель Case V Терстоуна ([2214], 1927). Модель TrueSkill от Microsoft (Гербрих и др. [1014], 2007; Минка и др. [1580], 2018) основана на байесовской версии Эло Марка Гликмана ([868], 1999) и в настоящее время работает на базе языка PPL системы Infer.NET.

Ассоциация данных при многоцелевом отслеживании впервые была описана в вероятностном окружении Ситтлером ([2080], 1964). Первым практическим алгоритмом для масштабных задач был алгоритм MHT — "Multiple Hypothesis Tracker"

(Рейд [1867], 1979). Важные публикации по этой теме были собраны Бар-Шаломом и Фортманном (1988) и Бар-Шаломом (1992). Разработка алгоритма MCMC для задачи ассоциации данных была выполнена Пасулой и соавт. ([1742], 1999), применившими его к задачам наблюдения за дорожным движением. Ох и соавт. ([1708], 2009) предоставили более формальный анализ и результаты экспериментальных сравнений с другими методами. Шульц и соавт. ([2008], 2003) описывают подход к реализации ассоциации данных, основанный на методе фильтрации частиц.

Ингемар Кокс проанализировал сложность задачи ассоциации данных (Кокс [486], 1993; Кокс и Хингорани [487], 1994) и привлек к этой теме внимание сообщества компьютерного зрения. Он также отметил применимость венгерского алгоритма, характеризующегося полиномиальным временем, к задаче поиска наиболее вероятных назначений, которая в сообществе отслеживания долгое время считалась неразрешимой. Сам этот алгоритм был опубликован Куном ([1320], 1955) на основании переводов работ, опубликованных в 1931 году двумя венгерскими математиками, Денесом Кенигом и Йено Эгервари. Однако основная теорема была выведена ранее и впервые упоминалась в неопубликованной латинской рукописи знаменитого математика Карла Густава Якоба (1804–1851).

Идея о том, что вероятностные программы могут также представлять сложные вероятностные модели, была предложена Коллером и соавт. ([1267], 1997). Первым работающим языком PPL был IBAL Ави Пфеффера ([1783], 2001; [1787], 2007), основанный на простом функциональном языке. Язык BLOG можно рассматривать как декларативный язык PPL. Возможность соединения декларативного и функционального языков PPL исследовалась Макаллестером и соавт. ([1526], 2008). Язык CHURCH (Гудман и др. [900], 2008) является языком PPL, построенным на базе языка Scheme; в нем впервые была реализована идея совмещения с уже существующим языком программирования. В CHURCH также представлен первый MCMC-алгоритм вероятностного вывода для моделей со случайными функциями высшего порядка, что вызвало интерес у сообщества когнитивных наук как способ моделирования сложных форм обучения человека (Лейк и др. [1342], 2015). Языки PPL также связаны с интересными методами в теории вычислимости (Аккерман и др. [12], 2013) и исследованиями в области языков программирования.

В 2010-х годах появились десятки языков PPL, построенных на базе широкого диапазона языков программирования. Язык Figaro, основанный на языке Scala, использовался в широком диапазоне разнообразных приложений (Пфеффер [1785], 2016). Язык Gen (Кусумано-Тоунер и др. [509], 2019), основанный на языках Julia и TensorFlow, использовался в системах машинного восприятия реального времени наряду с байесовской структурой обучения для анализа данных временных рядов. Языки PPL, построенные на основе структур глубокого обучения, включают язык Pyro (Бингам и др. [218], 2019) (построен на базе системы PyTorch) и язык Edward (Тран и др. [2225], 2017) (построен на базе языка TensorFlow).

Было предпринято немало усилий, чтобы сделать вероятностное программирование доступным для большего числа людей, таких как пользователи баз данных

и электронных таблиц. Язык Tabular (Гордон и др. [903], 2014) представляет собой язык описания реляционной схемы с интерфейсом, подобным электронной таблице, реализованный поверх языка Infer.NET. Язык BayesDB (Саад и Мансингха [1951], 2017) предоставляет пользователям возможность комбинировать вероятностные программы и делать запросы к ним с помощью языка, подобного SQL.

Вероятностный вывод в вероятностных программах обычно полагается на приближенные методы, поскольку точные алгоритмы не масштабируются до уровня моделей тех видов, которые способны представлять языки PPL. Языки с замкнутым миром, такие как BUGS, LIBBI (Мюррей [1643], 2013) и STAN (Карпентер и др. [375], 2017), обычно работают путем построения полной эквивалентной байесовской сети, а затем запускают в ней вероятностный вывод: выборку Гиббса — в случае языка BUGS, последовательный метод Монте-Карло — в случае языка LIBBI и гамильтониан Монте-Карло — в случае языка STAN. Программы на этих языках можно читать как инструкции по построению базовой байесовской сети. Бриз в работе [296] (1992) показал, как, учитывая запрос и свидетельство, генерировать только соответствующий случаю фрагмент всей сети.

Работа с базовой байесовской сетью означает, что возможные миры, посещаемые алгоритмом MCMC, представлены вектором значений переменных в байесовской сети. Идея прямого отбора возможных миров первого порядка принадлежит Расселу ([1941], 1999). На языке FACTORIE (Маккаллум и др. [1528], 2009) возможные миры в процессе работы алгоритма MCMC представлены в стандартной системе реляционной базы данных. В этих же двух статьях предлагается инкрементальная переоценка запросов как способ избежать полной оценки запросов в каждом возможном мире.

Методы вероятностного вывода, построенные на использовании базовых элементов, являются аналогами самых ранних пропозициональных методов логического вывода первого порядка (Девис и Путнам [544], 1960). В логическом выводе как приложения доказательства теорем резолюции, так и системы логического программирования полагаются на принцип **подъема** (см. раздел 9.2), чтобы избежать создания экземпляров логических переменных, не являющихся необходимыми.

Пфеффер и соавт. ([1784], 1999) предложили алгоритм устранения переменной, в котором каждый вычисленный фактор кешируется для повторного использования при последующих вычислениях, включающих те же отношения, но иные объекты, тем самым реализуя некоторые вычислительные преимущества подъема. Первый действительно поднятый алгоритм вероятностного вывода являлся формой алгоритма устранения переменной и был описан Пулом в [1809] (2003), а затем улучшен Сальво Бразом и соавт. ([562], 2007). Дальнейшие успехи, в том числе случаи, когда некоторые совокупные вероятности могут быть вычисленными в замкнутой форме, были описаны Милхом и соавт. ([1573], 2008), а также Кисински и Пулом ([1231], 2009). В настоящее время существует довольно хорошее понимание, когда подъем возможен, а также его сложности (Грибков и др. [922], 2014; Каземи и др. [1204], 2017).

Методы ускорения вероятностного вывода приходят из разных направлений исследований, как это было отмечено в начале главы. В нескольких проектах изучались более сложные алгоритмы в сочетании с методами построения компиляторов и/или обученных вспомогательных распределений. В системе LIBBI (Мюррей [1643], 2013) впервые введен вероятностный вывод по методу частиц Гиббса для вероятностных программ; реализован один из первых трансляторов вероятностного вывода, с поддержкой GPU для массовых параллельных SMC, и предусмотрено использование языка моделирования для определения пользовательских вспомогательных распределений алгоритма MCMC. Результаты изучения компиляции вероятностного вывода также представлены Вингейтом и соавт. ([2359], 2011), Пейджем и Вудом ([1722], 2014), Ву и соавт. ([2391], 2016). В работах Кларе и соавт. ([442], 2013), Хура и соавт. ([1100], 2014), а также Кусумано-Тоунера и соавт. ([509], 2019) демонстрируются методы статического анализа для преобразования вероятностных программ в более эффективные формы. Язык PICTURE (Кулкарни и др. [1324], 2015) является первым языком PPL, предоставляющим пользователям возможность применять обучение с помощью предварительных выполнений генерирующих программ для тренировки быстрых восходящих вспомогательных распределений. Ли и соавт. ([1368], 2017) описывают использование методов глубокого обучения для эффективной выборки по важности в PPL. На практике алгоритмы вероятностного вывода для сложных вероятностных моделей часто используют несколько методов для различных подмножеств переменных в модели. Мансингка и соавт. ([1489], 2013) подчеркнули важность идеи о программах вероятностного вывода, которые будут применять различную тактику вероятностного вывода к подмножествам переменных, выбранных в процессе выполнения вывода.

Сборник, отредактированный Гетуром и Таскаром ([842], 2007), включает в себя много важных статей о вероятностных моделях первого порядка и их использовании в машинном обучении. Статьи о вероятностном программировании появляются в материалах всех крупных конференций по вопросам машинного обучения и вероятностных рассуждений, в том числе NeurIPS, ICML, UAI и AISTATS. Регулярные семинары по языкам PPL проводятся в рамках конференций NeurIPS и POPL (Principles of Programming Languages), а в 2018 году была проведена первая международная конференция по вероятностному программированию — International Conference on Probabilistic Programming (ICPP).

ГЛАВА 16

Принятие простых решений

В этой главе показано, как агент должен принимать решения, чтобы получать то, что ему необходимо, по крайней мере насколько это возможно и в среднем количестве случаев.

В этой главе речь пойдет о том, как теорию полезности можно объединить с теорией вероятности, чтобы получить агента, действующего на основе теории принятия решений, т.е. агента, способного принимать рациональные решения, исходя из своих убеждений и целей. Такой агент сможет принимать решения в условиях, в которых неопределенность и противоречивость целей не позволят логическому агенту найти какое-либо решение. Агент, действующий на основе цели, способен лишь на бинарную оценку: отличить хорошее состояние (цель) от плохого состояния (не цель), тогда как агент, действующий на основе теории принятия решений, способен оценивать состояния в непрерывном диапазоне значений и имеет возможность выбрать лучшее состояние, даже когда наилучшее состояние недоступно.

В разделе 16.1 обсуждается основной принцип теории принятия решений — максимизация ожидаемой полезности. В разделе 16.2 показано, что модель поведения любого рационального агента можно построить, определив функцию полезности, которую он будет максимизировать. В разделе 16.3 подробно обсуждаются характерные особенности функций полезности и, в частности, их связь с отдельными величинами, такими как деньги. В разделе 16.4 показано, как обращаться с функциями полезности, которые зависят от нескольких величин. В разделе 16.5 описывается реализация систем принятия решений, в частности — рассматривается формальный подход, называемый **сетями принятия решений** (известный также под названием **диаграммы влияния**). Такие сети представляют собой расширение байесовских сетей за счет включения в них действий и показателей полезности. В разделе 16.6 объясняется, как агент, действующий на основе теории принятия решений, может рассчитать стоимость приобретения новой информации для улучшения своих решений.

В то время как в разделах 16.1–16.6 предполагается, что агент всегда действует на основании заданной, известной ему функции полезности, в разделе 16.7 это предположение ослабляется. Мы обсудим последствия неопределенности предпочтений со стороны машины, наиболее важным из которых является уважение к людям.

16.1. Сочетание убеждений и желаний в условиях неопределенности

Мы начнем с агента, который, как и все агенты, должен принять решение. Ему доступны некоторые действия a. В отношении его текущего состояния может иметь место неопределенность, поэтому предположим, что агент назначил вероятность $P(s)$ каждому возможному текущему состоянию s. Также у него может существовать неопределенность в отношении результатов действий: модель перехода задана как $P(s' \mid s, a)$, т.е. вероятность того, что действие a в состоянии s позволит достичь состояния s'. Поскольку в первую очередь нас интересует достижение результата s', для вероятности достижения состояния s' за счет выполнения действия a в текущем состоянии мы будем использовать сокращенное обозначение $P(\text{RESULT}(a) = s')$. Эта вероятность и модель перехода связаны следующим образом:

$$P(\text{RESULT}(a) = s') = \sum_s P(s)P(s' \mid s, a).$$

Теория принятия решений в своей простейшей форме связана с осуществлением выбора среди действий, основанных на желательности получения *немедленных* результатов их выполнения, при этом предполагается, что среда является эпизодической в смысле, определенном в разделе 2.3.2. (Это предположение смягчается в главе 17.) Предпочтения агента определяются ►**функцией полезности**, $U(s)$, которая присваивает состоянию единственное числовое значение, определяющее, насколько оно желательно. ►**Ожидаемая полезность** действия с учетом свидетельства, $EU(a)$, — это просто среднее значение полезности всех возможных исходов, взвешенных по вероятности того, что этот исход будет иметь место:

$$EU(a) = \sum_{s'} P(\text{RESULT}(a) = s')U(s'). \qquad (16.1)$$

Принцип **максимальной ожидаемой полезности** (*Maximum Expected Utility* — ►**MEU**) гласит, что рациональный агент должен выбирать действие, которое максимизирует ожидаемую полезность для агента:

$$action = \text{argmax}_a EU(a).$$

В некотором смысле принцип MEU может рассматриваться как определение всего искусственного интеллекта. Все, что должен делать интеллектуальный агент, сводится к вычислению различных количественных величин, определению максимальной полезности по своим действиям и осуществлению этих действий. Но сказанное не означает, что тем самым проблема искусственного интеллекта *решена* по определению!

Принцип MEU *формализует* общее представление о том, что интеллектуальный агент всегда должен "поступать правильно", но не дает никаких рекомендаций о том, как это можно *реализовать*. Для оценки распределения вероятностей $P(s)$ для

всех возможных состояний в мире, охватываемых в $P(\text{RESULT}(a) = s')$, требуются восприятие, обучение, представление знаний и вероятностный вывод. Собственно вычисление $P(\text{RESULT}(a) = s')$ также требует построения причинной модели мира. Может потребоваться рассмотреть много действий, а вычисление результирующей полезности $U(s')$ может само по себе потребовать дальнейшего поиска или планирования, поскольку агент не способен определить, насколько хорошим является состояние, пока не узнает, чего он сможет достичь из этого состояния. Система ИИ, действующая от имени человека, может не знать истинной функции полезности именно для человека, а значит, может иметь место неопределенность относительно значения U. Поэтому теория принятия решений — это вовсе не панацея, позволяющая решить все проблемы искусственного интеллекта, но она предоставляет истоки основных математических конструкций, достаточно общих для определения задачи ИИ.

Понятно, что принцип MEU связан с идеей показателей производительности, представленной в главе 2. Эта основная идея очень проста. Рассмотрим различные среды, действия в которых могут привести к получению агентом заданной истории восприятий, и предположим, что может быть спроектировано несколько разных агентов. ➔ *Если агент максимизирует функцию полезности, правильно отражающую показатели производительности, то этот агент достигнет наивысших возможных значений показателей производительности (усредненных по всем возможным вариантам среды).* Это определение является также главным обоснованием для самого принципа MEU. Хотя на первый взгляд такое заявление может показаться тавтологией, фактически оно воплощает в себе очень важный переход от внешнего критерия рациональности к внутреннему критерию, функции полезности. Мера производительности дает оценку для истории — некоторой последовательности состояний, а значит, она применяется ретроспективно, уже после того, как агент завершил выполнение данной последовательности действий. Функция полезности применяется непосредственно к следующему состоянию, поэтому ее можно использовать для пошагового руководства действиями.

16.2. Основы теории полезности

Интуитивно понятно, что принцип максимальной ожидаемой полезности (MEU) можно выбрать в качестве разумной основы для принятия решений, но вовсе не очевидно, что это *единственный* рациональный способ. В конце концов, почему именно максимизации *средней* полезности следует придавать такое значение? Что плохого в том, если агент будет максимизировать взвешенную сумму кубов возможных полезностей или попытается минимизировать наихудшие возможные потери? Сможет ли агент действовать рационально, просто выражая свои предпочтения между состояниями и не присваивая им числовых значений? Наконец, почему вообще должна существовать функция полезности с требуемыми свойствами? Давайте посмотрим.

16.2.1. Ограничения, налагаемые на рациональные предпочтения

На эти вопросы можно ответить, записав некоторые ограничения, налагаемые на предпочтения, которые должен иметь рациональный агент, а затем показав, что принцип MEU можно вывести из этих ограничений. Для описания предпочтений агента будем использовать следующую нотацию.

$A \succ B$	Для агента вариант A предпочтительнее, чем B
$A \sim B$	Агент безразличен к выбору между вариантами A и B
$A \succsim B$	Агент предпочитает вариант A варианту B или безразличен к выбору между ними

Теперь напрашивается очевидный вопрос: к какого рода понятиям относятся A и B? Это могут быть состояния мира, но гораздо чаще существует неопределенность в отношении того, что действительно предлагается. Например, пассажир авиакомпании, которому на выбор предлагают "спагетти" или "вареную курицу", не знает, что скрывается под фольгой.[1] Спагетти могут быть вкусными или оказаться застывшей слипшейся массой; курица может представлять собой сочный кусок или оказаться разваренной до неузнаваемости. Множество результатов каждого действия можно воспринимать как ▶**лотерею**, а само действие рассматривать как ее билет. Лотерея L с возможными исходами $S_1, \ldots, S_n$, которые имеют место с вероятностями $p_1, \ldots, p_n$, записывается как

$$L = [p_1, S_1; p_2, S_2; \ldots p_n, S_n].$$

В общем случае каждым результатом лотереи может быть атомарное состояние или другая лотерея. Основная проблема теории полезности состоит в том, чтобы понять, как предпочтения между сложными лотереями связаны с предпочтениями между состояниями, лежащими в основе этих лотерей. Чтобы решить эту проблему, перечислим шесть ограничений, которые должны всегда соблюдаться.

- ▶**Обязательность**. Если даны две лотереи, то рациональный агент должен либо предпочесть одну другой, либо рассматривать их обе как одинаково предпочтительные. Это означает, что агент *не может* избежать принятия решений. Как было указано в разделе 12.2.3, отказ делать ставку подобен отказу позволить времени двигаться:

  Из $(A \succ B)$, $(B \succ A)$ или $(A \sim B)$ выбирается точно одно.

- ▶**Транзитивность**. При наличии трех лотерей, если агент предпочитает лотерею A лотерее B и лотерею B лотерее C, он должен предпочесть лотерею A лотерее C:

[1] Мы приносим извинения читателям, чьи местные авиалинии больше не предлагают питание на дальних рейсах.

$$(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C).$$

- ▶**Непрерывность**. Если некоторая лотерея B находится в порядке предпочтений между лотереями A и C, то существует некоторая вероятность p того, что рациональный агент будет безразличен к тому, чтобы определенно выбрать лотерею B или лотерею, результатом которой является лотерея A с вероятностью p и лотерея C с вероятностью $1-p$:

$$A \succ B \succ C \Rightarrow \exists p\ [p, A;\ 1-p, C] \sim B.$$

- ▶**Заменяемость**. Если агент безразличен к выбору между двумя лотереями, A и B, то агент безразличен и к выбору между двумя более сложными лотереями, которые являются одинаковыми, за исключением того, что в одной из них подставлена лотерея B вместо лотереи A. Такое свойство сохраняется независимо от вероятностей и от других результатов в лотереях:

$$A \sim B \Rightarrow [p, A;\ 1-p, C] \sim [p, B;\ 1-p, C].$$

  Если в этой аксиоме заменить $\succ$ символом $\sim$, она также будет верна.

- ▶**Монотонность**. Предположим, что две лотереи имеют два одинаковых результата, A и B. Если агент предпочитает состояние A состоянию B, то агент должен предпочесть лотерею, которая имеет более высокую вероятность для состояния A (и наоборот):

$$A \succ B \Rightarrow (p > q \Leftrightarrow [p, A;\ 1-p, B] \succ [q, A;\ 1-q, B]).$$

- ▶**Декомпонуемость**. Сложные лотереи можно свести к простым, используя законы вероятностей. Это свойство получило название "правило «экономии количества ставок»", поскольку согласно ему две последовательные лотереи могут быть сжаты в одну эквивалентную лотерею[2] (как показано на рис. 16.1, *б*):

$$[p, A;\ 1-p, [q, B;\ 1-q, C]] \sim [p, A;\ (1-p)q, B;\ (1-p)(1-q), C].$$

Эти ограничения известны как аксиомы теории полезности. Каждую аксиому можно обосновать, показав, что нарушающий ее агент в определенных ситуациях будет демонстрировать явно нерациональное поведение. Например, можно обосновать свойство транзитивности, заставив агента с нетранзитивными предпочтениями отдать нам все свои деньги. Предположим, что этот агент имеет нетранзитивные предпочтения A $\succ$ B $\succ$ C $\succ$ A, где A, B и C — товары, которые можно свободно обменивать. Если агент в настоящее время имеет A, то можно предложить ему обменять наше C на его A плюс один цент. Агент предпочитает C и поэтому согласится совершить эту сделку. Далее можно предложить ему

[2] Можно учесть саму привлекательность игры на деньги, включив события игры в описание состояния; например, действие "взять с собой 10 долл. и сделать ставку" может рассматриваться как более предпочтительное, чем "взять с собой 10 долл. и не сделать ставку".

обменять наше B на его C при тех же условиях и забрать у него еще один цент. Наконец, аналогичным образом предлагаем ему обменять A на B и возвращаемся в исходную ситуацию за исключением того, что агент отдал нам три цента (рис. 16.1, *а*). Теперь можно продолжить повторение этого цикла до тех пор, пока у агента вообще не останется денег. Очевидно, что в этом случае агент действовал нерационально.

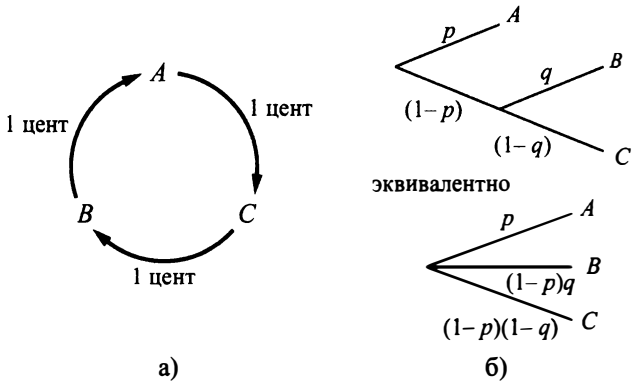


Рис. 16.1. а) Нетранзитивные предпочтения $A \succ B \succ C \succ A$ могут привести к нерациональному поведению: цикл из трех повторяющихся обменов, каждый из которых обходится в один цент. **б)** Графическая иллюстрация аксиомы декомпонуемости

16.2.2. Рациональные предпочтения ведут к полезности

Обратите внимание, что аксиомы теории полезности в действительности являются аксиомами о предпочтениях — в них ничего не говорится о функции полезности. Но на самом деле из этих аксиом полезности можно вывести важные следствия, приведенные ниже (доказательство представлено в работе Фон Неймана и Моргенштерна [2282] (1944)).

- **Наличие функции полезности**. Если предпочтения агента подчиняются аксиомам полезности, то существует функция U, такая, что $U(A) > U(B)$ тогда и только тогда, когда A предпочтительнее, чем B, и $U(A) = U(B)$ тогда и только тогда, когда агент безразличен к выбору между A и B:

$$U(A) > U(B) \Leftrightarrow A \succ B \quad \text{и} \quad U(A) = U(B) \Leftrightarrow A \sim B.$$

- **Ожидаемая полезность лотереи**. Полезность лотереи равна сумме произведений вероятности каждого результата на полезность этого результата:

$$U([p_1, S_1; \ldots ; p_n, S_n]) = \sum_i p_i U(S_i).$$

Другими словами, как только определены вероятности и полезности всех возможных результирующих состояний, полезность сложной лотереи, охватывающей эти состояния, становится полностью определенной. Поскольку результатом недетерминированного действия является лотерея, из этого следует, что агент может действовать рационально — т.е. в соответствии с его предпочтениями — только путем выбора действий, который максимизирует ожидаемую полезность в соответствии с уравнением (16.1).

В приведенных выше теоремах утверждается, что (в предположении о соблюдении ограничений на рациональные предпочтения) функция полезности *существует* для любого рационального агента. Однако эти теоремы не утверждают, что такая функция полезности является *единственной*. В действительности легко увидеть, что поведение агента не изменится, если его функцию полезности $U(S)$ преобразовать следующим образом:

$$U'(S) = aU(S) + b, \tag{16.2}$$

где a и b — константы и $a > 0$; это положительное аффинное преобразование.[3] Данный факт уже был отмечен в главе 5 (раздел 5.5.1) для азартных игр для двух игроков, здесь же показано, что это преобразование применимо ко всем видам сценариев принятия решений.

Как и в играх, в детерминированной среде агенту требуется только ранжирование предпочтений по состояниям — сами числовые величины не имеют значения. Это называется ►**функцией ценности** или ►**порядковой функцией полезности**.

Важно помнить, что из существования функции полезности, описывающей поведение агента по выбору предпочтений, вовсе необязательно следует, что агент *явно* максимизирует эту функцию полезности в собственных рассуждениях. Как было показано в главе 2, рациональное поведение может быть выработано многими различными способами. Рациональный агент может быть реализован даже с использованием поиска в таблице функций, когда количество возможных состояний достаточно мало.

Наблюдая за поведением рационального агента, можно достаточно узнать о функции полезности, чтобы составить представление о том, чего агент действительно пытается достичь (даже если агент сам этого не знает). К данной теме мы еще вернемся в разделе 16.7.

[3] В этом смысле полезность напоминает температуру: значение температуры в градусах Фаренгейта в 1,8 раза больше значения температуры в градусах Цельсия плюс 32, но переход от одной шкалы к другой не делает объект горячее или холоднее.

16.3. Функции полезности

Функции полезности отображают лотереи на действительные числа. Мы знаем, что они должны подчиняться аксиомам обязательности, транзитивности, непрерывности, заменяемости, монотонности и декомпонуемости. И это все, что можно сказать о функциях полезности? Строго говоря, да: агент может иметь любые предпочтения, какие пожелает. Например, он может предпочесть держать на своем банковском счете количество долларов, выражаемое простым числом: в этом случае, если у него на счету 16 долл., 3 долл. он должен куда-то деть. Это может казаться необычным, но это нельзя назвать иррациональным. Агент вправе предпочесть "зубастый" форд Pinto выпуска 1973 года сверкающему новому мерседесу. Агент может предпочитать иметь на счету количество долларов, выражающееся простыми числами, только когда он владеет автомобилем Pinto, а когда в его собственности мерседес, он может выбрать другое предпочтение: иметь больше долларов, чем меньше. К счастью, предпочтения реальных агентов обычно носят более систематический характер и поэтому с ними легче иметь дело.

16.3.1. Оценка полезности и шкалы полезности

Если наша цель — построить систему, реализующую теорию принятия решений, которая будет помогать человеку принимать решения или действовать от его имени, сначала необходимо выяснить, что представляет собой функция полезности человека. Этот процесс, чаще всего называемый ▶**выявлением предпочтений**, включает предоставление человеку возможности выбора и использование наблюдаемых предпочтений для выявления лежащей в его основе функции полезности.

Уравнение (16.2) говорит, что для полезности не существует абсолютной шкалы, тем не менее имеет смысл установить *некоторую* шкалу, которая позволит записывать и сравнивать полезности для любой конкретной задачи. Шкала может быть создана посредством фиксации полезности любых двух конкретных исходов, — так же, как была принята шкала измерения температуры по Цельсию за счет фиксации на точке замерзания и на точке кипения воды. Как правило, фиксируются полезность "наилучшего возможного выигрыша" при $U(S) = u_\top$ и "наихудшей возможной катастрофы" при $U(S) = u_\bot$. (Обе эти величины должны быть конечными.) Для ▶**нормализованных полезностей** используют шкалу с $u_\bot = 0$ и $u_\top = 1$. При такой шкале футбольный фанат в Англии может назначить полезность 1 для команды Англии, выигравшей Кубок мира, и полезность 0 — для команды Англии, не прошедшей квалификационный отбор.

При заданной шкале полезности между $u_\bot$ и $u_\top$ можно оценить полезность любого конкретного выигрыша S, попросив агента выбрать между S и ▶**стандартной лотереей** $[p, u_\top; (1-p), u_\bot]$. Значение вероятности p будет уточняться до тех пор, пока агенту не станет безразличен выбор между S и стандартной лотереей. При принятии нормализованной полезности полезность S будет задаваться

значением p. Повторив эти действия для каждого возможного выигрыша, можно определить полезность для всех лотерей, включающих эти выигрыши. Предположим, например, что необходимо узнать, как наш футбольный болельщик из Англии расценивает результат, когда команда Англии выйдет в полуфинал, а затем проиграет. Сравним этот результат со стандартной лотереей с вероятностью p выиграть кубок и вероятностью $1 - p$ для позорной неспособности пройти даже квалификационные игры. Если при $p = 0{,}3$ у болельщика наблюдается равнодушие, то 0,3 — это значение полезности для случая достижения полуфинала и проигрыша.

В медицинских, транспортных, экологических и других задачах принятия решения на карту часто ставится жизнь людей. (Да, есть вещи, более важные, чем победа команды Англии на Всемирном чемпионате.) В таких случаях как $u_{\perp}$ назначается значение, соответствующее немедленной смерти (или в действительно худшем случае — многих смертей). ➔ *Несмотря на то что никто не обладает правом устанавливать цену человеческой жизни, это факт, что компромиссы по вопросу о жизни и смерти принимались и принимаются постоянно.* Самолеты отправляются на полный капитальный ремонт с некоторыми интервалами, а не после каждого вылета. Автомобили производятся таким образом, чтобы обеспечить некий компромисс между стоимостью и вероятностью выживания при несчастном случае. Все мы миримся с таким уровнем загрязнения воздуха, который убивает четыре миллиона человек в год.

Как это ни парадоксально, но отказ определить стоимость человеческой жизни в виде денежной суммы может означать, что жизнь является *недооцененной*. Росс Шахтер описывает государственное агентство, заказавшее исследование по проблеме удаления асбеста из школ. Проводившие исследование аналитики, на которых было возложено принятие решения, установили определенную стоимость в долларах для жизни ребенка школьного возраста и в конце исследования пришли к заключению, что рациональным выбором в данной ситуации будет принятие мер по удалению асбеста. Заказавшее исследование агентство, морально возмущенное самой идеей денежной оценки жизни, немедленно отклонило отчет, а затем просто отказалось от удаления асбеста, этим неявно утверждая, что ценность жизни ребенка ниже, чем оценка, которую приняли аналитики.

В настоящее время несколько органов правительства США, в том числе Агентство по охране окружающей среды, Агентство контроля пищевых продуктов и медикаментов, а также Министерство транспорта, используют показатель ▶**статистической стоимости жизни** для определения затрат и выгоды от регулирования и вмешательств. Типичные значения этого показателя в 2019 году составляли примерно 10 млн долл.

Были предприняты некоторые попытки выяснить, какую ценность сами люди придают своей жизни. Одна общая "валюта", используемая в медицинском анализе и анализе безопасности, — это ▶**микроморт**, вероятность смерти один на миллион. Если вы спросите людей, сколько они готовы заплатить, чтобы избежать этого риска — например, избежать игры в русскую рулетку с револьвером, имеющим

миллион стволов, — их ответ будет включать достаточно большие суммы, возможно, на уровне десятков тысяч долларов, но их фактическое поведение отражает гораздо более низкую денежную стоимость для микроморта.

Например, в Великобритании принято, что поездка на машине длиной 230 миль сопряжена с риском в один микроморт. За время жизни вашего автомобиля его пробег может составить, скажем, 92 000 миль, а это уже 400 микромортов. Есть данные, что люди, похоже, готовы заплатить примерно на 12 000 долл. больше за более безопасный автомобиль, который вдвое снижает риск смерти. Тогда из сказанного выше можно заключить, что их действия при покупке автомобиля свидетельствуют о согласии со стоимостью в 60 долл. за микроморт. Ряд исследований подтвердил оценку примерно в этом диапазоне для многих лиц и разных типов риска. Тем не менее государственные органы США, такие как Министерство транспорта, как правило, выбирают более низкие значения; они готовы потратить на ремонт дорог всего лишь около 6 долл. на ожидаемую спасенную жизнь. Конечно, эти расчеты верны только для небольших рисков. Большинство людей не согласится убить себя даже за 60 млн долл.

Еще одной мерой является ►**QALY**, или *скорректированный на качество год жизни*. Пациенты готовы принять более короткую продолжительность жизни, чтобы избежать инвалидности. Например, пациенты, страдающие болезнями почек, в среднем безразличны к выбору между двумя годами жизни при диализе и одним годом полного здоровья.

16.3.2. Полезность денег

Корни теории полезности скрываются в экономике, а экономика предоставляет единственного очевидного кандидата для использования в качестве меры полезности — деньги (или, более конкретно, общий суммарный капитал агента). Почти универсальная способность денег к обмену на всевозможные товары и услуги подсказывает, что деньги играют важную роль в функциях полезности людей.

Чаще всего можно обнаружить, что при прочих равных условиях агент, как правило, предпочитает иметь больше денег, а не меньше. А значит, можно утверждать, что агент проявляет ►**монотонное предпочтение** в отношении бо́льших сумм денег. Однако это еще не означает, что деньги всегда можно использовать в качестве меры функции полезности, поскольку в этом определении ничего не сказано о предпочтениях между *лотереями*, включающими денежные ставки.

Предположим, что вы одержали победу над всеми соперниками в телевизионном игровом шоу и ведущий предлагает вам выбор: забирайте свой приз в 1 000 000 долл. или сделайте на него ставку, подбросив монету. Если выпадет орел, вы ничего не получите, а если выпадет решка, то получите 2 500 000 долл. Если вы не склонны к риску, как и большинство людей, то откажетесь от этой ставки и положите в карман миллион. Является ли это решение нерациональным?

При условии, что монета не подделана, ▶**ожидаемая денежная ценность** (*Expected Monetary Value* — **EMV**) этой ставки равно $\frac{1}{2}$(0 долл.) + $\frac{1}{2}$ (2 500 000 долл.) = 1 250 000 долл., что больше, чем исходные 1 000 000 долл. Но такой расчет не обязательно означает, что принятие предложения сделать подобную ставку является лучшим решением. Допустим, что мы используем запись S_n для обозначения состояния, соответствующего обладанию всей суммой в n долл., а ваши текущие накопления составляют k долл. В таком случае ожидаемые полезности двух действий, соответствующих принятию (*Accept*) и отказу (*Decline*) от предложения сделать ставку, будут следующими:

$$EU(Accept) = \frac{1}{2}\,U(S_k) + \frac{1}{2}\,U(S_{k+2\,500\,000}),$$
$$EU(Decline) = U(S_{k+1\,000\,000}).$$

Чтобы определить, что делать, необходимо присвоить значения полезности результирующим состояниям. Полезность не является прямо пропорциональной денежной ценности, поскольку полезность вашего первого миллиона очень высока (по крайней мере, все так говорят), тогда как полезность еще одного миллиона будет уже меньше. Предположим, что вы присвоили значение полезности 5 своему текущему финансовому состоянию (S_k), значение 9 — состоянию $S_{k+2\,500\,000}$ и значение 8 — состоянию $S_{k+1\,000\,000}$. В таком случае рациональным действием будет отказ от предложения сделать ставку, поскольку ожидаемая полезность его принятия равна только 7,5 (меньше 8, что соответствует отказу от этого предложения). С другой стороны, миллиардер, скорее всего, будет обладать функцией полезности, практически линейной в диапазоне добавления еще нескольких миллионов, а значит, он охотно примет эту ставку.

В своем первопроходческом исследовании фактически применяемых функций полезности Грейсон ([917], 1960) обнаружил, что полезность денег почти точно пропорциональна *логарифму* их количества (предположение об этом впервые высказал Бернулли ([191], 1738); см. упражнение 16.4). Одна конкретная кривая полезности, характеризующая предпочтения некоего мистера Берда, показана на рис. 16.2, *а*. Полученные Грейсоном данные о предпочтениях мистера Берда совместимы со следующей функцией полезности:

$$U(S_{k+n}) = -263{,}31 + 22{,}09 \log(n + 150\,000)$$

для диапазона от $n = -150\,000$ долл. до $n = 800\,000$ долл.

Не следует полагать, что это безусловно верная функция полезности для денежных значений, но создается впечатление, что большинство людей руководствуются функцией полезности, которая является вогнутой в области положительных значений денежных накоплений. Брать в долг обычно считается плохим решением, но предпочтения между различными уровнями задолженности могут показывать обратное поведение по отношению к вогнутости, связанной с положительными денежными накоплениями. Например, если некто уже имеет долг 10 000 000 долл.,

то вполне может принять предложение сделать ставку на подбрасывание подлинной монеты с выигрышем в 10 000 000 долл. при выпадении орла и проигрышем в 20 000 000 долл. при выпадении решки.[4] Такое поведение соответствует S-образной кривой, показанной на рис. 16.2, *б*.

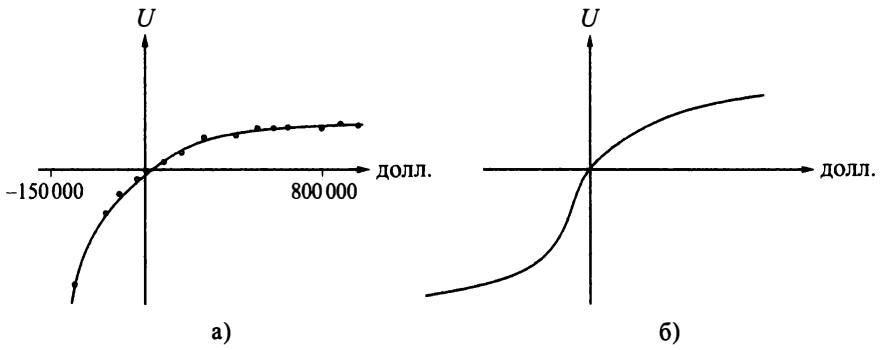


Рис. 16.2. Кривая полезности денег. **а)** Эмпирические данные о предпочтениях мистера Берда, представленные в ограниченном диапазоне. **б)** Типичная кривая для всего диапазона

Если ограничить рассмотрение лишь положительной частью таких кривых, где уклон постепенно уменьшается, то для любой лотереи L полезность решения, при котором придется сделать выбор в этой лотерее, будет меньше, чем полезность получения ожидаемой денежной ценности в этой лотерее без принятия условий:

$$U(L) < U(S_{EMV(L)}).$$

Это означает, что агенты с кривыми полезности такой формы ▶**не склонны к риску**: они предпочитают вариант гарантированной выплаты, пусть даже меньшей по сравнению с ожидаемой денежной ценностью возможной ставки. С другой стороны, в "безнадежной" области кривой с большими отрицательными накоплениями (см. рис. 16.2, *б*) для поведения агентов характерно уже ▶**стремление к риску**. Сумма, приобретаемая агентом вместо лотереи, называется ▶**эквивалентом определенности** лотереи. Исследования показали, что большинство людей предпочитают забрать 400 долл. вместо того, чтобы сделать ставку, розыгрыш которой позволит в половине случаев получить 1000 долл. и 0 долл. — в другой половине. Это означает, что эквивалентом определенности для этой лотереи является 400 долл., тогда как ее EMV равно 500 долл.

[4] Такое поведение можно назвать отчаянным, но оно рационально, если человек уже находится в безнадежной ситуации.

Разность между ожидаемой денежной ценностью лотереи и ее эквивалентом определенности называется ▶**страховой премией**. Неприятие риска является основой индустрии страхования, поскольку такое поведение означает, что страховые премии становятся положительными. Люди предпочитают заплатить небольшую страховую премию, нежели делать ставку на всю стоимость своего дома против вероятности его потери в огне. С точки зрения страховой компании цена отдельного дома весьма мала по сравнению с общими резервами этой компании. Это означает, что кривая полезности страховщика в таком небольшом регионе остается приблизительно линейной и для этой компании ставка на стоимость дома против страховой премии является почти беспроигрышной.

Обратите внимание, что при *небольших* изменениях в уровне денежных накоплений по сравнению с текущим размером накоплений почти любая кривая полезности должна быть приблизительно линейной. Агент, который руководствуется такой линейной кривой, называется ▶**нейтрально относящимся к риску**. Поэтому в случае ставок на небольшие суммы предполагается соблюдение свойства нейтрального отношения к риску. В определенном смысле это свойство является обоснованием упрощенной процедуры, в которой применяются небольшие ставки для оценки вероятностей, а также обоснованием аксиом вероятностей, приведенных в разделе 12.2.3.

16.3.3. Ожидаемая полезность и разочарование после принятия решения

Рациональный способ выбора наилучшего действия a^* заключается в максимизации ожидаемой полезности:

$$a^* = \text{argmax}_a EU(a).$$

Если ожидаемая полезность была правильно рассчитана в соответствии с вероятностной моделью и если вероятностная модель правильно отражает лежащие в основе стохастические процессы, определяющие генерируемые результаты, то в среднем ожидаемая полезность будет достигнута, если весь процесс будет повторен много раз.

Однако в действительности выбранная модель, как правило, слишком упрощает реальную ситуацию либо потому, что имеющихся знаний недостаточно (например, при принятии сложного инвестиционного решения), либо потому, что вычисление истинной ожидаемой полезности является слишком сложным (например, при выборе хода в нардах необходимо принимать во внимание все возможные в будущем варианты выпадения очков на паре игральных костей). В этом случае мы работаем действительно с *оценками* истинной ожидаемой полезности $\widehat{EU}(a)$. Будем предполагать, возможно излишне благожелательно, что оценки являются ▶**несмещенными**, т.е. ожидаемое значение ошибки $M(\widehat{EU}(a) - EU(a))$ является

нулевым. В этом случае все еще кажется разумным выбирать действия с самой высокой оценкой полезности и рассчитывать получить эту полезность, в среднем, после выполнения действия.

К сожалению, реальный результат, как правило, будет значительно *хуже*, чем это следует из оценки, несмотря даже на то, что оценка была несмещенной! Чтобы понять, почему, рассмотрим задачу принятия решения, в которой есть k вариантов, каждый из которых имеет истинную оценочную полезность 0. Предположим, что ошибки в оценке полезности каждого результата независимы и характеризуются единичным нормальным распределением, т.е. гауссианом с нулевым средним и единичной дисперсией, представленным утолщенной кривой на рис. 16.3. Теперь, если начать генерировать оценки полезности, некоторые ошибки будут отрицательными (пессимистичными), а некоторые — положительными (оптимистичными). Поскольку выбирается действие с самой *высокой* оценкой полезности, неизбежно проявится предпочтение в отношении чрезмерно оптимистичных оценок, что и послужит источником смещения.

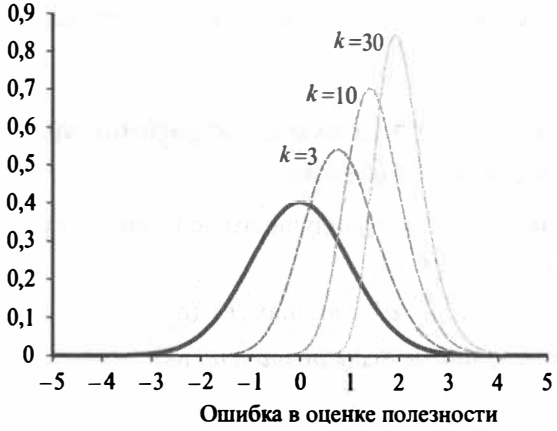


Рис. 16.3. Необоснованный оптимизм, вызванный выбором самых лучших из имеющихся k вариантов: предполагается, что каждый вариант имеет истинную полезность 0, а результаты оценки полезности подчиняются единичному нормальному распределению (утолщенная кривая). Остальные кривые показывают распределение для максимума из k оценок при $k = 3$, 10 и 30

Было совсем несложно вычислить распределение максимума для k оценок и, таким образом, количественно оценить степень нашего разочарования. (Это вычисление является частным случаем вычисления ▶ **порядковой статистики**, распределения любого конкретного ранжированного элемента выборки.) Предположим, что каждая оценка X_i имеет функцию плотности вероятности $f(x)$ и

кумулятивное распределение $F(x)$. (Как объясняется в приложении А, кумулятивное распределение F определяет вероятность того, что стоимость будет меньше или равна любой заданной величине, т.е. она интегрирует исходную плотность вероятности f.) Теперь пусть X^* будет наибольшей оценкой, т.е. равна $\max\{X_1,\dots, X_k\}$. Тогда кумулятивное распределение для X^* вычисляется как

$$P(\max\{X_1,\dots, X_k\} \le x) = P(X_1 \le x,\dots, X_k \le x) = \\ = P(X_1 \le x)\dots P(X_k \le x) = F(x)^k.$$

Функция плотности вероятности является производной кумулятивной функции распределения, поэтому плотность для X^*, т.е. для максимума из k оценок, будет равна

$$P(x) = \frac{d}{dx}\left(F(x)^k\right) = kf(x)(F(x))^{k-1}.$$

На рис. 16.3 эти плотности показаны для различных значений k в случае, когда $f(x)$ является единичным нормальным распределением. Для $k=3$ плотность вероятности для X^* имеет среднее значение примерно 0,85, так что среднее разочарование будет составлять около 85% от стандартного отклонения в оценке полезности. При большем количестве возможностей выбора крайне оптимистичные оценки с большой вероятностью возрастут: для $k=30$ разочарование будет примерно вдвое превышать стандартное отклонение для оценок.

Эту тенденцию — оказываться слишком высокой — для оценочной ожидаемой полезности при выборе наилучшей оценки называют ▶ **проклятием оптимизатора** (Смит и Винклер [2094], 2006). Оно поражает даже самых опытных аналитиков и статистиков. Серьезные проявления включают в себя необоснованную уверенность, что восхитительный новый препарат, который якобы излечивает 80% пациентов, при испытании действительно вылечит 80% пациентов (эта цифра была выбрана при k = "тысячи потенциальных лекарственных препаратов"), или фонд взаимопомощи, рекламируемый как имеющий доходность выше среднего уровня будет действительно ее иметь (для упоминания в рекламе это заявление было выбрано из k = "дюжины фондов" в общем портфолио компании). Может даже так оказаться, что то, что кажется лучшим выбором, может вовсе им не быть, если дисперсия у оценки полезности будет высока: препарат, который вылечил 9 из 10 пациентов и был выбран из тысяч опробованных, вероятно, будет *хуже*, чем тот, который вылечил 800 пациентов из 1000.

Проклятие оптимизатора проявляется везде и всюду по причине повсеместного распространения способа выбора по максимальной полезности, поэтому принимать оценки полезности по их "номинальной стоимости" — это плохая идея. Проклятия можно избежать с помощью байесовского подхода, в котором используется явная вероятностная модель $\mathbf{P}(\widehat{EU} \mid EU)$ для ошибки в оценках полезности. Имея эту модель и априорное распределение того, что можно обоснованно ожидать

относительно величин полезностей, можно рассматривать оценку полезности как свидетельство и вычислять апостериорное распределение для истинной полезности, используя правило Байеса.

16.3.4. Субъективные суждения и иррациональность

Теория принятия решений является ▶**нормативной теорией** — она описывает, как *должен* действовать рациональный агент. С другой стороны, ▶**описательная теория** описывает, как реальные агенты — например, люди — действуют на практике. Область применения экономической теории удалось бы значительно расширить, если бы вторая совпадала с первой, однако имеются некоторые экспериментальные свидетельства того, что все обстоит не так. Эти свидетельства показывают, что люди "предсказуемо иррациональны" (Ариэли [69], 2009).

Наиболее известным примером является парадокс экономиста Алле (Алле [30], 1953). Людям был предоставлен выбор между лотереями A и B, а затем между лотереями C и D, которые имели следующие призы.

A: шанс 80% на получение 4000 долл.
B: шанс 100% на получение 3000 долл.
C: шанс 20% на получение 4000 долл.
D: шанс 25% на получение 3000 долл.

Большинство участников предпочли лотерею B лотерее A (принимая гарантированный выигрыш) и лотерею C лотерее D (выбирая более высокую полезность). Однако нормативная теория требует иного! Почему — легче всего увидеть, если воспользоваться вытекающей из уравнения (16.2) свободой установить $U(0 \text{ долл.}) = 0$. В этом случае из $B \succ A$ для первого выбора следует, что $U(3000 \text{ долл.}) > 0{,}8U(4000 \text{ долл.})$, тогда как из $C \succ D$ следует прямо противоположное. Другими словами, не существует функции полезности, которая была бы совместимой с этими выборами.

Одним из объяснений для этих очевидно нерациональных предпочтений является ▶**эффект уверенности** (Канеман и Тверски [1173], 1979): людей чрезвычайно притягивает выигрыш, который они получат наверняка. Есть несколько причин, по которым это может происходить.

Во-первых, люди могут предпочесть снять с себя всю вычислительную нагрузку, — при выборе надежных результатов, им не потребуется выполнять каких-либо расчетов с вероятностями. И этот эффект сохраняется даже тогда, когда необходимые вычисления в действительности очень просты.

Во-вторых, люди могут не доверять законности заявленных вероятностей. Я поверю, что результаты бросков монеты действительно будут примерно 50/50 только в том случае, если я контролирую и монету, и броски, но я могу не поверить результатам, если броски выполнял кто-то другой, имеющий личные интересы в

полученных результатах.[5] При наличии недоверия лучшим выбором может оказаться гарантированный вариант.[6]

В-третьих, люди могут беспокоиться о своем эмоциональном состоянии точно так же, как и о финансовом. Люди знают, что испытают большое сожаление, если откажутся от гарантированной награды (лотерея *B*) в пользу шанса 80% на более высокую награду (лотерея *A*), а затем проиграют.

Другими словами, если будет выбрана лотерея *A*, то будет существовать шанс 20% вообще не получить никаких денег *и почувствовать себя полным идиотом*, что еще хуже, чем просто не получить денег. Так что вполне возможно, что те люди, которые предпочитают лотерею B лотерее A и лотерею C лотерее D, не являются нерациональными, — просто они готовы отказаться от лишних 200 долл. из EMV, чтобы избежать шанса 20% почувствовать себя полным идиотом.

Близкой проблемой является парадокс Эллсберга. В этом случае приз фиксирован, а ограничения относятся к вероятностям. Выплата будет зависеть от цвета шара, выбранного из урны. Известно, что урна содержит 1/3 красных шаров и 2/3 черных или желтых шаров, но остается неизвестным, сколько из них шаров черных и шаров желтых. И вновь, участнику предлагается выбор между лотереями *A* и *B*, а затем между лотереями *C* и *D*:

A: 100 долл. за красный шар
B: 100 долл. за черный шар
C: 100 долл. за красный или желтый шар
D: 100 долл. за черный или желтый шар

Должно быть понятно, что если вы полагаете, что красных шаров больше, чем черных, то вам лучше предпочесть лотерею *A* лотерее *B* и лотерею *C* лотерее *D*. Но если вы считаете, что красных шаров меньше, чем черных, то должны предпочесть обратное. Однако оказалось, что большинство людей предпочитают лотерею *A* лотерее *B*, но также предпочитают лотерею *D* лотерее *C*, несмотря на то что в этом мире нет такого состояния, для которого этот выбор был бы рациональным. Кажется, что люди испытывают ►**отвращение к неоднозначности**: лотерея A дает 1/3 шансов на победу, в то время как лотерея *B* может дать шансов где-то между 0 и 2/3. Точно так же лотерея *D* дает 2/3 шансов, в то время как лотерея C может дать шансов где-то между 1/3 и 3/3. Большинство людей выбирают известную вероятность, а не неизвестную неизвестность.

Еще одна проблема заключается в том, что точная формулировка задачи принятия решения может оказать большое влияние на выбор агента, — это называют ►**эффектом фрейминга**. Эксперименты показывают, что медицинская процедура,

[5] Например, математик и маг Перси Диаконис каждый раз может заставить монетку упасть так, как ему хочется (Ландхуис [1349], 2004).

[6] Даже гарантированные вещи могут не обладать полной достоверностью. Несмотря на "чугунные" обещания, никто еще так и не получил 27 млн долл. с нигерийского банковского счета ранее неизвестного умершего родственника.

которая описывается как имеющая "выживаемость 90%", нравится людям примерно вдвое больше, чем описанная как имеющая "смертность 10%", несмотря на то что оба эти утверждения означают одно и то же. Подобное расхождение в суждениях было обнаружено во множестве экспериментов, и оно проявляло себя примерно одинаково независимо от того, кто именно принимал участие в эксперименте: пациенты клиники, статистически подкованные студенты бизнес-школ или опытные врачи.

Обычно люди чувствуют себя комфортнее, делая *относительные* суждения о полезности, а не абсолютные. Клиент может иметь очень слабое представление о том, насколько хороши разные сорта вин, предлагаемых в ресторане. И ресторан пользуется этим, предлагая вино по 200 долл. за бутылку, которое никто не будет покупать, но которое способствует взвинчиванию оценки клиента в отношении стоимости остальных предлагаемых вин, делая вино по 55 долл. за бутылку вполне приемлемым выбором. Это явление называют ▶**эффектом привязки**.

Если люди-информаторы настаивают на противоречивых суждениях о предпочтениях, не существует ничего, что автоматизированный агент мог бы сделать, чтобы стать совместимым с ними. К счастью, суждения о предпочтениях, сделанные людьми, часто открыты для пересмотра в свете дальнейшего рассмотрения. Парадоксальность выбора в таких экспериментах, как лотереи Алле и Эллсберга, значительно сокращается (но не исчезает), если доступные варианты выбора лучше объясняются. Работая в Гарвардской школе бизнеса по оценке полезности денег, Кини и Райффа ([1213], 1976, с. 210) обнаружили следующее:

> Субъекты, как правило, слишком склонны к риску в малом, и в результате... встроенные функции полезности демонстрируют неприемлемо большие премии за риск в случае лотерей с широким распространением... Однако большинство испытуемых способны убедиться в своей непоследовательности и чувствуют, что усвоили важный урок в отношении того, как им следует поступать. Как следствие некоторые субъекты отменяют свою краткосрочную страховку на случай автомобильной аварии и заключают договор о страховании жизни на более длительный срок.

Свидетельства иррациональности человека также изучались исследователями в области ▶**эволюционной психологии**, которые указывают на тот факт, что механизмы принятия решений нашего мозга еще не эволюционировали в достаточной степени, чтобы решать устные задачи с вероятностями и призами, приведенными в виде десятичных чисел. Допустим в качестве аргумента, что мозг имеет встроенные нейронные механизмы для выполнения вычислений с вероятностями и полезностями или что-то функционально эквивалентное. Если это так, то необходимые входные данные поступали бы за счет накопленного опыта по результатам и наградам, а не посредством лингвистических представлений числовых значений.

Далеко не очевидно, что есть возможность непосредственно получить доступ к встроенным в мозг нейронным структурам, представляя задачи принятия решений

в лингвистической/числовой форме. Сам тот факт, что различные словесные варианты представления *одной и той же задачи принятия решения* приводят к различным вариантам выбора, предполагает, что сама задача принятия решения остается не воспринятой. Вдохновленные этим наблюдением, психологи уже пытались представлять задачи, требующие рассуждений в условиях неопределенности и принятия решений, в "эволюционно подходящих" формах. Например, вместо того чтобы сказать "шансы на выживание равны 90%" экспериментатор может показать 100 контурных изображений операции, на десяти из которых пациент умирает и остается жив на оставшихся 90. В случае, когда задачи принятия решений предлагаются подобным образом, поведение людей, кажется, намного ближе к стандарту рациональности.

16.4. Многоатрибутные функции полезности

Принятие решений в области государственной политики предполагает высокие ставки как в финансовом отношении, так и в вопросах жизни и смерти. Например, принимая решение о том, какие уровни вредных выбросов считать допустимыми для электростанций, лица, устанавливающие эти ограничения, должны сопоставить гарантии предотвращения смертей и инвалидности с выгодой для предприятий и экономическим бременем снижения выбросов. При поиске места для строительства нового аэропорта приходится учитывать, какой вред окружающей среде будет нанесен этим строительством, стоимость земельного участка, расстояние от центров сосредоточения большого количества населения, шум, связанный с деятельностью аэропорта, проблемы безопасности, обусловленные местной топографией и погодными условиями, и т.д. Задачи, подобные этим, в которых результаты характеризуются двумя или несколькими атрибутами, рассматриваются в ▶ **теории многоатрибутной полезности**. По своей сути это теория сравнения яблок с апельсинами.

Обозначим эти атрибуты как $\mathbf{X} = X_1, \ldots, X_n$, и пусть $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ будет полным вектором присваиваний, где каждый x_i представляет собой либо числовое, либо дискретное значение с предполагаемым упорядочением по значениям. Анализ упрощается, если все устроить так, чтобы более высокие значения атрибута всегда соответствовали более высоким полезностям: полезности должны монотонно возрастать. Это означает, например, что в качестве атрибута нельзя использовать количество смертей d, — в данном случае следует использовать значение $-d$. Это также означает, что недопустимо использовать как атрибут температуру в помещении. Если функция полезности для комнатной температуры имеет пик при 22°C и монотонно падает по обе стороны от него, то следует разделить подобный атрибут на две части: использовать значение $t - 22$ для измерения, достаточно ли в комнате тепло, и $22 - t$ — для измерения, не слишком ли в ней холодно. Каждый из этих атрибутов будет монотонно возрастать, пока не достигнет своей максимальной

полезности при значении 0. От этой точки и далее кривые полезности остаются горизонтальными, и это означает, что уже нельзя получить большее значение "достаточно тепло" при увеличении температуры выше 22°C или "достаточно прохладно" при ее снижении ниже 22°C.

В задаче о строительстве нового аэропорта можно выделить следующие атрибуты.

- **Пропускная способность** (*Throughput*), измеряемая количеством рейсов в день.
- **Безопасность** (*Safety*), измеряемая ожидаемым количеством смертей в год, взятым со знаком минус.
- **Беспокойство** (*Quietness*), измеряемое количеством людей, над жилищами которых пролетают самолеты, взятое со знаком минус.
- **Экономичность** (*Frugality*), измеряемая стоимостью строительства, взятой со знаком минус.

Начнем с анализа случаев, в которых решения могут быть приняты *без* комбинирования значений атрибутов в единственное значение полезности. Затем рассмотрим случаи, в которых полезности комбинаций атрибутов могут быть определены очень кратко.

16.4.1. Доминирование

Предположим, что строительство аэропорта на площадке S_1 стоит меньше, обеспечивает меньшее шумовое загрязнение и характеризуется большей безопасностью, чем в случае выбора площадки S_2. Значит, можно фактически без колебаний отвергнуть вариант с площадкой S_2. О такой ситуации говорят, что имеет место ▶**строгое доминирование** варианта S_1 над вариантом S_2. В общем случае, если некоторый вариант характеризуется меньшими значениями всех атрибутов по сравнению с каким-то другим вариантом, нет необходимости продолжать его дальнейшее рассмотрение. Выявление строгого доминирования часто бывает очень полезно, когда требуется сократить перечень вариантов выбора, оставив лишь реальных претендентов, хотя его применение редко приводит к тому, что остается единственный вариант. На рис. 16.4, *а* показана диаграмма для случая с двумя атрибутами — X_1 и X_2.

Такой подход, безусловно, применим для детерминированного случая, когда значения всех атрибутов точно известны. А как поступать в более общем случае, когда следствия представлены неопределенными значениями? В этих случаях можно применять прямой аналог отношения строгого доминирования, в котором, несмотря на неопределенность, все возможные конкретные результаты для S_1 строго доминируют над всеми возможными результатами для S_2 (рис. 16.4, *б*). Конечно, это, вероятно, происходит даже реже по сравнению с детерминированным случаем.

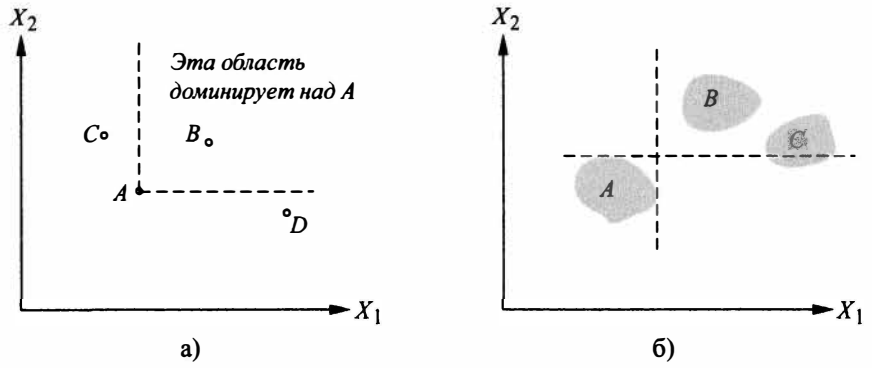


Рис. 16.4. Строгое доминирование. **а)** *Детерминированный случай*: над вариантом *A* строго доминирует вариант *B*, но этого нельзя сказать о вариантах *C* и *D*. **б)** *Неопределенный случай*: над вариантом *A* строго доминирует вариант B, но не вариант *C*

К счастью, есть и более полезное обобщение, называемое ►**стохастическим доминированием**, которое очень часто встречается в реальных задачах. Принцип стохастического доминирования проще понять в контексте задачи с одним атрибутом. Допустим, известно, что стоимость строительства аэропорта на площадке S_1 равномерно распределена в пределах от 2,8 млрд долл. до 4,8 млрд долл., а стоимость его строительства на площадке S_2 равномерно распределена в пределах от 3 млрд долл. до 5,2 млрд долл. Определим атрибут *Экономичность* как стоимость строительства со знаком "минус". На рис. 16.5, *а* показаны распределения для атрибутов экономичности площадок S_1 и S_2. Теперь, руководствуясь лишь информацией о том, что более экономичный выбор имеет большую полезность (все остальные показатели равны), можно сказать, что вариант S_1 стохастически доминирует над S_2 (а значит, вариант S_2 можно отбросить). Важно отметить, что такой вывод *не следует* из сравнения ожидаемых затрат. Например, если бы было известно, что стоимость варианта S_1 равна *точно* 3,8 млрд долл., то мы *не смогли* бы принять решение без дополнительной информации о полезности денег. (Может показаться странным, что *больше* информации о стоимости S_1 может *уменьшить* способность агента принимать решения. Парадокс разрешается, если отметить, что при отсутствии точной информации о стоимости решение принять легче, но с большей вероятностью его ошибочности.)

Точное соотношение между распределениями атрибутов, необходимое для определения стохастического доминирования, проще всего оценить, исследуя кумулятивные распределения, показанные на рис. 16.5, *б*. (Напомним, что в кумулятивном распределении измеряется вероятность того, что стоимость меньше или равна какой-либо заданной сумме.) Если кумулятивное распределение для S_1 всегда находится справа от кумулятивного распределения для S_2, то с точки зрения

стохастической оценки вариант S_1 дешевле, чем S_2. Говоря формальным языком, если два действия, A_1 и A_2, приводят к созданию распределений вероятностей $p_1(x)$ и $p_2(x)$ по атрибуту X, то действие A_1 стохастически доминирует над действием A_2 по атрибуту X, если справедливо следующее соотношение:

$$\forall x \int_{-\infty}^{x} p_1(x')dx' \leq \int_{-\infty}^{x} p_2(x')dx' .$$

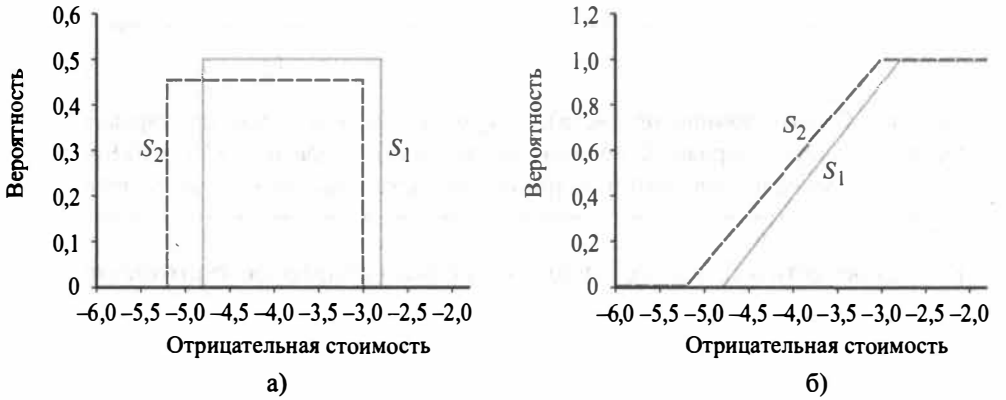


Рис. 16.5. Стохастическое доминирование. **а)** Вариант S_1 стохастически доминирует над вариантом S_2 по стоимости. **б)** Кумулятивные распределения для отрицательной стоимости вариантов S_1 и S_2

Возможность применения этого определения для выбора оптимальных решений вытекает из следующего свойства: ➡ *если действие A_1 стохастически доминирует над действием A_2, то для любой не убывающей монотонно функции полезности $U(x)$ ожидаемая полезность действия A_1 является по меньшей мере такой же высокой, как и ожидаемая полезность действия A_2*. Чтобы понять, почему это так, рассмотрим две ожидаемые полезности, $\int p_1(x)U(x)dx$ и $\int p_2(x)U(x)dx$. Вначале не очевидно, почему первый интеграл больше второго, если известно, что условие стохастического доминирования утверждает, что p_1-интеграл должен быть меньше p_2-интеграла.

Однако, вместо того чтобы рассматривать интеграл по x, давайте рассмотрим интеграл по y, т.е. кумулятивную вероятность, показанную на рис. 16.5, *б*. При любом значении y соответствующее значение x (а следовательно, и $U(x)$) для S_1 будет больше, чем для S_2. Поэтому, при интегрировании больших значений по всему диапазону изменения y, безусловно, будет получен больший результат. Говоря формально, это просто подстановка $y = P_1(x)$ в интеграл для ожидаемого значения S_1 и

$y = P_2(x)$ — в интеграл для ожидаемого значения S_2. При такой замене получим $dy = \frac{d}{dx}(P_1(x))\,dx = p_1(x)dx$ для S_1 и $dy = p_2(x)dx$ — для S_2, следовательно,

$$\int_{-\infty}^{\infty} p_1(x)U(x)dx = \int_0^1 U(P_1^{-1}(y))dy \geq \int_0^1 U(P_2^{-1}(y))dy = \int_{-\infty}^{\infty} p_2(x)U(x)dx\,.$$

Это неравенство позволяет нам A_1 предпочесть A_2 в задаче с одним атрибутом. В более общем случае, если в задаче с несколькими атрибутами над некоторым действием по *всем* атрибутам стохастически доминирует другое действие, то первое можно отбросить.

Состояние стохастического доминирования может показаться довольно специфическим, и оценить его, возможно, будет не так и просто без громоздких вероятностных расчетов. Но на самом деле прийти к подобному заключению очень легко во многих случаях. Например, что вы предпочтете: упасть головой на бетон с высоты 3 миллиметра или с высоты 3 метра? Полагаем, что вы выбрали 3 миллиметра — правильный выбор! Но почему это обязательно лучшее решение? Существует значительная неопределенность в отношении степени ущерба, который будет получен в обоих случаях; но для любого заданного уровня повреждений вероятность того, что вы получите повреждения не менее этого уровня, всегда будет выше при падении с высоты три метра, а не три миллиметра. Другими словами, высота 3 миллиметра стохастически доминирует над высотой 3 метра по атрибуту безопасности.

Подобный вид рассуждений совершенно естественен для человека. Это настолько очевидно, что мы даже не задумываемся об этом. Примеров стохастического доминирования достаточно и в задаче о строительстве аэропорта. Предположим, например, что транспортные расходы при строительстве зависят от расстояний до поставщиков. Сама по себе стоимость остается неопределенной, но чем больше эти расстояния, тем выше стоимость. Если площадка S_1 ближе к поставщикам, чем S_2, то вариант S_1 будет доминировать над S_2 по показателю экономичности. Хотя они здесь не приводятся, существуют алгоритмы распространения качественной информации такого рода среди неопределенных переменных в ►**качественных вероятностных сетях**, позволяющие системе вырабатывать рациональные решения на основе отношений стохастического доминирования без использования каких-либо числовых значений.

16.4.2. Структура предпочтений и многоатрибутная полезность

Предположим, что имеется n атрибутов, каждый из которых имеет d различных возможных значений. Чтобы определить полную функцию полезности $U(x_1, \ldots, x_n)$, в худшем случае потребуется d^n значений. Теория многоатрибутной полезности ставит целью выявление регулярных структур в предпочтениях людей, чтобы

устранить необходимость определения всех значений d^n в отдельности. После выявления некоторой закономерности в выборе предпочтений выводятся ▶**теоремы представления** — для обоснования того, что агент со структурой предпочтений определенного рода имеет следующую функцию полезности:

$$U(x_1, \ldots, x_n) = F[f_1(x_1), \ldots, f_n(x_n)],$$

где F представляет собой (мы надеемся) простую функцию, такую как сложение. Обратите внимание на сходство с использованием байесовских сетей для декомпозиции совместного распределения вероятностей нескольких случайных переменных.

В качестве примера предположим, что каждый атрибут x_i — это сумма денег, которую агент имеет в определенной валюте: доллары, евро, марки, лиры и т.д. Тогда функции f_i могут выполнять конвертирование этих сумм в общую валюту, а функция F будет простым суммированием.

Предпочтения без неопределенности

Начнем с детерминированного случая. В разделе 16.2.2 отмечалось, что для детерминированных вариантов среды у агента имеется функция ценности, записываемая как $V(x_1, \ldots, x_n)$, поэтому здесь наша цель состоит лишь в том, чтобы представить эту функцию в более краткой форме. Основное свойство регулярности, которое наблюдается в детерминированных структурах предпочтений, называется ▶**независимостью предпочтений**. Два атрибута, X_1 и X_2, являются независимыми по предпочтениям от третьего атрибута, X_3, если предпочтение между состояниями $\langle x_1, x_2, x_3\rangle$ и $\langle x_1', x_2', x_3\rangle$ не зависит от конкретного значения x_3 атрибута X_3.

Возвращаясь к примеру с аэропортом, в котором нужно было рассмотреть (кроме других атрибутов) атрибуты *Quietness* (беспокойство), *Frugality* (экономичность) и *Safety* (безопасность), можно предположить, что атрибуты *Quietness* и *Frugality* независимы по предпочтениям от атрибута *Safety*. Например, если мы предпочтем состояние с 20 000 людей, проживающих в районах, над которыми выполняются полеты, и стоимостью строительства 4 млрд долл. состоянию с 70 000 людей, проживающих в районах полетов, и стоимостью 3,7 млрд долл., при том что уровень безопасности в обоих случаях равен 0,006 смертей в расчете на миллиард миль перевозок пассажиров, то будем иметь то же предпочтение, когда уровень безопасности равен 0,012 или когда он равен 0,003, и то же отношение независимости сохранится для предпочтений между любыми другими парами значений атрибутов *Quietness* и *Frugality*. Также очевидно, что атрибуты *Frugality* и *Safety* независимы по предпочтениям от атрибута *Quietness*, а атрибуты *Quietness* и *Safety* независимы по предпочтениям от атрибута *Frugality*.

В подобных случаях говорят, что множество атрибутов {*Quietness*, *Frugality*, *Safety*} обнаруживает ▶**взаимную независимость по предпочтениям** (*Mutual Preferential Independence* — ▶**MPI**). Согласно свойству MPI, независимо от того,

насколько важен каждый атрибут, он не влияет на отношения, в которых другие атрибуты сопоставляются друг с другом.

Взаимная независимость по предпочтениям в определенной степени представляет собой идеализацию, но на ее основе можно вывести очень простую форму для функции ценности агента (Дебре [576], 1960): ➔ *если атрибуты* $X_1, \ldots, X_n$ *являются взаимно независимыми по предпочтениям, то предпочтения агента можно представить функцией ценности*

$$V(x_1, \ldots, x_n) = \sum_i V_i(x_i),$$

где каждое слагаемое V_i *ссылается только на атрибут* X_i. Например, вполне допустим такой вариант, что решение по размещению аэропорта может быть принято на основе следующей функции ценности:

$$V(quietness, frugality, safety) = quietness \times 10^4 + frugality + safety \times 10^{12}.$$

Функция ценности такого типа называется ►**аддитивной функцией ценности**. Аддитивные функции представляют собой исключительно естественный способ описания предпочтений агента и действительно правильно описывают многие реальные ситуации. Для n атрибутов оценка аддитивной функции ценности требует оценки n отдельных одномерных функций ценности вместо одной n-мерной функции. Как правило, это означает экспоненциальное уменьшение количества необходимых экспериментов с предпочтениями. Даже если свойство MPI не соблюдается строго, что может иметь место при крайних значениях атрибутов, аддитивная функция ценности все еще может предоставлять хорошую аппроксимацию для предпочтений агента. Такое утверждение особенно полно оправдывается, когда нарушения свойства MPI возникают в тех частях диапазонов значений атрибутов, которые редко встречаются на практике.

Чтобы лучше понять взаимную независимость по предпочтениям (MPI), будет полезно обратиться к случаям, когда она *не соблюдается*. Предположим, вы находитесь на средневековом рынке, рассматривая возможность покупки нескольких охотничьих собак, нескольких цыплят и нескольких плетеных клеток для цыплят. Охотничьи собаки очень ценны, но если у вас не будет достаточного количества клеток для цыплят, собаки съедят их. Следовательно, компромисс между собаками и цыплятами сильно зависит от количества клеток, которые вы сможете купить, и это нарушает MPI. Существование подобных видов взаимодействий между различными атрибутами значительно усложняет оценку общей функции ценности.

Предпочтения с неопределенностью

Если в рассматриваемой проблемной области присутствует неопределенность, то необходимо также рассмотреть структуру предпочтений между лотереями и понять результирующие свойства функций полезности, а не просто функций ценности. Математические основы решения этой проблемы могут оказаться достаточно

сложными, поэтому здесь мы представим только один из основных результатов, чтобы дать представление о том, как эта проблема может быть решена.

Основное понятие ▶**независимости полезностей** позволяет расширить понятие независимости предпочтений так, чтобы оно охватывало лотереи: множество атрибутов **X** является независимым по полезности от множества атрибутов **Y**, если предпочтения между лотереями по атрибутам **X** независимы от конкретных значений атрибутов **Y**. Множество атрибутов является ▶**взаимно независимым по полезностям** (*Mutually Utility Independent* — MUI), если каждое из его подмножеств является независимым по полезностям от остальных атрибутов. И вновь, предположение о том, что атрибуты задачи о размещении аэропорта обладают свойством MUI, кажется вполне резонным.

Из взаимной независимости по полезностям следует, что поведение агента может быть описано с помощью ▶**мультипликативной функции полезности** (Кини [1212], 1974). Общую форму мультипликативной функции полезности проще всего понять, обратившись к случаю с тремя атрибутами. Для краткости воспользуемся записью U_i для обозначения $U_i(x_i)$:

$$U = k_1U_1 + k_2U_2 + k_3U_3 + k_1k_2U_1U_2 + k_2k_3U_2U_3 + k_3k_1U_3U_1 + \\ + k_1k_2k_3U_1U_2U_3.$$

Хотя это соотношение не выглядит простым, оно содержит лишь три одноатрибутные функции полезности и три константы. В общем случае любую n-атрибутную задачу, характеризующуюся наличием свойства MUI, можно смоделировать с использованием n одноатрибутных полезностей и n констант. Каждая из одноатрибутных функций полезности может быть разработана независимо от других атрибутов, а применение комбинации этих функций гарантирует формирование правильных общих предпочтений. Для получения чисто аддитивной функции полезности потребуется ввести некоторые дополнительные предположения.

16.5. Сети принятия решений

В этом разделе рассматривается общий механизм принятия рациональных решений. Описанную здесь систему обозначений часто называют ▶**диаграммами влияния** (Ховард и Мафесон [1076], 1984) но в этой книге будет использоваться более описательный термин ▶**сети принятия решений**. В сетях принятия решений байесовские сети комбинируются с узлами дополнительных типов, представляющими действия и полезности. В качестве примера будет рассматриваться задача выбора места для строительства аэропорта.

16.5.1. Представление задачи принятия решений с помощью сети принятия решений

В своей наиболее общей форме любая сеть принятия решений представляет информацию о текущем состоянии агента, его возможных действиях, о состоянии, которое станет результатом данного действия агента, и о полезности этого состояния. Таким образом, данная сеть может служить основой для реализации агентов, действующих с учетом полезности, такого типа, который был впервые представлен в разделе 2.4.5. На рис. 16.6 показана сеть принятия решений для задачи выбора площадки для строительства аэропорта. Этот рисунок служит иллюстрацией того, как используются узлы трех описанных ниже типов.

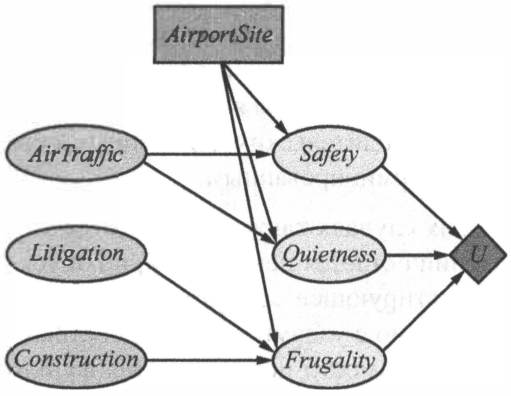


Рис. 16.6. Сеть принятия решений для задачи выбора площадки для строительства аэропорта

- ►**Узлы жеребьевки** (овалы). Представляют случайные переменные, как и в байесовских сетях. Агент может не иметь определенной информации о стоимости строительства (*Construction*), интенсивности воздушного трафика (*AirTraffic*) и возможностях получения разрешения на строительство (*Litigation*), а также о значениях переменных *Safety*, *Quietness* и *Frugality*, поскольку каждое из этих значений зависит от особенностей выбранной площадки. Каждый узел жеребьевки имеет связанное с ним распределение условных вероятностей, которое проиндексировано по состояниям его родительских узлов. В сетях принятия решений родительскими узлами могут быть узлы принятия решений, а также узлы жеребьевки. Обратите внимание, что каждый из узлов жеребьевки в текущем состоянии может входить в состав более крупной байесовской сети, предназначенной для оценки затрат на строительство, интенсивности воздушного трафика или возможностей урегулирования формальностей.

- ▶**Узлы принятия решений** (прямоугольники). Представляют собой точки, в которых лицу, принимающему решение, предоставляется выбор вариантов действий. В данном случае действие *AirportSite* может принимать разные значения для каждой площадки, подлежащей рассмотрению. Этот выбор влияет на безопасность, шумовое загрязнение и общую стоимость строительства. В данной главе предполагается, что нам придется иметь дело только с единственным узлом принятия решений. В главе 17 рассматриваются случаи, в которых потребуется принимать более одного решения.
- ▶**Узлы полезности** (ромбы). Представляют функцию полезности агента.[7] Родительскими переменными узла полезности являются все переменные, описывающие результат, который непосредственно влияет на полезность. С узлом полезности связано описание полезности агента как функции от родительских атрибутов. Это описание может представлять собой табуляцию функции или может быть выражено в виде параметризованной аддитивной или линейной функции от значений атрибутов. На данный момент будем предполагать, что данная функция является детерминированной, т.е. при заданных значениях ее родительских переменных значение полезности узла будет полностью детерминированным.

Кроме того, во многих случаях также применяется упрощенная форма. При этом система обозначений остается неизменной, но опускаются узлы жеребьевки, описывающие результирующее состояние. Вместо этого узел полезности связывается непосредственно с узлами текущего состояния и с узлом принятия решений. В данном случае вместо представления функции полезности от результирующих состояний узел полезности представляет *ожидаемую* полезность, связанную с каждым действием, как было определено в уравнении (16.1), т.е. узел связывается с ▶**функцией "действие–полезность"** (также известной как **Q-функция** в области обучения с подкреплением, как описывается в главе 22). На рис. 16.7 показано представление задачи о размещении аэропорта в форме "действие–полезность".

Обратите внимание на то, что узлы жеребьевки *Quietness*, *Safety* и *Frugality*, показанные на рис. 16.6, ссылаются на будущее состояние, поэтому их значения никогда не должны определяться в виде переменных свидетельства. Следовательно, к упрощенной версии, в которой эти узлы исключены, можно обращаться во всех случаях, когда допустимо использование более общей формы. Однако, несмотря на то, что в упрощенной форме содержится меньше узлов, исключение явного описания результатов решения по выбору площадки означает, что такая сеть является менее гибкой по отношению к возможным изменениям обстоятельств.

Например, на рис. 16.6 изменение допустимых уровней шума от самолетов можно отразить в виде изменения в таблице условных вероятностей, связанной с узлом *Quietness*, тогда как изменение веса, касающегося компонента с описанием

[7] Такие узлы в литературе также иногда называют **узлами ценности**.

шумового загрязнения в функции полезности, может быть отражено с помощью изменения в таблице полезности. С другой стороны, в схеме "действие–полезность", приведенной на рис. 16.7, все такие изменения должны быть отражены как изменения в таблице "действие–полезность". По сути формулировка на основе подхода "действие–полезность" представляет собой *откомпилированную* версию первоначальной формулировки, полученную суммированием переменных состояния результата.

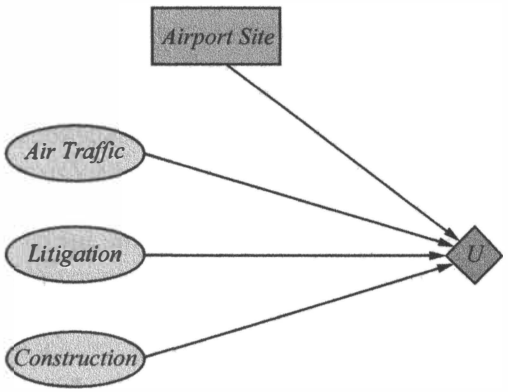


Рис. 16.7. Упрощенное представление задачи выбора площадки для строительства аэропорта. Исключены узлы жеребьевки, соответствующие результирующим состояниям

16.5.2. Вычисления в сетях принятия решений

Действия выбираются посредством проведения в сети принятия решений соответствующих вычислений для каждого возможного ряда значений узла принятия решений. После того как значение узла принятия решений определено, он ведет себя полностью аналогично узлу жеребьевки, которому были присвоены значения по тому же принципу, что и переменной свидетельства. Алгоритм проведения вычислений в сетях принятия решений описан ниже.

1. Задаются значения переменных свидетельства для текущего состояния.
2. Для каждого возможного значения узла принятия решений:
   а) вводится это значение в узел принятия решений;
   б) вычисляются апостериорные вероятности для родительских узлов узла полезности с использованием стандартного алгоритма вероятностного вывода;
   в) вычисляется результирующее значение полезности для данного действия.
3. Возвращается действие с наибольшей полезностью.

Этот алгоритм представляет собой непосредственное расширение алгоритма вычислений в байесовской сети и может быть внедрен непосредственно в проект агента, приведенный на рис. 12.1. Как будет показано в главе 17, эта задача становится намного более интересной, когда существует возможность последовательного выполнения нескольких действий.

16.6. Стоимость информации

В приведенном выше анализе предполагалось, что агенту, прежде чем он приступает к принятию решения, предоставляется вся относящаяся к делу информация или по меньшей мере вся доступная информация. Но на практике такая ситуация возникает чрезвычайно редко. ➔ *Одной из наиболее важных составляющих процесса принятия решений является знание о том, какие вопросы следует задавать.* Например, врач не может рассчитывать на то, что ему будут предоставлены результаты всех возможных диагностических тестов и опросов к тому времени, как пациент впервые войдет в его кабинет. Медицинские тесты часто являются дорогостоящими, а иногда даже опасными (как непосредственно, так и из-за связанных с ними задержек). Важность проведения этих тестов зависит от двух факторов: от того, приведет ли получение результатов этих тестов к выработке существенно лучшего плана лечения, и от того, насколько велика вероятность различных результатов тестов.

В этом разделе описывается ▶ **теория стоимости информации**, которая позволяет агенту выбирать, какую информацию он должен получить. Предполагается, что до выбора "реального" действия, представленного узлом принятия решения, агент может получить значение любой из потенциально наблюдаемых случайных величин в модели. Таким образом, теория стоимости информации включает упрощенную форму последовательного принятия решений, — упрощенную, поскольку действия наблюдения влияют только на **доверительное состояние** агента, а не на внешнее физическое состояние. Ценность любого конкретного наблюдения должна основываться на его способности оказать влияние на возможное физическое действие агента, и эта способность может быть оценена непосредственно из самой модели принятия решения.

16.6.1. Простой пример

Предположим, что нефтедобывающая компания надеется приобрести права на один из n равнозначных по своей перспективности участков для проведения разведочных работ на океанском шельфе. Также примем дополнительное предположение, что из этих участков ровно один обладает запасами нефти, позволяющими получить чистую прибыль в размере C долл., тогда как все остальные не содержат ничего. Запрашиваемая цена участка равна C/n долл. Если

эта компания нейтрально относится к риску, то она должна быть безразлична к выбору между покупкой лицензии на один из участков и отказом от такой покупки.

Теперь допустим, что некий сейсмолог доложил руководству этой компании результаты проведенного им исследования участка номер 3, которые, определенно, указывают на то, что на этом участке имеется нефть. Какую сумму компания должна быть готова заплатить за эту информацию? Один из способов получить ответ на этот вопрос состоит в анализе того, какие действия предприняла бы компания, обладая указанной ниже информацией.

- С вероятностью $1/n$ исследование покажет наличие нефти на участке 3. В этом случае компания купит участок 3 за C/n долл. и получит прибыль в размере $C - C/n = (n-1)C/n$ долл.
- С вероятностью $(n-1)/n$ исследование покажет, что участок не содержит нефти, и в этом случае компания купит другой участок. Теперь вероятность обнаружения нефти на одном из других участков измеряется значением от $1/n$ до $1/(n-1)$, поэтому компания получит ожидаемую прибыль в размере $C/(n-1) - C/n = C/n(n-1)$ долл.

Теперь можно рассчитать ожидаемую прибыль при наличии доступа к информации о результатах исследования:

$$\frac{1}{n} \times \frac{(n-1)C}{n} + \frac{n-1}{n} \times \frac{C}{n(n-1)} = C/n.$$

Таким образом, для компании информация стоит C/n долл. и компания должна быть готова выплатить сейсмологу некоторую значительную часть этой суммы.

Стоимость информации определяется тем фактом, что *при наличии* такой информации можно изменить собственную стратегию таким образом, чтобы она соответствовала *действительной* ситуации. Наличие информации позволяет выявить отличительные особенности рассматриваемой ситуации, а без этой информации в лучшем случае можно лишь найти среднее значение по всем возможным ситуациям. Вообще говоря, стоимость данного конкретного фрагмента информации определяется той разницей в ожидаемых значениях между наилучшими действиями, которая возникает до и после получения этой информации.

16.6.2. Общая формула для полной информации

Общую математическую формулу для стоимости информации получить несложно. Предполагается, что можно получить точное свидетельство о значении некоторой случайной переменной E_j (т.е. мы узнаем, что $E_j = e_j$), поэтому

используется выражение ▶**стоимость полной информации** (*Value of Perfect Information* — VPI).[8]

Пусть исходные знания агента — E, тогда стоимость текущего наилучшего действия α, согласно формуле (16.1), определяется как

$$EU(\alpha) = \max_a \sum_{s'} P(\text{RESULT}(a) = s')U(s'),$$

а стоимостью нового наилучшего действия (после получения нового свидетельства $E_j = e_j$) будет

$$EU(\alpha_{e_j} \mid e_j) = \max_a \sum_{s'} P(\text{RESULT}(a) = s' \mid e_j)U(s').$$

Но E_j — это случайная переменная, значение которой *в настоящее время* неизвестно, поэтому для определения стоимости обнаружения E_j необходимо выполнить усреднение по всем возможным значениям e_j, которые могут быть обнаружены для переменной E_j, с использованием *текущих* степеней уверенности о ее значении:

$$VPI(E_j) = \left(\sum_{e_j} P(E_j = e_j)EU(\alpha_{e_j} \mid E_j = e_j) \right) - EU(\alpha).$$

Чтобы получить определенное интуитивное представление о смысле этой формулы, рассмотрим простой случай, когда имеются только два действия, a_1 и a_2, из которых делается выбор. Текущими ожидаемыми полезностями этих действий являются U_1 и U_2. Получение информации $E_j = e_j$ приведет к некоторым новым значениям ожидаемых полезностей U_1' и U_2' для этих действий, но прежде чем информация E_j будет получена, уже будут существовать определенные распределения вероятностей по всем возможным значениям U_1' и U_2' (которые, согласно принятому предположению, являются независимыми).

Предположим, что действия a_1 и a_2 представляют собой два разных маршрута проезда через горную цепь зимой. Действие a_1 — это движение по скоростной трассе через туннель, а действие a_2 предусматривает проезд по извилистой грунтовой дороге, пересекающей горную цепь. Даже при наличии только этой информации очевидно, что предпочтительным является действие a_1, поскольку весьма вероятно, что проезд по второму маршруту может оказаться невозможным из-за снежных

[8] Требование полной информации не связано с потерей выразительности. Предположим, требуется построить модель для случая, когда появилось больше уверенности в отношении переменной. Можно сделать это, введя другую переменную, о которой известна полная информация. Например, предположим, что изначально имеет место значительная неопределенность в отношении переменной *Temperature* (температура). Затем поступает полная информация о том, что переменная *Thermometer* (термометр) имеет значение 17. Это дает нам неполную информацию об истинном значении переменной *Temperature*, а неопределенность за счет погрешности измерения будет закодирована в модели восприятия **P**(*Thermometer* | *Temperature*). В упражнении 16.22 приведен еще один пример.

заносов, тогда как очень маловероятно, что появятся какие-либо препятствия для проезда по первому маршруту. Следовательно, вполне очевидно, что полезность U_1 выше полезности U_2. Однако возможно, что результаты наблюдений со спутника E_j относительно фактического состояния каждой дороги приведут к получению новых оценок двух маршрутов через горный хребет, U_1' и U_2'. Распределения вероятностей для исходных ожиданий показаны на рис. 16.8, *а*. Очевидно, что в этом случае нет смысла оплачивать получение данных со спутника, поскольку маловероятно, что полученная с их помощью информация приведет к изменению плана. В случае отсутствия изменений информация не имеет какой-либо стоимости.

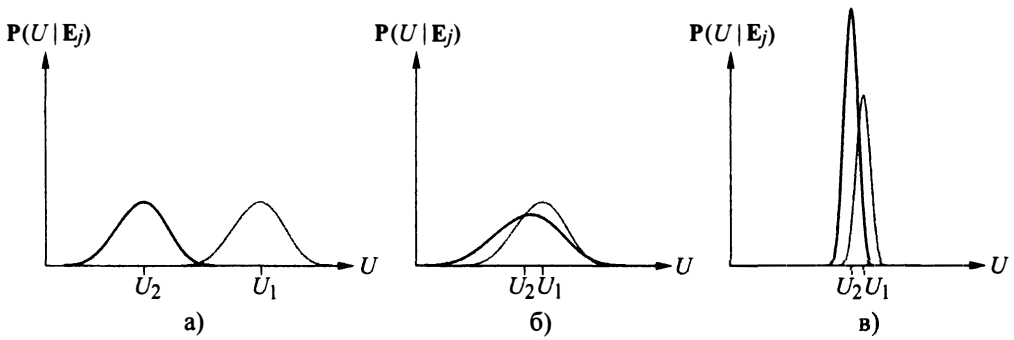


Рис. 16.8. Три типичные ситуации при определении стоимости информации. **а)** Действие a_1 почти с полной определенностью останется лучшим по сравнению с действием a_2, поэтому дополнительная информация не нужна. **б)** Выбор неясен, поэтому информация очень важна. **в)** Выбор неясен, но поскольку разница между полезностями действий невелика, информация не имеет большой ценности. (*Примечание.* Тот факт, что в варианте *в* полезность U_2 имеет существенно более высокий пик, означает, что ее ожидаемое значение известно с гораздо большей достоверностью, чем для полезности U_1)

Теперь предположим, что необходимо выбрать одну из двух извилистых грунтовых дорог, немного различающихся по длине, чтобы вывезти пострадавшего с серьезной травмой. В таком случае, даже если полезности U_1 и U_2 относительно близки, распределения вероятностей U_1' и U_2' будут очень широкими. Существует значительная вероятность того, что второй маршрут окажется свободным, а первый — заблокированным, и в этом случае разность между полезностями будет очень большой. Формула VPI показывает, что в данном случае вполне оправданно получение отчета со спутника. Данная ситуация показана на рис. 16.8, *б*.

Наконец, предположим, что необходимо выбрать одну из двух грунтовых дорог летом, когда вероятность снежных заносов невелика. В этом случае отчеты со спутника могут показать, что одна дорога будет более живописной, чем другая,

поскольку вдоль нее расположены цветущие альпийские луга, или, возможно, одна из дорог будет влажной из-за недавнего дождя. Поэтому весьма вероятно, что при получении подобной информации вполне возможно изменение исходного плана. Однако в этом случае разница в значении полезности между двумя маршрутами, по-видимому, все так же остается небольшой, а значит, в получении отчетов со спутника нет большого смысла. Подобная ситуация показана на рис. 16.8, *в*.

Подводя итог, можно сказать, что ➔ *стоимость информации определяется той степенью, в которой она может вызвать изменение плана, а также той степенью, в какой новый план окажется значительно лучше по сравнению со старым.*

16.6.3. Свойства стоимости информации

Может возникнуть вопрос: возможно ли, чтобы информация была вредной, т.е. может ли она в действительности иметь отрицательную ожидаемую стоимость? Интуиция подсказывает, что такая ситуация невозможна. В конце концов, в наихудшем случае можно просто проигнорировать ненужную информацию и сделать вид, что она так и не была получена. Это представление подтверждается приведенной ниже теоремой, которая применима к любому агенту, действующему на основе теории принятия решений и использующему сеть принятия решений с возможными наблюдениями E_j.

➔ *Ожидаемая стоимость информации является неотрицательной:*

$$\forall j \;\; VPI(E_j) \geq 0.$$

Эта теорема следует непосредственно из определения VPI, и мы оставляем ее доказательство читателю в качестве упражнения (упражнение 16.23). Безусловно, это теорема об *ожидаемой* стоимости, а не о *фактической* стоимости. Дополнительная информация легко может привести к плану, который, как *потом выясняется*, оказывается хуже, чем первоначальный план, если эта информация в действительности вводит в заблуждение. Например, медицинский тест, показавший ложный положительный результат, может привести к ненужной операции; но это вовсе не означает, что тест не следовало делать.

Важно помнить, что стоимость полной информации зависит от текущего состояния информации и что она может изменяться по мере получения дополнительной информации. Для любого заданного элемента свидетельства E_j стоимость его получения может уменьшаться (например, если другая переменная сильно ограничивает апостериорное распределение для E_j) или возрастать (например, если другая переменная дает информацию, на которой основывается E_j, позволяя разработать новый и лучший план). Следовательно, стоимость полной информации не аддитивна, и это означает, что справедливо следующее соотношение:

$$VPI(E_j, E_k) \neq VPI(E_j) + VPI(E_k) \qquad \text{(в общем случае)}.$$

Однако VPI не зависит от порядка следования переменных, т.е. справедливо соотношение

$$VPI(E_j, E_k) = VPI(E_j) + VPI(E_k \mid E_j) = VPI(E_k) + VPI(E_j \mid E_k) = VPI(E_k, E_j),$$

где нотация $VPI(\mid E)$ обозначает стоимость полной информации, рассчитанную в соответствии с апостериорным распределением, где E уже наблюдалось. Независимость по упорядоченности отличает действия восприятия от обычных действий и упрощает задачу расчета значения последовательности действий восприятия. Мы вернемся к этому вопросу в следующем разделе.

16.6.4. Реализация агента, собирающего информацию

Рассудительный агент должен задавать вопросы в разумном порядке, должен избегать вопросов, не имеющих отношения к делу, должен принимать во внимание важность каждого фрагмента в сопоставлении с его стоимостью и должен прекращать задавать вопросы, когда это будет уместно. Все эти возможности могут быть достигнуты за счет использования в качестве критерия стоимости информации.

На рис. 16.9 приведена общая конструкция агента, способного осуществлять интеллектуальный сбор информации, прежде чем приступать к действиям. На данный момент будем предполагать, что с каждой наблюдаемой переменной свидетельства E_j связана соответствующая стоимость $C(E_j)$, отражающая стоимость получения этого свидетельства с помощью проведения тестов, организации консультаций, получения ответов на вопросы или других подобных действий. Агент запрашивает то, что представляется ему наиболее целесообразным наблюдением в терминах полезности, получаемой на единицу стоимости. Предполагается, что результатом действия выдачи запроса $Request(E_j)$ является то, что следующий акт восприятия предоставит значение E_j. Если ни одно наблюдение не оправдывает его стоимость, агент выбирает "реальное" действие.

---

function INFORMATION-GATHERING-AGENT(*percept*) **returns** действие
  persistent: D, сеть принятия решений

  включить данные о восприятии *percept* в сеть D
  $j \leftarrow$ значение, максимизирующее выражение $VPI(E_j) / C(E_j)$
  if $VPI(E_j) > C(E_j)$
    then return $Request\ (E_j)$
  else return наилучшее действие из D

Рис. 16.9. Структура простого близорукого агента, действующего на основе сбора информации. Агент функционирует, выбирая наблюдение с наибольшим информационным значением, и повторяет эти действия до тех пор, пока стоимость следующего наблюдения не превысит его ожидаемую полезность

Представленный здесь алгоритм агента реализует подход к сбору информации, который называют ►**близоруким**. Это связано с тем, что в данном случае формула VPI используется без дальновидных расчетов и значение информации определяется так, как будто было бы достаточно получить значение единственной переменной свидетельства. Близорукий подход основан на той же эвристической идее, что и жадный поиск, и часто хорошо работает на практике. (Например, было показано, что подобные системы превосходят по своей производительности опытных врачей в подборе диагностических тестов.) Однако, если не существует единственной переменной свидетельства, позволяющей достичь значительных результатов, близорукий агент может опрометчиво прейти к действиям, когда было бы лучше сначала сделать запросы по двум или более переменным, а уже потом переходить к действиям. В следующем разделе рассматривается возможность получения многих наблюдений.

16.6.5. Сбор информации по нескольким наблюдениям

Тот факт, что значение последовательности наблюдений не зависит от перестановок в порядке их следования, интригует, но сам по себе не приводит к эффективным алгоритмам оптимального сбора информации. Даже если мы ограничимся выбором для сбора заранее определенного подмножества наблюдений, то существует 2^n таких возможных подмножеств из n потенциальных наблюдений. В общем случае мы сталкиваемся с еще более сложной проблемой нахождения оптимального *условного плана* (как описано в разделе 11.5.2), согласно которому будут выбираться наблюдения, а затем приниматься решения о выполнении действия или продолжении выбора следующих наблюдений — в зависимости от результата. Подобные планы имеют форму деревьев, а количество этих деревьев суперэкспоненциально относительно n.[9]

Как оказалось, для сетей принятия решений задача наблюдения многих переменных является неразрешимой даже в том случае, когда сеть представляет собой полидерево. Однако есть особые случаи, в которых эта задача может быть решена эффективно. Здесь мы приводим один такой случай: задача ►**охоты за сокровищем** (или ►**тестирования последовательности по наименьшей стоимости** (*least-cost testing sequence*) — для менее склонных к романтике). Имеется n ячеек $1,\ldots,n$, и каждая ячейка i может содержать сокровище с независимой вероятностью $P(i)$, а проверка ячейки i имеет стоимость $C(i)$. Это описание соответствует сети принятия решений, где все потенциальные переменные свидетельства $Treasure_i$ являются абсолютно независимыми. Агент проверяет ячейки в некотором порядке, пока не найдет сокровище, и вопрос заключается в том, какой порядок будет оптимальным.

[9] Общая проблема генерации последовательного поведения в частично наблюдаемой среде относится к категории **частично наблюдаемых марковских процессов принятия решений**, которые описываются в главе 17.

Чтобы ответить на этот вопрос, нужно рассмотреть ожидаемые затраты и вероятности успеха различных последовательностей наблюдений, предполагая, что агент останавливается, как только обнаруживает сокровище. Пусть **x** будет такой последовательностью, **xy** будет конкатенацией последовательностей **x** и **y**, $C(\mathbf{x})$ будет ожидаемой стоимостью последовательности **x**, $P(\mathbf{x})$ будет вероятностью того, что последовательность **x** успешно приводит к обнаружению сокровища, а $F(\mathbf{x}) = 1 - P(\mathbf{x})$ будет вероятностью того, что поиск будет неудачным. С учетом всех этих определений, получим

$$C(\mathbf{xy}) = C(\mathbf{x}) + F(\mathbf{x})C(\mathbf{y}), \tag{16.3}$$

т.е. последовательность **xy** будет, безусловно, включать стоимость **x** и, если поиск в ней будет неудачным, будет также включать стоимость **y**.

Основная идея в любой задаче оптимизации последовательности состоит в том, чтобы посмотреть на изменение стоимости, определяемое разностью $\Delta = C(\mathbf{wxyz}) - C(\mathbf{wyxz})$, когда переставляются две соседние подпоследовательности, **x** и **y**, в общей последовательности **wxyz**. Если последовательность является оптимальной, любые подробные перестановки делают ее хуже. Первый этап — показать, что знак эффекта (увеличение или уменьшение стоимости) не зависит от контекста, предоставляемого подпоследовательностями **w** и **z**. Итак, мы имеем

$$\begin{aligned}
\Delta &= [C(\mathbf{w}) + F(\mathbf{w})C(\mathbf{yxz})] - [C(\mathbf{w}) + F(\mathbf{w})C(\mathbf{yxz})] = && \text{(уравнение (16.3))} \\
&= F(\mathbf{w})[C(\mathbf{xyz}) - C(\mathbf{yxz})] = \\
&= F(\mathbf{w})[(C(\mathbf{xy}) + F(\mathbf{xy})C(\mathbf{z})) - (C(\mathbf{yx}) + F(\mathbf{yx})C(\mathbf{z}))] = && \text{(уравнение (16.3))} \\
&= F(\mathbf{w})[C(\mathbf{xy}) - C(\mathbf{yx})] && \text{(поскольку } F(\mathbf{xy}) = F(\mathbf{yx})\text{).}
\end{aligned}$$

Таким образом, мы показали, что направление изменения стоимости по всей последовательности зависит только от направления изменения стоимости пары переставляемых элементов, окружение пары не имеет значения. Это позволяет отсортировать последовательность путем парных сравнений для получения оптимального решения. В частности, теперь у нас есть

$$\begin{aligned}
\Delta &= F(\mathbf{w})[(C(\mathbf{x}) + F(\mathbf{x})C(\mathbf{y})) - (C(\mathbf{y}) + F(\mathbf{y})C(\mathbf{x}))] = && \text{(уравнение (16.3))} \\
&= F(\mathbf{w})[C(\mathbf{x})(1 - F(\mathbf{y})) - C(\mathbf{y})(1 - F(\mathbf{x}))] = \\
&= F(\mathbf{w})[C(\mathbf{x})P(\mathbf{y}) - C(\mathbf{y})P(\mathbf{x})].
\end{aligned}$$

Это выражение справедливо для любых последовательностей **x** и **y**, в том числе и тогда, когда **x** и **y** являются одиночными наблюдениями в ячейках i и j соответственно. Значит, для того чтобы i и j были смежными в оптимальной последовательности, необходимо иметь $C(i)P(j) \le C(j)P(i)$, или $\dfrac{P(i)}{C(i)} \ge \dfrac{P(j)}{C(j)}$. Другими словами, оптимальный порядок ранжирует ячейки согласно вероятности успеха на единицу стоимости.

16.6.6. Анализ чувствительности и надежные решения

Практика проведения ▶**анализа чувствительности** широко распространена в технологических дисциплинах: под этим подразумевается анализ того, насколько результаты процесса изменяются при небольших изменениях параметров модели. Анализ чувствительности в вероятностных системах и системах принятия решений особенно важен, поскольку используемые вероятности обычно либо изучаются на основе данных, либо оцениваются экспертами-людьми, а это означает, что они сами по себе подвержены значительной неопределенности. Лишь в редких случаях — например, как при бросках игральных костей при игре в нарды — эти вероятности объективно известны.

Для процесса принятия решений на основе полезности полученный результат можно понимать либо как фактическое принятие решения, либо как ожидаемую полезность от этого решения. Сначала рассмотрим последний случай: поскольку ожидания зависят от вероятностей в модели, можно вычислить производную от ожидаемой полезности любого заданного действия по отношению к каждому из этих значений вероятности. (Например, если все распределения условных вероятностей в модели явно сведены в таблицу, вычисление ожидания включает в себя вычисление отношения двух выражений, в которых суммируются произведения; подробнее об этом читайте в главе 20.) Следовательно, можно определить, какие параметры в модели оказывают наибольшее влияние на ожидаемую полезность окончательного решения.

Если, напротив, мы заинтересованы в фактическом принятии решения, а не в оценке его полезности согласно модели, то можно просто систематически изменять параметры (возможно, с помощью бинарного поиска), чтобы увидеть, будет ли меняться решение, и если будет, то что именно является наименьшим возмущением, вызывающим такое изменение. Конечно, можно полагать, что не важно, какое именно решение принято, важна только его полезность. Это так, но на практике разница между *реальной* полезностью решения и его полезностью *в соответствии с моделью* может оказаться очень существенной.

Если все разумные возмущения параметров оставляют оптимальное решение без изменений, то разумно предположить, что это решение является хорошим, даже если оценка полезности для этого решения является по существу неверной. С другой стороны, если оптимальное решение значительно изменяется при изменении параметров в модели, то есть большая вероятность, что модель выдает решение, которое в действительности является существенно неоптимальным. В этом случае стоит приложить дополнительные усилия для доработки модели.

Эти интуитивно понятные рассуждения были формализованы в нескольких областях (теория управления, анализ решений, управление рисками) с введением понятия ▶**надежного** или **минимаксного** решения, т.е. такого, которое дает наилучший результат в наихудшем случае. Здесь "наихудший случай" означает наихудший по отношению ко всем правдоподобным изменениям значений

параметров модели. Если принять, что θ обозначает все параметры модели, то надежные решения определяются следующим образом:

$$a^* = \text{argmax}_a \min_\theta EU(a; \theta).$$

Во многих случаях, особенно в теории управления, надежный подход приводит к решениям, которые очень устойчиво и бесперебойно работают на практике. В других случаях это приводит к чрезмерно консервативным решениям. Например, при проектировании беспилотного автомобиля надежный подход требовал бы предполагать наихудший вариант поведения со стороны всех других транспортных средств на дороге, т.е. все они должны восприниматься как управляемые маньяками-убийцами. В такой ситуации самым оптимальным решением для машины будет остаться в гараже.

Байесовская теория принятия решений предлагает альтернативу надежным методам: если существует неопределенность в отношении параметров модели, то включите эту неопределенность в модель, используя гиперпараметры.

В то время как надежный подход мог бы показать, что некоторая вероятность θ_i в модели может иметь значение где-то между 0,3 и 0,7, с фактическим значением, выбранным противником так, чтобы сделать происходящее настолько плохим, насколько это возможно, байесовский подход построил бы априорное распределение вероятностей для θ_i, а затем продолжил бы работу, как прежде. Такой подход требует бо́льших усилий при моделировании, например разработчик модели байесовской сети должен решить, будут ли параметры θ_i и θ_j независимыми, но на практике это часто приводит к более высокой производительности.

Помимо параметрической неопределенности, приложения теории принятия решений в реальном мире также страдают от *структурной* неопределенности. Например, допущение о независимости переменных *AirTraffic*, *Litigation* и *Construction* в сети на рис. 16.6 может быть неверным, а также могут существовать дополнительные переменные, которые в модели просто опущены. В настоящее время еще нет достаточно хорошего понимания в отношении того, как принять этот вид неопределенности во внимание. Одна из возможностей состоит в том, чтобы сохранить ансамбль моделей, возможно, сгенерированных с помощью алгоритмов машинного обучения, в надежде на то, что этот ансамбль захватит все значительные вариации, которые имеют значение.

16.7. Неизвестные предпочтения

В этом разделе мы обсудим, что происходит, когда существует неопределенность в отношении функции полезности, ожидаемое значение которой должно быть оптимизировано. Есть две версий этой проблемы: одна, когда агент (машина или человек) находится в состоянии неопределенности относительно его *собственной* функции полезности, и другая, в которой машина, как она полагает,

должна помочь человеку, но находится в неопределенности относительно того, что человек хочет.

16.7.1. Неопределенность в отношении собственных предпочтений

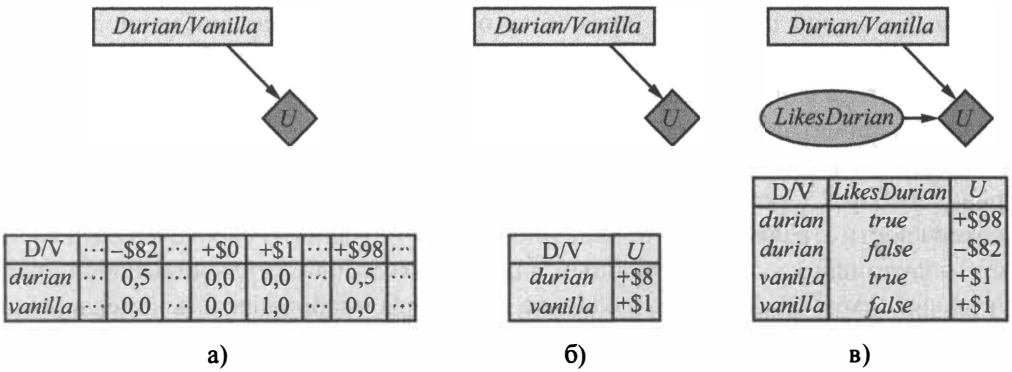
Представьте, что вы находитесь в магазине мороженого в Таиланде, и у них осталось только два его сорта: ваниль и дуриан. Оба стоят 2 долл. Вы знаете о своей умеренной симпатии к ванили и готовы платить до 3 долл. за ванильное мороженое в такой жаркий день, так что вас ждет "чистая прибыль" в 1 долл. при выборе ванили. С другой стороны, вы не имеете представления о том, понравится вам мороженое с дурианом или нет, но в Википедии вы читали, что в отношении дуриана мнения разных людей сильно расходятся: одни считают, что "его вкус превосходит вкус всех других плодов мира", в то время как другие уподобляют его "сточным водам, несвежей рвоте, зловонию скунса и использованным хирургическим тампонам".

Чтобы связать с этим сценарием какие-то цифры, давайте договоримся, что с вероятностью 50% вы найдете вкус этого мороженого восхитительным (+ 100 долл.) и с вероятностью 50% — что вы его возненавидите (– 80 долл., если его привкус сохранится на всю вторую половину дня). Здесь нет никакой неопределенности в отношении того, какой приз вы собираетесь выиграть — в любом случае это то же самое мороженое с дурианом, — но здесь есть неопределенность по поводу ваших личных предпочтений в отношении этого приза.

Можно расширить формальное математическое представление сетей принятия решений так, чтобы допустить возможность наличия неопределенности в отношении полезности, как показано на рис. 16.10, *а*. Однако если больше нет никакой возможности получить информацию по поводу ваших предпочтений относительно дуриана — например, если в магазине не соглашаются позволить вам сначала попробовать это мороженое, — то задача принятия решения будет идентична той, которая показано на рис. 16.10, *б*. Можно просто заменить неопределенную стоимость дуриана ожидаемой "чистой прибылью" в (0,5 × 100 долл.) – (0,5 × × 80 долл.) – 2 долл. = 8 долл., и ваше решение останется неизменным.

Если случится так, что вы измените свои представления о дуриане — скажем, вам удастся попробовать крошечный кусочек или вы вспомните, что все ваши ближайшие родственники любят дуриан, — то трансформированная сеть, представленная на рис. 16.10, *б*, станет недействительной. Однако, как выяснилось, все еще есть возможность найти эквивалентную модель, в которой функция полезности будет детерминированной. Вместо того чтобы говорить о наличии неопределенности в отношении функции полезности, мы, так сказать, перемещаем эту неопределенность "в мир". Для этого создается новая случайная переменная *LikesDurian* с теми же вероятностями 0,5 для значений *true* и *false*, как показано на рис. 16.10, *в*. При наличии этой дополнительной переменной функция полезности становится

детерминированной, но при этом появляется возможность учесть вероятные изменения в ваших представлениях о своих предпочтениях относительно дуриана.



D/V	···	–$82	···	+$0	+$1	···	+$98	···
durian	···	0,5	···	0,0	0,0	···	0,5	···
vanilla	···	0,0	···	0,0	1,0	···	0,0	···

а)

D/V	*U*
durian	+$8
vanilla	+$1

б)

D/V	*LikesDurian*	*U*
durian	*true*	+$98
durian	*false*	–$82
vanilla	*true*	+$1
vanilla	*false*	+$1

в)

Рис. 16.10. а) Сеть принятия решения для задачи выбора мороженого при неопределенности в отношении функции полезности. **б)** Сеть с ожидаемой полезностью каждого действия. **в)** Перемещение неопределенности из функции полезности в новую случайную величину

Тот факт, что неизвестные предпочтения можно смоделировать с помощью обычных случайных переменных, означает, что мы можем продолжать использовать весь аппарат и все теоремы, разработанные для случая известных предпочтений. С другой стороны, это не означает, что мы всегда можем полагать, что предпочтения известны. Неопределенность все еще существует и все еще влияет на то, как должны вести себя агенты.

16.7.2. Уважение к людям

Теперь давайте обратимся ко второму случаю, упомянутому выше: к машине, которая должна помогать человеку, но не уверена в том, чего именно хочет человек. Полное рассмотрение этой ситуации следует отложить до главы 18, в которой обсуждаются решения, выполняемые с участием более чем одного агента. Здесь же мы зададим лишь один простой вопрос: при каких обстоятельствах такая машина будет подчиняться человеку?

Чтобы изучить этот вопрос, давайте рассмотрим очень простой сценарий, представленный на рис. 16.11. Здесь Робби (***R***) — это программный робот, работающий на Гарриет (***H***), занятую бизнес-леди, в качестве ее личного помощника. Гарриет нужен номер в гостинице для ее следующей деловой встречи в Женеве. В данный момент Робби может действовать — допустим, может забронировать для Гарриет номер в очень дорогом отеле рядом с местом проведения встречи. Но он не имеет уверенности в том, насколько Гарриет понравится этот отель или его цены.

Допустим, что в отношении возможной оценки отеля со стороны Гарриет он принял равномерное распределение вероятностей в диапазоне от –40 до +60 со средним +10. Также у него есть возможность "отключиться" — говоря менее театрально, просто выйти из процедуры бронирования гостиничного номера, — исход, который мы определяем (не теряя общего смысла) как имеющий ценность 0 для Гарриет. Если бы это были два единственных варианта его выбора, он начал бы действовать и забронировал номер в гостинице, взяв на себя значительный риск сделать Гарриет несчастной. (Если бы диапазон был от –60 до +40 при среднем –10, он бы вместо этого просто отключился.) Однако мы дадим Робби третий вариант выбора: доложить свой план и перейти в ожидание, предоставив Гарриет возможность отключить его от выполнения этого задания. В свою очередь, Гарриет, ознакомившись с предложенным планом, может либо отключить Робби, либо позволить ему продолжить выполнение процедуры и забронировать номер в отеле. Может возникнуть вопрос: а что во всем этом хорошего, если известно, что Робби мог бы сделать оба эти выбора и сам?

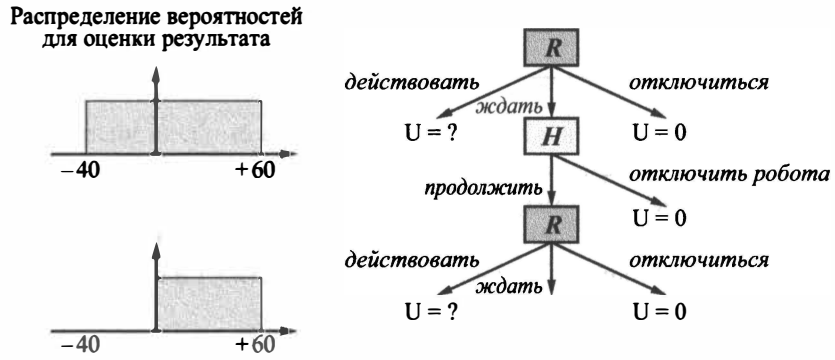


Рис. 16.11. Игра с отключением. ***R***, робот, может выбрать действовать сейчас при крайне неопределенном результате, может отключиться от выполнения задания и может переложить решение на ***H***, человека. У ***H*** есть выбор — отключить ***R*** от выполнения задания или позволить ему продолжить работу. В последнем случае у ***R*** вновь будут те же возможности выбора, однако, хотя результат действия все еще остается неопределенным, теперь ***R*** точно знает, что он не будет отрицательным

Дело в том, что выбор Гарриет — отключить Робби от выполнения задания или позволить ему продолжить — предоставляет Робби важную информацию о предпочтениях Гарриет. В данном случае мы предполагаем, что поведение Гарриет рационально, поэтому, если она разрешила Робби продолжить выполнение задания, то это означает, что Гарриет положительно оценивает его действия. Теперь, как показано на рис. 16.11, представления Робби о возможной оценке его действий меняются: это равномерное распределение в диапазоне от 0 до +60 со средним значением +30.

Итак, давайте оценим первоначальный выбор Робби, исходя из его точки зрения.

1. Решение действовать немедленно и забронировать номер в отеле имеет ожидаемое значение оценки +10.
2. Решение отключиться от выполнения задания имеет ожидаемое значение оценки 0.
3. Решение перейти в ожидание и позволить Гарриет отключить его от выполнения задания приводит к двум возможным результатам:
   а) исходя из выбранного Робби распределения вероятностей в отношении предпочтений Гарриет, с вероятностью 40% предложенный план ей не понравится, и она отключит Робби от выполнения задания, что даст ожидаемое значение оценки 0.
   б) с вероятностью 60% предложенный план Гарриет понравится, и она позволит Робби продолжить выполнение задания, что дает ожидаемое значение оценки +30.

Таким образом, переход в ожидание имеет ожидаемое значение оценки $(0{,}4 \times 0) + (0{,}6 \times 30) = +18$, что лучше, чем +10, ожидающих Робби, если он сразу начнет действовать.

В результате у Робби появляется положительный стимул предоставить Гарриет возможность сделать выбор, т.е. позволить себе стать отключенным ею. Этот стимул исходит непосредственно из неуверенности Робби в отношении предпочтений Гарриет. Робби знает, что есть шанс (40% в данном примере), что он может принять решение сделать что-то, что сделает Гарриет несчастной, и в этом случае отключение от выполнения задания предпочтительнее его выполнения. Если бы Робби уже был в полной уверенности о предпочтениях Гарриет, он просто принял бы решение и выполнил задание (или отключился от его выполнения), — ему было бы абсолютно нечего получить от консультации с Гарриет, поскольку, согласно определенным убеждениям Робби, он уже может точно предсказать, какое решение она собирается принять.

В действительности тот же результат можно доказать и для общего случая: пока Робби не совсем уверен в том, что он собирается сделать то, что сама Гарриет сделала бы, ему лучше позволить ей отключить его. Интуитивно понятно, что ее решение предоставляет Робби информацию, и ожидаемая ценность этой информации всегда будет неотрицательной. И наоборот, если Робби уверен в том, какое решение примет Гарриет, ее решение не предоставит ему никакой новой информации, и поэтому у Робби нет никакого стимула позволить ей принимать решения.

При формальном подходе пусть $P(u)$ будет априорной плотностью вероятности Робби в отношении полезности предлагаемого действия a для Гарриет. Тогда значением полезности в случае выполнения им действия a будет

$$EU(a) = \int_{-\infty}^{\infty} P(u) \cdot u du = \int_{-\infty}^{0} P(u) \cdot u du + \int_{0}^{\infty} P(u) \cdot u du.$$

(Скоро будет показано, почему интеграл разделяется именно таким образом.) С другой стороны, значение полезности действия d, решение о выполнении которого возлагается на Гарриет, будет состоять из двух частей: если $u > 0$, то Гарриет позволит Робби продолжить работу, поэтому значением полезности является u, но если $u < 0$, то Гарриет отключит Робби от выполнения задания и значение полезности будет равно 0:

$$EU(d) = \int_{-\infty}^{\infty} P(u) \cdot 0 du = \int_{0}^{\infty} P(u) \cdot u du.$$

Сравнив выражения для $EU(a)$ и $EU(d)$, сразу можно заметить, что

$$EU(d) \geq EU(a),$$

поскольку выражение для $EU(d)$ имеет нулевую полезность в области отрицательных значений. Эти два варианта имеют одинаковое значение только в том случае, когда отрицательная область имеет нулевую вероятность, т.е. когда Робби уже уверен, что Гарриет нравится предлагаемое им действие.

Существуют некоторые очевидные уточнения модели, которые стоит обсудить немедленно. Первым уточнением является наложение на Робби штрафов за расходование времени Гарриет. В этом случае Робби будет менее склонен беспокоить Гарриет, если риск отрицательной оценки будет небольшим. Именно так и должно быть. А если Гарриет при этом очень раздражается, когда ее отрывают от дела, то она не должна слишком удивляться тому, что Робби иногда будет делать что-то, что ей не понравится.

Вторым уточнением является допущение некоторой вероятности ошибки человека, т.е. Гарриет может иногда отключать Робби даже в тех случаях, когда предлагаемое им действие разумно, и наоборот, иногда она может позволить Робби продолжить работу, даже если предложенное им действие нежелательно. Включить эту вероятность ошибки в модель совсем несложно. Как и следовало ожидать, такое решение приводит к тому, что Робби становится менее склонным считаться с иррациональностью Гарриет, которая иногда действует против ее собственных интересов. Чем более непредсказуемым становится ее поведение, тем больше должна возрастать неуверенность Робби в отношении ее предпочтений, прежде чем возложить решение на нее. И вновь, именно так и должно быть. Например, если Робби представляет собой беспилотный автомобиль, а Гарриет — это его озорной двухлетний пассажир, Робби не должен позволить Гарриет отключить его посреди скоростного шоссе.

Резюме

В этой главе показано, как объединить теорию полезности с теорией вероятностей, чтобы предоставить агенту возможность выбрать действия, которые максимизируют его ожидаемую производительность.

- **Теория вероятностей** описывает, в чем должен быть уверен агент согласно полученному свидетельству, **теория полезности** показывает, к чему должен стремиться агент, а **теория принятия решений** позволяет объединить подходы этих двух теорий для определения того, что должен делать агент.
- Теорию принятия решений можно использовать для создания систем, которые принимают решения, рассматривая все возможные действия и выбирая из них именно то, которое приводит к наилучшему ожидаемому результату. Такая система известна под названием **рациональный агент**.
- Теория полезности показывает, что агент, руководствующийся отношениями предпочтения между лотереями, совместимыми с множеством простых аксиом, может быть описан как обладающий функцией полезности. Более того, этот агент выбирает действия так, чтобы можно было максимизировать его ожидаемую полезность.
- **Теория многоатрибутной полезности** связана с изучением полезности, которая зависит от нескольких разных атрибутов состояний. **Стохастическое доминирование** представляет собой особенно удобный метод принятия непротиворечивых решений даже при отсутствии точных значений полезности для атрибутов.
- **Сети принятия решений** являются простым формальным математическим представлением для описания и решения задач принятия решений. Они являются естественным расширением байесовских сетей и, кроме узлов жеребьевки, содержат узлы решения и узлы полезности.
- Иногда при решении задачи, прежде чем принимать решение, приходится заниматься поиском дополнительной информации. **Стоимость информации** определена как ожидаемое повышение полезности по сравнению с принятием решений без этой информации. Этот подход особенно полезен для управления процессом сбора информации до принятия окончательного решения.
- Когда, как это часто бывает, невозможно полностью и корректно определить функцию полезности для человека, машинам приходится работать в условиях неопределенности о его истинной цели. Ситуация существенно меняется в том случае, если у машины имеется возможность получить больше информации о предпочтениях человека. На простом примере было показано, что неопределенность относительно предпочтений обеспечивает тот факт, что машины полагаются на человека, — вплоть до того, что позволяют себе предоставить ему возможность их отключения.

Библиографические и исторические заметки

В трактате Арно *L'art de Penser* или *Port-Royal Logic* XVII века ([74], 1662) говорится:

> Для того чтобы судить о том, что нужно сделать, чтобы получить хорошее или избежать плохого, необходимо учитывать не только хорошее и плохое в самом себе, но и вероятность того, что это произойдет или не произойдет, и геометрически рассматривать пропорции, которые все они имеют совместно.

Современные тексты говорят о *полезности*, а не о хорошем и плохом, но в этом утверждении правильно отмечается, что, чтобы "судить, что нужно сделать", следует умножить ("геометрически рассмотреть") полезность на вероятность, получив ожидаемую полезность, и максимизировать это по всем результатам ("все они"). Примечательно то, насколько Арно был прав более 350 лет назад и всего через 8 лет после того, как Паскаль и Ферма впервые показали, как правильно использовать вероятности.

Даниил Бернулли ([191], 1738), исследуя Санкт-Петербургский парадокс (см. упражнение 16.4), впервые понял важность измерения предпочтений в отношении лотерей и написал такие слова: "*Ценность* любого предмета должна быть основана не на его *стоимости*, а на той *пользе*, которую он может принести" (курсив Бернулли). Философ-утилитарист Джереми Бентам ([176], 1823) предложил ▶**гедонистическое исчисление** для взвешивания "удовольствий" и "неприятностей", доказывая, что все решения (а не только касающиеся денег) можно свести к сравнению полезностей.

Введение Бернулли понятия полезности как внутреннего, субъективного качества для объяснения поведения человека с помощью математической теории было чрезвычайно примечательным предложением для своего времени. И оно было тем более примечательным благодаря тому факту, что, в отличие от денежных сумм, значения полезности различных ставок и призов непосредственно не наблюдались. Вместо этого полезности следовало выводить из предпочтений, проявляемых индивидом. Потребовалось два столетия, чтобы полностью разработать все следствия этой идеи и чтобы она во всей широте была принята статистиками и экономистами.

Определение числовых значений полезности на основе предпочтений было впервые выполнено Рамзеем ([1848], 1931); аксиомы предпочтений, приведенные в этой книге, по своей форме ближе всего тем, которые были вновь открыты в книге *Theory of Games and Economic Behavior* (фон Нейман и Моргенштерн [2282], 1944). Рамзей предложил способ вычисления субъективных вероятностей (а не только полезностей) на основе предпочтений агента; Сэведж ([1984], 1954) и Джеффри ([1129], 1983) предложили более современные вычислительные конструкции такого рода. Бердо и соавт. ([150], 2002) показали, что функции полезности недостаточно для представления нетранзитивных предпочтений и других аномальных ситуаций.

В послевоенный период теория принятия решений стала стандартным инструментом в экономике, финансах и науке управления. Возникла новая область ▶**анализа решений**, направленная на поиск более рациональных решений при выборе линии поведения в таких областях, как военная стратегия, медицинская диагностика, здравоохранение, инженерные проекты и управление ресурсами. Процесс включает ▶**принимающего решения**, утверждающего предпочтения между результатами, и ▶**аналитика решений**, определяющего возможные действия, вычисляющего их результаты и получающего сведения о предпочтениях от принимающего решения с целью определения наилучшей последовательности действий. Фон Винтерфельдт и Эдвардс ([2283], 1986) предоставили детальное исследование области анализа решений и ее связи со структурами предпочтений человека. Смит в [2093] (1988) дает обзор методологии в области анализа решений.

До 1980-х годов основным инструментом, применяемым при работе с многомерными задачами принятия решений, было построение "деревьев решений", включающих все возможные конкретизации переменных. Сети принятия решений, или диаграммы влияния, использующие все преимущества свойств условной независимости, предоставляемые байесовскими сетями, были разработаны Говардом и Матесоном ([10769], 1984) на основе одной из ранних работ, выполненных группой специалистов (включая Говарда и Матесона) в институте SRI (Миллер и др. [1577], 1976). Алгоритм Говарда и Матесона обеспечивает формирование полного (имеющего экспоненциальные размеры) дерева решений на основе сети принятия решений. Шахтер ([2028], 1986) разработал метод принятия решений, основанный на использовании непосредственно сети принятия решений, без создания промежуточного дерева решений. Этот алгоритм оказался также одним из первых алгоритмов, обеспечивающих полный вероятностный вывод в многосвязных байесовских сетях. В работе Нильссона и Лауритцена [1685] (2000) показано, как алгоритмы для сетей принятия решений связаны с продолжающимися разработками в области алгоритмов кластеризации для байесовских сетей. В сборнике статей Оливера и Смита [1712] (1990) приведен целый ряд полезных статей по сетям принятия решений, как и в специальном выпуске журнала Networks, вышедшем в 1990 году. Учебник Фентона и Нейла [727] (2018) представляет собой отличное руководство по решению реальных задач принятия решений. Статьи по сетям принятия решений и моделированию полезностей также регулярно публикуются в журналах *Management Science* и *Decision Analysis*.

Как это ни удивительно, лишь немногие исследователи в области искусственного интеллекта взяли на вооружение инструментальные средства теории принятия решений после появления первых приложений принятия решений в медицине, ранее описанных в главе 12. Одним из немногих исключений был Джерри Фельдман, применивший теорию принятия решений в задачах машинного зрения (Фельдман и Якимовски [718], 1974) и планирования (Фельдман и Спроулл [717], 1977). Экспертные системы на базе правил конца 1970-х и начала 1980-х годов в большей степени предназначались для предоставления ответов на вопросы, чем для

принятия решений. А те системы, которые действительно давали рекомендации в отношении действий, обычно работали на базе правил "условие–действие", а не явного представления результатов и предпочтений.

Сети принятия решений предлагают гораздо более гибкий подход, например, позволяя предпочтениям изменяться, при сохранении модели переходов неизменной, или наоборот. Они также принципиально допускают расчеты, какую информацию искать дальше. В конце 1980-х годов, отчасти из-за работы Перла по байесовским сетям, получили широкое распространение теоретико-экспертные системы принятия решений (Хорвиц и др. [1068], 1988; Коуэлл и др. [485], 2002). Примечателен тот факт, что с 1991 года на обложке журнала *Искусственный интеллект* была изображена сеть принятия решений, хотя с направлениями стрелок на ней все же были допущены некоторые художественные вольности.

Практические попытки измерения оценок полезности у людей начались еще в послевоенные годы на первой стадии разработки теории анализа принятия решений (см. выше). Такая единица измерения полезности, как микроморт, обсуждалась Говардом ([1075], 1989). Талер в работе [2197] (1992) установил, что при шансе погибнуть 1/1000 респондент обычно отказывается платить больше чем 200 долл. за устранение этого риска, но при этом не соглашается получить 50 000 долл., чтобы взять на себя этот риск.

Использование такого показателя, как **QALY** (Quality-Adjusted Life Years — *годы жизни с поправкой на качество*), для анализа экономической выгоды от медицинского вмешательства и соответствующей социальной политики можно проследить в прошлое по крайней мере до работы Кларман и соавт. [1237] (1968), хотя сам по себе этот термин впервые был использован Зекхаузером и Шепардом в работе [2421] (1976). Как и деньги, показатель QALY прямо соответствует полезности только при соблюдении довольно строгих допущений, таких как нейтральное отношение к риску, что в действительности часто нарушается (Березняк и др. [178], 2015); тем не менее этот показатель очень широко используется на практике, например при формировании политики Национальной службы здравоохранения в Великобритании. Расселл в работе [1937] (1990) приводит типичный пример аргументации в пользу серьезных изменений в политике общественного здравоохранения на основании ожидаемой полезности, измеряемой в QALY.

Кини и Райффа в [1213] (1976) дают введение в **многоатрибутную теорию полезности**. Они описывают ранние компьютерные реализации методов выявления необходимых параметров для многоатрибутной функции полезности и приводят множество сообщений о реальных приложениях теории. Аббас в [4] (2018) рассматривает многие достижения начиная с 1976 года. Эта теория впервые была введена в области ИИ в работе Вэллмана [2315] (1985), который также исследовал использование стохастического доминирования и качественных вероятностных моделей (Вэллман [2315], (1988); [2316], 1990). Вэллман и Дойл ([2319], 1992) предоставили предварительный набросок того, как сложное множество независимых по полезности отношений может быть использовано для представления

структурированной модели функции полезности, — во многом аналогично тому, как байесовские сети могут быть использованы для представления распределений совместных вероятностей. Бакхус и Гроув ([97], 1995; [98], 1996) и Ла Мура и Шохам ([1333], 1999) освещают дальнейшие результаты работ по этим направлениям. Бутилье и соавт. в [269] (2004) описывают CP-сети, полностью разработанную графическую модель формального математического представления для условных *ceteribus paribus* (при прочих равных условиях) утверждений предпочтения.

К понятию **проклятия оптимизатора** внимание аналитиков решения с большим напором было привлечено Смитом и Винклерлом ([2094], 2006), указавшими, что прогнозируемые аналитиками финансовые выгоды клиента для предлагаемых ими вариантов действий почти никогда не материализуются. Они связали это непосредственно с предвзятостью, введенной путем выбора оптимального действия, и показали, что более полный байесовский анализ устраняет эту проблему.

Харрисон и Март ([969], 1984) ту же основную концепцию назвали ▶**разочарованием после принятия решения**, а Браун ([319], 1974) выявил эту проблему в контексте анализа проектов капиталовложений. Проклятие оптимизатора также тесно связано с ▶**проклятием победителя** (Капе и др. [366], 1971; Талер [2197], 1992), которое применимо к конкурсным торгам на аукционах: кто бы ни выиграл аукцион, очень вероятно, что он переоценил стоимость рассматриваемого объекта. В своей работе Капе и соавт. процитировали слова инженера-нефтяника на тему проведения торгов по правам: "Если кто-то выиграет пакет у двух или трех противников, то он сможет почувствовать себя счастливым в связи со своей удачей. Но что он почувствует, если выиграет у 50 противников? Огорчение".

Парадокс Алле, предложенный лауреатом Нобелевской премии Морисом Алле ([30], 1953), был экспериментально проверен, и результаты показали, что люди постоянно непоследовательны в своих суждениях (Тверски и Канеман [2239], 1982; Конлиск [469], 1989). Парадокс Эллсберга о неоднозначности неприятия был представлен в докторской диссертации Даниэля Эллсберга ([682], 1962).[10] Фокс и Тверски ([765], 1995) описывают последующее исследование в отношении отвращения к неоднозначности. Махина ([1471], 2005) приводит общий обзор в отношении выбора в условиях неопределенности и объясняет, как он может варьироваться согласно теории ожидаемой полезности. В отношении углубленного анализа предпочтений при неопределенности стоит обратить внимание на классический учебник Кини и Райффы ([1213], 1976) и недавнюю работу Аббаса ([4], 2018).

Большим годом для популярных книг о человеческой ▶**иррациональности** стал 2009 год — вышли такие издания, как *Predictably Irrational* (Ариэли [69], 2009), *Sway* (Брафман и Брафман [286], 2009), *Nudge* (Талер и Сунстейн [2198], 2009), *Kluge* (Маркус [1491], 2009), *How We Decide* (Лерер [1380], 2009) и *On Being Certain*

[10] Позже Эллсберг стал военным аналитиком корпорации RAND, и связанная с этим утечка информации, известная как "документы Пентагона", способствовала окончанию Вьетнамской войны.

(Бартон [351], 2009). Все эти издания дополняют классическую книгу *Judgment Under Uncertainty* (Канеман и др. [1172], 1982) и статью, с которой все началось (Канеман и Тверски [1173], 1979). Сам Канеман предложил глубокое и легко читаемое обсуждение предмета в *Thinking: Fast and Slow* (Канеман [1171], 2011).

С другой стороны, в области эволюционной психологии было высказано противоположное относительно этой литературы мнение: в работе Басса [352] (2005) утверждается, что люди вполне рациональны в эволюционно соответствующих контекстах. Его сторонники указывают, что в эволюционном контексте иррациональность наказывается по определению, и показывают, что в некоторых случаях отмеченные факты являются следствием экспериментальной процедуры (Камминс и Аллен [507], 1998). В свое время было отмечено возрождение интереса к байесовской модели познания после десятилетий пессимизма (Элио [680], 2002; Чатер и Оаксфорд [401], 2008; Гриффитс и др. [923], 2008); однако это возрождение не осталось без внимания недоброжелателей (Джонс и Лав [1144], 2011).

Теория стоимости информации вначале исследовалась в контексте статистических экспериментов, в которых использовалась квазиполезность (сокращение энтропии) (Линдли [1416], 1956). Специалист в области теории управления, Руслан Стратонович, в [2144] (1965) разработал более общую теорию, представленную в этой книге, где стоимость информации устанавливается в соответствии с ее способностью влиять на решения. Работа Стратоновича была неизвестна на Западе, где Рон Ховард ([2144], 1966) развил ту же идею. Его статья завершалась следующим замечанием: "Если теория стоимости информации и связанные с ней структуры теории принятия решений в будущем не охватят бо́льшую часть образования инженеров, то в отношении профессии инженера вскоре обнаружится, что ее традиционная роль управления научными и экономическими ресурсами на благо человека была утрачена в пользу другой профессии". На сегодняшний день подобной революции в методах управления так и не произошло.

Близорукий алгоритм сбора информации, описанный в этой главе, повсеместно упоминается в литературе по анализу решений; его основные контуры можно разглядеть в оригинальной статье о диаграммах влияния (Говард и Матесон [1076], 1984). Эффективные расчетные методы изучались Диттмером и Йенсеном ([625], 1997). Ласки ([1356], 1995) и Нильсен и Йенсен ([1681], 2003) обсуждают методы анализа чувствительности в байесовских сетях и сетях принятия решений соответственно. В классическом учебнике *Robust and Optimal Control* (Чжоу и др. [2437], 1995) предоставляется полное раскрытие темы и приводится сравнение надежных и теоретических подходов к принятию решений при неопределенности.

Задача поиска сокровищ была независимо решена многими авторами, по крайней мере начиная с работ по последовательному тестированию Гласса ([871], 1959) и Миттена ([1601], 1960). Стиль доказательства в этой главе опирается на основной результат, представленный Смитом ([2101], 1956), при котором значение последовательности связывается со значением той же последовательности при перестановке двух смежных элементов. Эти результаты для независимых тестов были расширены

на более общие задачи поиска по дереву и графику (где тесты частично упорядочены) Каданом и Симоном ([1163], 1977). Результаты по сложности не близоруких (*nonmyopic*) вычислений стоимости информации были получены Краузе и Гестрином ([1307], 2009). Краузе и соавт. ([1308], 2008) идентифицировали случаи, когда субмодулярность (*submodularity*) приводит к разрешимой аппроксимации алгоритма, опираясь на плодотворную работу Немхаузера и соавт. ([1662], 1978) о субмодулярных функциях. Краузе и Гестрин ([1306], 2005) выявили случаи, когда точный алгоритм динамического программирования предоставляет эффективное решение как для выбора подмножества свидетельства, так и для генерации условного плана.

Гарсани ([971], 1967) исследовал проблему *неполной* информации в теории игр, когда игроки могут не знать точно функцию вознаграждения друг друга. Он показал, что такие игры были идентичны играм с *частично наблюдаемой* информацией, в которых игроки находятся в неопределенности о состоянии мира, с помощью приема добавления переменных состояния, ссылающихся на вознаграждение игроков. Сайерт и де Гроот ([512], 1979) разработали теорию ▶**адаптивной полезности**, в которой агент может быть не уверен в своей функции полезности и может получить больше информации через опыт.

Работа по выявлению байесовских предпочтений (Чаевска и др. [383], 2000; Бутилье [265], 2002) начинается с принятия функции априорной вероятности полезности агента. Ферн и соавт. ([732], 2014) предлагают базирующуюся на теории принятия решений модель ▶**помощи**, в которой робот пытается выяснить цель человека, относительно которой он изначально находится в неопределенности, и помочь в ее достижении. Пример с отключением в разделе 16.7.2 взят из статьи Хетфилда-Менелла и соавт. ([943], 2017). Рассел ([1942], 2019) предлагает общую структуру для благотворного ИИ, в которой игра на отключение является основным примером.

Упражнения

16.1. Это упражнение построено на игре Almanac Game (игра на знание фактов из справочника), используемой аналитиками решений для калибровки свих числовых оценок. На каждый из приведенных ниже вопросов дайте наилучший предполагаемый вами ответ, т.е. найдите число, которое, по вашему мнению, с такой же вероятностью является слишком большим, с какой оно может быть слишком малым. Кроме того, приведите свою гипотезу с оценкой на уровне 25-й процентили, т.е. такое число, которое, по вашему мнению, имеет 25% шансов на то, чтобы быть слишком высоким, и 75% шансов на то, чтобы быть слишком низким. Наконец, приведите такую же оценку значения и для 75-й процентили. (Таким образом, вы должны дать всего три оценки для каждого вопроса — низкую, среднюю и высокую.)

а) Количество пассажиров, которые совершали полеты между Нью-Йорком и Лос-Анджелесом в 1989 году.

б) Население Варшавы в 1992 году.

в) Год, в который испанский конкистадор Коронадо открыл реку Миссисипи.

г) Количество голосов, полученных Джимми Картером во время президентских выборов в 1976 году.
д) Возраст самого старого живого дерева по состоянию на 2002 год.
е) Высота плотины Гувера (Hoover Dam) в футах.
ж) Количество яиц, произведенных в штате Орегон в 1985 году.
з) Количество буддистов в мире в 1992 году.
и) Количество смертных случаев из-за СПИДа в Соединенных Штатах в 1981 году.
к) Количество американских патентов, выданных в 1901 году.

Правильные ответы приведены в конце этой главы. С точки зрения анализа решений интересно не то, насколько ваши средние предположения подошли к реальным ответам, а скорее то, насколько часто реальный ответ попадал в установленные вами границы 25 и 75%. Если такая ситуация возникала примерно в половине случаев, то указанные вами границы были достаточно точными. Но если вы похожи на большинство людей, то проявите больше самоуверенности, чем следует, и более половины ответов выйдет за пределы этих границ. Постоянно практикуясь, вы сможете откалибровать свои оценки, чтобы устанавливаемые пределы стали более реалистичными и тем самым приносили больше пользы при предоставлении информации для принятия решений. Попробуйте ответить на второй ряд вопросов и определите, достигнуты ли вами какие-либо улучшения.

а) Год рождения актрисы За За Габор (Zsa Zsa Gabor).
б) Максимальное расстояние от Марса до Солнца в милях.
в) Стоимость в долларах пшеницы, экспортированной из Соединенных Штатов в 1992 году.
г) Количество тонн груза, обработанных в порту Гонолулу в 1991 году.
д) Годовая заработная плата в долларах губернатора Калифорнии в 1993 году.
е) Население города Сан-Диего в 1990 году.
ж) Год, в котором Роджер Уильямс основал г. Провиденс, штат Род-Айленд.
з) Высота горы Килиманджаро в футах.
и) Длина Бруклинского моста в футах.
к) Количество смертных случаев из-за автомобильных аварий в Соединенных Штатах в 1992 году.

16.2. Крис рассматривает пять подержанных автомобилей, прежде чем купить тот, который принесет ему максимальную полезность. Пэт рассматривает десять автомобилей с то же самой целью. При прочих равных условиях у кого из них больше шансов купить машину лучше, чем у другого? Кто из них с большей вероятностью будет разочарован качеством своего автомобиля? И на сколько именно (с точки зрения стандартных отклонений от ожидаемого качества)?

16.3. Крис рассматривает пять подержанных автомобилей, прежде чем купить тот, который принесет ему максимальную полезность. Пэт рассматривает одиннадцать автомобилей с то же самой целью. При прочих равных условиях, у кого из них больше шансов купить машину лучше, чем у другого? Кто из них с большей вероятностью будет разочарован качеством своего автомобиля? И насколько именно (с точки зрения стандартных отклонений от ожидаемого качества)?

16.4. В 1713 году Николай Бернулли исследовал санкт-петербургский парадокс, который заключается в следующем. У вас есть возможность сыграть в игру, в которой подлинная монета подбрасывается повторно до тех пор, пока не выпадет орлом вверх. Если орел впервые появится на n-м броске, вы выигрываете 2^n монет.

а) Покажите, что ожидаемая денежная ценность этой игры является бесконечно большой.

б) Сколько бы лично вы заплатили за участие в этой игре?

в) В 1738 году Даниил Бернулли, двоюродный брат Николая, разрешил кажущийся парадокс, связанный с нежеланием людей участвовать в этой игре, несмотря на ее привлекательность, выдвинув предположение, что полезность денег измеряется логарифмической шкалой (т.е. $U(S_n) = a\log_2 n + b$, где S_n — денежное состояние человека, т.е. наличие у него n монет). Какова ожидаемая полезность этой игры согласно указанному предположению?

г) Каковая максимальная сумма, которую было бы разумно заплатить, чтобы сыграть в эту игру, при условии, что исходное состояние равно k монет?

16.5. Напишите компьютерную программу для автоматизации процесса, описанного в упражнении 16.11. Проверьте работу вашей программы на нескольких людях с разным собственным капиталом и различными политическими взглядами. Прокомментируйте результаты сравнения согласованности полученных данных как для отдельного лица, так и для разных лиц.

16.6. Компания "Карамельки с сюрпризом" выпускает конфеты двух вкусов: 75% — со вкусом клубники и 25% — со вкусом анчоуса. Каждой новой карамельке придается шарообразная форма, после чего она попадает на производственный конвейер. По мере продвижения ленты машина случайным образом выбирает определенный процент конфет, которым придается кубическая форма. Далее все конфеты оборачивается в обертку, цвет которой выбирается случайным образом — красный или коричневый. В итоге 70% клубничных конфет имеют круглую форму и 70% имеют красную обертку, тогда как 90% анчоусных конфет имеют квадратную форму и 90% имеют коричневую обертку. Все конфеты продаются поштучно в одинаковых черных закрытых коробочках. Итак, теперь вы — покупатель в магазине и только что купили конфету с сюрпризом, но еще не открыли коробочку. Рассмотрим три байесовские сети, приведенные на рис. 16.12, с узлами *Wrapper* (обертка), *Flavor* (вкус), *Shape* (форма).

а) Какая сеть (сети) может правильно представить вероятность $P(Flavor, Wrapper, Shape)$?

б) Какая сеть лучше всего подходит для этой задачи?

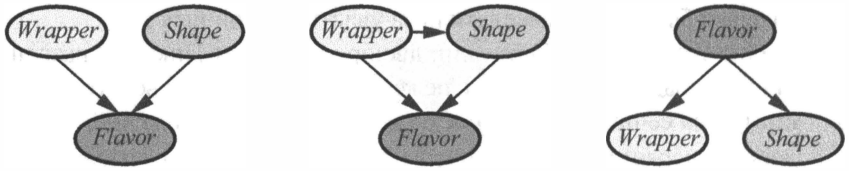
в) Утверждает ли сеть на рис. 16.12, *a*, что $P(Wrapper \mid Shape) = P(Wrapper)$?

г) Какова вероятность того, что ваша конфета имеет красную обертку?

д) В коробочке находится круглая конфетка с красной оберткой. Какова вероятность того, что ее вкус клубничный?

е) Развернутая конфета с клубничным вкусом на уличном рынке стоит s, а развернутая конфета со вкусом анчоуса — a. Напишите выражение для стоимости нераскрытой коробочки с конфетой.

ж) Новый закон запрещает торговлю конфетами без обертки, но по-прежнему разрешает торговать конфетами без упаковки (вынутыми из коробочки). Будет ли теперь нераскрытая коробочка с конфетой стоить больше либо меньше, или же она останется в той же цене, что и раньше?



а) б) в)

Рис. 16.12. Три варианта байесовской сети, предлагаемые для задачи о карамельках с сюрпризом

16.7. Компания "Карамельки с сюрпризом" выпускает конфеты двух вкусов: 70% — со вкусом клубники и 30% — со вкусом анчоуса. Каждой новой карамельке придается шарообразная форма, после чего она попадает на производственный конвейер. По мере продвижения ленты машина случайным образом выбирает определенный процент конфет, которым придается кубическая форма. Далее все конфеты оборачивается в обертку, цвет которой выбирается случайным образом — красный или коричневый. В итоге 80% клубничных конфет имеют круглую форму и 80% имеют красную обертку, тогда как 90% анчоусных конфет имеют квадратную форму и 90% имеют коричневую обертку. Все конфеты продаются поштучно в одинаковых черных закрытых коробочках. Итак, теперь вы — покупатель в магазине и только что купили конфету с сюрпризом, но еще не открыли коробочку. Рассмотрим три байесовские сети, приведенные на рис. 16.12, с узлами *Wrapper* (обертка), *Flavor* (вкус), *Shape* (форма).

а) Какая сеть (сети) может правильно представить вероятность $P(Flavor, Wrapper, Shape)$?

б) Какая сеть лучше всего подходит для этой задачи?

в) Утверждает ли сеть на рис. 16.12, *а*, что $P(Wrapper \mid Shape) = P(Wrapper)$?

г) Какова вероятность того, что ваша конфета имеет красную обертку?

д) В коробочке находится круглая конфетка с красной оберткой. Какова вероятность того, что ее вкус клубничный?

е) Развернутая конфета с клубничным вкусом на уличном рынке стоит *s*, а развернутая конфета со вкусом анчоуса — *a*. Напишите выражение для стоимости нераскрытой коробочки с конфетой.

ж) Новый закон запрещает торговлю конфетами без обертки, но по-прежнему разрешает торговать конфетами без упаковки (вынутыми из коробочки). Будет ли теперь нераскрытая коробочка с конфетой стоить больше либо меньше, или же она останется в той же цене, что и раньше?

16.8. Докажите, что суждения $B \succ A$ и $C \succ D$ в парадоксе Алле (раздел 16.3.4) нарушают аксиому заменяемости.

16.9. Рассмотрим парадокс Алле, описанный в разделе 16.3.4: агент, который предпочитает лотерею B лотерее A (принимая гарантированный выигрыш) и лотерею C — лотерее D (выбирая более высокую полезность), согласно теории полезности действует нерационально. Как вы думаете, это указывает на проблему для агента, на проблему для теории или здесь вообще нет проблемы? Поясните свой ответ.

16.10. Лотерейные билеты стоят 1 долл. В этой лотерее возможны два приза: выплата 10 долл. с вероятностью 1/50 и выплата 1 000 000 долл. с вероятностью 1/2 000 000. Какова ожидаемая денежная ценность лотерейного билета? Когда (если вообще когда-либо) будет рационально купить такой билет? Дайте точный ответ — представьте уравнение, включающее полезности. При этом можете принять размер текущего состояния k и полезность $U(S_k) = 0$, а также можете предположить, что полезность $U(S_{k+10}) = 10 \times U(S_{k+1})$, но не можете делать каких-либо предположений в отношении полезности $U(S_{k+1\,000\,000})$. Социологические исследования показывают, что люди с более низким доходом покупают непропорционально большое количество лотерейных билетов. Как вы думаете, это потому, что в качестве принимающих решения они действуют хуже, или потому, что они имеют другую функцию полезности? Рассмотрите ценность обдумывания возможности выиграть в лотерею в сравнении с обдумыванием возможности стать героем происходящего на экране во время просмотра приключенческого фильма.

16.11. Проанализируйте свою оценку полезности различных постепенно увеличивающихся сумм денег, выполнив ряд проверок предпочтения между некоторой определенной суммой M_1 и лотереей $[p, M_2; (1-p), 0]$. Выбирайте различные значения M_1 и M_2 и варьируйте вероятность p до тех пор, пока для вас выбор из этих двух вариантов не станет безразличным. Представьте полученную в результате функцию полезности в виде графика.

16.12. Какова стоимость микроморта лично для вас? Разработайте определенный протокол, позволяющий узнать это значение. Задавайте вопросы, основанные как на том, сколько вы готовы заплатить, чтобы избежать риска смерти, так и на том, сколько вы готовы принять в качестве платы за то, чтобы взять на себя этот риск.

16.13. Пусть непрерывные переменные $X_1, \ldots, X_k$ имеют независимое распределение в соответствии с одной и той же функцией плотности вероятности $f(x)$. Докажите, что функция плотности вероятности для $\max\{X_1, \ldots, X_k\}$ задается как $kf(x)(F(x))^{k-1}$, где F — кумулятивное распределение для f.

16.14. Экономисты часто используют экспоненциальную функцию полезности для денег: $U(x) = -e^{-x/R}$, где R — положительная константа, представляющая толерантность конкретного человека к риску. Толерантность к риску отражает, насколько вероятно, что человек может принять лотерею с определенной ожидаемой денежной ценностью (EMV) в сравнении с определенной выплатой за участие

в ней. По мере того как значение R (измеряемое в тех же единицах, что и x) становится больше, человек становится менее склонным к риску.

а) Предположим, что для Мэри в экспоненциальной функции полезности $R = 500$ долл. Мэри предоставляется право сделать выбор между гарантированным получением 500 долл. (вероятность 1) и участием в лотерее с вероятностью 60% выиграть 5000 долл. и вероятностью 40% не выиграть ничего. Если предположить, что Мэри действует рационально, то какой вариант она выберет? Покажите, как вы пришли к своему ответу.

б) Проанализируйте выбор между гарантированным получением 100 долл. (вероятность 1) и участием в лотерее с вероятностью 50% выигрыша 500 долл. и вероятностью 50% не выиграть ничего. Укажите приблизительное (до трех значащих цифр) значение коэффициента R в экспоненциальной функции полезности, при котором человек будет безразличен к этим двум альтернативам. (Возможно, имеет смысл написать короткую программу, которая поможет решить эту задачу.)

16.15. Экономисты часто используют экспоненциальную функцию полезности для денег: $U(x) = -e^{-x/R}$, где R — положительная константа, представляющая толерантность конкретного человека к риску. Толерантность к риску отражает, насколько вероятно, что человек может принять лотерею с определенной ожидаемой денежной ценностью (EMV) в сравнении с определенной выплатой за участие в ней. По мере того как значение R (измеряемое в тех же единицах, что и x) становится больше, человек становится менее склонным к риску.

а) Предположим, что для Мэри в экспоненциальной функции полезности $R = 400$ долл. Мэри предоставляется право сделать выбор между гарантированным получением 400 долл. (вероятность 1) и участием в лотерее с вероятностью 60% выиграть 5000 долл. и вероятностью 40% не выиграть ничего. Если предположить, что Мэри действует рационально, то какой вариант она выберет? Покажите, как вы пришли к своему ответу.

б) Проанализируйте выбор между гарантированным получением 100 долл. (вероятность 1) и участием в лотерее с вероятностью 50% выигрыша 500 долл. и вероятностью 50% не выиграть ничего. Укажите приблизительное (до трех значащих цифр) значение коэффициента R в экспоненциальной функции полезности, при котором человек будет безразличен к этим двум альтернативам. (Возможно, имеет смысл написать короткую программу, которая поможет решить эту задачу.)

16.16. Алексу предложено сделать выбор между двумя играми. В игре 1 подбрасывается честная монета, и, если выпадает орел, Алекс получает 100 долл. Если выпадает решка, Алекс ничего не получает. В игре 2 честная монета подбрасывается дважды. Каждый раз, когда выпадает орел, Алекс получает 50 долл., и каждый раз, когда выпадает решка, Алекс не получает ничего. Предположим, что для Алекса функция полезности денег в диапазоне [0, 100] долл. является монотонно возрастающей. Математически покажите, что если Алекс предпочитает игру 2 игре 1, то он не склонен к риску (по крайней мере в отношении этого

диапазона денежных сумм). Покажите, что если X_1 и X_2 являются независимыми по предпочтению от X_3 и если X_2 и X_3 являются независимыми по предпочтению от X_1, то X_3 и X_1 являются независимыми по предпочтению от X_2.

16.17. Выполните упражнение 16.21, используя представление "действие–полезность", приведенное на рис. 16.7.

16.18. Для какой из двух диаграмм размещения аэропорта, упоминаемых в упражнениях 16.21 и 16.17, значения в таблице условной вероятности являются собственно точной полезностью с учетом имеющихся свидетельств?

16.19. Модифицируйте и дополните программы байесовской сети, приведенные в репозитории кода, чтобы обеспечить создание и оценку сетей принятия решений, а также вычисление стоимости информации.

16.20. Рассмотрим ситуацию студента, которому нужно сделать выбор: купить или не купить учебник для курса. Смоделируем ее как задачу принятия решения с одним булевым узлом принятия решения, B, указывающим, принял ли агент решение купить книгу, и двумя булевыми узлами жеребьевки: узлом M, указывающим, освоил ли студент материал в учебнике, и узлом P, указывающим, прошел ли студент этот курс. Безусловно, имеется также узел полезности U. Некий студент Сэм имеет следующую аддитивную функцию полезности: 0 долл. — не покупать книгу и –100 долл., если купить ее, 2000 долл. — за прохождение курса и 0 долл. — за его не прохождение. Оценки условной вероятности Сэма следующие.

$$P(p \mid b, m) = 0{,}9 \qquad P(m \mid b) = 0{,}9$$
$$P(p \mid b, \neg m) = 0{,}5 \qquad P(m \mid \neg b) = 0{,}7$$
$$P(p \mid \neg b, m) = 0{,}8$$
$$P(p \mid \neg b, \neg m) = 0{,}3$$

Можно решить, что P является независимым от B при заданном M, однако при сдаче экзамена по этому курсу разрешается использовать литературу, так что наличие учебника может оказаться полезным.

а) Нарисуйте сеть принятия решений для этой задачи.

б) Вычислите ожидаемую полезность от покупки книги и от отказа от ее покупки.

в) Как следует поступить Сэму?

16.21. В этом упражнении завершается анализ задачи выбора площадки для размещения аэропорта, приведенной на рис. 16.6.

а) Приведите приемлемые области определения переменных, значения вероятностей и полезностей для этой сети при условии, что имеются три возможные площадки для постройки аэропорта.

б) Решите эту задачу принятия решений.

в) Что произойдет, если будут внедрены такие технологические усовершенствования, что каждый самолет будет издавать вдвое меньше шума?

г) Что произойдет, если требования по снижению уровня шума станут в три раза более жесткими?

д) Рассчитайте значение VPI для переменных *AirTraffic* (интенсивность воздушного трафика), *Litigation* (возможности получения разрешения на строительство) и *Construction* (стоимость строительства) в вашей модели.

16.22. Покупатель подержанного автомобиля может принять решение выполнить различные проверки с разной стоимостью (например, постучать по шинам, показать автомобиль квалифицированному механику), а затем, в зависимости от результата этих проверок, принять решение о том, какой автомобиль следует купить. Предположим, что покупатель должен принять решение о покупке автомобиля c_1 и при этом готов провести не более одной проверки t_1 автомобиля c_1 стоимостью 50 долл. Автомобиль может находиться в хорошем состоянии (качество q^+) или в плохом состоянии (качество q^-), а проверка может показать, в каком состоянии находится автомобиль. Автомобиль c_1 стоит 1500 долл., а его рыночная стоимость равна 2000 долл., если он находится в хорошем состоянии. Если же нет, то потребуется ремонт стоимостью 700 долл., чтобы привести его в хорошее состояние. По оценке покупателя, вероятность того, что автомобиль c_1 находится в хорошем состоянии, составляет 70%.

а) Нарисуйте сеть принятия решений, представляющую эту задачу.

б) Рассчитайте ожидаемую чистую прибыль от покупки автомобиля c_1, если проверка не проводится.

в) Проверку можно описать посредством оценки вероятности того, что автомобиль пройдет или не пройдет данную проверку, если известно, что автомобиль находится в хорошем или плохом состоянии. Имеется следующая информация:

$$P(pass(c_1, t_1) \mid q^+(c_1)) = 0{,}8;$$
$$P(pass(c_1, t_1) \mid q^-(c_1)) = 0{,}35.$$

Примените теорему Байеса для вычисления вероятности того, что автомобиль успешного пройдет (или не пройдет) проверку, и, следовательно, вероятности того, что он находится в хорошем (или плохом) состоянии, с учетом каждого возможного результата проверки.

а) Рассчитайте оптимальные решения при условии прохождения или не прохождения проверки, а также их ожидаемые полезности.

б) Рассчитайте стоимость информации от проверки и разработайте оптимальный условный план для покупателя.

16.23. Вспомните определение стоимости информации, приведенное в разделе 16.6.

а) Докажите, что стоимость информации неотрицательна и не зависит от порядка следования восприятий.

б) Объясните, почему некоторые предпочли бы не получать определенную информацию, например не желают узнать пол своего ребенка при проведении УЗИ.

в) Функция f на множествах является субмодулярной, если для любого элемента x и любых множеств A и B, таких, что $A \subseteq B$, добавление элемента x к множеству A дает большее увеличение значения f, чем добавление x к мно-

жеству B: $A \subseteq B \Rightarrow (f(A \cup \{x\}) - f(A)) \geq (f(B \cup \{x\}) - f(B))$. Субмодулярность отражает интуитивное понятие *убывающей отдачи*. Является ли стоимость информации, рассматриваемой как функция f на множествах возможных наблюдений, субмодулярной? Докажите это или приведите контрпример.

Ответы к упражнению 16.1

- Первый ряд вопросов: 3 000 000; 1 600 000; 1541; 41 000 000; 4768; 221; 649 000 000; 295 000 000; 132; 25 546.
- Второй ряд вопросов: 1917; 155 000 000; 4 500 000 000; 11 000 000; 120 000; 1 100 000; 1636; 19 340; 1595; 41 710.

ГЛАВА 17

Принятие сложных решений

В данной главе рассматриваются методы принятия решений о том, что следует делать сегодня, учитывая, что завтра может потребоваться принять другое решение.

В этой главе обсуждаются расчеты, связанные с принятием решений в стохастическом окружении. В главе 16 речь шла о задачах принятия единоразовых или эпизодических решений, в которых полезность результата каждого действия была вполне известна, а в данной главе рассматриваются ▶**задачи последовательного принятия решений**, в которых полезность действий агента зависит от последовательности решений. Задачи последовательного принятия решений объединяют обработку полезностей, неопределенностей и результатов восприятия и включают особые случаи задач поиска и планирования. В разделе 17.1 объясняется, как определяются задачи последовательного принятия решений, а в разделе 17.2 описываются методы их решения с целью выработки оптимальных правил поведения, соответствующих стохастической окружающей среде. В разделе 17.3 рассматриваются задачи "**многоруких бандитов**" — особый и привлекательный класс задач последовательного принятия решений, возникающих во многих контекстах. В разделе 17.4 исследуются задачи принятия решений в частично наблюдаемых вариантах среды, а в разделе 17.5 описывается, как эти задачи решаются.

17.1. Задачи последовательного принятия решений

Предположим, что агент находится в среде размером 4×3, показанной на рис. 17.1, *а*. Начиная с исходного состояния START, в каждом временном интервале он должен выбирать какое-то действие. Взаимодействие со средой оканчивается, когда агент достигает одного из целевых состояний, обозначенных на рисунке как +1 и −1. Как и в задачах поиска, действия, доступные агенту в каждом состоянии, задаются функцией ACTIONS(s), которую мы иногда будем сокращать до $A(s)$. В среде 4×3 в каждом состоянии возможны действия *Up* (вверх), *Down* (вниз), *Left* (влево) и *Right* (вправо). Будем предполагать, что эта среда является **полностью наблюдаемой**, поэтому агент всегда знает, где он находится.

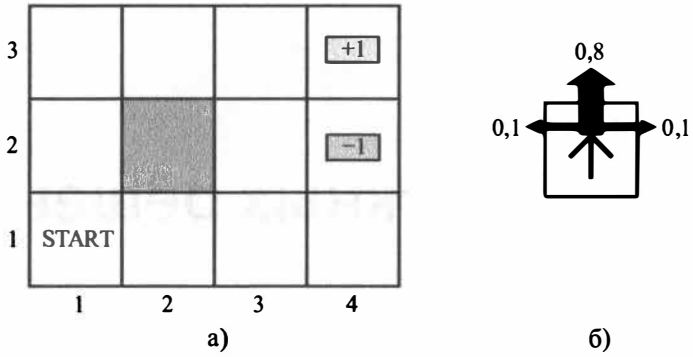


Рис. 17.1. а) Простая стохастическая среда размером 4 × 3, в которой перед агентом поставлена задача последовательного принятия решений. **б)** Модель перехода для этой среды: "намеченный" результат достигается с вероятностью 0,8, а с вероятностью 0,2 агент движется под прямым углом влево или вправо от намеченного направления. Столкновение со стеной приводит к прекращению движения. Перемещение в два конечных состояния приводит к получению вознаграждений +1 и −1 соответственно, а перемещение в любое другое состояние связано с вознаграждением −0,04

Если бы эта среда была детерминированной, то достичь требуемого результата было бы несложно: [*Up*, *Up*, *Right*, *Right*, *Right*]. К сожалению, среда не всегда способствует осуществлению этого решения, поскольку действия в ней являются ненадежными. Конкретная, принятая в данном случае модель стохастического движения показана на рис. 17.1, *б*. Каждое действие достигает намеченной цели с вероятностью 0,8, но в остальных случаях в результате его выполнения агент перемещается под прямым углом к выбранному направлению, а если при этом агент ударяется в стену, то остается в том же квадрате. Например, действие *Up*, выполняемое в начальном квадрате (1,1), перемещает агента в квадрат (1,2) с вероятностью 0,8, но с вероятностью 0,1 агент может переместиться вправо, в квадрат (2,1), а с вероятностью 0,1 он двинется влево, ударится в стену и останется в квадрате (1,1). В подобной среде последовательность действий [*Up*, *Up*, *Right*, *Right*, *Right*] позволяет обойти препятствие и достичь целевого состояния в квадрате (4,3) с вероятностью $0{,}8^5 = 0{,}32768$. В этом случае есть также небольшой шанс случайно достичь цели, обойдя препятствие с другой стороны — с вероятностью $0{,}1^4 \times 0{,}8$, поэтому суммарная вероятность достижения цели равна 0,32776 (см. также упражнение 17.1).

Как и в главе 3, **модель перехода** (или просто "модель", когда смысл слова очевиден) описывает результаты каждого действия в каждом состоянии. В этом случае результат является стохастическим, поэтому для обозначения вероятности достижения состояния s', если в состоянии s было выполнено действие a, мы будем

использовать запись $P(s' \mid s, a)$. Предполагается, что эти переходы являются **марковскими**: вероятность достижения состояния s' из состояния s зависит только от s и не зависит от истории предыдущих состояний.

В завершение данного определения среды задачи требуется сформулировать функцию полезности для агента. Поскольку эта задача принятия решений является последовательной, функция полезности должна зависеть от последовательности состояний и действий — **истории пребывания в среде**, — а не от отдельного состояния. Ниже в этом разделе будет рассмотрена природа функций полезности по истории, а сейчас просто оговорим, что за каждый переход из состояния s в состояние s' посредством выполнения действия a агент получает ▶ **вознаграждение** $R(s, a, s')$. Вознаграждение может быть положительным или отрицательным, но должно быть ограничено величиной $\pm R_{\max}$.[1]

В данном конкретном примере вознаграждение равно –0,04 для всех переходов, кроме тех, которые приводят в конечные состояния (с ними связаны вознаграждения +1 и –1). Полезность, связанная с историей пребывания в среде (на текущий момент), рассматривается просто как *сумма* полученных вознаграждений. Например, если агент достиг состояния +1 после выполнения 10 переходов, суммарная полезность его действий будет $(9 \times -0{,}04) + 1 = 0{,}64$. Отрицательное вознаграждение –0,04 побуждает агента быстрее достичь квадрата (4,3), поэтому данная среда представляет собой стохастическое обобщение задач поиска в главе 3. Еще один способ описать эту ситуацию — сказать, что агенту "не нравится" находиться в этой среде, поэтому он стремится выйти из нее как можно быстрее.

Подведем итог: задача последовательного принятия решений для полностью наблюдаемой стохастической среды с марковской моделью перехода и аддитивными вознаграждениями называется ▶ **марковским процессом принятия решений**, или **MDP** (*Markov Decision Process*). Любая задача MDP состоит из множества состояний (с начальным состоянием s_0), функции ACTIONS(s), определяющей множество действий, доступных в каждом состоянии, модели перехода $P(s' \mid s, a)$ и функции вознаграждения $R(s, a, s')$. Методы решения задач MDP обычно включают ▶ **динамическое программирование**: упрощение задачи посредством ее рекурсивного разбиения на более мелкие фрагменты и запоминания оптимальных решений для этих фрагментов.

Следующий вопрос состоит в том, как должно выглядеть решение этой задачи. Из сказанного выше уже понятно, что какая-либо фиксированная последовательность действий не может служить искомым решением, поскольку после ее выполнения агент может оказаться в состоянии, отличном от целевого. Следовательно, решение должно определять, что следует делать агенту в *любом* состоянии, которого он может достичь. Решение такого рода — это так называемая ▶ **стратегия**.

[1] Также допускается использовать стоимость $c(s, a, s')$, как это было сделано в определении задач поиска в главе 3. Однако использование вознаграждения является стандартной практикой в литературе по последовательным решениям в условиях неопределенности.

По традиции для обозначения стратегии принято использовать символ π, а нотация $\pi(s)$ определяет действие, рекомендованное в стратегии π для состояния s. Не имеет значения, каким будет результат этого действия, — результирующее состояние будет входить в стратегию и агент будет знать, что делать дальше.

Всякий раз, когда определенная стратегия выполняется, начиная с начального состояния, стохастический характер среды приводит к формированию другой истории пребывания в среде. Поэтому качество определения стратегии измеряется по *ожидаемой* полезности возможных историй пребывания в среде, создаваемых согласно этой стратегии. ►**Оптимальной стратегией** называется такая стратегия, которая позволяет достичь максимальной ожидаемой полезности, которую принято обозначать как π^*. Следуя заданной стратегии π^*, агент принимает решение, что делать, учитывая текущие результаты своего восприятия, сообщающие ему, что он находится в состоянии s, а затем выполняет действие $\pi^*(s)$. В любой стратегии функция агента представлена явно, поэтому стратегия является описанием простого рефлексного агента, вычисленным на основе информации, используемой агентом, действующим на основе полезности.

Оптимальные стратегии для мира, представленного на рис. 17.1, приведены на рис. 17.2, *а*. Здесь есть две стратегии, поскольку агенту абсолютно безразлично, куда идти из квадрата (3,1) — влево или вверх: идти влево безопаснее, но дольше, а идти вверх быстрее, но есть риск случайно попасть в квадрат (4,2). В общем случае довольно часто возможно наличие нескольких оптимальных стратегий.

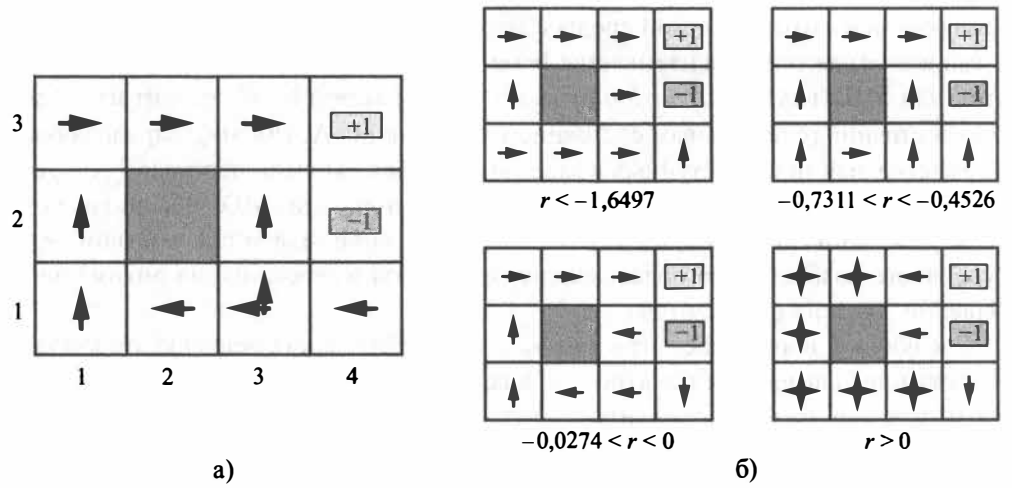


Рис. 17.2. а) Оптимальные стратегии для стохастической среды с $r = -0{,}04$ для переходов в нетерминальные состояния. Существует две стратегии, поскольку в состоянии (3,1) оба перехода, влево и вверх, являются оптимальными. **б)** Оптимальные стратегии для четырех различных диапазонов значений r

Равновесие между риском и вознаграждением изменяется в зависимости от значения вознаграждения $r = R(s, a, s')$ для переходов между нетерминальными состояниями. Стратегии, показанные на рис. 17.2, *а*, являются оптимальными для $-0{,}0850 < r < -0{,}0273$. На рис. 17.2, *б* показаны оптимальные стратегии для четырех других диапазонов значения r. Если $r < -1{,}6497$, жизнь агента настолько мучительна, что он сразу направляется прямо к ближайшему выходу, даже если стоимость этого выхода равна -1. Когда $-0{,}7311 < r < -0{,}4526$, жизнь агента заметно дискомфортна, и он выбирает кратчайший маршрут к состоянию $+1$ из состояний (2,1), (3,1) и (3,2). Однако из состояния (4,1) стоимость достижения $+1$ является настолько высокой, что агент предпочитает направиться прямо в -1. А когда жизнь агента не столь уж неприятна ($-0{,}0274 < r < 0$), оптимальная стратегия состоит в том, чтобы *вообще не рисковать*. В состояниях (4,1) и (3,2) агент направляется буквально прочь от состояния -1, чтобы случайно не попасть туда ни при каких обстоятельствах, даже несмотря на то, что из-за этого ему приходится несколько раз удариться головой о стену. Наконец, если $r > 0$, то жизнь агента становится, определенно, приятной и он избегает *обоих* выходов. При использовании действий, показанных для состояний (4,1), (3,2) и (3,3), любая стратегия является оптимальной и агент получает бесконечно большое суммарное вознаграждение, поскольку никогда не попадает в терминальное состояние. Как оказалось, существует девять оптимальных стратегий для различных диапазонов значений r, — в упражнении 17.7 предлагается найти эти стратегии.

Введение неопределенности делает задачи MDP гораздо ближе к реальному миру по сравнению с задачами детерминированного поиска. По этой причине возможности МДП уже исследовались в нескольких научных областях, включая искусственный интеллект, исследование операций, экономику и теорию управления. Были предложены десятки алгоритмов поиска решений и некоторые из них обсуждаются в разделе 17.2. Но прежде необходимо более обстоятельно разобраться в определениях полезности, оптимальной стратегии и моделях для марковских процессов принятия решений (MDP).

17.1.1. Определение полезности с учетом времени

В примере задачи MDP, представленном на рис. 17.1, производительность агента оценивалась как сумма вознаграждений за выполненные переходы. Такой выбор показателя производительности нельзя назвать произвольным, но он не является единственно возможным для функций полезности по историям пребывания в среде, которые могут записываться как $U_h([s_0, a_0, s_1, a_1 \ldots, s_n])$.[2]

Первый вопрос, на который нужно найти ответ, — что имеет место при принятии решений: ►**конечный горизонт** или ►**бесконечный горизонт**? Наличие

[2] В этой главе для функции полезности используется обозначение U (с целью соответствия остальной части книги), но во многих работах по MDP вместо него используется обозначение V (от слова *value*).

конечного горизонта означает, что есть такое *фиксированное* время N, после которого все теряет смысл, — можно сказать, что игра уже окончена. Таким образом,

$$U_h([s_0, a_0, s_1, a_1, \ldots, s_{N+k}]) = U_h([s_0, a_0, s_1, a_1, \ldots, s_N])$$

для всех $k > 0$. Например, предположим, что агент начинает движение из состояния (3,1) в мире с размерами 4×3, показанном на рис. 17.1, а также допустим, что $N = 3$. Тогда, чтобы получить хоть малейший шанс достичь состояния +1, агент должен направиться непосредственно к нему, и оптимальным действием будет переход в направлении *Up*. С другой стороны, если $N = 100$, то запас времени у агента настолько велик, что лучше выбрать безопасный маршрут в направлении *Left*. Поэтому ➡ *при наличии конечного горизонта оптимальное действие в данном состоянии может зависеть от того, сколько времени осталось.* Стратегия, которая зависит от времени, называется ▶**нестационарной**.

С другой стороны, если нет заданного предела времени, то нет смысла вести себя по-разному в одном и том же состоянии в разное время. Поэтому здесь оптимальное действие зависит только от текущего состояния, а оптимальная стратегия является ▶**стационарной**. Таким образом, стратегии для случая с бесконечным горизонтом будут проще по сравнению со стратегиями, которые применяются в случае с конечным горизонтом, и в данной главе будет в основном рассматриваться случай с бесконечным горизонтом. (Позднее будет показано, что для частично наблюдаемых сред случай с бесконечным горизонтом уже не будет таким простым.) Обратите внимание, что понятие "бесконечного горизонта" не обязательно означает, что все последовательности состояний являются бесконечными, просто в этом случае для их выполнения не устанавливаются фиксированные сроки. Конечные последовательности состояний могут существовать в любой задаче MDP с бесконечным горизонтом, содержащей терминальное состояние.

Следующий вопрос, на который необходимо найти ответ, — как рассчитать полезность последовательностей состояний? Везде в этой главе мы будем использовать ▶**аддитивные обесцениваемые вознаграждения** (*additive discounted rewards*): полезность истории будет следующей:

$$\begin{aligned} U_h([s_0, a_0, s_1, a_1, s_2, \ldots]) &= \\ &= R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \cdots , \end{aligned}$$

где γ — это ▶**коэффициент обесценивания**, представляющий собой число в диапазоне от 0 до 1. Коэффициент обесценивания описывает степень предпочтения агентом текущих вознаграждений перед будущими вознаграждениями. Если коэффициент γ близок к 0, вознаграждения в отдаленном будущем рассматриваются им как малозначащие. Когда коэффициент γ близок к 1, агент проявляет больше желания ожидать получения отдаленных вознаграждений. Если же коэффициент γ равен 1, то аддитивные обесцениваемые вознаграждения сводятся к особому случаю чисто ▶**аддитивных вознаграждений**. Обратите внимание, что свойство

аддитивности уже было определено неявно в используемых нами функциях стоимости пути для алгоритмов эвристического поиска (глава 3).

Использование аддитивных обесцениваемых вознаграждений имеет смысл по нескольким причинам. Одна из них является эмпирической: как люди, так и животные склонны оценивать получение вознаграждения в ближайшей перспективе более высоко, чем его получение в отдаленном будущем. Другая — экономическая: если вознаграждение денежное, то действительно лучше получить его раньше, а не позже, поскольку полученное раньше вознаграждение может быть инвестировано и уже принести прибыль, пока придется ожидать получения более поздних вознаграждений. В этом контексте коэффициент обесценивания γ является эквивалентом процентной ставки размером $(1/\gamma) - 1$. Например, коэффициент обесценивания $\gamma = 0{,}9$ является эквивалентом процентной ставки 11,1%.

Третьей причиной является неопределенность в отношении истинных вознаграждений: их можно так никогда и не получить по самым разным причинам, которые не были приняты во внимание в модели перехода. При определенных допущениях коэффициент обесценивания *гамма* является эквивалентом добавления вероятности $1 - \gamma$ случайного останова истории на любом этапе выполнения стратегии, независимо от выбранного действия.

Четвертое обоснование вытекает из естественного свойства предпочтений относительно историй. В терминологии многоатрибутной теории полезности (см. раздел 16.4) каждый переход $s_t \xrightarrow{a_t} s_{t+1}$ можно рассматривать как **атрибут** истории $[s_0, a_0, s_1, a_1, s_2 ...]$. В принципе, функция полезности может зависеть от этих атрибутов сколь угодно сложным образом. Однако существует возможность сделать весьма правдоподобное допущение по поводу независимости предпочтений, а именно — что в отношении последовательностей состояний агент имеет ►**стационарные предпочтения**.

Предположим, что две истории, $[s_0, a_0, s_1, a_1, s_2, \ldots]$ и $[s_0', a_0', s_1', a_1', s_2', \ldots]$, начинаются с одного и того же перехода (т.е. $s_0 = s_0'$, $a_0 = a_0'$ и $s_1 = s_1'$). Тогда стационарность предпочтений означает, что эти две истории должны быть упорядочены по предпочтительности тем же способом, что и $[s_1, a_1, s_2, \ldots]$ и $[s_1', a_1', s_2', \ldots]$. На обычном языке это означает, что если вы предпочитаете одно будущее другому, начиная с завтрашнего дня, то вы должны также предпочесть это будущее, если оно начнется сегодня, а не завтра. Стационарность — допущение, которое выглядит довольно безобидно, но аддитивное обесценивание — это единственная форма полезности историй, которая ему удовлетворяет.

Последним обоснованием использования аддитивных обесцениваемых вознаграждений является тот факт, что это позволяет удобным образом устранить некоторые неприятные бесконечности. При бесконечных горизонтах имеет место потенциальное затруднение: если среда не содержит терминального состояния или если оно есть, но агент никогда не сможет его достичь, то все истории в этой среде будут бесконечно длинными, а их полезности при аддитивном необесцениваемом

вознаграждении будут в общем случае бесконечными. Хотя мы можем согласиться, что $+\infty$ — это лучше, чем $-\infty$, сравнить две последовательности состояний с полезностью $+\infty$ будет сложнее. Здесь возможны три решения, с двумя из которых мы уже встречались.

1. При обесцениваемых вознаграждениях полезность любой бесконечной последовательности будет *конечной*. Так, если $\gamma < 1$, а величина вознаграждения ограничена значением $\pm R_{\max}$, то, с использованием стандартной формулы суммы бесконечных геометрических рядов будет справедливо следующее соотношение:

$$U_h([s_0, a_0, s_1, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \leq \sum_{t=0}^{\infty} \gamma^t R_{\max} = \frac{R_{\max}}{1-\gamma}. \qquad (17.1)$$

2. Если среда содержит терминальные состояния *и если в конечном итоге гарантируется достижение агентом одного из этих состояний*, то нам никогда не придется сравнивать бесконечные последовательности действий. Стратегия, которая гарантирует достижение терминального состояния, называется ▶**правильной стратегией**. При наличии правильных стратегий можно использовать $\gamma = 1$ (т.е. аддитивные необесцениваемые вознаграждения). Первые три стратегии, показанные на рис. 17.2, *б*, являются правильными, а четвертая — неправильной. В ней достигается бесконечное суммарное вознаграждение за счет предотвращения попадания в терминальные состояния, и при этом вознаграждение за пребывание в нетерминальных состояниях является положительным. Существование таких неправильных стратегий в сочетании с использованием аддитивных необесцениваемых вознаграждений может стать причиной неудачного завершения стандартных алгоритмов решения задач MDP, что само по себе является весомым доводом в пользу применения обесцениваемых вознаграждений.
3. Бесконечные последовательности можно сравнивать в терминах ▶**среднего вознаграждения**, получаемого в расчете на временной интервал. Предположим, что в нашем мире 4×3 с квадратом (1,1) связано вознаграждение 0,1, тогда как для других нетерминальных состояний предусмотрено вознаграждение 0,01. В таком случае стратегия, в которой агент предпочтет оставаться в квадрате (1,1), позволит получать более высокое среднее вознаграждение по сравнению с той стратегией, в которой агент находится в каком-то другом квадрате. Среднее вознаграждение является полезным критерием для некоторых задач, но анализ алгоритмов со средним вознаграждением слишком сложен.

Аддитивные обесцениваемые вознаграждения обеспечивает наименьшее количество трудностей в оценке историй, поэтому мы будем использовать их в дальнейшем.

17.1.2. Оптимальные стратегии и полезность состояний

Решив, что полезность определенной истории является суммой обесцениваемых вознаграждений, можно сравнивать отдельные стратегии, сравнивая *ожидаемые* полезности, которые будут получены при их выполнении. Предполагается, что агент находится в некотором начальном состоянии s и определена случайная переменная S_t, представляющая состояние, которое агент достигнет на этапе t при выполнении определенной стратегии π. (Очевидно, что $S_0 = s$ — это то состояние, в котором агент находится на данный момент.) Распределение вероятностей по последовательности состояний $S_1, S_2, \ldots$ определяется начальным состоянием s, стратегией π и моделью перехода окружающей среды.

Ожидаемая полезность, которую можно получить при выполнении стратегии π, начиная с состояния s, определяется как

$$U^{\pi}(s) = M\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1})\right], \tag{17.2}$$

где M — математическое ожидание относительно распределения вероятностей последовательности состояний, определяемой начальным состоянием s и стратегией π. Теперь из всех стратегий, которые агент может выбрать для выполнения начиная с состояния s, одна (или более) будет иметь более высокую ожидаемую полезность, чем все остальные. Обозначив эту лучшую стратегию как π_s^*, можно сформулировать следующее определение:

$$\pi_s^* = \underset{\pi}{\operatorname{argmax}} U^{\pi}(s). \tag{17.3}$$

Не забывайте, что π_s^* — это стратегия, поэтому она рекомендует действие для *каждого* состояния. В частности, ее связь с состоянием s состоит в том, что это оптимальная стратегия, когда s является начальным состоянием. Замечательное последствие использования обесцениваемых утилит при бесконечных горизонтах заключается в том, что оптимальная стратегия *не зависит* от исходного состояния. (Безусловно, *последовательность действий* не может быть независимой; напомним, что стратегия является функцией, определяющей действие для каждого состояния.) Этот факт кажется интуитивно очевидным: если стратегия π_a^* является оптимальной стратегией, начинающейся в состоянии a, а стратегия π_b^* есть оптимальная стратегия, начинающаяся в состоянии b, то когда они достигают третьего состояния, c, нет никаких оснований для того, чтобы они противоречили друг другу или стратегии π_c^* в отношении того, что следует делать дальше.[3]

[3] Хотя это утверждение кажется очевидным, оно не соблюдается для стратегий при конечных горизонтах или для других способов комбинирования вознаграждений во времени, таких как выбор максимального значения. Доказательство следует непосредственно из уникальности функции полезности в состояниях, как показано в разделе 17.2.1.

Поэтому в дальнейшем мы можем обозначать оптимальную стратегию просто как π^*.

Из этого определения следует, что истинная полезность любого состояния есть просто $U^{\pi^*}(s)$, т.е. ожидаемая сумма обесцениваемых вознаграждений, если агент осуществляет оптимальную стратегию. Мы обозначаем ее как $U(s)$, придерживаясь нотации, использовавшейся в главе 16 для полезности результата. (Обратите внимание, что $U(s)$ и $R(s)$ — разные величины: $R(s)$ является "кратковременным" вознаграждением за пребывание в состоянии s, а $U(s)$ является "долговременным" суммарным вознаграждением, начинающимся с состояния s и продолжающимся далее.) На рис. 17.3 показаны значения полезности для мира 4×3. Обратите внимание, что значения полезности становятся выше по мере приближения состояний к выходу +1, — результат уменьшения количества шагов, необходимых для достижения этого выхода.

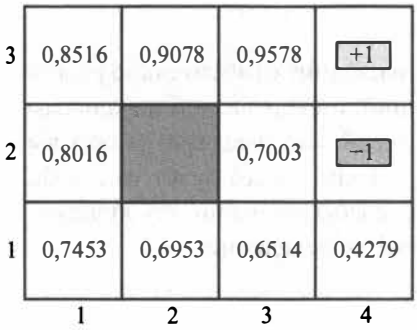


Рис. 17.3. Полезности состояний в мире 4×3, рассчитанные при $\gamma = 1$ и $r = -0{,}04$ для нетерминальных состояний

Функция полезности $U(s)$ позволяет агенту выбирать действия, используя принцип максимальной ожидаемой полезности, обсуждавшийся в главе 16, т.е. выбирать действие, максимизирующее вознаграждение за следующий этап плюс ожидаемая обесцениваемая полезность следующего состояния:

$$\pi^*(s) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')]. \qquad (17.4)$$

Итак, полезность некоторого состояния $U(s)$ определена как ожидаемая сумма обесцениваемых вознаграждений от данной точки и дальше. Из этого следует, что существует прямая связь между полезностью состояния и полезностью его соседних состояний: ➡ *полезность состояния равна ожидаемому вознаграждению за следующий переход плюс ожидаемая обесцениваемая полезность следующего состояния, при*

условии, что агент выбирает оптимальное действие. Это означает, что полезность любого состояния можно определить следующим образом:

$$U(s) = \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')]. \tag{17.5}$$

Уравнение (17.5) называется ▶**уравнением Беллмана** в честь Ричарда Беллмана ([169], 1957). Полезности состояний — определяемые с помощью уравнения (17.2) как ожидаемые полезности дальнейших последовательностей состояний — являются решениями множества уравнений Беллмана. В действительности, как будет показано в разделе 17.2.1, они являются *уникальными* решениями.

Рассмотрим одно из уравнений Беллмана для мира 4 × 3. Выражение для $U(1,1)$ будет выглядеть следующим образом:

$$\begin{aligned}\max\{&[0{,}8(-0{,}04 + \gamma U(1, 2)) + 0{,}1(-0{,}04 + \gamma U(2, 1)) + 0{,}1(-0{,}04 + \gamma U(1, 1))],\\ &[0{,}9(-0{,}04 + \gamma U(1, 1)) + 0{,}1(-0{,}04 + \gamma U(1, 2))],\\ &[0{,}9(-0{,}04 + \gamma U(1, 1)) + 0{,}1(-0{,}04 + \gamma U(2, 1))],\\ &[0{,}8(-0{,}04 + \gamma U(2, 1)) + 0{,}1(-0{,}04 + \gamma U(1, 2)) + 0{,}1(-0{,}04 + \gamma U(1, 1))]\},\end{aligned}$$

где элементы этого выражения в каждой из четырех строк соответствуют движениям *Up*, *Left*, *Down* и *Right* соответственно. После подстановки в это уравнение чисел, приведенных на рис. 17.3, при $\gamma = 1$ выясняется, что наилучшим действием является *Up*.

Другой важной величиной является ▶**функция значения действия** (*action-utility function*), или ▶**Q-функция**: $Q(s, a)$ представляет ожидаемую полезность выбора данного действия в данном состоянии. Q-функция связана с полезностями очевидным образом:

$$U(s) = \max_{a} Q(s, a). \tag{17.6}$$

Более того, оптимальная стратегия также может быть извлечена из Q-функции следующим образом:

$$\pi^*(s) = \operatorname*{argmax}_{a} Q(s, a). \tag{17.7}$$

Также можно вывести уравнение Беллмана для Q-функций, вспомнив, что ожидаемым общим вознаграждением за принятие действия является его немедленное вознаграждение плюс обесцениваемая полезность результирующего состояния, что, в свою очередь, можно выразить в терминах Q-функции:

$$\begin{aligned} Q(s, a) &= \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')] = \\ &= \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma \max_{a'} Q(s', a')]. \end{aligned} \tag{17.8}$$

Решение уравнений Беллмана для U (или для Q) дает нам то, что необходимо, чтобы найти оптимальную стратегию. Q-функция появляется вновь и вновь в алгоритмах решения задач MDP, поэтому в дальнейшем мы будем использовать для нее следующее определение.

function Q-VALUE(*mdp*, *s*, *a*, *U*) **returns** значение полезности

return $\sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U[s']]$

17.1.3. Шкалы вознаграждений

В главе 16 отмечалось, что шкала полезностей может быть произвольной: аффинное преобразование оставляет оптимальное решение без изменений. Можно заменить $U(s)$ значением $U'(s) = mU(s) + b$, где m и b являются любыми константами при условии, что $m > 0$. Из определения полезностей как обесцениваемых сумм вознаграждений легко увидеть, что в MDP подобное преобразование вознаграждений также оставит оптимальную стратегию без изменений:

$$R'(s, a, s') = mR(s, a, s') + b.$$

Однако было установлено, что аддитивное разложение вознаграждения полезностей приводит к значительно большей свободе в определении вознаграждений. Пусть $\Phi(s)$ — *любая* функция состояния s. Тогда, в соответствии с ▶**теоремой формирования** (*shaping theorem*), следующая трансформация оставляет оптимальную стратегию неизменной:

$$R'(s, a, s') = R(s, a, s') + \gamma\Phi(s') - \Phi(s). \tag{17.9}$$

Чтобы показать, что это утверждение является истинным, нужно доказать, что два марковских процесса принятия решений (MDP), M и M', имеют идентичные оптимальные стратегии, пока они различаются только своими функциями вознаграждения, как это определено в уравнении (17.9). Начнем с уравнения Беллмана для Q — Q-функции для MDP M:

$$Q(s, a) = \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma \max_{a'} Q(s', a')].$$

Теперь пусть $Q'(s, a) = Q(s, a) - \Phi(s)$; включив эту подстановку в данное уравнение, получим

$$Q'(s, a) + \Phi(s) = \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma \max_{a'}(Q'(s', a') + \Phi(s'))].$$

Затем это уравнение упрощается до

$$\begin{aligned} Q'(s, a) &= \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma\Phi(s') - \Phi(s) + \gamma \max_{a'} Q'(s', a')] = \\ &= \sum_{s'} P(s' \mid s, a)[R'(s, a, s') + \gamma \max_{a'} Q'(s', a')]. \end{aligned}$$

Другими словами, $Q'(s, a)$ удовлетворяет уравнению Беллмана для MDP M'. Теперь можно извлечь оптимальную стратегию для M', используя уравнение (17.7):

$$\pi^*_{M'}(s) = \operatorname*{argmax}_a Q'(s, a) = \operatorname*{argmax}_a Q(s, a) - \Phi(s) = \operatorname*{argmax}_a Q(s, a) = \pi^*_M(s).$$

Функцию $\Phi(s)$ часто называют ▶**потенциалом**, по аналогии с электрическим потенциалом (напряжением), порождающим электрические поля. Член $\gamma\Phi(s') - \Phi(s)$ функционирует как градиент потенциала. Таким образом, если $\Phi(s)$ имеет более высокое значение в состояниях, которые имеют более высокую полезность, то добавление $\gamma\Phi(s') - \Phi(s)$ к вознаграждению приводит к тому, что агент "поднимается" в полезности.

На первый взгляд может показаться довольно нелогичным, что таким способом можно изменить вознаграждение, не изменяя оптимальной стратеги. Но сомнения рассеются, если вспомнить, что *все стратегии являются оптимальными* при функции вознаграждения, которая всюду является нулевой. А это означает, что в соответствии с теоремой формирования все стратегии являются оптимальными для любого вознаграждения, основанного на потенциале в виде $R(s, a, s') = \gamma\Phi(s') - \Phi(s)$. Интуитивно понятно, что это имеет место потому, что при таком вознаграждении не имеет значения, по какому пути агент переходит от A к B. (Это утверждение проще всего понять, когда $\gamma = 1$: по любому пути сумма вознаграждения свернется в $\Phi(B) - \Phi(A)$, поэтому все пути будут одинаково хороши.) Таким образом, добавление к вознаграждению, основанному на потенциале, любого другого вознаграждения не должно приводить к изменению оптимальной стратегии.

Гибкость, обеспечиваемая теоремой формирования, означает, что можно реально помочь агенту, сделав немедленное вознаграждение таким, чтобы оно более прямо отражало то, что агент должен делать. И действительно, если установить $\Phi(s) = U(s)$, то жадные стратегии π_G относительно модифицированного вознаграждения R' также будут оптимальной стратегией:

$$\begin{aligned}
\pi_G(s) &= \operatorname*{argmax}_a \sum_{s'} P(s' \mid s, a) R'(s, a, s') = \\
&= \operatorname*{argmax}_a \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma\Phi(s') - \Phi(s)] = \\
&= \operatorname*{argmax}_a \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s') - U(s)] = \\
&= \operatorname*{argmax}_a \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')] = \\
&= \pi^*(s) \qquad \text{(по уравнению (17.4)).}
\end{aligned}$$

Безусловно, чтобы установить $\Phi(s) = U(s)$, необходимо знать $U(s)$, поэтому бесплатных пирожков здесь нет, но все же есть значительная ценность в отношении определения функции вознаграждения, которая будет настолько полезной,

насколько это возможно. Данный метод в точности соответствует тому, что делают дрессировщики, давая животному небольшое угощение за каждый выполненный им этап в намеченной последовательности.

17.1.4. Представление MDP

Простейшим способом представления $P(s' \mid s, a)$ и $R(s, a, s')$ являются большие трехмерные таблицы размером $|S|^2|A|$. Это хорошо для небольших задач, таких как мир 4×3, для которого таблицы будут иметь по $11^2 \times 4 = 484$ записи. В некоторых случаях таблицы будут **разреженными** — большинство записей в них будет равно нулю, поскольку из каждого состояния s переход возможен только в ограниченное количество состояний s', а это означает, что таблицы имеют размер $O(|S||A|)$. Однако для более крупных задач даже разреженные таблицы оказываются слишком большими.

Как и в главе 16, где для создания сетей принятия решений байесовские сети были расширены с помощью узлов действий и полезности, можно представить задачи MDP, расширив динамические байесовские сети (DBN, см. главу 14) с помощью узлов решений, вознаграждений и полезности для создания ▶**динамических сетей принятия решений** (*dynamic decision networks* — DDN). Сети DDN являются **развернутым представлением** в соответствии с терминологией, представленной в главе 2, поэтому они, как правило, обеспечивают экспоненциальное снижение вычислительной сложности относительно атомарного представления и позволяют моделировать довольно существенные реальные задачи.

На рис. 17.4 представлена DDN, построенная на основе DBN, приведенной на рис. 14.13, *б* (раздел 14.5). Она включает некоторые элементы более реалистичной модели для мобильного робота, который способен заряжать свой аккумулятор. Здесь состояние S_t декомпоновано на четыре следующих переменных состояния.

- Переменная $\mathbf{X}_t$ включает две координаты местоположения в клеточном мире плюс сведения об ориентации.
- Переменная $\dot{\mathbf{X}}_t$ определяет скорость изменения $\mathbf{X}_t$.
- Переменная $Charging_t$ имеет значение *true*, когда робот подключен к зарядному устройству.
- Переменная $Battery_t$ характеризует уровень заряда аккумуляторной батареи робота, который моделируется целым числом в диапазоне от 0 до 5.

Пространство состояний для задачи MDP представляет собой декартово произведение диапазонов этих четырех переменных. Действие здесь является множеством $\mathbf{A}_t$ переменных действия, состоящее из переменной *Plug/Unplug*, имеющей три значения (*plug* (*подключен*), *unplug* (*отключен*) и *poop* (*нет данных*)), переменной *LeftWheel* (*левое колесо*), представляющей мощность, потребляемую двигателем левого колеса, и переменной *RightWheel* (*правое колесо*), представляющей

мощность, потребляемую двигателем правого колеса. Множество действий этой задачи представляет собой декартово произведение диапазонов этих трех переменных. Обратите внимание, что каждая переменная действия влияет только на некоторое подмножество переменных состояния.

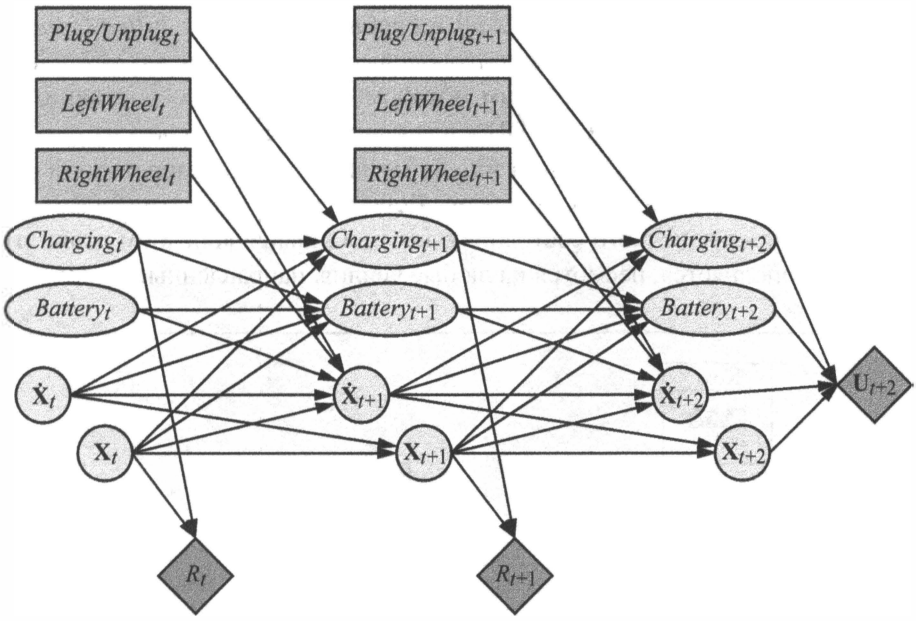


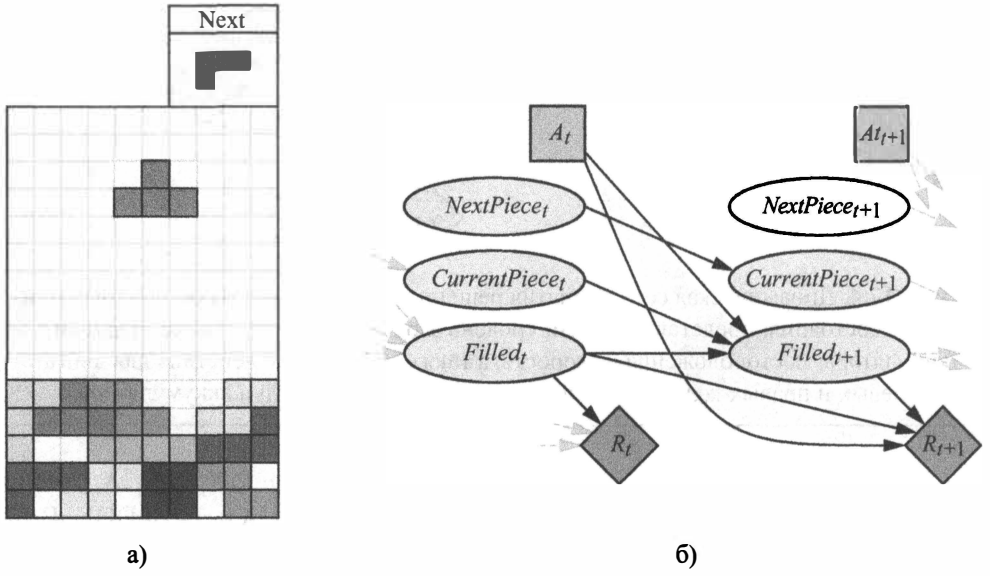
Рис. 17.4. Динамическая сеть принятия решения для мобильного робота с переменными состояния, представляющими уровень заряда батареи, состояние зарядки аккумулятора, местоположение и скорость, а также переменные действия для двигателей левых и правых колес и подключения к устройству зарядки аккумулятора

Общая модель перехода — это условное распределение $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{X}_t, \mathbf{A}_t)$, которое можно вычислить как произведение условных вероятностей из сети DDN. Вознаграждением в данном случае является единственная переменная, которая зависит только от переменной местоположения $\mathbf{X}$ (скажем, за прибытие в пункт назначения) и от переменной *Charging*, поскольку робот должен платить за использованную им электроэнергию, т.е. в данной конкретной модели вознаграждение не зависит от действия или состояния результата.

Сеть на рис. 17.4 была спроецирована на три этапа в будущее. Обратите внимание, что эта сеть включает узлы *вознаграждения* для этапов t, $t+1$ и $t+2$, но узел *полезности* есть только для этапа $t+3$. Так сделано потому, что агент должен максимизировать (обесцениваемую) сумму всех будущих вознаграждений, а $U(\mathbf{X}_{t+3})$ представляет собой вознаграждение наперед для всех этапов от $t+3$ и далее. Если

доступно эвристическое приближение для U, таким способом оно может быть включено в представление сети MDP и использовано вместо ее дальнейшего расширения. Подобный подход тесно связан с использованием поиска с ограничением глубины и функций эвристической оценки для игр, представленных в главе 5.

Другой интересной и хорошо изученной задачей MDP является игра ***Тетрис*** (рис. 17.5, *а*). Переменными состояния для этой игры являются *CurrentPiece* (*текущий блок*), *NextPiece* (*следующий блок*) и особая переменная *Filled* (*заполнено*), представляющая собой битовый вектор, включающий один бит для каждой из клеток на игровом поле 10×20. Таким образом, пространство состояний имеет $7 \times 7 \times 2^{200} \approx 10^{62}$ состояний. Сеть DDN для игры ***Тетрис*** показана на рис. 17.5, *б*. Обратите внимание, что переменная $Filled_{t+1}$ является детерминированной функцией от $Filled_t$ и A_t. Как оказалось, для игры ***Тетрис*** любая стратегия является допустимой (достигает конечного состояния): так или иначе в конечном итоге игровое поле всегда заполняется, несмотря на любые усилия, направленные на его очистку.



а) б)

Рис. 17.5. а) Игра ***Тетрис***. Т-образный блок, находящийся в средине верхней части игрового поля, может быть опущен вниз в любой ориентации и в любой горизонтальной позиции. Если строка оказывается заполненной, она удаляется, содержимое строк над ней опускается, а агент получает одно очко. Следующий блок (здесь L-образный блок вверху справа) становится текущим блоком, и появляется новый следующий блок, тип которого будет случайным образом выбран из семи возможных. Игра заканчивается, когда игровое поле оказывается заполненным блоками до самого верха. **б)** Динамическая сеть принятия решений (DDN) для задачи MDP, представляющей игру ***Тетрис***

17.2. Алгоритмы для задач MDP

В этом разделе представлены четыре различных алгоритма для решения задач MDP. Первые три, **итерация по значениям**, **итерация по стратегиям** и **линейное программирование**, позволяют получить точные решения в автономном режиме. Четвертым типом является семейство неавтономных приближенных алгоритмов, которое включает ▶ **планирование по методу Монте-Карло**.

17.2.1. Алгоритм итерации по значениям

Основой алгоритма ▶ **итерации по значениям**, применяемого для решения задач MDP, является уравнение Беллмана (17.5). Если существует n возможных состояний, то количество уравнений Беллмана также равно n, по одному для каждого состояния. Эти n уравнений содержат n неизвестных — полезностей состояний. Поэтому можно было бы заняться поиском решений системы этих уравнений, чтобы определить полезности. Однако существует одна проблема: эти уравнения являются *нелинейными*, поскольку оператор "max" — нелинейный оператор. В то время как системы линейных уравнений могут быть быстро решены с использованием методов линейной алгебры, поиск решения систем нелинейных уравнений более проблематичен. Один из возможных вариантов — использовать *итеративный* подход. В этом случае нужно начать с произвольных исходных значений полезностей, вычислить правую часть уравнения и подставить ее в левую, тем самым обновляя значение полезности каждого состояния с учетом полезностей его соседних состояний. Подобная операция повторяется до тех пор, пока не достигается равновесие.

Пусть $U_i(s)$ — значение полезности для состояния s на i-й итерации. Каждый этап итерации, называемый ▶ **обновлением Беллмана**, выглядит следующим образом:

$$U_{i+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U_i(s')], \tag{17.10}$$

где предполагается, что это обновление применяется одновременно ко всем состояниям на каждой итерации. Если обновление Беллмана используется неопределенно большое количество раз, то гарантируется достижение равновесия (см. приведенный ниже раздел "Сходимость алгоритма итерации по значениям"), и в этом случае конечные значения полезности должны представлять собой решения уравнений Беллмана. В действительности они также представляют собой *уникальные* решения, и соответствующая стратегия (полученная с помощью уравнения (17.4)) является оптимальной. Детальный алгоритм, включающий условие завершения, когда полезности будут "достаточно близкими", представлен на рис. 17.6. Обратите внимание на использование в нем функции Q-VALUE, определенной в конце раздела 17.1.2.

function VALUE-ITERATION(mdp, ϵ) **returns** функция полезности
  inputs: mdp, задача MDP с состояниями S, действиями $A(s)$, моделью перехода $P(s' \mid s, a)$, вознаграждениями $R(s, a, s')$, коэффициентом обесценивания γ
    ϵ, максимально допустимая ошибка определения полезности любого состояния
  local variables: U, U', векторы полезностей для состояний из S, исходно равные нулю
    δ, максимальное относительное изменение полезности любого состояния во время итерации

  repeat
    $U \leftarrow U'$; $\delta \leftarrow 0$
    for each состояние s **in** S **do**
      $U'[s] \leftarrow \max_{a \in A(s)}$ Q-VALUE(mdp, s, a, U)
      if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
  until $\delta \leq \epsilon(1 - \gamma)/\gamma$
  return U

Рис. 17.6. Алгоритм итерации по значениям для вычисления полезностей состояний. Условие завершения работы взято из уравнения (17.12)

Можно применить алгоритм итерации по значениям к миру 4×3, представленному на рис. 17.1, *a*. Начиная с исходных значений, равных нулю, полезности изменяются, как показано на рис. 17.7, *a*. Обратите внимание на то, как состояния, находящиеся на различных расстояниях от квадрата (4,3), накапливают отрицательное вознаграждение до тех пор, пока в какой-то момент не обнаруживается путь к состоянию (4,3), после чего значения полезности начинают возрастать. Алгоритм итерации по значениям может рассматриваться как способ *распространения информации* через пространство состояний с помощью локальных обновлений.

Сходимость алгоритма итерации по значениям

Выше уже говорилось, что алгоритм итерации по значениям в конечном итоге всегда сходится к уникальному множеству решений уравнений Беллмана. В этом разделе показано, почему так происходит. По ходу обсуждения будут представлены некоторые полезные математические идеи и получены определенные методы оценки ошибки в значении функции полезности, возвращаемом при преждевременном завершении работы алгоритма. Это полезно, поскольку означает, что количество выполняемых итераций алгоритма не обязательно должно стремиться к бесконечности. Изложение материала в этом разделе является достаточно формальным.

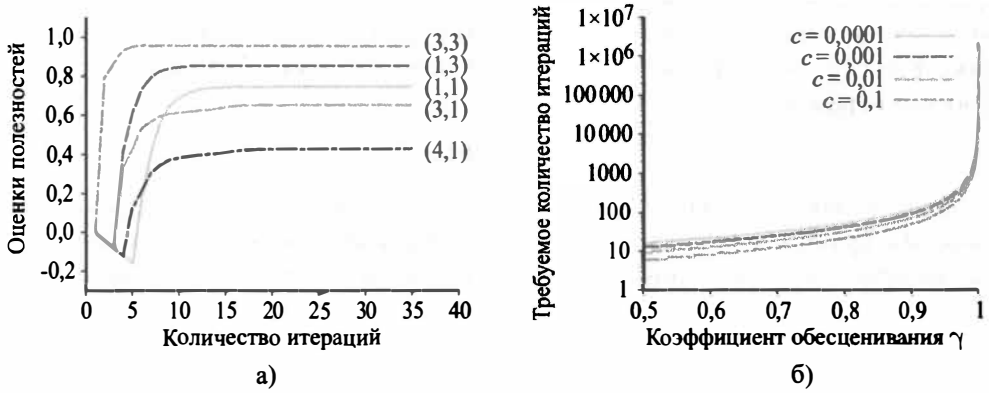


Рис. 17.7. а) График, показывающий изменение полезностей выбранных состояний в процессе работы алгоритма итерации по значениям. **б)** Количество итераций по значениям, необходимое для гарантированного получения ошибки, не превышающей $\epsilon = c \cdot R_{max}$ для различных значений c, как функция от коэффициента обесценивания γ

Основной концепцией, используемой при доказательстве того, что алгоритм итерации по значениям сходится, является ▶**сжатие** (*contraction*). Грубо говоря, функция сжатия — это функция одного аргумента, которая после ее последовательного применения к двум различным входным значениям вырабатывает два выходных значения, которые "ближе друг к другу" по меньшей мере на некоторую постоянную величину, чем первоначальные входные значения. Например, функция "деления на два" представляет собой функцию сжатия, поскольку после деления двух чисел на два разница между ними уменьшается наполовину. Обратите внимание, что функция "деления на два" имеет фиксированную точку, а именно — нуль, которая остается неизменной в результате применения этой функции. На основании данного примера можно установить два важных свойства функций сжатия.

- Функция сжатия имеет только одну фиксированную точку; если бы было две фиксированные точки, они бы не приближались друг к другу после применения функции, поэтому такая функция не соответствовала бы определению функции сжатия.
- После применения функции к любому аргументу полученное значение должно стать ближе к фиксированной точке (поскольку фиксированная точка не смещается), поэтому в пределе многократное повторное применение функции сжатия всегда приводит к достижению фиксированной точки.

Теперь предположим, что обновление Беллмана (уравнение (17.10)) рассматривается как оператор B, который используется для одновременного обновления значений полезности каждого состояния. В результате уравнение Беллмана принимает вид $U = BU$, а уравнение обновления Беллмана может быть переписано следующим образом:

$$U_{i+1} \leftarrow BU_i.$$

Далее нужно найти способ измерения расстояний между векторами полезностей. Мы будем использовать ▶ **нормализованный максимум** (*max norm*), позволяющий измерить "длину" вектора по абсолютному значению его наибольшего компонента:

$$\| U \| = \max_s |U(s)|.$$

Исходя из этого определения, можно утверждать, что "расстоянием" между двумя векторами, $\| U - U' \|$, будет максимальная разность между любыми двумя соответствующими элементами. Основным математическим результатом данного раздела является следующее утверждение: пусть U_i и U_i' — любые два вектора полезностей. В таком случае получим следующее:

$$\| BU_i - BU_i' \| \leq \gamma \| U_i - U_i' \|. \tag{17.11}$$

Это означает, что ➡ *обновление Беллмана представляет собой функцию сжатия на коэффициент* γ, *применяемую к пространству векторов полезностей.* (В упражнении 17.8 даются некоторые рекомендации в отношении доказательства этого утверждения.) Следовательно, алгоритм итерации по значениям всегда сходится к уникальному решению уравнений Беллмана, при условии, что $\gamma < 1$.

Свойство сжатия также можно использовать для анализа *скорости* сходимости к решению. В частности, можно заменить значение U_i' в уравнении (17.11) *истинными* полезностями U, для которых $BU = U$. В этом случае будет получено следующее неравенство:

$$\| BU_i - U \| \leq \gamma \| U_i - U \|.$$

Поэтому, если $\| U_i - U \|$ рассматривать как *ошибку* в оценке U_i, то можно увидеть, что при каждой итерации эта ошибка уменьшается на коэффициент, по меньшей мере равный γ. Это означает, что процедура итерации по значениям сходится экспоненциально быстро. Можно рассчитать количество требуемых итераций следующим образом. Прежде всего вспомним, что из уравнения (17.1) следует, что полезности всех состояний ограничены значением $\pm R_{\max}/(1-\gamma)$. Это означает, что максимальная начальная ошибка определяется соотношением $\| U_0 - U \| \leq 2R_{\max}/(1-\gamma)$. Если предположить, что для достижения ошибки, не превышающей ϵ, необходимо выполнить N итераций, то, поскольку ошибка на каждой итерации уменьшается по

меньшей мере на величину γ, можно записать следующее соотношение: $\gamma^N \cdot 2R_{\max} / (1-\gamma) \leq \epsilon$. Взяв логарифмы, можно определить, скольких итераций будет достаточно:

$$N = \lceil \log(2R_{\max} / \epsilon(1-\gamma)) / \log(1/\gamma) \rceil .$$

На рис. 17.7, *б* показано, как количество итераций N изменяется в зависимости от γ при различных значениях отношения $\epsilon / R_{\max}$. Положительной особенностью здесь является то, что по причине экспоненциально быстрой сходимости значение N не очень сильно зависит от отношения $\epsilon / R_{\max}$, а отрицательной особенностью — то, что N быстро возрастает по мере приближения значения γ к 1. Следовательно, можно добиться ускорения сходимости, если сделать γ малым, но это сильно сужает горизонт агента, практически лишая его возможности обнаруживать долговременные последствия своих действий.

Приведенный выше анализ предельной ошибки позволяет получить некоторое представление о том, какие факторы влияют на продолжительность выполнения данного алгоритма, но иногда сам этот подход оказывается слишком консервативным способом принятия решения о прекращении итераций. Для этой цели также можно использовать предел, связывающий ошибку с размерами обновления Беллмана в каждой конкретной итерации. На основании свойства сжатия (уравнение (17.11)) можно показать, что если обновление невелико (т.е. не происходит значительного изменения полезности ни одного состояния), то ошибка также является небольшой по сравнению с истинным значением функции полезности. Точнее,

$$\text{если} \quad \|U_{i+1} - U_i\| < \epsilon(1-\gamma)/\gamma), \quad \text{то} \quad \|U_{i+1} - U\| < \epsilon. \tag{17.12}$$

Именно это условие завершения используется в алгоритме VALUE-ITERATION, приведенном на рис. 17.6.

До сих пор мы анализировали ошибку в значении функции полезности, возвращаемом алгоритмом итерации по значениям. ➔ *Однако фактически для агента гораздо важнее то, насколько успешно он будет действовать, принимая решения на основе данной функции полезности.* Предположим, что при выполнении алгоритма итерации по значениям после выполнения i итераций агент получает оценку U_i истинной полезности U и определяет максимальную ожидаемую полезность (MEU) стратегии π_i на основе прогнозирования на один шаг вперед с использованием значения U_i (как в уравнении (17.4)). Будет ли выбранное в итоге поведение почти столь же хорошим, как и оптимальное поведение? Это крайне важный вопрос для любого реального агента, и было показано, что ответ на него является положительным. Значение $U^{\pi_i}(s)$ — это полезность, достигаемая, если, начиная с состояния s, осуществляется стратегия π_i, а ► **убыточность стратегии** $\|U^{\pi_i} - U\|$ — это самая большая часть полезности, которую агент может потерять, осуществляя

стратегию π_i вместо оптимальной стратегии π^*. Убыточность стратегии π_i связана с ошибкой в значении полезности U_i следующим неравенством:

$$\text{если} \quad \|U_i - U\| < \epsilon \quad \text{то} \quad \|U^{\pi_i} - U\| < 2\epsilon. \tag{17.13}$$

На практике часто происходит так, что стратегия π_i становится оптимальной задолго до того, как сходится значение U_i. На рис. 17.8 показано, как максимальная ошибка в значении U_i и убыточность стратегии приближаются к нулю по мере выполнения итераций алгоритма итерации по значениям для среды 4×3 со значением $\gamma = 0{,}9$. Стратегия π_i становится оптимальной при $i = 5$, даже несмотря на то, что при этом максимальная ошибка в значении U_i все еще остается равной 0,51.

Рис. 17.8. Максимальная ошибка $\|U_i - U\|$ в оценках полезности и убыточность стратегии $\|U^{\pi_i} - U\|$ как функции от количества выполненных итераций в алгоритме итерации по значениям для клеточного мира 4×3

Теперь у нас есть все необходимое для использования алгоритма итерации по значениям на практике. Известно, что этот алгоритм сходится к правильным значениям полезности; ошибку в оценках полезностей можно ограничить, даже если выполнение алгоритма итерации по значениям будет остановлено после некоторого количества итераций; а также можно ограничить убыточность стратегии, связанную с осуществлением соответствующей стратегии с максимальной ожидаемой полезностью. В качестве заключительного замечания отметим, что все результаты, приведенные в данном разделе, соответствуют случаю применения обесценивания полезностей с коэффициентом $\gamma < 1$. Если же $\gamma = 1$ и среда содержит терминальные состояния, то можно вывести аналогичное множество результатов оценки сходимости и определения предельных значений ошибок, если выполняются некоторые формальные условия.

17.2.2. Алгоритм итерации по стратегиям

В предыдущем разделе было показано, что есть возможность выработать оптимальную стратегию, даже если оценка функции полезности является неточной. Если очевидно, что одно действие лучше по сравнению со всеми остальными, то нет необходимости точно определять истинные значения величины полезности всех рассматриваемых состояний. Эта идея подсказывает альтернативный метод поиска оптимальных стратегий. В алгоритме ►**итерации по стратегиям**, начиная с некоторой исходной стратегии π_0, чередуется выполнение следующих двух этапов.

- ►**Оценка стратегии**. При заданной стратегии π_i вычисляется $U_i = U^{\pi_i}$ — полезность каждого состояния, если стратегия π_i будет осуществлена.
- ►**Усовершенствование стратегии**. Вычисляется новая стратегия с максимальной ожидаемой полезностью π_{i+1}, с использованием прогноза на один этап, основанного на значении U_i (как в уравнении (17.4)).

Алгоритм завершает свою работу после того, как этап усовершенствования стратегии не приводит к изменению значений полезности. Известно, что в этот момент функция полезности U_i представляет собой фиксированную точку обновления Беллмана и, следовательно, является решением уравнений Беллмана, а это значит, что стратегия π_i должна быть оптимальной. Поскольку в каждом конечном пространстве состояний количество возможных стратегий является конечным и можно показать, что каждая итерация приводит к определению лучшей стратегии, алгоритм итерации по стратегиям должен всегда завершать свою работу. Этот алгоритм приведен на рис. 17.9. Как и в случае алгоритма итерации по значениям, в нем используется функция Q-VALUE, определенная в конце раздела 17.1.2.

Но как можно реализовать процедуру POLICY-EVALUATION? Как оказалось, сделать это намного проще по сравнению с решением стандартных уравнений Беллмана (а именно это происходит в алгоритме итерации по значениям), поскольку действие, применяемое в каждом состоянии, зафиксировано в соответствии с выбранной стратегией. Стратегия π_i определяет действие $\pi_i(s)$, выполняемое в состоянии s на i-й итерации. Это означает, что можно воспользоваться упрощенной версией уравнения Беллмана (уравнение (17.5)), где полезность состояния s (в соответствии со стратегией π_i) связывается с полезностями его соседних состояний:

$$U_i(s) = \sum_{s'} P(s' \mid s, \pi_i(s))[R(s, \pi_i(s), s') + \gamma U_i(s'). \qquad (17.14)$$

Например, предположим, что π_i — это стратегия, показанная на рис. 17.2, *а*. В таком случае имеет место $\pi_i(1,1) = Up$, $\pi_i(1,2) = Up$ и так далее, а упрощенные уравнения Беллмана принимают следующий вид:

$$U_i(1,1) = 0{,}8[-0{,}04 + U_i(1,2)] + 0{,}1[-0{,}04 + U_i(2,1) + 0{,}1[-0{,}04 + U_i(1,1)]],$$
$$U_i(1,2) = 0{,}8[-0{,}04 + U_i(1,3)] + 0{,}2[-0{,}04 + U_i(1,2)]$$

и так далее для всех состояний. Важным моментом является то, что эти уравнения *линейные*, поскольку в них оператор "max" был удален. Для n состояний имеется n линейных уравнений с n неизвестными, которые могут быть решены точно за время $O(n^3)$ с помощью стандартных методов линейной алгебры. Если модель перехода является разреженной — т.е. из любого состояния переход возможен только в небольшое число других состояний, — процесс решения может потребовать еще меньше времени.

---

function POLICY-ITERATION(*mdp*) **returns** стратегия
  inputs: *mdp*, задача MDP с состояниями S, действиями $A(s)$, моделью перехода $P(s' \mid s, a)$
  local variables: U, вектор полезностей для состояний из S, исходно равный нулю
    π, вектор стратегии, индексированный по состоянию, со случайными исходными значениями

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, *mdp*)
    unchanged? $\leftarrow$ истина
    for each состояние s **in** S **do**
      $a^* \leftarrow \underset{a \in A(s)}{\text{argmax}}$ Q-VALUE(*mdp*, s, a, U)
      if Q-VALUE(*mdp*, s, a^*, U) > Q-VALUE(*mdp*, s, $\pi[s]$, U) **then**
        $\pi[s] \leftarrow a^*$; *unchanged*? $\leftarrow$ ложь
  until *unchanged*?
  return π

Рис. 17.9. Алгоритм итерации по стратегиям для вычисления оптимальной стратегии

Для небольших пространств состояний оценка стратегии с использованием точных методов решения часто является наиболее эффективным подходом, но для больших пространств состояний затраты времени $O(n^3)$ могут оказаться чрезмерными. К счастью, *точная* оценка стратегии вовсе не обязательна. Вместо этого можно выполнить некоторое количество упрощенных этапов алгоритма итерации по значениям (они будут упрощенными, поскольку стратегия зафиксирована) и получить достаточно хорошую аппроксимацию полезности. Упрощенное обновление Беллмана для этого процесса определяется таким соотношением:

$$U_{i+1}(s) \leftarrow \sum_{s'} P(s' \mid s, \pi_i(s))[R(s, \pi_i(s), s') + \gamma U_i(s')],$$

и, несколько раз повторив определяемую в нем операцию подстановки, можно успешно получить следующую оценку полезности. Результирующий алгоритм получил название ▶**модифицированная итерация по стратегиям**.

Алгоритмы, описанные в данной главе до этого момента, требуют одновременного обновления полезности или стратегии для всех состояний. Как выяснилось, применение такой организации работы не является строго необходимым. В действительности на каждой итерации можно выбирать *любое подмножество* состояний и применять к нему *либо тот, либо другой* вид обновления (усовершенствование стратегии или упрощенную итерацию по значениям). Такой наиболее общий алгоритм называется ▶**асинхронной итерацией по стратегиям**. При соблюдении определенных условий выбора исходной стратегии и функции полезности гарантируется сходимость асинхронной итерации по стратегиям к некоторой оптимальной стратегии. При этом свобода выбора для работы любых состояний означает, что могут быть разработаны гораздо более эффективные эвристические алгоритмы, например алгоритмы, которые сосредоточиваются на обновлении значений тех состояний, которые с наибольшей вероятностью будут достигнуты при осуществлении хорошей стратегии. Не имеет смысла заниматься планированием результатов действий, которые никогда не будут выполнены.

17.2.3. Линейное программирование

Линейное программирование (*Linear programming* — LP), которое кратко упоминалось в главе 4 (раздел 4.2), является общим подходом для постановки задач оптимизации с ограничениями. В настоящее время существует много доступных решателей LP промышленного уровня. Учитывая, что уравнения Беллмана включают множество операций суммирования и нахождения максимумов, видимо, не вызовет удивления тот факт, что решение задачи MDP может быть сведено к решению подходящим образом сформулированной задачи ЛП.

Основная идея правильной формулировки задачи ЛП состоит в принятии в качестве переменных полезностей $U(s)$ для каждого состояния s, если при этом отметить, что полезности для оптимальной стратегии будут самыми высокими из числа достижимых, что согласуется с уравнениями Беллмана. На языке линейного программирования это означает, что мы стремимся минимизировать $U(s)$ для всех s с учетом приведенных ниже неравенств для каждого состояния s и каждого действия a:

$$U(s) \geq \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')].$$

В результате устанавливается связь между динамическим программированием и линейным программированием, для которого алгоритмы и вопросы сложности уже были изучены достаточно глубоко. Например, исходя из того факта, что задачи линейного программирования решаются за полиномиальное время, можно показать, что и задачи МДП можно решить за полиномиальное время относительно

количества состояний и действий, а также количества битов, необходимых для определения модели. Практика показала, что использование решателей LP редко оказывается столь же эффективным, как применение методов динамического программирования для решения задач MDP. Более того, хотя полиномиальное время обычно считается хорошим показателем, количество состояний часто бывает слишком большим. И наконец, не следует забывать, что даже самый простой и наименее информированный поисковый алгоритм из главы 3 выполняется за линейное время в зависимости от количества состояний и действий.

17.2.4. Неавтономные алгоритмы для задач MDP

Алгоритмы итерации по значениям и итерации по стратегиям являются *автономными*: подобно алгоритму A* из главы 3, они находят оптимальное решение проблемы, которое затем может быть выполнено простым агентом. Но в случае достаточно больших задач МДП, таких как игра ***Тетрис***, модель MDP которой включает 10^{62} состояний, найти точное решение в автономном режиме, даже при использовании алгоритма с полиномиальными временными затратами, невозможно. Поэтому было разработано несколько методов приближенного автономного решения задач МДП, — некоторые из них рассматриваются в разделе "Библиографические и исторические заметки" в конце этой главы и в главе 22, "Обучение с подкреплением".

В этом разделе обсуждаются неавтономные алгоритмы, подобные тем игровым алгоритмам из главы 5, в которых агент выполняет значительный объем вычислений в каждой точке принятия решения, вместо того чтобы действовать в первую очередь согласно заранее просчитанной информации.

Наиболее прямолинейным подходом фактически является упрощение алгоритма EXPECTIMINIMAX для деревьев игр с узлами жеребьевки: алгоритм EXPECTIMAX строит дерево попеременно из узлов MAX и узлов жеребьевки, как показано на рис. 17.10. (Здесь есть небольшое отличие от стандартного EXPECTIMINIMAX: вознаграждения имеются на нетерминальных, равно как и на терминальных переходах.) К нетерминальным листьям дерева может быть применена функция оценки, или им могут быть присвоены значения по умолчанию. Решение может быть извлечено из дерева поиска путем запоминания значения полезности из листьев с выбором среднего значения для узлов жеребьевки и максимального значения в узлах принятия решения.

Для задач, в которых коэффициент обесценивания γ не слишком близок к 1, полезным понятием будет ϵ-горизонт. Пусть ϵ будет заданной границей абсолютной ошибки в полезностях, вычисленной по дереву алгоритма EXPECTIMAX ограниченной глубины, в сравнении с точными полезностями в модели MDP. Тогда ϵ-горизонтом будет глубина дерева H — такая, что сумма всех вознаграждений ниже любого листа на этой глубине будет меньше ϵ. Грубо говоря, все из того, что может произойти ниже H, будет нам безразлично, поскольку находится слишком далеко в

будущем. Так как сумма вознаграждений за пределами H ограничена $\gamma^H R_{\max}/(1-\gamma)$, глубины $H = \lceil \log_\gamma \epsilon(1-\gamma)/R_{\max} \rceil$ будет достаточно. Следовательно, построение дерева на эту глубину дает нам почти оптимальные решения. Например, при $\gamma = 0{,}5$; $\epsilon = 0{,}1$ и $R_{\max} = 1$ получаем $H = 5$, что кажется вполне разумным. С другой стороны, если при тех же прочих условиях $\gamma = 0{,}9$; то $H = 44$, что кажется гораздо менее разумным!

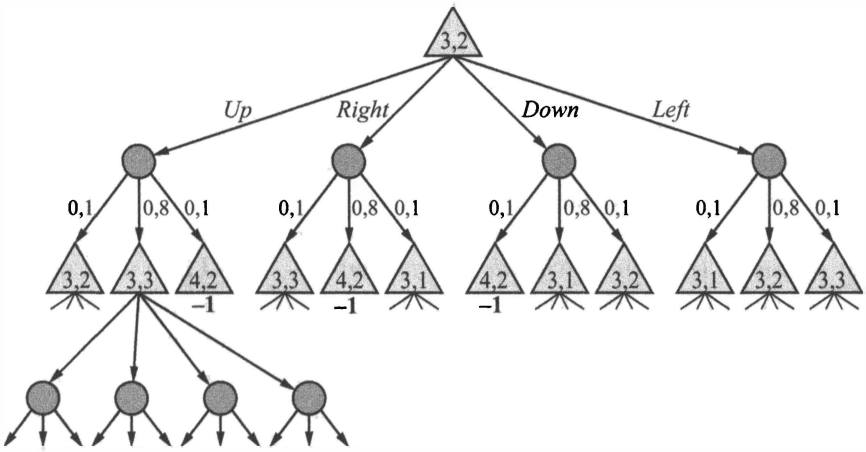


Рис. 17.10. Часть дерева алгоритма EXPECTIMAX для задачи MDP мира 4 × 3 с корнем в квадрате (3,2). Треугольные узлы — это узлы MAX, а круглые — узлы жеребьевки

В дополнение к ограничению глубины также возможно избежать потенциально огромного фактора ветвления в узлах жеребьевки. (Например, если все условные вероятности в модели перехода DBN отличны от нуля, то вероятности перехода, задаваемые как произведения условных вероятностей, также будут ненулевыми, а это означает, что каждое состояние имеет *некоторую* вероятность перехода в любое другое состояние.)

Как отмечалось в разделе 13.4, ожидания относительно распределения вероятностей P можно аппроксимировать с помощью генерации N выборок из P и использования выборочного среднего. В математической форме это выглядит так:

$$\sum_x P(x) f(x) \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i).$$

Поэтому, если коэффициент ветвления является очень большим, а это означает, что существует очень много возможных значений x, хорошее приближение к значению узла жеребьевки может быть получено путем выборки ограниченного количества результатов действия. Как правило, выборки будут ориентированы на

наиболее вероятные результаты, поскольку именно они, скорее всего, и будут сгенерированы.

Если внимательно посмотреть на дерево, приведенное на рис. 17.10, то можно заметить нечто удивительное: на самом деле это не дерево. Например, корневой узел (3,2) также является и листовым узлом, поэтому следует рассматривать это построение как граф, а также ограничить значение листа (3,2) той же величиной, что и значение корня (3,2), поскольку они представляют одно и то же состояние. Фактически эта цепочка размышлений быстро возвращает нас к уравнениям Беллмана, которые связывают значения всех состояний с значениями соседних с ними состояний. Разведанные состояния фактически представляют собой подзадачу MDP исходной задачи MDP, и эти подзадачи MDP могут быть решены с использованием любого из алгоритмов, обсуждавшихся в этой главе, что предоставит решение для текущего состояния. (Граничным состояниям обычно присваивается фиксированное оценочное значение.)

Этот общий подход называется ►**динамическим программированием в реальном времени** (*real-time dynamic programming* — ►**RTDP**) и он весьма близок к алгоритму LRTA*, обсуждавшемуся в главе 4. Алгоритмы такого типа могут быть весьма эффективными в проблемных областях среднего размера, таких как клеточные миры; в больших проблемных областях, подобных игре ***Тетрис***, имеют место две проблемы. Во-первых, пространство состояний таково, что любое управляемое множество разведанных состояний содержит очень мало повторяющихся состояний, поэтому можно также использовать простое дерево алгоритма EXPECTIMAX. Во-вторых, простой эвристики для пограничных узлов может оказаться недостаточно, чтобы направлять действия агента, особенно если вознаграждения сильно разрежены.

Одним из возможных решений является применение обучения с подкреплением для получения гораздо более точной эвристики (см. главу 22). Другой подход заключается в том, чтобы смотреть в модели МДП дальше, используя вариант поиска по дереву методом Монте-Карло, обсуждавшийся в разделе 5.4. В действительности алгоритм UCT, представленный на рис. 5.10, первоначально был разработан именно для задач MDP, а не для игр. Изменения, которые потребуется внести в него для решения задач MDP, а не игровых задач, минимальны: они нужны главным образом из-за того, что противник (здесь это природа) является стохастическим, а также из-за необходимости отслеживать вознаграждения, а не только выигрыши и проигрыши.

Применительно к миру 4×3 производительность алгоритма UCT не особенно впечатляет. Как показано на рис. 17.11, требуется в среднем 160 прогонов, чтобы достичь общего вознаграждения в размере 0,4, тогда как оптимальная стратегия имеет ожидаемое общее вознаграждение в размере 0,7453 при движении из начального состояния (см. рис. 17.3). Одна из причин, по которой алгоритм UCT может испытывать затруднения применительно к этой задаче, состоит в том, что он строит дерево, а не граф, и использует (для аппроксимации) алгоритм EXPECTIMAX,

а не динамическое программирование. Данный мир 4×3 является очень "запутанным": хотя в нем существует только девять нетерминальных состояний, прогоны алгоритма UCT часто включают более 50 действий.

Рис. 17.11. Производительность алгоритма UCT, представленная в виде функции от количества прогонов на один ход для мира 4×3, с использованием случайной стратегии в каждом прогоне. Приведенные величины являются средними значениями для 1000 выполнений алгоритма для каждой точки данных

Алгоритм UCT, кажется, лучше подходит для игры ***Тетрис***, в которой прогоны проходят достаточно далеко в будущее, чтобы дать агенту представление о том, сработает ли в конечном счете потенциально рискованный ход или же приведет к огромному скоплению блоков. Вот один из особенно интересных вопросов: насколько полезной может оказаться стратегия простой модели, например та, которая позволит избегать нагромождения блоков друг на друга и обеспечит их укладку настолько низко, насколько это возможно.

17.3. Задачи о бандитах

Название этого класса задач происходит от сленгового прозвища игровых автоматов, — в Лас-Вегасе такие автоматы (слот-машины) называют *однорукими бандитами*. Игроку предоставляется возможность бросить в автомат монетку, потянуть за рычаг и забрать свой выигрыш (если он будет). Обобщенный вариант, ►***n*-рукий бандит**, имеет *n* рычагов. За каждым рычагом закреплено фиксированное, но неизвестное игроку распределение вероятностей выигрыша, и каждый раз, когда игрок тянет за какой-либо из рычагов, делается выборка из соответствующего этому рычагу неизвестного распределения.

Задача игрока — при каждом броске в автомат очередной монетки выбрать, за какой из рычагов потянуть: за тот, который уже дал наибольшую выплату, или,

может быть, за тот, который еще не опробован? Это пример вездесущего компромисса между **эксплуатацией** текущего наилучшего действия с целью получения вознаграждения и **исследованием** ранее неизвестных состояний и действий для получения информации, которая в некоторых случаях может быть использована для выработки лучшей стратегии, обеспечивающей долгосрочное получение большего вознаграждения. В реальном мире нам постоянно приходится выбирать между продолжением текущего, относительно комфортного существования и прыжком в неизвестность в надежде обрести лучшую жизнь.

Задача n-рукого бандита — это формальная модель для реальных проблем во многих жизненно важных областях, таких как принятие решения, какой из n возможных новых методов лечения лучше применить, чтобы исцелиться от болезни, какую из n возможных инвестиций выбрать, чтобы вложить часть своих сбережений, какой из n предлагаемых исследовательских проектов профинансировать или какое из n имеющихся рекламных объявлений показать пользователю, зашедшему на определенную веб-страницу.

Первые работы по этой проблеме начали проводить в США во время Второй мировой войны, и она показала себя настолько неподдающейся, что ученые стран-союзников предложили "подбросить эту задачу ученым Германии в качестве мощного инструмента интеллектуального саботажа" (Виттле [2332], 1979).

Как оказалось, ученые как во время войны, так и после нее, пытались доказать "очевидно истинные" факты в отношении задач о бандитах, которые на самом деле являлись ошибочными. (Как сказал Брэдт и соавт. ([285], 1956), "Есть много хороших свойств, которыми оптимальные стратегии не обладают".) Например, обычно предполагалось, что оптимальная стратегия в конечном итоге будет опираться на рычаг, наилучший в долгосрочной перспективе; тогда как на самом деле существует конечная вероятность того, что оптимальная стратегия опирается на неоптимальный рычаг. Сейчас у нас имеется четкое теоретическое понимание задач о бандитах, а также разработаны полезные алгоритмы их решения.

Имеется несколько разных определений ▶**задачи о бандите**, — одно из самых ясных и общих приведено ниже.

- Каждая рука (рычаг) M_i является ▶**марковским процессом вознаграждения** (*Markov reward process* — MRP), т.е. задачей MDP с только одним возможным действием a_i. Он имеет состояния S_i, модель перехода $P_i(s' \mid s, a_i)$ и вознаграждение $R_i(s, a_i, s')$. Рука определяет распределение для последовательностей вознаграждений $R_{i,0}$, $R_{i,1}$, $R_{i,2}$,..., где каждый $R_{i,t}$ является случайной величиной.
- Общая задача о бандите является задачей MDP: пространство состояний задается декартовым произведением $S = S_1 \times \cdots \times S_n$; действиями являются $a_1, \cdots, a_n$; модель перехода обновляет состояние в зависимости от того, какая рука M_i была выбрана, в соответствии с ее конкретной моделью перехода, оставляя другие руки без изменений; коэффициент обесценивания равен γ.

Данное определение является очень общим и охватывает широкий спектр вариантов. Ключевым свойством является то, что руки независимы в сочетании с тем фактом, что одновременно агент может работать только с одной рукой. Можно определить еще более общую версию, в которой частичные усилия могут быть применены ко всем рукам одновременно, но суммарное усилие для всех рук ограниченно. Основные результаты, описываемые в этом разделе, могут быть перенесены и на этот случай.

Ниже показано, как в указанных рамках можно сформулировать типичную задачу о бандите, но для начала мы рассмотрим более простой частный случай детерминированных последовательностей вознаграждения. Пусть $\gamma = 0{,}5$, и предположим, что у автомата есть два рычага, обозначенных как M и M_1. Многократное повторное нажатие рычага M приводит к последовательности вознаграждений 0; 2; 0; 7,2; 0; 0;..., тогда как многократные повторные нажатия рычага M_1 дают результат 1, 1, 1,... (рис. 17.12, *а*). Если сначала необходимо выбрать тот или иной рычаг, а затем придерживаться этого выбора, то принять обоснованное решение можно посредством вычисления полезности (суммарного обесцениваемого вознаграждения) для каждого из рычагов.

$$U(M) = (1{,}0 \times 0) + (0{,}5 \times 2) + (0{,}5^2 \times 0) + (0{,}5^3 \times 7{,}2) = 1{,}9$$

$$U(M_1) = \sum_{t=0}^{\infty} 0{,}5^t = 2{,}0$$

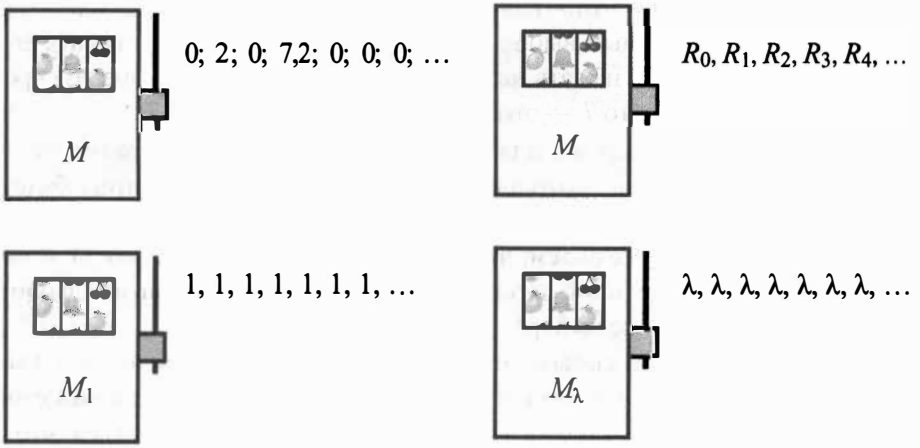


Рис. 17.12. а) Простая детерминированная задача о двуруком бандите. Рычаги можно нажимать в любом порядке, при этом каждый из них реализует приведенные последовательности вознаграждений. **б)** Более общий случай задачи о бандите, определенной на рис. *а*, где первый рычаг выдает произвольную последовательность вознаграждений, а второй всегда выдает фиксированное вознаграждение λ

Можно решить, что в данном случае лучшим выбором является рычаг M_1, но если немного подумать, то станет понятно, что следует начать с M, а затем переключиться на рычаг M_1, — после получения четвертого вознаграждения. В этом случае получим последовательность вознаграждений $S = 0; 2; 0; 7{,}2; 1; 1; 1;...$, полезность которой будет равна

$$U(S) = (1{,}0 \times 0) + (0{,}5 \times 2) + (0{,}5^2 \times 0) + (0{,}5^3 \times 7{,}2) + \sum_{t=4}^{\infty} 0{,}5^t = 2{,}025.$$

Следовательно, стратегия S, предусматривающая переключение с рычага M на рычаг M_1 в нужный момент времени, будет лучше, чем постоянный выбор любого из двух рычагов. Фактически стратегия S является оптимальной для этой задачи: переключение в любой другой момент времени приводит к меньшему суммарному вознаграждению.

Давайте немного обобщим эту задачу: пусть теперь первый рычаг M выдает произвольную последовательность вознаграждений R_0, R_1, R_2,... (которая может быть известна или неизвестна), а второй рычаг M_λ выдает вознаграждения λ, λ, λ,... где λ — некоторая известная фиксированная константа (см. рис. 17.12, *б*). В литературе этот вариант называют задачей ►**однорукого бандита**, поскольку формально он эквивалентен случаю, когда у автомата есть только один рычаг M, который выдает вознаграждения R_0, R_1, R_2,... при стоимости λ каждого его нажатия. (Нажатие рычага M является эквивалентом не нажатия рычага M_λ, так что каждый раз оно отменяет вознаграждение λ.) При наличии только одного рычага единственный возможный выбор состоит в том, чтобы вновь нажать его или прекратить нажатия. Если нажать на первый рычаг T раз (т.е. в моменты времени 0, 1,..., $T-1$), то говорят, что T — это ►**время останова**.

Возвращаясь к нашей версии задачи с M и M_λ, давайте предположим, что после T нажатий на первый рычаг оптимальной стратегией в конечном итоге будет первый раз нажать второй рычаг. Поскольку никакой новой информации от этого действия не поступит (мы уже знаем, что вознаграждение будет равно λ), в момент $T+1$ мы будем находиться в той же ситуации, и, следовательно, оптимальной стратегией будет сделать тот же выбор.

Таким образом, можно сказать, что оптимальной стратегией является многократное нажатие рычага M до момента времени T, а затем переход к нажатию рычага M_λ в течение всего остального времени. Вполне может оказаться, что $T = 0$, если стратегия предполагает немедленный выбор рычага M_λ, или же $T = \infty$, если стратегия вообще не предполагает выбора рычага M_λ, либо значение T будет находиться где-то между этими крайностями. Теперь давайте рассмотрим такое значение λ, что оптимальная стратегия будет *точно безразлична* между двумя вариантами: *а*) нажатие рычага M до наилучшего времени останова с последующим переключением на рычаг M_λ в течение всего остального времени и *б*) немедленный выбор рычага M_λ. На переломном этапе мы имеем

$$\max_{T>0} E\left[\left(\sum_{t=0}^{T-1}\gamma^t R_t\right)+\sum_{t=T}^{\infty}\gamma^t\lambda\right]=\sum_{t=0}^{\infty}\gamma^t\lambda,$$

что можно упростить до

$$\lambda=\max_{T>0}\frac{E\left(\sum_{t=0}^{T-1}\gamma^t R_t\right)}{E\left(\sum_{t=0}^{T-1}\gamma^t\right)}. \tag{17.15}$$

Это уравнение определяет своего рода "значение" для M с точки зрения его способности обеспечить поток регулярных вознаграждений. Числитель дроби представляет полезность, тогда как знаменатель может рассматриваться как "обесцениваемое время", поэтому значение описывает максимальную получаемую полезность на единицу обесцениваемого времени. (Важно помнить, что T в уравнении — это время останова, которое определяется правилом останова, а не просто целым числом; оно сводится к простому целому числу только тогда, когда M является детерминированной последовательностью вознаграждения.) Значение, определенное в уравнении (17.15), называется ►**индексом Гиттинса** для M.

Замечательным свойством индекса Гиттинса является то, что он предоставляет очень простую оптимальную стратегию для любой задачи о бандитах: ➔ *нажать рычаг, который имеет самый высокий индекс Гиттинса, а затем обновить индексы Гиттинса*. Более того, поскольку индекс рычага M_i зависит только от свойств этого рычага, оптимальное решение о первой итерации может быть вычислено за время $O(n)$, где n — количество рычагов. А поскольку индексы Гиттинса для тех рычагов, которые не выбирались, остаются неизменными, каждое решение после первого можно вычислить за время $O(1)$.

17.3.1. Вычисление индекса Гиттинса

Для того чтобы получить полное представление об индексе Гиттинса, давайте вычислим значения числителя, знаменателя и дроби в уравнении (17.15) для различных возможных значений времени останова для детерминированной последовательности вознаграждений 0, 2, 0, 7,2, 0, 0, 0,...

T	1	2	3	4	5	6
R_t	0	2	0	7,2	0	0
$\Sigma\gamma^t R_t$	0,0	1,0	1,0	1,9	1,9	1,9
$\Sigma\gamma^t$	1,0	1,5	1,75	1,875	1,9375	1,9687
Дробь	0,0	0,6667	0,5714	1,0133	0,9806	0,9651

Очевидно, что с этого момента дробь будет только уменьшаться, поскольку числитель остается постоянным, тогда как знаменатель продолжает увеличиваться. Таким образом, индекс Гиттинса для этого рычага равен 1,0133, максимальному значению, достигаемому дробью. В сочетании с рычагом M_λ с фиксированным вознаграждением $0 < \lambda \leq 1{,}0133$, оптимальная стратегия включает получение первых четырех вознаграждений от M с последующим постоянным переключением на M_λ. Для $\lambda > 1{,}0133$ оптимальная стратегия сразу предусматривает выбор M_λ.

Для того чтобы рассчитать индекс Гиттинса для обобщенного рычага M с текущим состоянием s, достаточно просто сделать следующее наблюдение: в переломный момент, когда оптимальная стратегия становится безразличной к выбору между рычагом M и рычагом M_λ с фиксированным вознаграждением, значение выбора M является таким же, как и значение выбора бесконечной последовательности λ-вознаграждений.

Предположим, что M было расширено так, что в каждом состоянии M агенту предоставляется два варианта действий: либо продолжать работу с M на прежних условиях, либо выйти из игры и получить бесконечную последовательность λ-вознаграждений (рис. 17.13, *а*). Это превращает M в модель MDP, оптимальная стратегия которой состоит лишь в определении оптимального момента остановки правила для M. Следовательно, значение оптимальной стратегии в этой новой модели MDP будет равно значению бесконечной последовательности из λ-вознаграждений, определяемой как $\lambda / (1 - \gamma)$. Поэтому достаточно просто решить эту задачу MDP... но, к сожалению, значение λ, которое нужно вставить в MDP, нам неизвестно, поскольку это именно то, что мы пытаемся вычислить. Однако нам *точно* известно, что в переломный момент оптимальная стратегия становится безразличной к выбору между M и M_λ, поэтому вариант получения бесконечной последовательности λ-вознаграждений можно заменить вариантом возврата и перезапуска работы с M из начального состояния s. (Говоря более точно, в каждом состоянии мы добавляем новое действие, которое имеет те же вознаграждение и исходы, что и действие, доступное в s.) Этот новый вариант MDP, M^s, называется ▶ **перезапущенной MDP** и приведен на рис. 17.13, *б*.

Мы получили обобщенный результат: индекс Гиттинса для рычага M в состоянии s равен $1 - \gamma$, умноженному на значение оптимальной стратегии для перезапущенной MDP M^s. Эту задачу MDP можно решить с помощью любого из алгоритмов, обсуждавшихся в разделе 17.2. Алгоритм итерации по значениям, примененный к модели M^s, приведенной на рис. 17.13, *б*, дает значение 2,0266 для начального состояния, поэтому мы имеем $\lambda = 2{,}0266 \cdot (1 - \gamma) = 1{,}0133$, как и в предыдущем расчете.

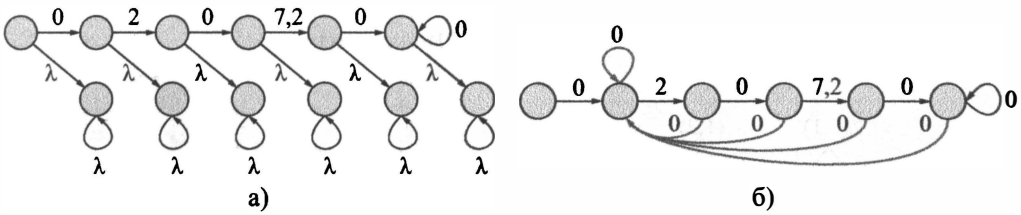


Рис. 17.13. а) Последовательность вознаграждений $M = 0; 2; 0; 7{,}2; 0; 0; 0;\ldots$, расширенная в каждой точке возможностью выбора переключения на рычаг с постоянным вознаграждением M_λ. **б)** Перезапущенная MDP, оптимальное значение которой в точности эквивалентно оптимальному значению для MDP на рис. *а*, по меньшей мере в точке, где оптимальная стратегия является безразличной к выбору между M и M_λ

17.3.2. Многорукий бандит по Бернулли

Возможно, простейший и самый известный пример задач о бандитах — это ►**бандит по Бернулли**, у которого каждая рука M_i выдает вознаграждение 0 или 1 с фиксированной, но неизвестной вероятностью μ_i. Состояние руки M_i определяется с помощью счетчиков s_i и f_i, фиксирующих количество успехов (единиц) и неудач (нулей) соответственно на данный момент для данной руки. Вероятности перехода предсказывают, что следующий результат будет 1 с вероятностью $(s_i) / (s_i + f_i)$ или 0 — с вероятностью $(f_i) / (s_i + f_i)$. Счетчики инициализируются значением 1, так что начальными вероятностями являются 1/2, а не 0/0.[4] Марковский процесс вознаграждения представлен на рис. 17.14, *а*.

Мы не можем просто применить трансформации из предыдущего раздела с целью расчета индекса Гиттинса для руки бандита по Бернулли, потому что она имеет бесконечно много состояний. Однако можно получить очень точное приближение за счет решения усеченной модели MDP с состояниями вплоть до $s_i + f_i = 100$ и $\gamma = 0{,}9$. Эти результаты приведены на рис. 17.14, *б*, — интуитивно они кажутся разумными: мы видим, что, вообще говоря, руки с более высокими вероятностями получения выплаты являются предпочтительными, но здесь также есть ►**бонус за разведку**, связанный с руками, которые были опробованы лишь несколько раз. Например, индекс для состояния (3,2) выше, чем индекс для состояния (7,4) (0,7057 против 0,6922), хотя значение оценки для состояния (3,2) ниже (0,6 против 0,6364).

[4] Здесь вероятности — это вероятности байесовского процесса обновления с априорным распределением *Beta*(1,1) (см. раздел 20.2.5).

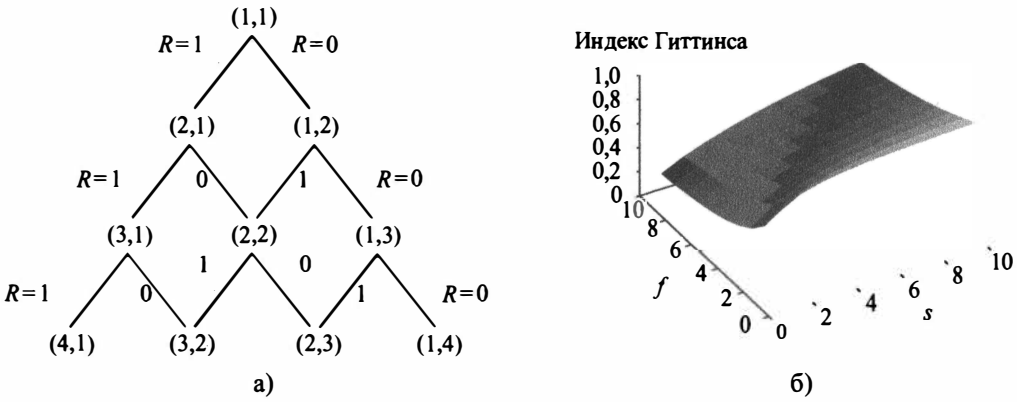


Рис. 17.14. а) Состояния, вознаграждения и вероятности перехода для бандита по Бернулли. **б)** Индексы Гиттинса для состояний процесса бандита по Бернулли

17.3.3. Приблизительно оптимальные стратегии в задачах о бандитах

Вычисление индексов Гиттинса для более реалистичных задач редко бывает простым. К счастью, общие свойства, отмеченные в предыдущем разделе (а именно — желательность некоторой комбинации оценочной стоимости и неопределенности), допускают создание простых стратегий, которые оказываются "почти такими же хорошими", как оптимальная стратегия.

В первом классе методов используется ▶ **верхняя граница достоверности** или эвристика UCB (*Upper Confidence Bound*), введенная ранее, при поиске по дереву методом Монте-Карло (рис. 5.11 в разделе 5.4). Основная идея состоит в том, чтобы использовать выборки из каждой руки для установления **доверительного интервала** значения руки, т.е. диапазона, в пределах которого это значение может быть оценено с высокой степенью достоверности. Затем выбирается рука с самой высокой верхней границей ее доверительного интервала. Верхняя граница — это текущее среднее значение оценки $\hat{\mu}_i$ плюс некоторое кратное стандартному отклонению неопределенности в значении. Стандартное отклонение пропорционально $\sqrt{1/N_i}$, где N_i — количество выборок для руки M_i. Отсюда получаем приблизительное значение индекса для руки M_i, определяемое как

$$UCB(M_i) = \hat{\mu}_i + g(N)/\sqrt{N_i},$$

где $g(N)$ — надлежащим образом выбранная функция от N, общего количества выборок, взятых для всех рук. Стратегия UCB — это просто выбор руки с самым высоким UCB-значением. Обратите внимание, что UCB-значение не является

индексом в строгом понимании, поскольку оно зависит от N, общего количества выборок, выполненных для всех рук, а не только этой руки самой по себе.

Точное определение функции g обусловливает **сожаление** относительно ясновидящей стратегии, в которой просто выбирается лучшая рука, что ведет к оптимальному среднему вознаграждению μ^*. Известный результат Лея и Роббинса ([1338], 1985) показывает, что для случая без обесценивания не существует возможных алгоритмов, имеющих сожаление, возрастающее медленнее, чем $O(\log N)$. Несколько различных вариантов функции g приводят к стратегии UCB, соответствующей этому росту, например можно использовать вариант $g(N) = (2\log(1 + N\log^2 N))^{1/2}$.

Второй метод, ►**выборка Томпсона** (Томпсон [2207], 1933), предусматривает выбор руки случайным образом в соответствии с вероятностью, что эта рука фактически является оптимальной, исходя из выборок, уже сделанных на текущий момент. Предположим, что $P_i(\mu_i)$ является текущим распределением вероятностей для истинного значения руки M_i. Тогда простой способ реализации выборки Томпсона заключается в генерации одной выборки из каждого P_i, а затем отбора наилучшей выборки. Этот алгоритм также имеет сожаление, возрастающее как $O(\log N)$.

17.3.4. Неиндексируемые варианты

Исследование задач о бандитах частично мотивировалось необходимостью тестирования новых методов лечения тяжелобольных пациентов. Для подобных задач цель максимизации общего числа успешных результатов за установленный период времени имеет вполне очевидный смысл: каждый успешный тест означает, что жизнь спасена, а каждый неуспешный — что жизнь потеряна.

Однако если слегка изменить принятые допущения, возникает другая проблема. Предположим, что вместо определения наилучшего способа лечения для каждого нового пациента-человека решено испытывать различные новые препараты на образцах бактериальных культур с целью принять решение, какой из этих препаратов наилучший. Затем выбранный препарат будет запущен в производство, а все остальные варианты будут отклонены. В подобном сценарии не существует никаких дополнительных расходов, связанных с гибелью бактерий, — бактериальная культура для каждого теста имеет фиксированную стоимость, однако наша задача состоит не в том, чтобы минимизировать количество неудачных тестов: на самом деле мы просто пытаемся принять правильное решение настолько быстро, насколько это возможно.

Задачу выбора наилучшего варианта при таких условиях называют ►**задачей отбора**. Задачи подобного типа очень широко распространены в промышленных и кадровых контекстах. Очень часто требуется принять решение, кого из поставщиков следует выбрать для обеспечения того или иного технологического процесса либо кого из кандидатов взять на работу. Задача отбора внешне схожа с задачей

о бандитах, но они имеют различные математические свойства. В частности, ➨ *для задач отбора не существует индексной функции*. Доказательство этого факта требует представить какой-либо сценарий, в котором оптимальная стратегия меняет прежнее предпочтение между двумя руками, M_1 и M_2, когда добавляется третья рука, M_3.

В главе 5 было введено понятие **метарассуждений** при решении задач, таких как выбор, какие вычисления выполнить в процессе поиска по дереву игры, прежде чем сделать ход. Такого рода метарассуждения также являются задачей отбора, а не задачей о бандитах. Очевидно, что развертывание или оценка узла *требует* одинакового количества времени независимо от того, какое выходное значение будет получено, высокое или низкое. Возможно, может показаться удивительным, что алгоритм поиска по дереву методом Монте-Карло (см. раздел 5.4) показал себя настолько успешным, принимая во внимание тот факт, что он пытается решить задачу отбора, используя эвристику UCB, которая была разработана для задач о бандитах. Вообще говоря, можно ожидать, что оптимальные алгоритмы решения задач о бандитах будут требовать выполнения гораздо меньшего количества исследований, чем оптимальные алгоритмы выбора, поскольку в первых предполагается, что неудачное испытание стоит реальных денег.

Важным обобщением процесса задачи о бандитах является ▶**суперпроцесс задачи о бандитах** (*bandit superprocess* — ▶**BSP**), в котором каждая рука представлена полным марковским процессом принятия решения в собственном праве вместо ее представления как марковского процесса вознаграждения только с одним возможным действием. Все остальные свойства остаются теми же: руки независимы, только с одной рукой (или с ограниченным их количеством) можно работать одновременно и существует единственный коэффициент обесценивания.

В качестве примеров задач BSP можно привести саму нашу повседневную жизнь, когда в каждый момент приходится заниматься лишь одной задачей, даже если несколько задач требуют внимания; проект-менеджмент при нескольких проектах; совместное обучение нескольких учеников, нуждающихся в индивидуальном руководстве, и т.д. Общий термин для обозначения всего этого — ▶**многозадачность**, и она настолько вездесуща, что на это уже никто не обращает внимания: в реальном мире при постановке задачи принятия решения аналитики редко спрашивают, есть ли у их клиента другие задачи, не связанные с этой задачи.

Можно рассуждать следующим образом: "Если существует n непересекающихся задач MDP, то очевидно, что в целом оптимальная стратегия строится на основе оптимальных решений отдельных MDP. При известной ее оптимальной стратегии π_i каждая задача MDP превращается в марковский процесс вознаграждения, где в каждом состоянии s существует только одно действие $\pi_i(s)$. Поэтому суперпроцесс n-рукого бандита сокращается до процесса n-рукого бандита". Например, если строительная компания имеет только одну команду строителей и при этом должна построить несколько торговых центров, то, просто следуя здравому смыслу, можно заключить, что необходимо разработать оптимальный план

строительства для каждого торгового центра, а затем решить задачу о бандите для определения, куда отправлять команду строителей в каждый из дней.

Хотя это предположение звучит весьма правдоподобно, оно ошибочно. В действительности глобально оптимальная стратегия для BSP может включать действия, которые будут локально неоптимальны с точки зрения той входящей в нее MDP, в которой они будут предприняты. Причиной здесь является наличие других MDP, в которых действие изменяет баланс между краткосрочным и долгосрочным вознаграждениями в некотором компоненте MDP. Фактически это ведет к тенденции к жадному поведению в каждой MDP (поиск краткосрочных вознаграждений), поскольку нацеливание на получение долгосрочного вознаграждения в одной задаче MDP может вызвать задержку получения вознаграждения во всех остальных MDP.

Например, предположим, что оптимальный локальный график строительства одного торгового центра обеспечивает сдачу в аренду первой торговой площадки через 15 недель, тогда как неоптимальный график предполагает больший уровень затрат, но обеспечивает сдачу первой торговой площадки в аренду через 5 недель. Если предполагается построить четыре торговых центра, то может быть лучше для каждого из них воспользоваться неоптимальным графиком, поскольку тогда арендная плата от них начнет поступать через 5, 10, 15 и 20 недель, а не через 15, 30, 45 и 60. Другими словами, то, что является задержкой на 10 недель для первой MDP, для четвертой MDP превращается в задержку на 40 недель. В общем случае глобально и локально оптимальные стратегии обязательно совпадают только тогда, когда коэффициент обесценивания равен 1, — в этом случае нет никаких потерь от задержки в получении вознаграждений в любой MDP.

Следующий вопрос — как решать задачи BSP. Очевидно, что глобально оптимальное решение для BSP может быть вычислено путем его преобразования в глобальное MDP на декартовом произведении пространств состояний рук. Количество состояний будет возрастать экспоненциально относительно количества рук в BSP, а значит, такой подход будет ужасно непрактичным.

Вместо этого можно воспользоваться разреженной природой взаимодействий между руками. Это взаимодействие возникает исключительно из-за ограниченной способности агента обслуживать руки одновременно. В некоторой степени такое взаимодействие может быть смоделировано понятием ▶**альтернативной стоимости**: сколько полезности будет утеряно на данном этапе времени, если не выделить этот этап другой руке. Чем выше альтернативная стоимость, тем больше необходимость генерации ранних вознаграждений для данной руки. В некоторых случаях оптимальная стратегия в данной руке не зависит от альтернативной стоимости. (Это тривиально верно для марковского процесса вознаграждения, поскольку существует только одна стратегия.) В этом случае оптимальная стратегия может быть применена путем преобразования руки в марковский процесс вознаграждения.

Такая оптимальная стратегия, если она существует, называется ▶**доминирующей стратегией**. Как оказалось, добавляя действия к состояниям, всегда можно создать ослабленную версию MDP (см. раздел 3.6.2) таким образом, чтобы у нее

была доминирующая стратегия, которая в конечном счете даст верхнюю границу для значения действия руки. Нижняя граница может быть вычислена путем решения каждой руки по отдельности (что может дать общую неоптимальную стратегию), а затем вычисления индексов Гиттинса. Если нижняя граница действия одной руки будет выше, чем верхние границы всех других действий, то задача решена. Если нет, то сочетание поиска с опережением (*look-ahead*) и перерасчета границ гарантированно позволит в конечном счете определить оптимальную стратегию для BSP. При таком подходе относительно большие BSP (10^{40} состояний или более) могут быть решены за несколько секунд.

17.4. Марковские процессы принятия решений в частично наблюдаемых средах

В описании марковских процессов принятия решений, приведенном в разделе 17.1, предполагалось, что среда является **полностью наблюдаемой**. При использовании этого предположения агент всегда знает, в каком состоянии он находится. Это предположение, в сочетании с предположением о марковости модели перехода, означает, что оптимальная стратегия зависит только от текущего состояния.

Когда среда является лишь **частично наблюдаемой**, то очевидно, что ситуация становится менее ясной. Агент не всегда точно знает, в каком состоянии он находится, поэтому не может выполнить действие $\pi(s)$, рекомендуемое для этого состояния. Более того, полезность состояния s и оптимальное действие в состоянии s зависят не только от s, но и от того, *насколько много агент знает*, находясь в состоянии s. По этим причинам ▶**задачи MDP в частично наблюдаемой среде** (*Partially Observable MDP* — **POMDP**, читается как "пом-ди-пи") обычно рассматриваются как намного более сложные по сравнению с обычными задачами MDP. Однако невозможно игнорировать необходимость решения задач POMDP, поскольку реальный мир полон таких задач.

17.4.1. Определение задач POMDP

Чтобы найти подход к решению задач POMDP, вначале необходимо должным образом их определить. Любая задача POMDP состоит из тех же компонентов, что и задача MDP — модели перехода $P(s' \mid s, a)$, действий $A(s)$ и функции вознаграждения $R(s, a, s')$, — но также имеет **модель восприятия** $P(e \mid s)$. Здесь, как и в главе 14, модель восприятия задает вероятность получения свидетельства e в состоянии s.[5] Например, клеточный мир 4×3, представленный на рис. 17.1, можно преобразовать в POMDP, добавив зашумленный или частичный датчик вместо предположения, что агент точно знает свое местоположение. Так, можно

[5] Модель восприятия также может зависеть от действия и результирующего состояния, но это изменение не является фундаментальным.

использовать зашумленный четырехбитовый датчик, предложенный в разделе 14.3.2, сообщающий о наличии или отсутствии стены с каждой из четырех сторон света с точностью $1-\epsilon$.

Как и в случае задач MDP, компактные представления для больших задач POMDP можно получить, используя динамические сети принятия решений (см. раздел 17.1.4). Мы добавляем переменные восприятия $\mathbf{E}_t$, исходя из предположения, что переменные состояния $\mathbf{X}_t$ не могут быть непосредственно наблюдаемы. Тогда модель восприятия задачи POMDP задается как $\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$. Например, можно добавить переменные восприятия к сети DDN, представленной на рис. 17.4, такие как $BatteryMeter_t$ для оценки фактического заряда $Battery_t$ и $Speedometer_t$ для оценки величины вектора скорости $\dot{\mathbf{X}}_t$. Ультразвуковой датчик $Walls_t$ может дать оценку расстояния до ближайшей стены в каждом из четырех основных направлений относительно текущей ориентации робота, — эти значения будут зависеть от его текущего положения и ориентации $\mathbf{X}_t$.

В главах 4 и 14 рассматривались задачи планирования в недетерминированных и частично наблюдаемых вариантах среды и было определено **доверительное состояние** — множество фактических состояний, в которых может находиться агент — как ключевая концепция для описания и вычисления решений. В задачах POMDP доверительное состояние b превращается в *распределение вероятностей* по всем возможным состояниям, как в главе 14. Например, начальное доверительное состояние в задаче POMDP для клеточного мира 4×3 может представлять собой равномерное распределение для девяти нетерминальных состояний и нули — для терминальных состояний, т.е.

$$\langle \tfrac{1}{9}, \tfrac{1}{9}, \tfrac{1}{9}, \tfrac{1}{9}, \tfrac{1}{9}, \tfrac{1}{9}, \tfrac{1}{9}, \tfrac{1}{9}, \tfrac{1}{9}, 0, 0\rangle.$$

Мы будем использовать обозначение $b(s)$ для ссылок на вероятность, присвоенную фактическому состоянию s в доверительном состоянии b. Агент может вычислить свое текущее доверительное состояние как распределение условных вероятностей по фактическим состояниям при заданной последовательности восприятий и действий, имевших место до сих пор. Такая задача, по сути, сводится к задаче **фильтрации**, рассматривавшейся в главе 14. Основное рекурсивное уравнение фильтрации (см. уравнение (14.5) в разделе 14.2.1) показывает, как вычислить новое доверительное состояние из предыдущего доверительного состояния и нового свидетельства. Для решения задач POMDP необходимо также учитывать действие, но результат, по сути, остается тем же самым. Если предыдущим доверительным состоянием было b, затем агент выполнил действие a и как результат восприятия получил свидетельство e, то новое доверительное состояние определяется вычислением вероятностей нынешнего пребывания в состоянии s' для каждого s' по следующей формуле:

$$b'(s') = \alpha P(e \mid s') \sum_s P(s' \mid s, a) b(s),$$

где α — нормализующая константа, при использовании которой сумма вероятностей доверительных состояний становится равной 1. По аналогии с оператором обновления для фильтрации (см. раздел 14.2.1) это уравнение можно сокращенно записать как

$$b' = \alpha \, \text{FORWARD}(b, a, e). \tag{17.16}$$

В задаче POMDP для клеточного мира 4×3 предположим, что агент выполняет действие *Left* и его восприятие сообщает о наличии одной стены рядом. Тогда вполне вероятно (хотя и не гарантированно, поскольку и исполнительный механизм перемещения, и датчик зашумлены), что агент в настоящее время находится в квадрате (3,1). В упражнении 17.15 предлагается вычислить точные значения вероятностей для нового доверительного состояния.

Основная концепция, необходимая для понимания сути задач POMDP, состоит в следующем: ➔ *оптимальное действие зависит только от текущего доверительного состояния агента.* А это означает, что оптимальную стратегию можно описать путем отображения $\pi^*(b)$ из доверительных состояний на действия. Она *не* зависит от *фактического* состояния, в котором агент находится. И это очень хорошо, поскольку агент не знает своего фактического состояния и все, что ему известно, — это лишь его доверительное состояние. Следовательно, цикл принятия решения агентом в задаче POMDP можно разбить на следующие три этапа.

1. Учитывая текущее доверительное состояние b, выполнить действие $a = \pi^*(b)$.
2. Получить результаты восприятия — свидетельство e.
3. Установить текущее доверительное состояние равным $\text{FORWARD}(b, a, e)$ и повторить ту же процедуру.

Можно рассматривать задачи POMDP как требующие поиска в пространстве доверительных состояний, во многом аналогично тем методам, которые применялись для решения бессенсорных задач и задач в условиях непредвиденных ситуаций, описанных в главе 4. Основное различие состоит в том, что в задачах POMDP пространство доверительных состояний является *непрерывным*, поскольку доверительное состояние POMDP — это распределение вероятностей. Например, доверительное состояние для клеточного мира 4×3 представляет собой точку в 11-мерном непрерывном пространстве. Любое действие изменяет не только физическое состояние, но и доверительное состояние, поскольку оно влияет на получаемые агентом результаты восприятия. Следовательно, действие оценивается (по меньшей мере, частично) в соответствии с информацией, которую агент получает как результат его выполнения. Таким образом, задачи POMDP должны включать стоимость информации в качестве одного из компонентов задачи принятия решений (см. раздел 16.6).

Давайте внимательнее рассмотрим результаты действий. В частности, рассчитаем вероятность того, что агент, находящийся в доверительном состоянии b,

достигнет доверительного состояния b' после выполнения действия a. Если известны действие *и последующее восприятие*, то уравнение (17.16) должно обеспечить получение *детерминированного* обновления доверительного состояния: $b' = \text{FORWARD}(b, a, e)$. Безусловно, результаты последующего наблюдения еще не известны, поэтому агент может перейти в одно из нескольких возможных доверительных состояний b', в зависимости от результатов восприятия, которые будут получены после выполнения данного действия. Вероятность получения в результате восприятия свидетельства e, при условии, что действие a было выполнено в доверительном состоянии b, определяется путем суммирования по всем фактическим состояниям s', которых агент может достичь:

$$\begin{aligned} P(e|a,b) &= \sum_{s'} P(e|a,s',b)P(s'|a,b) \\ &= \sum_{s'} P(e|s')P(s'|a,b) \\ &= \sum_{s'} P(e|s')\sum_{s} P(s'|s,a)b(s). \end{aligned}$$

Обозначим вероятность достижения состояния b' из b, если дано действие a, как $P(b'|b,a)$. В таком случае вероятность можно рассчитать следующим образом:

$$\begin{aligned} P(b'|b,a) &= \sum_{e} P(b'|e,a,b)P(e|a,b) \\ &= \sum_{e} P(b'|e,a,b)\sum_{s'} P(e|s')\sum_{s} P(s'|s,a)b(s), \end{aligned} \tag{17.17}$$

где $P(b'|e,a,b)$ равно 1, если $b' = \text{FORWARD}(b,a,e)$, и 0 — в противном случае.

Уравнение (17.17) можно рассматривать как определение модели перехода для пространства доверительных состояний. Можем также определить функцию вознаграждения для доверительных состояний, которая выводится из ожидаемого вознаграждения для переходов в фактические состояния, в которых может оказаться агент. Здесь мы используем простую форму $\rho(b,a)$ — ожидаемое вознаграждение, если агент выполняет действие a в доверительном состоянии b:

$$\rho(b,a) = \sum_{s} b(s)\sum_{s'} P(s'|s,a)R(s,a,s').$$

Совместно $P(b'|b,a)$ и $\rho(b,a)$ определяют *полностью наблюдаемую* задачу MDP в пространстве доверительных состояний. Более того, можно показать, что оптимальная стратегия для этой задачи MDP, $\pi^*(b)$, является также оптимальной стратегией для исходной задачи POMDP. Другими словами, ➔ *решение любой задачи POMDP в пространстве физических состояний можно свести к решению задачи MDP в соответствующем пространстве доверительных состояний*. Этот факт, вероятно, станет менее удивительным, если вспомнить, что по определению доверительное состояние всегда является наблюдаемым для агента.

17.5. Алгоритмы для решения задач POMDP

Выше было показано, как свести задачу POMDP к задаче МДП, но полученная МДП будет иметь непрерывное (и, как правило, многомерное) пространство состояний. А это означает, что необходимо перепроектировать алгоритмы динамического программирования, представленные в разделах 17.2.1 и 17.2.2, где предполагалось конечное пространство состояний и конечное число действий. Здесь будет описан алгоритм итерации по значениям, разработанный специально для задач POMDP, а затем представлен неавтономный алгоритм принятия решений, подобный тем, которые были разработаны для игр в главе 5.

17.5.1. Алгоритм итерации по значениям для задач POMDP

В разделе 17.2.1 описывается алгоритм итерации по значениям, в котором вычисляется по одному значению полезности для каждого состояния. При бесконечно большом количестве доверительных состояний необходимо найти более разумный подход. Рассмотрим оптимальную стратегию π^* и ее применение в конкретном доверительном состоянии b: согласно стратегии генерируется действие, затем для каждого последующего восприятия доверительное состояние обновляется и генерируется новое действие, и т.д. Следовательно, для данного конкретного доверительного состояния b эта стратегия в точности эквивалентна **условному плану**, как он был определен в главе 4 для недетерминированных и частично наблюдаемых задач. Вместо того чтобы думать о стратегиях, давайте подумаем об условных планах, а также о том, как ожидаемая полезность от выполнения фиксированного условного плана будет изменяться в зависимости от исходного доверительного состояния. Можно сделать два следующих замечания.

1. Пусть полезность от выполнения *фиксированного* условного плана p, начиная с физического состояния s, будет $\alpha_p(s)$. Тогда ожидаемая полезность от выполнения плана p в доверительном состоянии b будет просто $\Sigma_s b(s)\alpha_p(s)$ или $b \cdot \alpha_p$, если рассматривать их в качестве векторов. Следовательно, ожидаемая полезность фиксированного условного плана изменяется *линейно* относительно b и, значит, соответствует гиперплоскости в пространстве доверительных состояний.
2. В любом заданном доверительном состоянии b оптимальная стратегия будет выбирать для выполнения условный план с самой высокой ожидаемой полезностью, т.е. при оптимальной стратегии ожидаемая полезность состояния b является просто полезностью этого условного плана: $U(b) = U^{\pi^*}(b) = \max_p b \cdot \alpha_p$. Если оптимальная стратегия π^* выбирает для выполнения план p, начиная с состояния b, то будет разумно ожидать, что она может выбрать для выполнения план p и в доверительных состояниях, которые очень близки к состоянию b. В действительности, если ограничить глубину

условных планов, то может иметь место только конечное множество таких планов и непрерывное пространство доверительных состояний в общем случае может быть разделено на *регионы*, каждый из которых соответствует конкретному условному плану, оптимальному для этого региона.

Из этих двух наблюдений следует, что функция полезности $U(b)$ доверительных состояний, представляя собой максимум коллекции гиперплоскостей, будет *кусочно-линейной* и *выпуклой*.

Чтобы проиллюстрировать этот вывод, воспользуемся простым миром с двумя состояниями, — обозначим их как A и B. В этом мире есть два действия: *Stay* — оставаться на месте с вероятностью 0,9 и *Go* — перейти в другое состояние с вероятностью 0,9. Также определены вознаграждения: $R(\cdot, \cdot, A) = 0$ и $R(\cdot, \cdot, B) = 1$; т.е. любой переход, оканчивающийся в состоянии A, имеет нулевое вознаграждение, а за любой переход, оканчивающийся в состоянии B, выдается вознаграждение 1. На данный момент примем коэффициент обесценивания $\gamma = 1$. Датчик выдает правильную информацию о текущем состоянии с вероятностью 0,6. Очевидно, что агенту следует выбрать действие *Stay*, когда он в состоянии B, и выполнить действие *Go*, когда он в состоянии A. Проблема в том, что он не знает наверняка, где находится!

Преимущество мира с двумя состояниями в том, что его пространство доверительных состояний может быть визуализировано в одном измерении, поскольку две вероятности, $b(A)$ и $b(B)$, в сумме дают 1. На рис. 17.15, *а* ось x представляет доверительное состояние, определяемое как $b(B)$, т.е. вероятность того, что агент находится в состоянии B. Теперь давайте рассмотрим два одноэтапных плана, план [*Stay*] и план [*Go*], выполнение каждого из которых приводит к получению следующего вознаграждения за один переход.

$$\alpha_{[Stay]}(A) = 0{,}9R(A, Stay, A) + 0{,}1R(A, Stay, B) = 0{,}1$$
$$\alpha_{[Stay]}(B) = 0{,}1R(B, Stay, A) + 0{,}9R(B, Stay, B) = 0{,}9$$
$$\alpha_{[Go]}(A) = 0{,}1R(A, Go, A) + 0{,}9R(A, Go, B) = 0{,}9$$
$$\alpha_{[Go]}(B) = 0{,}9R(B, Go, A) + 0{,}1R(B, Go, B) = 0{,}1$$

Гиперплоскости (в данном случае это линии) для ожидаемых полезностей $b \cdot \alpha_{[Stay]}$ и $b \cdot \alpha_{[Go]}$ показаны на рис. 17.15, *а*, а их общий максимум представлен утолщенной линией. Следовательно, эта утолщенная линия представляет собой функцию полезности для задачи с конечным горизонтом, допускающим только одно действие, и в каждом "кусочке" кусочно-линейной функции полезности оптимальное действие является первым действием соответствующего условного плана. В данном случае оптимальная одноэтапная стратегия состоит в том, чтобы выбирать действие *Stay*, когда $b(B) > 0{,}5$, и действие *Go* — в противном случае.

Когда у нас есть полезности $\alpha_p(s)$ для всех условных планов p глубины 1 в каждом физическом состоянии s, можно вычислить полезности для условных планов глубины 2, рассматривая каждое возможное первое действие, каждое возможное

последующее восприятие, а затем выполнение каждого варианта плана глубины 1 для каждого восприятия.

[*Stay*; **if** *Percept* = *A* **then** *Stay* **else** *Stay*]
[*Stay*; **if** *Percept* = *A* **then** *Stay* **else** *Go*]
[*Go*; **if** *Percept* = *A* **then** *Stay* **else** *Stay*]
...

Всего существует восемь различных планов глубины 2; их полезность приведена на рис. 17.15, *б*. Обратите внимание, что из этих восьми планов четыре, показанные пунктирными линиями, являются неоптимальными во всем пространстве доверительных состояний, — говорят, что эти планы ▶**доминируемые** (*dominated*), и поэтому нет необходимости продолжать их рассмотрение. Есть еще четыре недоминируемых плана, каждый из которых является оптимальным в определенной области, как показано на рис. 17.15, *в*. Эти области делят пространство доверительных состояний на части.

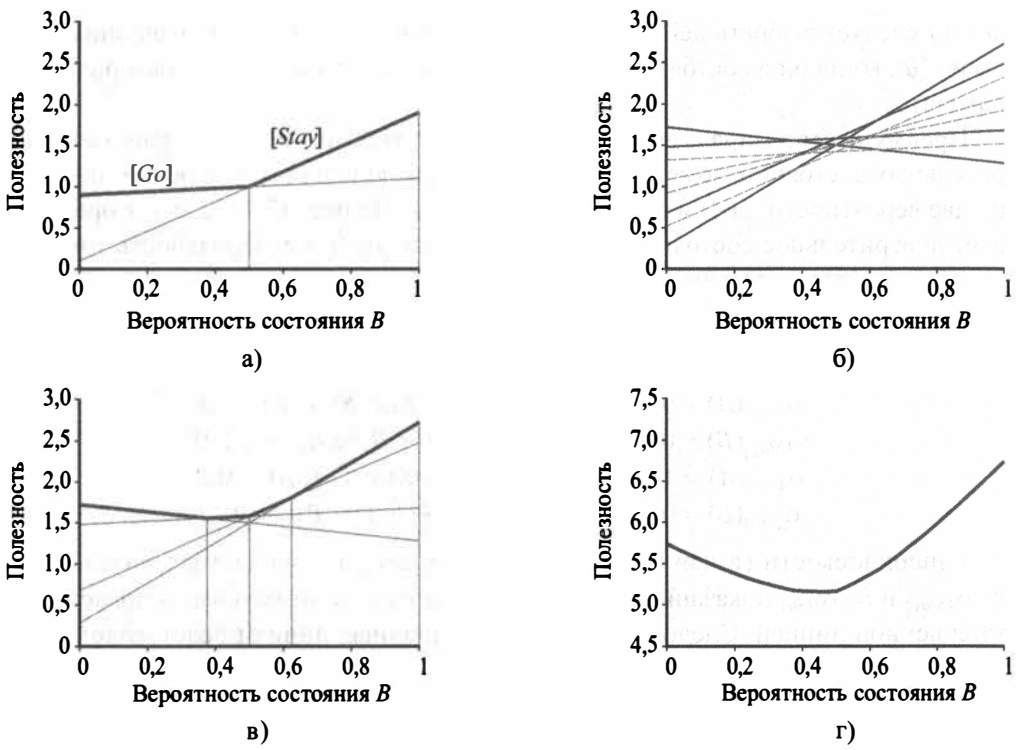


Рис. 17.15. а) Полезность двух одноэтапных планов как функция исходного доверительного состояния $b(B)$ для мира с двумя состояниями, — соответствующая функция полезности приведена утолщенной линией. **б)** Полезности для восьми различных двухэтапных планов. **в)** Полезности для четырех недоминируемых двухэтапных планов. **г)** Функция полезности для оптимальных восьмиэтапных планов

Повторим приведенную выше процедуру для планов глубины 3 и т.д. В общем случае пусть p — условный план глубины d, начальным действием которого является a, а подплан глубины $(d-1)$ для восприятия e есть $p.e$. Тогда можно записать:

$$\alpha_p(s) = \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma \sum_{e} P(e \mid s')\alpha_{p.e}(s')]. \qquad (17.18)$$

Это рекурсивное выражение естественным образом дает нам алгоритм итерации по значениям, который приведен на рис. 17.16. Структура этого алгоритма и его анализ ошибок аналогичны базовому алгоритму итерации по значениям, приведенному на рис. 17.6 в разделе 17.2.1. Основное различие в том, что вместо вычисления одного значения полезности для каждого состояния в алгоритме POMDP-VALUE-ITERATION поддерживается коллекция недоминируемых планов с их гиперплоскостями полезности.

---

function POMDP-VALUE-ITERATION(*pomdp*, ϵ) **returns** функция полезности
  inputs: *pomdp*, задача POMDP с множеством состояний S, множеством действий $A(s)$, моделью перехода $P(s' \mid s, a)$, моделью восприятия $P(e \mid s)$, вознаграждениями $R(s)$, коэффициентом дисконтирования γ
    ϵ, максимальная допустимая ошибка для полезности в любом состоянии
  local variables: U, U', множества планов p со связанными векторами полезности α_p

  $U' \leftarrow$ множество, содержащее только пустой план [], с вектором $\alpha_{[]}(s) = R(s)$
  repeat
    $U \leftarrow U'$
    $U' \leftarrow$ множество всех планов, состоящих из действия и, для каждого возможного следующего восприятия, плана в U с векторами полезности, вычисленными в соотвествии с уравнением (17.18)
    $U' \leftarrow$ REMOVE-DOMINATED-PLANS(U')
  until MAX-DIFFERENCE(U, U') $\leq \epsilon(1-\gamma)/\gamma$
  return U

Рис. 17.16. Высокоуровневый эскиз алгоритма итерации по значениям для задач POMDP. Этап REMOVE-DOMINATED-PLANS и тест MAX-DIFFERENCE обычно реализуются в виде линейных программ

Сложность алгоритма зависит, прежде всего, от того, сколько планов сгенерировано. При заданных $|A|$ действиях и $|E|$ возможных восприятиях, существует $|A|^{O(|E|^{d-1})}$ различных планов глубины d. Даже для скромного мира с двумя состояниями при $d = 8$ это 2^{255} планов. Устранение доминируемых планов исключительно важно для сокращения этого дважды экспоненциального роста: число недоминируемых планов при $d = 8$ составляет всего 144. Функция полезности для этих 144 планов представлена на рис. 17.15, *г*.

Обратите внимание, что промежуточные доверительные состояния имеют более низкое значение, чем состояние A и состояние B, поскольку в промежуточных состояниях агенту не хватает информации, необходимой для выбора хорошего действия. Вот почему информация имеет ценность в смысле, определенном в разделе 16.6, и оптимальные стратегии в задачах POMDP часто включают действия по сбору информации.

При наличии такой функции полезности выполняемая стратегия может быть извлечена за счет анализа, какая из гиперплоскостей является оптимальной при любом заданном доверительном состоянии b, и выполнения первого действия в соответствующем плане. На рис. 17.15, *г* соответствующая оптимальная стратегия все та же, что и для планов глубины 1: выбирать действие *Stay*, когда $b(B) > 0{,}5$, и действие *Go* — в противном случае.

На практике алгоритм итерации по значениям, приведенный на рис. 17.16, является безнадежно неэффективным для задач сколько-нибудь большего размера, — даже задача POMDP для клеточного мира 4×3 для него слишком сложна. Основной причиной является то, что при наличии n недоминируемых условных планов на уровне d этот алгоритм строит $|A| \cdot n^{|E|}$ условных планов на уровне $d+1$ и только затем исключает из них доминируемые. При четырехбитовом датчике $|E|$ равно 16, а количество недоминируемых планов n может исчисляться сотнями, что делает ситуацию совершенно безнадежной.

Поскольку этот алгоритм был разработан в 1970-х годах, за прошедшие годы уже было предложено несколько улучшений, включая более эффективные формы алгоритма итерации по значениям и различные варианты алгоритма итерации по стратегиям. Некоторые из них обсуждаются в разделе "Библиографические и исторические заметки" в конце этой главы. Однако в общем случае нахождение оптимальных стратегий для задач POMDP является очень сложной проблемой (класс PSPACE-hard, т.е. действительно очень трудно). В следующем разделе описывается другой, приблизительный метод решения задач POMDP, основанный на поиске с опережением.

17.5.2. Неавтономные алгоритмы решения задач POMDP

Базовая схема неавтономного POMDP-агента проста: он начинает работу с некоторого априорного доверительного состояния; он выбирает действие, полагаясь на некоторый процесс обдумывания, основанный на его текущем доверительном состоянии; после выполнения действия он получает результаты восприятия и обновляет свое доверительное состояние, используя алгоритм фильтрации; затем процесс повторяется вновь.

Одним очевидным выбором для процесса обдумывания является алгоритм EXPECTIMAX из раздела 17.2.4, — за исключением использования в его дереве доверительных состояний, а не физических состояний в качестве узлов принятия решения. Узлы жеребьевки в POMDP-дереве имеют ветви, помеченные возможными

результатами восприятия и ведущие к следующему доверительному состоянию, с переходными вероятностями, определяемыми по уравнению (17.17). Фрагмент дерева доверительных состояний алгоритма EXPECTIMAX для задачи POMDP клеточного мира 4×3 показан на рис. 17.17.

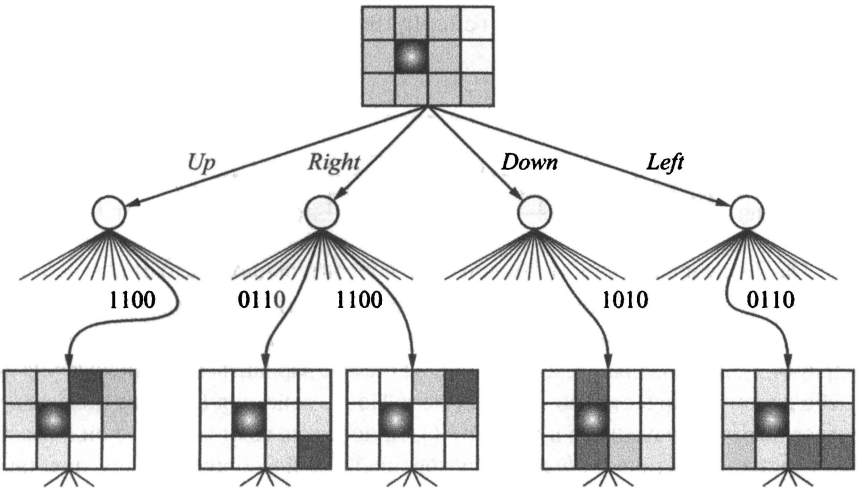


Рис. 17.17. Часть дерева алгоритма EXPECTIMAX для задачи POMDP клеточного мира 4×3 с равномерным исходным доверительным состоянием. Доверительные состояния представлены с затенением, пропорциональным вероятности нахождения в соответствующем квадрате

Временная сложность исчерпывающего поиска на глубину d составляет $O(|A|^d \cdot |\boldsymbol{E}|^d)$, где $|A|$ задает количество доступных действий, а $|\boldsymbol{E}|$ определяет количество возможных результатов восприятия. (Обратите внимание, что это намного меньше, чем количество возможных условных планов глубины d, генерируемых алгоритмом итерации по значениям.) Как и в случае полностью наблюдаемой среды, генерация выборок в узлах жеребьевки представляет собой хороший способ сокращения фактора ветвления без чрезмерного снижения точности окончательного решения. Следовательно, сложность приблизительного неавтономного принятия решений в задачах POMDP может оказаться не намного выше, чем в задачах МДП.

Для очень больших пространств состояний точная фильтрация невозможна, поэтому агент будет вынужден использовать приблизительный алгоритм фильтрации, например фильтрации частиц (см. раздел 14.5.3). В этом случае доверительными состояниями в дереве алгоритма EXPECTIMAX становятся коллекции частиц, а не точные распределения вероятностей. Также для задач с продолжительными горизонтами, возможно, потребуется применить вариант с многократным повторением прогонов в пределах отведенного времени, используемый в алгоритме UCT

(см. рис. 5.11). Комбинацию алгоритма фильтрации частиц и алгоритма UCT применительно к задачам POMDP принято называть частично наблюдаемым планированием по методу Монте-Карло, или ▶ **POMCP** (*Partially Observable Monte Carlo Planning*). При представлении модели в виде DDN алгоритм POMCP — по крайней мере, в принципе — применим к очень большим и реалистичным задачам POMDP. Короткий (и довольно удачный) пример того, насколько компетентное поведение алгоритм POMCP способен генерировать в задаче POMDP для клеточного мира 4×3, показан на рис. 17.18.

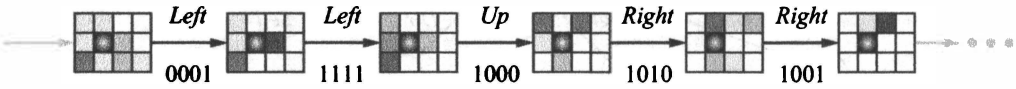


Рис. 17.18. Последовательность восприятий, доверительных состояний и действий в задаче POMDP для клеточного мира 4×3 с погрешностью датчика стены $\epsilon = 0{,}2$. Обратите внимание, насколько безопасны первые действия *Left* — они едва ли приведут в квадрат (4,2), но позволяют свести предполагаемые варианты местоположения агента к небольшому количеству возможных мест. После действия *Up* агент полагает, что, вероятнее всего, он находится в квадрате (3,3), но также возможно, что он в квадрате (1,3). К счастью, действие *Right* является хорошей идеей в обоих этих случаях, так что агент выполняет его и выясняет, что он находился в квадрате (1,3), а теперь находится в квадрате (2,3). Затем он еще раз выполняет действие *Right* и достигает цели

Неавтономные POMDP-агенты на основе динамических сетей принятия решений имеют ряд преимуществ по сравнению с агентами других, более простых конструкций, представленных в предыдущих главах. В частности, они могут действовать в частично наблюдаемых стохастических средах и способны легко пересматривать свои "планы" с целью учета неожиданных свидетельств. При соответствующих моделях восприятия они способны справляться с отказами датчиков и могут планировать действия по сбору информации. Они демонстрируют "изящную деградацию" под давлением времени и в сложных средах, используя различные методы аппроксимации.

Так чего не хватает? Основным препятствием для развертывания таких агентов в реальном мире является их неспособность генерировать успешное поведение в течение достаточно длительного промежутка времени. В случайных или почти случайных сценариях у них нет никакой надежды получить какое-либо положительное вознаграждение, скажем, за накрытие стола для обеда, что может потребовать десятки миллионов отдельных действий по управлению двигателями. Для исправления ситуации представляется необходимым позаимствовать некоторые идеи иерархического планирования, обсуждавшиеся в разделе 11.4. На момент написания данной главы еще не было найдено удовлетворительных и эффективных способов применения этих идей в частично наблюдаемых стохастических средах.

Резюме

В этой главе показано, как использовать знания о мире для принятия решений, даже если результаты действий являются неопределенными, а вознаграждения за действия могут оставаться недоступными до тех пор, пока не будет осуществлен целый ряд действий. Основные моменты заключаются в следующем.

- Задачи последовательного принятия решений в стохастических средах, называемые также **марковскими процессами принятия решений** (*Markov Decision Process* — **MDP**), определяются с помощью **моделей перехода**, задающих вероятностные результаты действий, и **функции вознаграждения**, указывающей, какое вознаграждение соответствует каждому состоянию.
- Полезность последовательности состояний представляет собой сумму всех вознаграждений вдоль этой последовательности, которая, возможно, со временем подвергается обесцениванию. Решением задачи MDP является **стратегия**, в которой с каждым состоянием, достижимым для агента, связано некоторое решение. Оптимальная стратегия максимизирует полезность встречающейся последовательности состояний при ее осуществлении.
- Полезностью состояния является ожидаемая сумма вознаграждений при осуществлении оптимальной стратегии, начиная с этого состояния. Алгоритм **итерации по значениям** действует по принципу итеративного решения уравнений, связывающих полезности каждого состояния с полезностями его соседних состояний.
- В алгоритме **итерации по стратегиям** чередуются этап вычисления полезностей состояний согласно текущей стратегии и этап усовершенствования текущей стратегии по отношению к текущим полезностям.
- Задачи MDP в частично наблюдаемой среде, или задачи POMDP, являются гораздо более трудными для решения, чем задачи MDP. Они могут быть решены путем преобразования в задачу MDP в непрерывном пространстве доверительных состояний и применения уже разработанных для этого случая алгоритмов итерации по значениям или итерации по стратегиям. Оптимальное поведение при решении задач POMDP должно предусматривать сбор информации для уменьшения неопределенности и, соответственно, принятия лучших решений в будущем.
- Для вариантов среды POMDP может быть создан агент, действующий на основе теории принятия решений. В таком агенте **динамическая сеть принятия решений** используется для представления модели перехода и модели восприятия, для обновления его доверительного состояния и для прогноза возможных последовательностей действий в прямом направлении.

Мы вернемся к задачам MDP и POMDP в главе 22, где рассматриваются методы **обучения с подкреплением**, позволяющие агенту совершенствовать свое поведение на основании собственного опыта.

Библиографические и исторические заметки

Идеи, лежащие в основе современного подхода к анализу задач последовательного принятия решений, разрабатывал Ричард Беллман во время работы в корпорации RAND начиная с 1949 года. Как сказано в его автобиографии (Беллман [167], 984), термин "динамическое программирование" он ввел, чтобы скрыть от известного свой фобией к научным исследованиям министра обороны Чарльза Уилсона тот факт, что его группа занималась математикой. (Это едва ли может быть абсолютно верно, потому что первая статья, в которой он использовал этот термин (Беллман [163], 1952), появилась еще до того, как Уилсон стал министром обороны в 1953 году.) В книге Беллмана *Dynamic Programming* ([169], 1957) был заложен прочный фундамент новой области исследований и введен алгоритм итерации по значениям.

Шепли ([2042], 1953) фактически также описал алгоритм итерации по значениям независимо от Беллмана, но его результаты не получили широкого признания в сообществе исследователей операций, — возможно, потому, что они были представлены в более общем контексте марковских игр. Хотя оригинальные формулировки уже включали обесценивание, его анализ с точки зрения стационарных предпочтений был предложен Купмансом ([1281], 1972). Теорема формирования была предложена в работе Нга и соавт. [1674] (1999).

В тезисах докторской диссертации Рона Говарда ([1073], 1960) были предложены алгоритм итерации по стратегиям и идея среднего вознаграждения при решении задач с бесконечным горизонтом. Несколько дополнительных результатов было предложено Беллманом и Дрейфусом ([168], 1962). Идея использования сжатых отображений в ходе анализа алгоритмов динамического программирования принадлежит Денардо ([601], 1967). Модифицированный алгоритм итерации по стратегиям описан в работах Ньюнена ([2259], 1976) и Путермана и Шина ([1828], 1978). Алгоритм асинхронной итерации по стратегиям был проанализирован Уильямсом и Бердом ([2352], 1993), которые также доказали свойство граничной убыточности стратегии, рассматриваемое в уравнении (17.13). Общее семейство алгоритмов **уборки по приоритетам** нацелено на ускорение сходимости к оптимальной стратегии посредством эвристического упорядочения по значению и расчетов обновления стратегии (Мур и Аткесон [1613], 1993; Андре и др. [52], 1998; Уингейт и Сеппи [2358], 2005).

Формулировка задач MDP как задач линейного программирования была рассмотрена де Геллинком ([556], 1960), Манне ([1485], 1960), и Д'Эпену ([608], 1963). Хотя линейное программирование традиционно считалось уступающим динамическому программированию в качестве метода точного решения задач MDP, де Фариас и Рой ([552], 2003) показали, что можно использовать линейное программирование и линейное представление функции полезности для получения доказуемо хороших приближенных решений очень больших задач MDP. Пападимитриу и Цициклис ([1727], 1987), а также Литтман и соавт. ([1420], 1995) приводят

общие результаты по вычислительной сложности задач MDP. Йинью Йе ([2401], 2011) проанализировал взаимосвязь между методом итерации по стратегиям и симплекс-методом линейного программирования и доказал, что для фиксированного коэффициента обесценивания γ время выполнения итерации стратегии зависит полиномиально от количества состояний и действий.

Плодотворные работы Саттона ([2156], 1988) и Уоткинса ([2299], 1989) по применению методов обучения с подкреплением для решения задач MDP сыграли важную роль в ознакомлении сообщества разработчиков в области искусственного интеллекта с задачами MDP. (В более ранней работе Уэрбоса [2324] (1977) содержались во многом аналогичные идеи, но они не были развиты до такой же степени.) Задачи MDP подтолкнули исследователей в области ИИ в направлении использования более выразительных представлений, позволяющих охватить гораздо большие задачи, чем при использовании традиционного атомарного представления на основе матриц перехода.

Основные идеи по созданию архитектуры агента с использованием динамических сетей принятия решений были предложены Дином и Канадзава ([570], 1989). Татмен и Шахтер ([2183], 1990) показали, как применять алгоритмы динамического программирования к моделям DDN. Несколько авторов установили связь между задачами MDP и задачами планирования ИИ, разработав вероятностные формы компактного представления STRIPS для моделей перехода (Веллман [2317], 1990; Кёниг [1257], 1991). В книге *Planning and Control* Дина и Уэллмана ([572], 1991) эта связь исследуется очень глубоко.

В более поздних работах над ►**структурными MDP** (Бутилье и др. [267], 2000; Коллер и Парр [1268], 2000; Гуэстрин и др. [931], 2003) используются структурные представления функции значения, а также модели перехода при доказуемом улучшении в сложности. В концепции ►**реляционных MDP** (Бутилье и др. [268], 2001; Гуэстрин и др. [930], 2003) сделан еще один шаг и используются уже структурные представления для работы в проблемных областях со многими связанными объектами. Задачи МДП и POMDP с открытой вселенной (Шривастава и др. [2120], 2014) также допускают неопределенность в отношении существования и идентичности объектов и действий.

Многие авторы разработали приближенные неавтономные алгоритмы для принятия решений в задачах MDP, часто явно заимствуя их из более ранних подходов в области ИИ к поиску в реальном времени и ведения игр (Уэрбос [2323], 1992; Дин и др. [569], 1993; Тэш и Расселл [2177], 1994). Работа Барто и соавт. ([138], 1995) над RTDP (*real-time dynamic programming* — динамическое программирование в режиме реального времени) предоставила общую основу для понимания работы таких алгоритмов и их связи с обучением с подкреплением и эвристическим поиском. Анализ ограниченного по глубине алгоритма EXPECTIMAX с выборкой в случайных узлах был проведен Кирнсом и соавт. ([1210], 2002). Алгоритм UCT, упоминавшийся в этой главе, был проанализирован Коксисом и Чепешвари ([1251], 2006) и заимствован из ранних работ о

случайных прогонах для оценки значения состояний (Абрамсон [8], 1990; Брюгманн [327], 1993; Чанг и др. [39], 2005).

Задачи о бандитах были предложены Томпсоном ([2207], 1933), но приобрели известность лишь после Второй мировой войны благодаря работе Герберта Роббинса ([1893], 1952). Брадт и соавт. ([285], 1956) подтвердили первые результаты, касающиеся правил останова для задачи однорукого бандита, что в конечном итоге привело к наиболее значимым результатам, полученным Джоном Гиттинсом (Гиттинс и Джонс [866], 1974; Гиттинс [865], 1989). Катехакис и Вейнотт ([1196], 1987) предложили перезапуск MDP как метод вычисления индексов Гиттинса. Учебник Берри и Фристедта ([195], 1985) охватывает множество вариантов основной задачи, в то время как в простом и понятном интерактивном учебнике Фергюсона ([730], 2001) задачи о бандитах связываются с задачами останова.

Лей и Роббинс ([1338], 1985) инициировали исследование асимптотического сожаления оптимальной стратегии в задачах о бандитах. Эвристика UCB была введена и проанализирована Ауэром и соавт. ([90], 2002). Суперпроцессы задачи о бандитах (*bandit superprocesses* — BSP) были впервые изучены Нэшем ([1659], 1973), но эти результаты оставались малоизвестными в области ИИ. Хедфилд-Менелл и Рассел ([944], 2015) описали эффективный алгоритм ветвей и границ, способный решать относительно большие задачи BSP. Задачи отбора были введены Беххофером ([152], 1954). Хей и соавт. ([988], 2012) разработали более формальную основу для задач метарассуждений, показав, что для отбора простые экземпляры подходят лучше, чем задачи о бандитах. Они также доказали удовлетворительный результат, что ожидаемая вычислительная стоимость оптимальной вычислительной стратегии никогда не бывает выше ожидаемого выигрыша в качестве решения, хотя существуют случаи, когда оптимальная стратегия может, с некоторой вероятностью, продолжать вычисления и после того, как точка любого возможного выигрыша уже была пройдена.

Обнаружение того факта, что любая частично наблюдаемая MDP может быть преобразована в обычную MDP в пространстве доверительных состояний, принадлежит Астрому ([87], 1965) и Аоки ([59], 1965). Первый полный алгоритм для точного решения POMDP — по сути, алгоритм итерации по значениям, представленный в этой главе — был предложен Эдвардом Сондиком ([2109], 1971) в тезисах его докторской диссертации. (Более поздняя журнальная статья Смоллвуда и Сондика ([2083], 1973) содержит некоторые ошибки, но более доступна.) Лавджой ([1446], 1991) дает обзор первых двадцати пяти лет исследований в области POMDP, делая несколько пессимистические выводы о целесообразности решения больших задач.

Первым значительным вкладом в рамках ИИ был алгоритм Witness (Кассандра и др. [378], 1994; Каэлблинг и др. [1165], 1998) — усовершенствованная версия алгоритма итерации по значениям для задач POMDP. Вскоре были разработаны другие алгоритмы, в том числе основанные на подходе, предложенном Хансеном ([959], 1998), который предусматривает инкрементное построение стратегии с

помощью конечного автомата, состояния которого определяют возможные доверительные состояния агента.

В последнее время работа в области ИИ была сосредоточена на **точечных методах** итерации по значениям, когда на каждой итерации генерируются условные планы и α-векторы для конечного множества доверительных состояний, а не для всего пространства доверительных состояний. Лавджой ([1446], 1991) предложил такой алгоритм для фиксированной сетки точек, — подход, также принятый Бонетом ([246], 2002). Во влиятельной статье Пино и соавт. [1790] (2003) предлагается генерация достижимых точек с помощью моделирования траектории некоторым жадным образом. По наблюдениям Спана и Влассиса ([2112], 2005), следует генерировать планы только для небольшого, случайно выбранного подмножества точек, что позволит улучшить планы от предыдущей итерации для всех точек в множестве. Шани и соавт. ([2037], 2013) предоставили обзор этих и других усовершенствований в точечных алгоритмах, которые уже привели к успешному решению задач с тысячами состояний. Поскольку задачи POMDP относятся к классу PSPACE-hard (Пападимитриу и Цициклис [1727], 1987), дальнейший прогресс в автономных методах их решения может потребовать использования преимуществ различных видов структуры функций значения, вытекающих из развернутого представления модели.

Неавтономный подход для задач POMDP — использование упреждающего поиска для выбора действия в текущем доверительном состоянии — впервые исследовался Сатией и Лейвом ([1980], 1973). Использование выборки в случайных узлах аналитически исследовалось Кернсом и соавт. ([1206], 2000) и Нгом и Джорданом ([1675], 2000). Алгоритм POMCP был предложен Сильвером и Венессом ([2062], 2011).

С разработкой достаточно эффективных алгоритмов аппроксимации для задач POMDP их использование в качестве моделей для задач реального мира постоянно возрастало, особенно в образовании (Рафферти и др. [1844], 2016), диалоговых системах (Янг и др. [2411], 2013), робототехнике (Хсяо и др. [1079], 2007; Хьюн и Рой [1108], 2009) и при разработке самоуправляемых автомобилей (Форбес и др. [750], 1995; Баи и др. [112], 2015). Важным крупномасштабным применением является система Airborne Collision Avoidance System X (ACAS X), которая удерживает самолеты и беспилотные летательные аппараты от столкновений в воздухе. В этой системе POMDP на нейронных сетях используются для выполнения функции аппроксимации. Система ACAS X значительно повышает безопасность полетов в сравнении с устаревшей системой TCAS, которая была построена в 1970-е годы с использованием технологии экспертных систем (Кохендерфер [1250], 2015; Джулиан и др. [1159], 2018).

Принятие комплексных решений также изучалось экономистами и психологами. Они выяснили, что лица, принимающие решения, не всегда рациональны и могут не придерживаться в точности того поведения, которое следует из моделей, обсуждавшихся в этой главе. Например, когда человеку предоставляется выбор, большинство предпочитает 100 долл. сегодня вместо гарантированных

200 долл. в течение двух лет, но те же самые люди предпочитают 200 долл. в течение восьми лет вместо 100 долл. в течение шести лет. Один из вариантов интерпретации этого результата заключается в том, что люди не используют аддитивные экспоненциально обесцениваемые вознаграждения; возможно, они используют ▶ **гиперболические вознаграждения** (для небольших значений гиперболическая функция уменьшается быстрее, чем экспоненциально убывающая функция). Эта и другие возможные интерпретации обсуждаются Рубинштейном ([1929], 2003).

В учебниках Берцекаса ([200], 1987) и Путермана ([1827], 1994) дается подробное введение в методы решения задач последовательного принятия решений и динамическое программирование. Учебник Берцекаса и Цицикаса ([201], 1996) дополнительно включает тему обучения с подкреплением. Саттон и Барто ([2159], 2018) освещают тот же круг вопросов, но в более доступном стиле. Сигу и Бюффе ([2058], 2010), Маусам и Колобов ([1517], 2012), а также Кочендерфер ([1250], 2015) рассматривают задачу последовательного принятия решений с точки зрения ИИ. Кришнамурти ([1311], 2016) предоставляет полное освещение области решения задач POMDP.

Упражнения

17.1. Для клеточного мира 4×3, представленного на рис. 17.1, рассчитайте, какие квадраты могут быть достигнуты из квадрата (1,1) с помощью последовательности действий [*Up*, *Up*, *Right*, *Right*, *Right*] и с какими вероятностями. Объясните, как эти вычисления связаны с задачей прогнозирования (см. раздел 14.3) для скрытой марковской модели.

17.2. Для клеточного мира 4×3, представленного на рис. 17.1, рассчитайте, какие квадраты могут быть достигнуты из квадрата (1,1) с помощью последовательности действий [*Right*, *Right*, *Right*, *Up*, *Up*] и с какими вероятностями. Объясните, как эти вычисления связаны с задачей прогнозирования (см. раздел 14.3) для скрытой марковской модели.

17.3. Выберите конкретный член множества стратегий, являющихся оптимальными для $R(s) > 0$, как показано на рис. 17.2, *б*, и вычислите долю времени, которое агент в пределе проводит в каждом состоянии, если стратегия выполняется бесконечно. (*Подсказка*. Создайте матрицу вероятностей перехода между состояниями, соответствующую выбранной стратегии, затем см. упражнение 14.2.)

17.4. Предположим, что в качестве полезности последовательности состояний определено максимальное вознаграждение, полученное в любом из состояний этой последовательности. Покажите, что данная функция полезности не приводит к формированию стационарных предпочтений между последовательностями состояний. Остается ли возможность определить такую функцию полезности на состояниях, что принятие решений на основе полезности MEU позволит сформировать оптимальное поведение?

17.5. Может ли любая задача конечного поиска быть точно преобразована в такую марковскую задачу принятия решений, что ее оптимальное решение будет также оптимальным решением исходной задачи поиска? Если да, то объясните, как преобразовать эту задачу и как выполнить обратное преобразование решения, а если нет, объясните, почему это невозможно (в частности, приведите контрпример).

17.6. Иногда задачи MDP формулируются на основе функции вознаграждения $R(s, a)$, которая зависит от выполненного действия, либо на основе функции вознаграждения $R(s, a, s')$, которая зависит и от результирующего состояния.

а) Запишите уравнения Беллмана для этих двух формулировок.

б) Покажите, как можно преобразовать задачу MDP с функцией вознаграждения $R(s, a, s')$ в другую задачу MDP с функцией вознаграждения $R(s, a)$ — такую, что оптимальные стратегии в новой задаче MDP будут точно соответствовать оптимальным стратегиям в первоначальной задаче MDP.

в) Выполните аналогичные действия для преобразования задачи MDP с функцией вознаграждения $R(s, a)$ в задачу MDP с функцией вознаграждения $R(s)$.

17.7. Для среды, показанной на рис. 17.1, найдите все пороговые значения для $R(s)$, такие что оптимальная стратегия изменяется при пересечении этого порога. Вам понадобится способ расчета оптимальной стратегии и ее значения для фиксированных $R(s)$. (*Подсказка.* Докажите, что значение любой фиксированной стратегии изменяется линейно в зависимости от $R(s)$.)

17.8. Уравнение (17.11) в разделе 17.2.1 показывает, что оператор Беллмана является функцией сжатия.

а) Покажите, что для любых функций f и g

$$|\max_a f(a) - \max_a g(a)| \leq \max_a |f(a) - g(a)|.$$

б) Запишите выражение для

$$|(BU_i - BU_i')(s)|,$$

а затем примените результат из п. *а*, чтобы завершить доказательство того, что оператор Беллмана является функцией сжатия.

17.9. В этом упражнении рассматриваются задачи MDP с двумя игроками, которые соответствуют играм с нулевой суммой и поочередными ходами, подобными описанным в главе 5. Примем, что игроки обозначены как A и B, и пусть $R(s)$ — вознаграждение игрока A в состоянии s. (Вознаграждение игрока B всегда равно ему и противоположно.)

а) Пусть $U_A(s)$ — полезность состояния s, когда очередь хода в состоянии s принадлежит игроку A, а $U_B(s)$ — полезность состояния s, когда очередь хода в состоянии s принадлежит игроку B. Все вознаграждения и полезности вычисляются с точки зрения игрока A (как в минимаксном дереве игры). Запишите уравнения Беллмана, определяющие $U_A(s)$ и $U_B(s)$.

б) Объясните, как с помощью этих уравнений реализовать алгоритм итерации по значениям для двух игроков, и определите подходящий критерий останова.

в) Рассмотрите игру, приведенную на рис. 5.18. Нарисуйте пространство состояний (а не дерево игры), показывая ходы игрока A сплошными линиями, а ходы игрока B — пунктирными линиями. Обозначьте каждое состояние значением $R(s)$. Вы обнаружите, что удобнее всего расположить состояния (s_A, s_B) в виде двухмерной решетки, используя в качестве "координат" значения s_A и s_B.

г) Теперь примените алгоритм итерации по значениям для двух игроков, чтобы найти решение этой игры, и определите оптимальную стратегию.

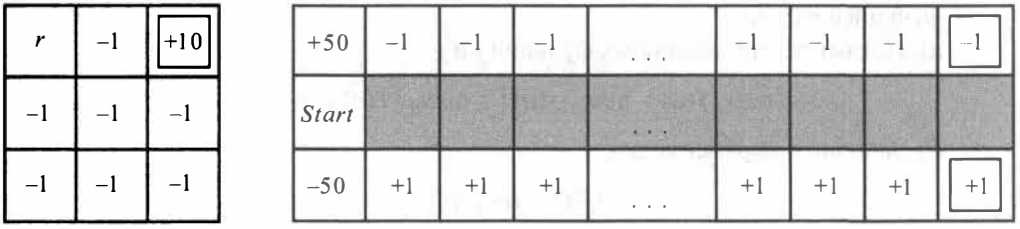
17.10. Рассмотрите клеточный мир 3 × 3, приведенный на рис. 17.19, *а*. Модель перехода такая же, как в случае клеточного мира 4 × 3, представленного на рис. 17.1: в 80% случаев агент перемещается в выбранном направлении, а в остальных случаях совершает переход под прямым углом к нему. Примените к этому миру алгоритм итерации по значениям для каждого значения r из числа приведенных ниже. Используйте обесценивание вознаграждения с коэффициентом обесценивания 0,99. Приведите стратегию, полученную для каждого случая. Объясните своими словами, почему данное значение r приводит к соответствующей стратегии.

а) $r = -100$

б) $r = -3$

в) $r = 0$

г) $r = +3$



а) б)

Рис. 17.19. а) Клеточный мир 3 × 3 для упражнения 17.10. Указано вознаграждение за каждое состояние. Верхний правый квадрат является конечным состоянием. **б)** Клеточный мир 101 × 3 для упражнения 17.11 (в середине пропущено 93 одинаковых столбца). Начальное состояние *Start* имеет вознаграждение 0

17.11. Рассмотрите клеточный мир 101 × 3, приведенный на рис. 17.19, *б*. В начальном состоянии *Start* у агента есть выбор из двух детерминированных действий, *Up* (вверх) или *Down* (вниз), но в других состояниях у агента есть лишь одно детерминированное действие — *Right* (вправо). При использовании обесцениваемой функции вознаграждения для каких значений коэффициента обесценивания γ агент должен выбрать действие *Up*, а для каких — действие *Down*? Вычислите полезность каждого действия как функцию от γ. (Обратите внимание,

что этот простой пример на самом деле отражает множество реальных ситуаций, в которых нужно взвесить значение немедленного действия в сравнении с возможными постоянными долгосрочными последствиями, такими как выбор сброса загрязняющих веществ в озеро.)

17.12. Рассмотрите задачу MDP без обесценивания, имеющую три состояния, (1, 2, 3), с вознаграждениями −1, −2 и 0 соответственно. Состояние 3 является терминальным. В состояниях 1 и 2 имеются два возможных действия, *a* и *b*. Модель перехода описана ниже.

- В состоянии 1 действие *a* переводит агента в состояние 2 с вероятностью 0,8 и оставляет агента в том же состоянии с вероятностью 0,2.
- В состоянии 2 действие *a* переводит агента в состояние 1 с вероятностью 0,8 и оставляет агента в том же состоянии с вероятностью 0,2.
- Как в состоянии 1, так и в состоянии 2 действие *b* переводит агента в состояние 3 с вероятностью 0,1 и оставляет агента в прежнем состоянии с вероятностью 0,9.

Дайте ответы на следующие вопросы.

а) Какие характеристики оптимальной стратегии в состояниях 1 и 2 могут быть определены качественно?

б) Примените алгоритм итерации по стратегиям для определения оптимальной стратегии и значения состояний 1 и 2, полностью демонстрируя каждый этап. Примите предположение, что исходная стратегия включает выполнение действия *b* в обоих состояниях.

в) Как повлияет на работу алгоритма итерации по стратегиям включение в исходную стратегию действия *a* в обоих состояниях? Поможет ли применение обесценивания? Зависит ли оптимальная стратегия от коэффициента обесценивания?

17.13. Рассмотрим клеточный мир 4 × 3, приведенный на рис. 17.1.

а) Реализуйте имитатор для этой среды, такой, чтобы можно было легко изменять ее географию. Некоторый код для решения указанной задачи уже находится в интерактивном репозитории кода.

б) Создайте агента с использованием алгоритма итерации по стратегиям и измерьте его производительность в имитаторе среды, начиная с различных начальных состояний. Выполните несколько экспериментов из каждого начального состояния и сравните среднее суммарное вознаграждение, полученное за каждый прогон, с полезностью состояния, как она определяется применяемым алгоритмом.

в) Проведите эксперименты в среде с увеличенными размерами. Как изменяется время прогона алгоритма итерации по стратегиям в зависимости от размеров среды?

17.14. Как можно использовать алгоритм определения значения для расчета ожидаемых потерь, которые понесет агент, используя заданное множество оценок полезности U и модель оценки P, по сравнению с агентом, использующим правильные значения?

17.15. Пусть начальное доверительное состояние b_0 для задачи POMDP клеточного мира 4×3, как она была определена в разделе 17.4.1, представляет собой равномерное распределение по нетерминальным состояниям, т.е.

$$\langle \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, 0, 0 \rangle.$$

Вычислите точное доверительное состояние b_1 после того, как агент выполнит действие *Left* и его датчик засвидетельствует наличие одной стены по соседству. Затем рассчитайте доверительное состояние b_2, предполагая, что вновь произошло то же самое.

17.16. Какова временн*ая* сложность выполнения d этапов алгоритма итерации по значениям для задачи POMDP в бессенсорной среде?

17.17. Рассмотрите такую версию приведенной в разделе 17.5.1 задачи POMDP с двумя состояниями, A и B, в которой датчик на 90% надежен в состоянии A, но не предоставляет информации в состоянии B (т.е. сообщает о нахождении в A или B с равной вероятностью). Проанализируйте, качественно или количественно, функцию полезности и оптимальную стратегию для этой задачи.

ГЛАВА 18

Принятие решений при наличии нескольких агентов

В этой главе рассматривается, что делать, если в проблемной среде присутствует более одного агента.

18.1. Свойства мультиагентной среды

До настоящего момента в основном предполагалось, что все процессы восприятия, планирования и действия осуществляет только один агент. Но в действительности это является чрезвычайно упрощающим предположением, не позволяющим охватить в теории ИИ многие установки реального мира. Поэтому в этой главе рассматриваются проблемы, возникающие в тех случаях, когда агент должен принимать решения в средах, содержащих несколько действующих лиц. Такие среды называются ►**мультиагентными системами**, а действующие в них агенты сталкиваются с ►**задачей мультиагентного планирования**. Тем не менее, как будет показано ниже, точный характер задачи мультиагентного планирования — а значит, и методы, которые подходят для ее решения, — будет зависеть, прежде всего, от отношений между агентами в проблемной среде.

18.1.1. Решения принимаются одним агентом

Особенность первого варианта мультиагентной среды состоит в том, что, хотя она и содержит несколько *действующих лиц*, есть только одно лицо, *принимающее решения*. В этом случае агент, принимающий решения, разрабатывает планы для всех других агентов и сообщает им, что делать. Предположение, что агенты будут просто делать то, что им говорят, называется ►**предположением о доброжелательности агента**. Однако даже в этом простом случае планы с участием нескольких исполнителей потребуют от них *синхронизации* своих действий. Исполнители *А* и *Б* должны будут действовать одновременно для выполнения совместных действий (таких, как пение дуэтом), в разное время — для выполнения взаимно исключающих действий (таких, как зарядка аккумулятора, когда есть только одна

розетка) и последовательно — когда один обеспечивает предварительные условия для действий другого (например, *A* моет посуду, после чего *B* вытирает ее насухо).

В одном особом случае имеется единственный принимающий решения агент с несколькими исполнительными механизмами (или *эффекторами*), которые могут работать одновременно, например человек, который может ходить и разговаривать в одно и то же время. Такой агент должен осуществлять ▶**многоэффекторное планирование** для успешного управления каждым эффектором, одновременно обрабатывая позитивные и негативные взаимодействия между исполнительными механизмами. Когда исполнительные механизмы физически разделены на отдельные единицы — как в случае парка роботов доставки материалов и комплектующих на заводе, — многоэффекторное планирование становится ▶**планированием для многих исполнителей** (*multibody planning*).

Планирование для многих исполнителей — это все еще "стандартная" задача одного агента до тех пор, пока необходимая информация от датчиков, поступающая от каждого исполнителя, может быть объединена — либо централизовано, либо в каждом исполнителе, — чтобы сформировать общую оценку состояния мира, позволяющую контролировать ход выполнения общего плана. В этом случае можно полагать, что все множество исполнителей действует как единый агент. Когда ограничения в отношении обмена информацией делают такой подход невозможным, ситуация превращается в то, что иногда называют задачей ▶**децентрализованного планирования**. Возможно, такое определение неточно, потому что фаза планирования как раз централизована, а вот фаза выполнения, по крайней мере частично, осуществляется исполнителями независимо. В таких случаях подпланы, созданные для каждого исполнителя, возможно, должны будут включать четкие указания о необходимых действиях по коммуникации с другими исполнителями. Например, несколько роботов-разведчиков, разбросанных по обширной территории, часто могут не иметь радиосвязи друг с другом и должны будут делиться своими результатами в то время, когда связь будет возможна.

18.1.2. Решения принимаются многими агентами

Второй возможный вариант проблемной среды — когда прочие действующие в ней исполнители также являются лицами, принимающими решения: у каждого из них есть свои предпочтения, и они выбирают и выполняют собственный план. Будем называть таких участников ▶**партнерами** (*counterparts*). В этом случае дополнительно можно выделить два возможных варианта.

- В первом случае, хотя имеется множество лиц, принимающих решения, все они преследуют ▶**общую цель**. Это примерно соответствует ситуации компании с несколькими работниками, в которой различные принимающие решения лица преследуют, можно надеяться, одну и ту же цель от имени компании. Основная проблема, с которой сталкиваются принимающие решения

исполнители в подобной ситуации, — это ▶**задача координации**: они должны быть уверены, что все участники всегда действуют в одном направлении и не нарушат, даже случайным образом, выполнение планов друг друга.

- Второй вариант — когда каждый принимающий решения исполнитель имеет собственные личные предпочтения, которым каждый из них и будет следовать в меру своих способностей. Возможно, эти предпочтения окажутся диаметрально противоположными, как в играх с нулевой суммой, таких как шахматы (см. главу 5). Однако в большинстве случаев ситуация в мультиагентных средах будет гораздо сложнее из-за более запутанных отношений между предпочтениями агентов.

Когда имеется несколько принимающих решения исполнителей, каждый из которых действует, исходя их собственных предпочтений, агент должен учитывать предпочтения других агентов, а также тот факт, что эти другие агенты *также* принимают во внимание предпочтения других агентов, и так далее. И это приводит нас в область ▶**теории игр** — теорию принятия стратегических решений. Именно этот *стратегический* аспект рассуждений — каждый игрок принимает во внимание, как могут действовать другие игроки — отличает теорию игр от теории принятия решений. Подобно тому, как теория принятия решений дает теоретическую основу для выработки решений в задачах ИИ с единственным агентом, теория игр предоставляет теоретическую основу для принятия решений в мультиагентных системах.

Использование слова "игра" в данном случае нельзя считать идеальным: напрашивается вполне естественный вывод, что теория игр по большей части касается сугубо развлекательных занятий или искусственных сценариев. Однако нет ничего более далекого от правды. Теория игр — это теория ▶**принятия стратегических решений**. Она используется при необходимости принятия решений в различных ситуациях, включая аукционы на получение прав на бурение нефтяных скважин и прав на использование участка спектра радиочастот, процедуры объявления банкротства, принятие решений о выпуске промышленной продукции и назначении цен на нее, а также национальную оборону, — ситуации, связанные с денежными потоками в миллиарды долларов и многими человеческими жизнями. В области ИИ теорию игр можно использовать двумя основными способами.

1. ▶**Проектирование агента**. Теория игр может использоваться агентом для анализа возможных решений и вычисления ожидаемой полезности для каждого из них (в предположении, что другие агенты также действуют оптимальным образом согласно теории игр). Следовательно, теория игр позволяет найти наилучшую стратегию против рационально действующего игрока и определить ожидаемый выигрыш для каждого игрока.
2. ▶**Проектирование механизма**. Если в среде присутствует много агентов, то может существовать возможность так определить правила действий в этой среде (т.е. правила игры, в которую должны играть агенты), чтобы

общее благосостояние всех агентов максимизировалось, если каждый агент принимает обоснованное теорией игр решение, максимизирующее его собственную полезность. Например, теория игр позволяет разрабатывать такие протоколы для некоторого набора маршрутизаторов Интернета, чтобы каждый маршрутизатор стремился действовать в направлении максимизации глобальной пропускной способности. Проектирование механизма может также использоваться для создания интеллектуальных мультиагентных систем, решающих сложные задачи распределенным образом.

Теория игр предлагает ряд различных моделей, в каждой из которых используется собственный набор базовых допущений, — и здесь важно выбрать правильную модель для каждого случая. Самое важное различие заключается в том, следует ли считать данный случай кооперативной игрой.

- В ▶**кооперативной игре** между агентами возможно заключение обязывающего соглашения (*binding agreement*), которое будет обеспечивать их надежное сотрудничество. В мире людей устанавливать такие обязывающие соглашения помогают юридические контракты и социальные нормы. В мире компьютерных программ может оказаться возможным выполнение проверки исходного кода, чтобы убедиться, что он будет следовать соглашению. Для анализа этой ситуации можно использовать теорию кооперативных игр.
- Если заключение между игроками обязывающих соглашений невозможно, мы имеем дело с ▶**некооперативной игрой**. Хотя сам этот термин предполагает, что данная игра по природе конкурентна и любое сотрудничество в ней не представляется возможным, это не тот случай: здесь "некооперативная" просто означает, что в игре нет центрального соглашения, которое связывает всех агентов и гарантирует их сотрудничество. Но вполне может оказаться, что в процессе игры агенты самостоятельно решат установить сотрудничество, потому что это будет в их собственных интересах. В подобных ситуациях используется теория некооперативных игр.

Некоторые проблемные среды могут объединять в себе несколько разных измерений. Например, компания доставки пакетов может применять независимое централизованное планирование для ежедневных маршрутов ее машин и самолетов, но оставить некоторые аспекты плана открытыми для автономных решений водителей и пилотов, которые в результате получают возможность самостоятельно реагировать на особенности дорожного движения и погодных условий. Кроме того, цели компании и ее сотрудников приведены в соответствие (до некоторой степени) благодаря выплате ▶**стимулов** (прибавок к зарплате и бонусов) — верный признак того, что это истинная мультиагентная система.

18.1.3. Мультиагентное планирование

В этом разделе мы не будем проводить различия между многоэффекторным окружением со многими исполнителями и мультиагентным окружением, понимая любое из них под общим термином ▶ **многосубъектное** (*multiactor*) окружение и используя общий термин ▶ **субъект** или **актор** (*actor*) для эффекторов, исполнительных механизмов и агентов. Целью данного раздела является разработка способов определения моделей перехода, корректных планов и эффективных алгоритмов планирования для многосубъектного окружения. Корректным является тот план, который после выполнения субъектами позволяет достичь цели. (Безусловно, в истинном мультиагентном окружении агенты могут не согласиться выполнять некоторый конкретный план, но по крайней мере они будут знать, какие планы *будут* работать, если они *согласятся* их выполнить.)

Основная трудность в попытках построить удовлетворительную модель мультиагентных действий заключается в том, что необходимо как-то справиться с щекотливым вопросом ▶ **взаимной совместимости** (*concurrency*), который здесь просто означает, что планы каждого из агентов могут быть выполнены одновременно. Чтобы рассуждать о выполнении многосубъектных планов, сначала нужно найти такую модель многосубъектных планов, которая будет включать удовлетворительную модель параллельных действий.

Кроме того, многосубъектное действие связано с целым рядом вопросов, которые не принимаются во внимание при планировании действий единственного агента. В частности, ➨ *агенты должны учитывать, каким образом их собственные действия могут влиять на действия других агентов*. Например, агенту будет необходимо учитывать, будут ли действия, выполняемые другими агентами, как-то искажать предусловия его собственных действий, являются ли ресурсы, которые он намерен использовать в процессе выполнения своей стратегии, разделяемыми и могут ли они быть полностью растрачены другими агентами; являются ли действия взаимоисключающими; а склонный к оказанию помощи агент мог бы также учесть, как его действия могут облегчить действия другим.

Чтобы ответить на эти вопросы, необходима модель параллельного действия, в рамках которой можно было бы эти действия правильно формулировать. Модели параллельных действий были основным направлением исследований в сообществе компьютерных наук в течение десятилетий, но ни одна окончательная, общепринятая модель пока не достигла абсолютного преобладания. Тем не менее три следующих подхода получили достаточно широкое распространение.

Первый подход предполагает ▶ **чередующееся выполнение** (*interleaved execution*) действий в соответствующих планах. Например, предположим, что есть два агента, A и B, со следующими планами.

$$A : [a_1, a_2]$$
$$B : [b_1, b_2]$$

Ключевая идея модели чередующегося выполнения состоит в том, что единственное, в чем мы можем быть уверены в отношении выполнения планов этих двух агентов, — порядок действий в соответствующих планах будет сохранен. Далее, если теперь предположить, что действия являются атомарными, то имеется шесть различных способов, которыми два приведенных выше плана могут быть выполнены одновременно.

$$[a_1, a_2, b_1, b_2]$$
$$[b_1, b_2, a_1, a_2]$$
$$[a_1, b_1, a_2, b_2]$$
$$[b_1, a_1, b_2, a_2]$$
$$[a_1, b_1, b_2, a_2]$$
$$[b_1, a_1, a_2, b_2]$$

Для того чтобы план в модели чередующегося выполнения был корректным, ➔ *он должен быть корректным по отношению ко всем возможным вариантам чередования планов.* Модель чередующегося выполнения широко применяется в сообществе специалистов по параллельным вычислениям, поскольку это разумная модель того, как несколько потоков вычислений по очереди могут работать на одном процессоре. Однако она не являются подходящей моделью для тех случаев, когда два действия выполняются в одно и то же время. Кроме того, количество последовательностей чередования будет экспоненциально возрастать с увеличением числа агентов и действий: как следствие проверка плана на корректность, вычислительно простая в среде с одним агентом, в модели чередующегося выполнения становится вычислительно сложной.

Вторым подходом является ▶ **истинный параллелизм**, в котором не предпринимается попыток создать полностью упорядоченную последовательность действий. Действия остаются лишь *частично упорядоченными*: известно, что действие a_1 будет выполнено до действия a_2, но ничего нельзя сказать в отношении упорядочения, скажем, действий a_1 и b_1: любое из них может произойти раньше другого или они произойдут одновременно. Мы всегда можем "выпрямить" частично упорядоченную модель параллельных планов в модель чередующегося выполнения, но в результате будет утрачена информация о частичной упорядоченности. Хотя модели с частичным упорядочением являются, возможно, более удовлетворительными, чем модели чередующегося выполнения, согласно теоретической оценке по параллельности действия, на практике они не получили широко признания.

Третий подход заключается в предположении идеальной ▶ **синхронизации**: имеются некие глобальные часы, к которым каждый агент имеет доступ, каждое действие выполняется за один и тот же промежуток времени и действия в каждой точке совместного плана являются одновременными. Таким образом, действия всех агентов выполняются синхронно, в ногу друг с другом (также может быть, что некоторые агенты выполняют действие "нет операции", т.е. просто ждут, пока другие

агенты завершат выполнение своих действий). Синхронное выполнение является не совсем полной моделью параллелизма реального мира, но имеет простую семантику, и по этой причине именно с этой моделью мы будем работать дальше.

Начнем с модели перехода. Для детерминированного случая с одним агентом это функция RESULT(s, a), возвращающая состояние, которое возникнет в результате выполнения действия a, когда среда находится в состоянии s. В окружении с одним агентом может существовать b различных вариантов действия, и при этом значение b может быть довольно большим, особенно для представлений первого порядка со многими объектами, на которые можно воздействовать, но схемы действий, тем не менее, обеспечивают достаточно краткое представление.

В многосубъектном окружении с n субъектами-исполнителями единственное действие заменяется ▶**совместным действием** $\langle a_1,\ldots, a_n\rangle$, где a_i является действием, предпринятым i-м субъектом. И сразу же можно отметить две проблемы: во-первых, необходимо описать модель перехода для b^n различных совместных действий; во-вторых, это задача совместного планирования с коэффициентом ветвления b^n.

Поскольку субъекты-исполнители все вместе помещены в многосубъектную систему с огромным коэффициентом ветвления, основным направлением исследований в многосубъектном планировании было *разъединение* субъектов, насколько это возможно, — так, чтобы (в идеале) сложность задачи возрастала линейно в зависимости от n, а не экспоненциально в зависимости от b^n.

Если субъекты никак не взаимодействуют друг с другом, например имеется n субъектов, каждый из которых раскладывает пасьянс солитер, то достаточно просто решить n отдельных задач. Если субъекты-акторы ▶**слабо связаны**, можем ли мы достичь чего-то близкого к подобному экспоненциальному улучшению? Безусловно, это центральный вопрос во многих областях ИИ. Мы уже встречались с успешными методами решения для слабо связанных систем в контексте задач УО, где "древовидность" графа ограничений позволяла использовать эффективные способы решения (см. раздел 6.5.2), а также в контексте непересекающихся шаблонов баз данных (раздел 3.6.3) и аддитивных эвристик для планирования (раздел 11.4).

Стандартный подход к слабосвязанным задачам состоит в том, чтобы принять допущение, что эти задачи полностью разъединены, а затем соответствующим образом исправить взаимодействия. Для модели перехода это означает написание схемы действий так, как если бы субъекты действовали независимо друг от друга.

Давайте посмотрим, как этот подход сработает для парной игры в теннис. В этом случае у нас есть два агента-теннисиста, которые образуют команду с общей целью — выиграть матч против команды соперников. Давайте предположим, что в какой-то момент игры команда имеет целью отбить мяч, который был направлен противником на их половину поля, и убедиться, что по крайней мере один из них находится под сеткой и контролирует ее. На рис. 18.1 представлены начальные условия, цель и схема действий для этой задачи. Легко увидеть, что из начальных условий достичь цели можно с помощью ▶**совместного плана** из

двух этапов, определяющего, что каждый из игроков должен сделать: игрок A должен переместиться в правый сектор к задней линии и отбить мяч, в то время как игрок B должен просто оставаться на месте у сетки.

PLAN 1: A: $[Go(A, RightBaseline), Hit(A, Ball)]$
B: $[NoOp(B), NoOp(B)]$.

$Actors(A, B)$
$Init(At(A, LeftBaseline) \wedge At(B, RightNet) \wedge$
$\quad \wedge Approaching(Ball, RightBaseline) \wedge Partner(A, B) \wedge Partner(B, A)$
$Goal(Returned(Ball) \wedge (At(x, RightNet) \vee At(x, LeftNet))$
$Action(Hit(actor, Ball),$
$\quad$ PRECOND: $Approaching(Ball, loc) \wedge At(actor, loc)$
$\quad$ EFFECT: $Returned(Ball))$
$Action(Go(actor, to),$
$\quad$ PRECOND: $At(actor, loc) \wedge to \neq loc,$
$\quad$ EFFECT: $At(actor, to) \wedge \neg At(actor, loc))$

Рис. 18.1. Задача парной игры в теннис. Два игрока, A и B, играют в одной команде и могут находиться в одном из четырех мест: *LeftBaseline* (слева, у задней линии), *RightBaseline* (справа, у задней линии), *LeftNet* (слева, под сеткой) и *RightNet* (справа, под сеткой). Мяч может быть обит, только если игрок находится в нужном месте. Действие *NoOp* является фиктивным, простым заполнителем, не имеющим никакого эффекта. Обратите внимание, что каждое действие должно включать субъект в качестве аргумента

Проблемы возникают в том случае, когда план диктует, что оба агента бьют по мячу в одно и то же время. В реальном мире это не сработает, но для действия *Hit* схема действий говорит, что мяч будет успешно отбит и в этом случае. Сложность в данной ситуации состоит в том, что предусловия ограничивают состояния, в которых данное действие может быть успешно выполнено, но не ограничивают другие параллельные действия, которые могут помешать его выполнению.

Решить эту проблему можно путем расширения схемы действий одним новым свойством — ▶**ограничением параллельности действия** (*concurrent action constraint*) с указанием, какие действия должны или не должны выполняться одновременно. Например, действие *Hit* в этом случае можно описать следующим образом:

$Action(Hit(actor, Ball),$
$\quad$ CONCURRENT: $\forall\, b \;\; b \neq actor \;\Rightarrow\; \neg Hit(b, Ball)$
$\quad$ PRECOND: $Approaching(Ball, loc) \wedge At(actor, loc)$
$\quad$ EFFECT: $Returned(Ball))$.

Другими словами, действие *Hit* будет иметь свой объявленный эффект только в том случае, если нет другого действия *Hit*, выполняемого другим агентом в это же самое время. (В подходе SATPLAN эта ситуация обрабатывалась бы с помощью частной **аксиомы запрета действия**.) И наоборот, для некоторых действий желаемый эффект будет достигнут *только в том случае*, когда одновременно выполняется другое аналогичное действие. Например, чтобы принести на теннисный корт кулер с напитками, необходимы совместные действия двух агентов:

$$\begin{aligned}&Action(Carry(\text{actor}, cooler, here, there),\\ &\quad \text{CONCURRENT: } \exists\, b \;\; b \neq actor \wedge Carry(b, cooler, here, there)\\ &\quad \text{PRECOND: } At(actor, here) \wedge At(cooler, here) \wedge Cooler(cooler)\\ &\quad \text{EFFECT: } At(actor, there) \wedge At(cooler, there) \wedge \neg At(actor, here) \wedge\\ &\qquad \wedge \neg At(cooler, here)).\end{aligned}$$

С такими типами схем действий любой из алгоритмов планирования, описанных в главе 11, при незначительных модификациях может быть адаптирован так, чтобы генерировать многосубъектные планы. В той степени, в которой связь между подпланами является слабой — это означает, что ограничения параллелизма редко вступают в силу в процессе построения плана, — можно ожидать, что различные эвристики, полученные для задач планирования с одним агентом, также будут эффективными и в многосубъектном контексте.

18.1.4. Планирование при нескольких агентах: кооперация и координация

Теперь давайте обратимся к истинному мультиагентному окружению, в котором каждый агент выполняет собственный план. Для начала предположим, что цели и база знаний у всех агентов являются общими. Можно решить, что такое предположение сводит ситуацию к многосубъектному случаю — каждый агент просто вычисляет общее совместное решение и выполняет свою часть этого решения. Увы, слово "общее" в определении совместного решения вводит в заблуждение. Проблема в том, что может существовать более одного плана, ведущего к достижению одной и той же цели. Например, вот второй план, который также достигает цели в приведенной выше задаче о командной игре в теннис:

$$\begin{aligned}&\text{PLAN 2: } A\text{: } [Go(A, LeftNet), NoOp(A)]\\ &\qquad\quad\;\; B\text{: } [Go(B, RightBaseline), Hit(B, Ball)].\end{aligned}$$

Если два агента смогут договориться о выполнении либо плана 1, либо плана 2, общая цель будет достигнута. Но если агент A выберет план 2, а агент B — план 1, то никто из них не отобьет мяч. И наоборот, если агент A выберет план 1, а агент B — план 2, то они, вероятно, столкнутся друг с другом и ни один из них не сможет отбить мяч. Агенты знают это, но как они смогут координировать свои действия, чтобы убедиться, что эти действия соответствуют общему плану?

Одним из вариантов является принятие некоторого ▶**соглашения** (*convention*) еще до начала совместной деятельности. Соглашение — это любое ограничение на выбор совместных планов. Например, соглашение "придерживаться своей стороны корта" исключает план 1, в результате чего оба партнера выбирают план 2. Перед водителями на всех дорогах стоит задача не сталкиваться друг с другом, — она (частично) решается принятием соглашения "придерживаться правой стороны дороги" в большинстве стран мира. Альтернативный вариант соглашения, "придерживаться левой стороны", действует одинаково хорошо до тех пор, пока все агенты, действующие в этой среде, его соблюдают. Аналогичные соображения применимы к развитию человеческого языка, где важно не то, на каком именно языке должен говорить каждый человек, а то, что все сообщество говорит на одном языке. Когда подобные соглашения общеприняты и широко распространены, их называют ▶**социальными законами**.

При отсутствии соглашения агенты могут использовать **общение** (*communication*) для получения общих знаний о выполняемом совместном плане. Например, теннисист может крикнуть "Мой!" или "Твой!" для указания на предпочтительный совместный план. Общение не обязательно предполагает словесный обмен. Например, один игрок может неявно сообщить о предпочтительном совместном плане другому, просто выполнив его первую часть. Если агент A направился к сетке, то агент B обязан отойти назад, к задней линии, чтобы отбить мяч, поскольку план 2 является единственным совместным планом, который начинается с того, что агент A направляется к сетке. Такой подход к координированию действия, иногда называемый ▶**распознаванием плана** (*plan recognition*), будет применим, если для безошибочного определения нужного совместного плана достаточно одного действия (или краткой последовательности действий).

18.2. Теория некооперативных игр

Теперь пришло время познакомиться с ключевыми концепциями и аналитическими методами теории игр — теории, лежащей в основе принципов принятия решений в мультиагентных средах. Мы начнем обсуждение с теории некооперативных игр.

18.2.1. Игры с единственным ходом: нормальная форма игры

Первая модель игры, которую мы рассмотрим, будет такой, в которой все игроки действуют одновременно, а результат игры определяется, исходя из общего результата действий, выбранных подобным образом. (В действительности совсем не важно, чтобы действия игроков выполнялись в одно и то же время, — главное, что имеет значение, это чтобы ни один из игроков до или во время выполнения действия не имел никаких сведений о выборе других игроков.) Подобные игры

называют ▶**играми в нормальной форме**. Игра в нормальной форме определяется с помощью трех компонентов.

- ▶**Игроки**, или агенты, которые должны принимать решения. Наибольшее внимание в исследованиях уделялось играм с двумя игроками, хотя достаточно часто рассматриваются также игры с n игроками, где $n > 2$. Имена игроков мы будем записывать с прописной буквы, например *Ali* и *Bo* или *O* и *E*.
- **Действия**, которые могут быть выбраны игроками. В этой главе названия действий записываются строчными буквами, например *one* или *testify*. В распоряжении игроков могут находиться одинаковые или неодинаковые множества действий.
- **Функция вознаграждений**, определяющая для каждого игрока полезность каждой возможной комбинации действий всех игроков. Для игр с двумя игроками функция вознаграждений для игрока может быть представлена матрицей, в которой имеется строка для каждого возможного действия одного игрока, и столбец — для каждого возможного выбора другого игрока. Пересечение выбранной строки и выбранного столбца определяет ячейку матрицы, в которой указано вознаграждение соответствующего игрока. В случае с двумя игроками принято объединять две матрицы в одну ▶**матрицу выплат**, в которой каждая ячейка содержит сведения о выплатах обоим игрокам.

Чтобы проиллюстрировать эти идеи, давайте рассмотрим пример игры в **чет-нечет на двух пальцах** (эту игру на пальцах называют также *morra*, от итальянского слова *camorra* — группа). В этой игре участвуют два игрока, *O* (от *odd* — нечетный) и *E* (от *even* — четный), которые одновременно показывают один или два пальца. Обозначим общее количество показанных пальцев как f. Если число f является нечетным, игрок *O* получает f долл. от игрока *E*, а если число f — четное, то игрок *E* получает f долл. от игрока *O*.[1] Матрица выплат для игры в чет-нечет на двух пальцах выглядит следующим образом.

	O: один	*O: два*
E: один	$E = +2, O = -2$	$E = -3, O = +3$
E: два	$E = -3, O = +3$	$E = +4, O = -4$

Говорят, что *E* — ▶**игрок строки**, а *O* — ▶**игрок столбца**. Так, например, в нижнем правом углу показано, что если игрок *O* выбирает действие *два*, а игрок E также выбирает действие *два*, то вознаграждение для *E* равно 4, а для *O* равно –4.

[1] Эта игра является развлекательной версией инспекционных игр. В таких играх инспектор выбирает день осмотра объекта (например, ресторана или завода биологического оружия), а управляющий объекта выбирает день, чтобы скрыть все, что инспектор не должен видеть. Инспектор выигрывает, если выбранные дни оказываются разными, а управляющий объекта выигрывает, если они совпадают.

Прежде чем анализировать игру в чет-нечет на двух пальцах, стоит рассмотреть вопрос о том, зачем нам вообще нужны идеи теории игр: почему мы не можем решить задачу, стоящую (скажем) перед игроком E, воспользовавшись аппаратом теории принятия решений и методом максимизации полезности, которые уже неоднократно использовались в этой книге? Чтобы понять, почему, нужно еще кое-что: давайте предположим, что E попытается найти лучшее действие для выполнения. Альтернативные варианты — действия *один* или *два*. Если игрок E выберет действие *один*, то выплата будет либо +2, либо –3. Однако какой именно выигрыш E получит *на самом деле*, будет зависеть от выбора, сделанного игроком O: самое большее, что может сделать E как игрок строки, — это своим выбором определить расположение исхода игры в той или иной строке. Точно так же игрок O выбирает только столбец.

Чтобы сделать оптимальный выбор между имеющимися возможностями, игрок E должен учесть, как игрок О будет действовать в качестве рационального агента, принимающего решения. Но игрок О, в свою очередь, должен учитывать тот факт, что Е также является рациональным агентом, принимающим решения. Таким образом, принятие решений в мультиагентном окружении весьма отличается по характеру от принятия решений в окружении с одним агентом, поскольку игрокам необходимо принимать во внимание рассуждения друг друга. Роль ►**концепций решения** в теории игр состоит в том, чтобы попытаться сделать этот вид рассуждений точным.

В теории игр термин *strategy* используется для обозначения того, что ранее в этой книге называлось *policy*, т.е. ►**стратегия**. ►**Чистая стратегия** — это детерминированная стратегия; для игр, состоящих из одного хода, чистая стратегия состоит всего лишь из одного действия. Как будет показано ниже, для многих игр агент может получить лучший результат, используя ►**смешанную стратегию**, которая представляет собой рандомизированную стратегию, предусматривающую выбор конкретных действий среди доступных в соответствии с некоторым распределением вероятностей. Смешанная стратегия, в которой действие a выбирается с вероятностью p, а в противном случае выбирается действие b, условно обозначается как $[p: a; (1-p): b]$. Например, смешанной стратегией для игры в чет-нечет на двух пальцах может быть [0,5: *один*; 0,5: *два*]. ►**Профилем стратегии** называется вариант присваивания стратегии каждому игроку. После того как задан профиль стратегии, **результат** игры для каждого игрока принимает определенное числовое значение, а если игроки используют смешанные стратегии, то следует использовать ожидаемую полезность.

Итак, как следует агенту решить действовать в таких играх, как чет-нечет на двух пальцах? Теория игр предоставляет ряд концепций решения, в которых предпринимаются попытки определить рациональное действие с учетом убеждений агента об убеждениях другого агента. К сожалению, не существует единой концепции идеального решения: сложно определить, что может означать "рациональный", когда каждый агент выбирает только часть профиля стратегии, определяющего общий результат.

Первая концепция решения будет представлена с помощью, вероятно, самой известной игры в канонической теории игр — ▶**"Дилемма заключенного"**. Эта игра построена на следующей истории: два отпетых грабителя, Али и Бо, были пойманы с поличным недалеко от места совершенного ими ограбления, и теперь следователи допрашивают их по отдельности. Каждому из них прокурор предлагает сделку: если он даст свидетельство против своего напарника, указав на него как на главаря шайки грабителей, то его освободят за сотрудничество, а напарника осудят на 10 лет тюрьмы. Однако если они оба дадут показания друг против друга, то оба получат по 5 лет тюрьмы. Али и Бо также знают, что если они оба просто откажутся давать показания, то получат только по одному году заключения каждый за менее грубое правонарушение — владение краденым имуществом. В результате Али и Бо сталкиваются с так называемой дилеммой заключенного: должны ли они свидетельствовать (*testify*) друг против друга или лучше отказаться (*refuse*) давать показания? Будучи рациональными агентами, Али и Бо хотят максимизировать свою ожидаемую полезность, что означает минимизацию количества лет, которые они проведут в тюрьме, — при этом каждый из них безразличен к благополучию другого игрока. Дилемма заключенного в этих условиях описывается следующей матрицей выплат.

	Али: *testify*	*Али*: *refuse*
Бо: *testify*	$A=-5, B=-5$	$A=-10, B=0$
Бо: *refuse*	$A=0, B=-10$	$A=-1, B=-1$

Теперь поставим себя на место Али. Он может проанализировать матрицу выплат следующим образом.

- Предположим, что Бо даст показания — сыграет *testify*. Тогда я получу 5 лет, если буду свидетельствовать против него, и 10 лет — если откажусь, поэтому в данном случае лучше свидетельствовать против него.
- С другой стороны, если Бо отказывается — играет *refuse*, я буду освобожден, если дам показания против него, и получу 1 год, если откажусь, так что свидетельствовать против него лучше и в этом случае.
- Поэтому *независимо от того, что решит делать Бо*, для меня будет лучше дать показания.

Али обнаружил, что свидетельство (*testify*) является ▶**доминирующей стратегией** в этой игре. Говорят, что стратегия s для игрока p ▶**строго доминирует** над стратегией s', если результат стратегии s для игрока p лучше, чем результат стратегии s' при любом выборе стратегии другим игроком (игроками). Стратегия s ▶**слабо доминирует** над стратегией s', если s лучше, чем s', по крайней мере в одном профиле стратегии, и не хуже — в любом другом. Доминирующей стратегией является такая стратегия, которая доминирует над всеми остальными. В теории игр обычным является предположение, что ➡*рациональный игрок всегда*

будет выбирать доминирующую стратегию и избегать доминируемой стратегии. Будучи рациональным — по крайней мере, не желая, чтобы его считали иррациональным, — Али выбирает доминирующую стратегию.

Нетрудно заметить, что рассуждения Бо будут идентичными: он также сделает вывод, что свидетельство против Али будет для него доминирующей стратегией, и примет решение играть *testify*. Результатом игры, в соответствии с анализом доминирующих стратегий, будет то, что оба игрока выберут свидетельство и как следствие получат по 5 лет тюрьмы.

В ситуации, подобной этой, когда все игроки выбирают доминирующую стратегию, результат игры называют ▶**равновесием доминирующих стратегий**. Термин "равновесие" выбран потому, что ни у одного игрока нет никакого стимула уклоняться от своей части выигрыша: по определению, если они так сделали, значит, они не могли добиться большего, но могли получить худший результат. В этом смысле равновесие доминирующих стратегий является очень сильной концепцией решения.

Возвращаясь к дилемме заключенного, мы видим, что *дилемма*, собственно, состоит в том, что исход равновесия доминирующих стратегий, когда оба игрока дают показания друг против друга, хуже для обоих игроков, чем результат, который бы они получили, если бы оба отказались давать показания. При выборе стратегии (*refuse*, *refuse*) каждый из игроков в результате получает всего по одному году тюрьмы, что лучше для них *обоих*, чем 5 лет, которые каждый получает при выборе равновесия доминирующих стратегий.

Существует ли какой-либо способ, с помощью которого Али и Бо могли бы формально прийти к результату (*refuse*, *refuse*)? Безусловно, для них обоих это *допустимый* вариант — отказаться от дачи показаний, но очень трудно представить, как рациональные агенты могли бы сделать такой выбор, учитывая то, как данная игра настроена. Не забывайте, что это некооперативная игра: игрокам запрещено разговаривать друг с другом, поэтому они не могут заключить обоюдное соглашение о выборе варианта отказа от показаний.

Однако все же можно было бы добраться до решения (*refuse*, *refuse*), если изменить условия игры. Можно превратить ее в кооперативную игру, когда агентам разрешается заключать обоюдные соглашения. Или же можно превратить ее в **повторяющуюся игру** (*repeated game*), когда игроки знают, что будут встречаться вновь; ниже будет показано, как это работает. Как альтернатива у игроков могут быть моральные убеждения, поощряющие сотрудничество и справедливость. Но это означало бы, что у них разные функции полезности, а значит, они и в этом случае будут играть уже в другую игру.

Наличие у определенного игрока доминирующей стратегии существенно упрощает для него процесс принятия решений. Как только Али понял, что дача свидетельских показаний является доминирующей стратегией, ему уже требуется прикладывать какие-либо усилия, чтобы попытаться выяснить, что будет делать Бо, — просто потому, что он знает, что *независимо от того, что сделает Бо*, дача показаний будет его ▶**лучшим ответом**. Тем не менее большинство игр не имеют

ни доминирующих стратегий, ни равновесий доминирующих стратегий. Очень редко единственная стратегия является лучшим ответом на все возможные стратегии партнеров.

Следующая концепция решения, которую мы рассмотрим, слабее, чем равновесие доминирующих стратегий, но она нашла гораздо более широкое применение. Она называется ▶**равновесием Нэша** и получила свое название в честь Джона Форбса Нэша-младшего (1928–2015), который представил доказательство соответствующей теоремы в своей докторской диссертации в 1950 году, — работа, за которую он был удостоен Нобелевской премии в 1994 году.

Профиль стратегии представляет собой равновесие Нэша, если ни один игрок не может в одностороннем порядке изменить свою стратегию и в результате получить более высокую выплату при условии, что все другие игроки сохранили свои стратегии выборов. Следовательно, при равновесии Нэша каждый из игроков одновременно выбирает лучший ответ на выбор всех своих противников. Равновесие Нэша представляет собой стабильную точку в игре: стабильную в том смысле, что для любого игрока не существует рационального стимула отклониться от него. Однако равновесие Нэша представляет *локальную* стабильную точку: как будет показано ниже, игра может включать несколько равновесий Нэша.

Поскольку доминирующая стратегия является наилучшим ответом на *все* стратегии противников, очевидно, что любое равновесие доминирующих стратегий также должно быть равновесием Нэша (упражнение 18.1). Следовательно, в дилемме заключенного существует уникальное равновесие доминирующих стратегий, которое также является уникальным равновесием Нэша.

Следующий пример игры демонстрирует, во-первых, что иногда игры могут не иметь ни одной доминирующей стратегии, а во-вторых, что некоторые игры имеют несколько равновесий Нэша.

	Али: l	*Али*: r
Бо: t	$A = 10, B = 10$	$A = 0, B = 0$
Бо: b	$A = 0, B = 0$	$A = 1, B = 1$

Легко удостовериться, что в этой игре нет доминирующих стратегий для каждого из игроков и, следовательно, нет равновесия доминирующих стратегий. Однако профили стратегии (t, l) и (b, r) являются равновесиями Нэша. Вполне очевидно, что в интересах обоих агентов стремиться к *одному и тому же* равновесию Нэша — либо (t, l), либо (b, r), но, поскольку речь идет о проблемной области *некооперативной* теории игр, игроки должны делать свой выбор независимо, не имея каких-либо знаний о выборе других игроков, и без какого-либо способа заключения с ними соглашений. Это пример **проблемы координации**: игроки хотят координировать свои действия глобально, чтобы все они выбирали действия, ведущие к одному и тому же равновесию Нэша, но должны делать это, используя только локальное принятие решений.

Был предложен ряд подходов к решению проблем координации. Одна идея заключается в понятии ▶ **узловых точек**. Узловая точка в игре — это результат, который некоторым образом выделяется для игроков как "очевидный" результат, с помощью которого возможно скоординировать их выборы. Это, конечно же, не точное определение — то, что это означает, будет зависеть от конкретной игры. В приведенном выше примере есть, кстати, одна узловая точка: результат выбора (t, l) даст обоим игрокам существенно более высокую полезность, чем та, которую они получат, если скоординируют свой выбор на (b, r). С точки зрения теории игр оба результата является равновесиями Нэша, но тот, кто предположит координацию действий на варианте (b, r), должен быть действительно неадекватным игроком.

В некоторых играх нет равновесий Нэша в чистых стратегиях, как в приведенной ниже игре под названием ▶ **"Соответствие пенни"** (*matching pennies*). В этой игре Али и Бо одновременно выбирают одну из сторон монеты, орла или решку. Если они делают один и тот же выбор, то Бо дает Али 1 долл., а если их выбор был разным, то Али отдает Бо 1 долл.

	Али: орел	*Ali*: решка
Бо: орел	$A = 1, B = -1$	$A = -1, B = 1$
Бо: решка	$A = -1, B = 1$	$A = 1, B = -1$

Мы предлагаем читателю самостоятельно убедиться в том, что эта игра не имеет доминирующих стратегий и что ни один из результатов не является равновесием Нэша при чистых стратегиях. При любом исходе один из игроков сожалеет о своем выборе и предпочел бы поступить наоборот при данном выборе другого игрока.

При поиске равновесия Нэша вся хитрость состоит в использовании смешанных стратегий, т.е. надо позволить игрокам внести в их выборы элемент случайности. Нэш доказал, что ◀ *в каждой игре есть хотя бы одно равновесие Нэша при смешанных стратегиях*. Это объясняет, почему равновесие Нэша является такой важной концепцией решения: для других концепций решения, таких как равновесие доминирующих стратегий, нет гарантий, что они имеют место в каждой игре, но всегда можно получить решение, если обратиться к равновесиям Нэша при смешанных стратегиях.

Для игры "Соответствие пенни" равновесие Нэша в смешанных стратегиях будет получено, если оба игрока выбирают орла или решку с равной вероятностью. Чтобы понять, что этот результат действительно является равновесием Нэша, предположим, что один из игроков выбирает результат с вероятностью, отличной от 0,5. Тогда другой игрок сможет использовать этот факт, вкладывая весь свой вес в конкретную стратегию. Например, предположим, что Бо выбирает орла с вероятностью 0,6 (и, соответственно, решку — с вероятностью 0,4). Тогда Али достигнет лучших результатов, если также будет постоянно выбирать орла. Легко увидеть, что в подобном случае Бо, выбирающий орла с вероятностью 0,6, не сможет сформировать часть какого-либо равновесия Нэша.

18.2.2. Общественное благо

В теории игр основная точка зрения состоит в том, что в процессе игры игроки всегда пытаются получить для себя наилучшие результаты, насколько это возможно. Однако иногда будет поучительно придерживаться иной точки зрения. Предположим, что вы — доброжелательный, всеведающий субъект, смотрящий на игру сверху и способный *выбрать* ее исход. Будучи доброжелательным, вы хотите выбрать лучший всеобщий результат — такой, который был бы лучше для *общества в целом*, если можно так сказать. Как его следует выбирать? Какие критерии в этом случае можно применить? Это как раз то, что охватывается понятием ▶**общественного блага** (*social welfare*).

Вероятно, наиболее важным и наименее спорным критерием общественного блага является то, что вам следует избегать результатов, которые *бесполезно расходуют* полезность. Это требование отражено в концепции ▶**эффективности по Парето** (*Pareto optimality*), получившей это название в честь итальянского экономиста Вильфредо Парето (1848–1923). Результат является эффективным по Парето, если нет другого результата, который улучшит положение одного игрока, не ухудшая положения кого-то другого. Если вы выбираете результат, который не является эффективным по Парето, то вы теряете полезность в том смысле, что могли бы дать больше полезности по крайней мере одному агенту, не затрагивая при этом интересов ни одного из других агентов.

▶**Утилитарное общественное благо** является показателем того, насколько хорош полученный результат в общем целом. Утилитарное общественное благо для результата — это просто сумма полезностей, предоставленных игрокам по этому результату. Однако в отношении утилитарного общественного блага есть два ключевых затруднения. Во-первых, в этом показателе учитывается сумма, а не *распределение* полезностей между игроками, так что может иметь место очень неравное распределение, если это будет способствовать максимизации общей суммы. Во-вторых, серьезное затруднение состоит в том, что в этом показателе предполагается использование *общей шкалы* для всех полезностей. Однако многие экономисты утверждают, что такой подход недопустим, потому что полезность (маловероятные деньги) является весьма субъективной величиной. Если требуется решить, как поделить пачку печенья, следует ли отдать ее всю целиком монстру полезности, который заявил, что любит печенье в тысячу раз больше, чем кто-либо другой? Это вполне может максимизировать общую полезность согласно ее самооценке, но вовсе не кажется правильным.

Вопрос о том, как полезность будет распределена среди игроков, рассматривается в исследованиях по ▶**эгалитарному общественному благу**. Например, в одном предложении предполагается, что следует максимизировать ожидаемую полезность для наиболее обездоленного члена общества — максиминный подход. Возможны и другие метрики, в том числе ▶**коэффициент Джини**, который подводит итог тому, насколько равномерно полезность будет распределена среди

игроков. Главное затруднение в таких предложениях связано с тем, что они могут пожертвовать большим количеством общего благосостояния ради небольших выигрышей в распределении и, как простой утилитаризм, все еще оставаться во власти монстра полезности.

Применение этих концепций к игре "Дилемма заключенного", обсуждавшейся выше, объясняет, почему она называется дилеммой. Напомним, что в этой игре профиль стратегии (*testify*, *testify*) является равновесием доминирующих стратегий и единственным равновесием Нэша. Тем не менее это также единственный результат, который не является эффективным по Парето. Результат профиля стратегии (*refuse*, *refuse*) максимизирует как утилитарное, так и эгалитарное общественное благо. Таким образом, дилемма в игре "Дилемма заключенного" возникает потому, что очень сильная концепция решения (равновесие доминантных стратегий) приводит к результату, который, по сути, не проходит любую проверку на то, что считается разумным результатом с точки зрения "общества". Тем не менее для отдельных игроков в ней нет очевидных способов достичь лучшего решения.

Вычисление равновесий

Теперь давайте рассмотрим ключевые вычислительные вопросы, связанные с обсуждавшимися выше концепциями. Сначала обратимся к чистым стратегиям, в которых отсутствуют случайные элементы.

Если игрокам доступно лишь конечное число возможных вариантов выбора, то для нахождения равновесий можно использовать исчерпывающий поиск: итерации по каждому возможному профилю стратегии с проверкой, получит ли какой-либо игрок благоприятное отклонение в этом профиле. Если нет, то это равновесие Нэша в чистых стратегиях. Доминирующие стратегии и равновесия доминирующих стратегий можно вычислить с помощью аналогичных алгоритмов. К сожалению, число возможных профилей стратегий для n игроков, каждому из которых доступно m возможных действий, определяется как m^n, т.е. нереально большое для исчерпывающего поиска.

Альтернативным подходом, который хорошо работает в некоторых играх, является ►**близорукий наилучший ответ** (также известный как **последовательный наилучший ответ**): начните с выбора случайного профиля стратегии, а затем, если какой-то игрок не будет использовать оптимальный выбор при заданных выборах других, переверните их выбор к оптимальному и повторите процедуру. Этот процесс будет сходиться, если он приводит к профилю стратегии, в котором каждый игрок делает оптимальный выбор при заданных выборах других игроков, т.е. к равновесию Нэша. Для некоторых игр алгоритм близорукого наилучшего ответа не сходится, но для других важных классов игр он сходится гарантированно.

Вычисление равновесий смешанной стратегии алгоритмически намного сложнее. Для простоты мы сосредоточимся на соответствующих методах для игр с

нулевой суммой и лишь кратко прокомментируем возможности их распространения на другие игры в конце этого раздела.

В 1928 году фон Нейман разработал метод поиска *оптимальной* смешанной стратегии для игр с двумя игроками, называемых ▶**играми с нулевой суммой**, — игр, в которых общая сумма выигрышей всегда равна нулю (или некоторой константе, как объяснялось в разделе 5.1.1). Очевидно, что игра в чет-нечет является именно такой. В играх с двумя игроками и нулевой суммой вознаграждения всегда являются равными и противоположными, поэтому достаточно рассмотреть вознаграждения только для одного игрока, который будет считаться максимизирующим (так же, как и в главе 5). Применительно к игре в чет-нечет примем в качестве максимизирующего игрока E, выбравшего себе в качестве выигрышного результата четное количество пальцев, поэтому можно определить матрицу вознаграждений на основе значений $U_E(e, o)$ — вознаграждения, получаемого игроком E, если игрок E выбирает действие e, а O — действие o. Метод фон Неймана называется методом ▶**максимина** и действует, как описано ниже.

- Предположим, что правила игры были изменены таким образом, что игрок E теперь вынужден раскрывать свою стратегию первым, а затем игрок O выбирает свою стратегию, зная стратегию игрока Е. Далее оцениваются ожидаемые результаты игры на основе выбранных стратегий. В результате мы получаем игру, в которой ходы выполняются поочередно и к которой можно применить стандартный алгоритм **минимакс** из главы 5. Допустим, что игра приводит к получению результата $U_{E,O}$. Очевидно, что эта игра дает преимущество игроку O, поэтому истинная полезность U данной игры (с точки зрения игрока E) равна *по меньшей мере* $U_{E,O}$. Например, если рассматриваются только чистые стратегии, то минимаксное дерево игры имеет корневое значение, равное –3 (рис. 18.2, *а*), поэтому мы знаем, что $U \geq -3$.
- Теперь предположим, что правила были изменены таким образом, что первым свою стратегию вынужден раскрывать игрок O, а за ним следует игрок E. В таком случае минимаксное значение этой игры становится равным $U_{O,E}$, а поскольку игра складывается в пользу игрока Е, то известно, что полезность U *самое большее* равна $U_{O,E}$. При использовании чистых стратегий это значение +2 (см. рис. 18.2, *б*), поэтому известно, что $U \leq +2$.

Рассматривая эти два предположения совместно, можно прийти к заключению, что истинная полезность U решения исходной игры должна удовлетворять следующему неравенству:

$$U_{E,O} \leq U \leq U_{O,E} \quad \text{или в нашем случае} \quad -3 \leq U \leq 2.$$

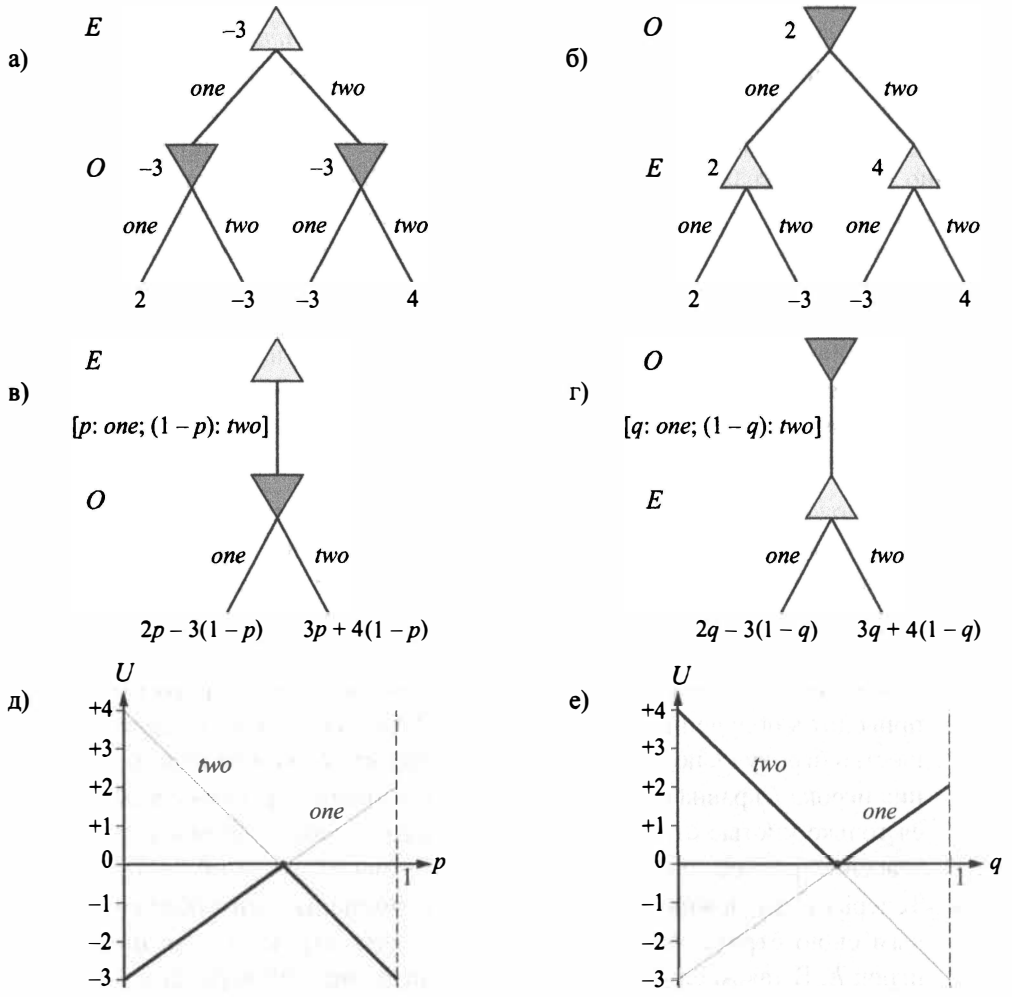


Рис. 18.2. а) и б) Минимаксные деревья игры в чет-нечет на двух пальцах, если игроки ходят по очереди, ведя игру на основе чистых стратегий. **в) и г)** Параметризованные деревья игры, в которой первый игрок использует смешанную стратегию. Выплаты зависят от параметра вероятности (p или q) в смешанной стратегии. **д) и е)** Для любого конкретного значения параметра вероятности второй игрок будет выбирать "наилучшее" из двух действий, поэтому значения для смешанной стратегии первого игрока задаются утолщенными линиями. Первый игрок выбирает параметр вероятности для смешанной стратегии в точке их пересечения

Чтобы точно определить значение U, необходимо перейти к анализу смешанных стратегий. Вначале отметим следующее: ➡*как только первый игрок раскроет свою стратегию, второй игрок может также выбрать чистую стратегию.* Причина этого проста — если второй игрок использует смешанную стратегию, $[p: one;(1-p): two]$, то ожидаемая полезность этой игры представляет собой линейную комбинацию $(p \cdot U_{one} + (1-p) \cdot U_{two})$ полезностей чистых стратегий U_{one} и U_{two}. Эта линейная комбинация ни при каких условиях не будет лучше по сравнению с лучшим из значений U_{one} и U_{two}, поэтому второй игрок может просто выбрать лучшее.

С учетом этого замечания минимаксные деревья можно рассматривать как имеющие бесконечное количество ветвей, исходящих от корня, соответствующих бесконечному количеству смешанных стратегий, доступных для выбора первым игроком. Каждая из этих ветвей ведет к узлу с двумя ветвями, соответствующими чистым стратегиям для второго игрока. Эти бесконечные деревья можно изобразить как конечные, предусмотрев один "параметризованный" выбор у корня, как описано ниже.

- Ситуация, когда игрок E ходит первым, показана на рис. 18.2, *в*. Игрок E делает из корневой позиции ход $[p: one; (1-p): two]$, а затем игрок O выбирает чистую стратегию (и, следовательно, ход) с учетом значения p. Если игрок O выбирает ход *one*, то ожидаемое вознаграждение (для E) становится равным $2p - 3(1-p) = 5p - 3$; если игрок O выбирает ход *two*, то ожидаемое вознаграждение равно $-3p + 4(1-p) = 4 - 7p$. Зависимости, выражающие величину этих двух вознаграждений, можно изобразить в виде прямых линий на графике, где p изменяется от 0 до 1 вдоль оси x, как показано на рис. 18.2, *д*. Игрок O, минимизирующий стоимость игры, должен всегда выбирать наименьшее значение на двух прямых линиях, как показано на этом рисунке утолщенными отрезками прямых. Поэтому наилучшее решение, которое может принять игрок E, выбирая ход из корневой позиции, состоит в том, чтобы выбрать значение p, соответствующее точке пересечения и определяемое следующим образом:

$$5p - 3 = 4 - 7\text{p} \quad \Rightarrow \quad p = 7/12.$$

  Полезность для игрока E в этой точке равна $U_{E,O} = -1/12$.
- Если первым ходит игрок O, то складывается ситуация, показанная на рис. 18.2, *г*. Игрок O в корневой позиции выбирает стратегию $[q: one; (1-q): two]$, после чего игрок E выбирает ход с учетом значения q. При этом выплаты определяются соотношениями $2q - 3(1-q) = 5q - 3$ и $-3q + 4(1-q) = 4 - 7q$.[2] И вновь, на рис. 18.2, *е* показано, что наилучший ход,

[2] Это всего лишь совпадение, что эти уравнения такие же, как для p. Это совпадение возникает потому, что $U_E(one, two) = U_E(two, one) = -3$. Это также объясняет, почему оптимальная стратегия одинакова для обоих игроков.

который может быть сделан игроком O из корневой позиции, заключается в выборе точки пересечения:

$$5q - 3 = 4 - 7q \quad \Rightarrow \quad q = 7/12.$$

Полезность для игрока E в этой точке равна $U_{O,E} = -1/12$.

Теперь мы знаем, что истинная полезность оригинальной игры находится в пределах от –1/12 до –1/12, т.е. равна точно –1/12! (Общий вывод состоит в том, что в эту игру лучше играть от имени игрока O, а не E.) Кроме того, истинная полезность достигается при использовании смешанной стратегии [7/12: *one*; 5/12: *two*], которой должны придерживаться оба игрока. Такая стратегия называется ▶**максиминным равновесием игры** и является равновесием Нэша. Обратите внимание, что каждая составляющая стратегия в равновесной смешанной стратегии имеет одну и ту же ожидаемую полезность. В данном случае и ход *one*, и ход *two* имеют ту же ожидаемую полезность –1/12, что и сама смешанная стратегия.

Приведенный выше результат для игры в чет-нечет на двух пальцах представляет собой пример общего результата, полученного фон Нейманом: ▶*каждая игра с нулевой суммой с двумя игроками имеет максиминное равновесие, если разрешены смешанные стратегии*. Более того, каждое равновесие Нэша в игре с нулевой суммой является максиминным равновесием для обоих игроков. Игрок, принимающий максиминную стратегию, получает две гарантии: во-первых, ни одна другая стратегия не может оказаться лучше против соперника, который играет хорошо (хотя некоторые другие стратегии могли бы быть лучше, если правильно использовать промахи противника, который делает иррациональные ошибки). Во-вторых, игрок продолжает делать то же самое, даже если его стратегия будет раскрыта противнику.

Общий алгоритм поиска максиминных равновесий в играх с нулевой суммой несколько сложнее по сравнению с тем, что схематично показано на рис. 18.2, *д* и *е*. Если количество возможных действий равно n, то смешанная стратегия представляет собой точку в n-мерном пространстве, а прямые линии становятся гиперплоскостями. Возможно также, что над некоторыми чистыми стратегиями для второго игрока будут доминировать другие стратегии, так что они перестанут быть оптимальными по отношению к *любой* стратегии для первого игрока. После удаления всех подобных стратегий (а эту операцию может потребоваться выполнить неоднократно) оптимальным вариантом хода из корневой позиции становится самая высокая (или самая низкая) точка пересечения оставшихся гиперплоскостей.

Поиск этого варианта представляет собой пример задачи **линейного программирования** — максимизация целевой функции с учетом линейных ограничений. Такие задачи могут быть решены с помощью стандартных методов за время, полиномиально зависящее от количества действий (а также от количества битов, используемых для определения функции вознаграждения, если углубиться в технические подробности).

Остается нерешенным следующий вопрос: как фактически должен действовать рациональный агент при ведении такой одноходовой игры, как игра в чет-нечет? Рациональный агент логическим путем выявил тот факт, что [7/12: *one*; 5/12: *two*] представляет собой максиминную равновесную стратегию и исходит из предположения, что знаниями об этом обладает и его рациональный противник. Агент может использовать игральную кость с 12 сторонами или генератор случайных чисел для выбора случайным образом хода, соответствующего этой смешанной стратегии, и в этом случае ожидаемое вознаграждение для игрока E будет равно $-1/12$. Или агент может просто решить сделать ход *one* или *two*. В любом случае для игрока E ожидаемое вознаграждение остается тем же $-1/12$. Любопытно то, что односторонний выбор конкретного действия не уменьшает ожидаемое вознаграждение для данного игрока, но если позволить другому агенту узнать, что этот игрок принял такое одностороннее решение, то ожидаемое вознаграждение *изменится*, поскольку противник сможет откорректировать свою стратегию соответствующим образом.

Найти равновесия в играх с ненулевой суммой несколько сложнее. Общий подход включает два этапа. На первом этапе необходимо перечислить все возможные подмножества из действий, которые могут образовывать смешанные стратегии. Например, сначала проверьте все профили стратегий, в которых каждый игрок выполняет одно действие, затем — те, в которых каждый игрок выполняет либо одно, либо два действия, и т.д. Количество таких проверок экспоненциально зависит от количества действий, поэтому может применяться только в относительно простых играх. На втором этапе для каждого профиля стратегий, включенного в список на первом этапе, необходимо провести проверку с целью определения, представляет ли он некоторое равновесие. Такая задача выполняется путем решения ряда уравнений и неравенств, аналогичных используемым в случае с нулевой суммой. В игре с двумя игроками эти уравнения являются линейными и могут быть решены с помощью основных методов линейного программирования, но в случае трех или более игроков они являются нелинейными и задача поиска их решения может оказаться очень сложной.

18.2.3. Повторяющиеся игры

До сих пор рассматривались только игры, состоящие из одного хода. Простейшей разновидностью игры, состоящей из нескольких ходов, является ▶**повторяющаяся игра**, в которой игроки снова и снова играют раунды из одноходовой игры, называемой ▶**игрой этапа**. Профиль стратегий для повторяющейся игры определяет выбор действия для каждого игрока на каждом временном интервале для всех возможных историй предыдущих выборов игроков.

Сначала давайте рассмотрим случай, когда игра этапа повторяется фиксированное, конечное и взаимно известное количество раундов — все эти условия необходимы для выполнения последующего анализа. Давайте предположим, что Али

и Бо предложено играть в повторяющуюся версию дилеммы заключенного и что оба они знают, что им придется провести ровно 100 раундов этой игры. В каждом раунде их будут спрашивать, какое действие они выбирают — *testify* или *refuse*, а затем они будут получать вознаграждение за этот раунд в соответствии с правилами игры, которые были приведены выше.

После завершения 100 раундов определяется общая выплата для каждого игрока — как сумма выплат этому игроку в каждом из 100 раундов. Какие стратегии должны выбрать Али и Бо, играя в эту игру? Рассмотрим следующий аргумент. Оба игрока знают, что 100-й раунд не будет повторяющейся игрой, т.е. что ее результат не сможет оказать влияния на будущие раунды. Таким образом, в сотом раунде они фактически играют в обычную игру "Дилемма заключенного".

Тогда, как было показано выше, они выберут в этом раунде профиль стратегии (*testify*, *testify*) — равновесие доминирующих стратегий для обоих игроков. Но как только будет определен результат 100-го раунда, 99-й раунд перестанет оказывать влияние на последующие раунды, поэтому в нем также будет выбран профиль действий (*testify*, *testify*). Следуя этому индуктивному аргументу, оба игрока должны выбирать действие *testify* в каждом раунде, заработав в сумме по 500 лет тюремного заключения на каждого. Этот тип рассуждений известен как ▶**обратная индукция**; он играет фундаментальную роль в теории игр.

Однако, если отбросить одно из трех принятых выше условий — фиксированность, конечность или взаимную известность, — то индуктивный аргумент утратит свою силу. Предположим, что игра будет повторяться *бесконечное* количество раз. С математической точки зрения стратегия игрока в бесконечно повторяющейся игре является функцией, отображающей каждую возможную конечную историю игры на выбор этого игрока в игре этапа в соответствующем раунде. Следовательно, стратегия рассматривает то, что произошло в игре ранее, и принимает решение, какой выбор сделать в текущем раунде. Но мы не можем хранить бесконечную таблицу в конечном компьютере. Нам необходима *конечная* модель стратегий для игр, в которых будет разыгрываться *бесконечное* количество раундов. По этой причине стандартом представления стратегий для бесконечно повторяющихся игр являются конечные автоматы (*finite state machines* — FSM) с выходом.

На рис. 18.3 представлен ряд стратегий конечных автоматов для повторяющейся версии игры "Дилемма заключенного". Рассмотрим стратегию ▶**Око за Око** (*Tit-for-Tat*). Каждый овал — это состояние автомата, а внутри овала — выбор, который будет сделан в соответствии с данной стратегией, если автомат будет находиться в этом состоянии. В каждом состоянии есть по одной исходящей дуге для каждого возможного выбора агента-партнера: переход для определения следующего состояния автомата выполняется по той исходящей дуге, которая соответствует выбору, сделанному другим игроком на данном этапе. Наконец, одно состояние помечено входящей стрелкой, указывающей, что это состояние является начальным. Таким образом, при стратегии Око за Око автомат начинает работу

в состоянии *refuse*; если противник выбирает действие *refuse*, то автомат остается в состоянии *refuse*, но если противник выбирает действие *testify*, то автомат также переходит в состояние *testify*. Он будет оставаться в состоянии *testify* до тех пор, пока его противник будет выбирать действие *testify*, но если в некоторый момент противник выберет действие *refuse*, автомат вернется в исходное состояние *refuse*. Другими словами, в стратегии Око за Око автомат начинает работу в состоянии *refuse*, а затем просто копирует то действие, которое противник выбрал в предыдущем раунде.

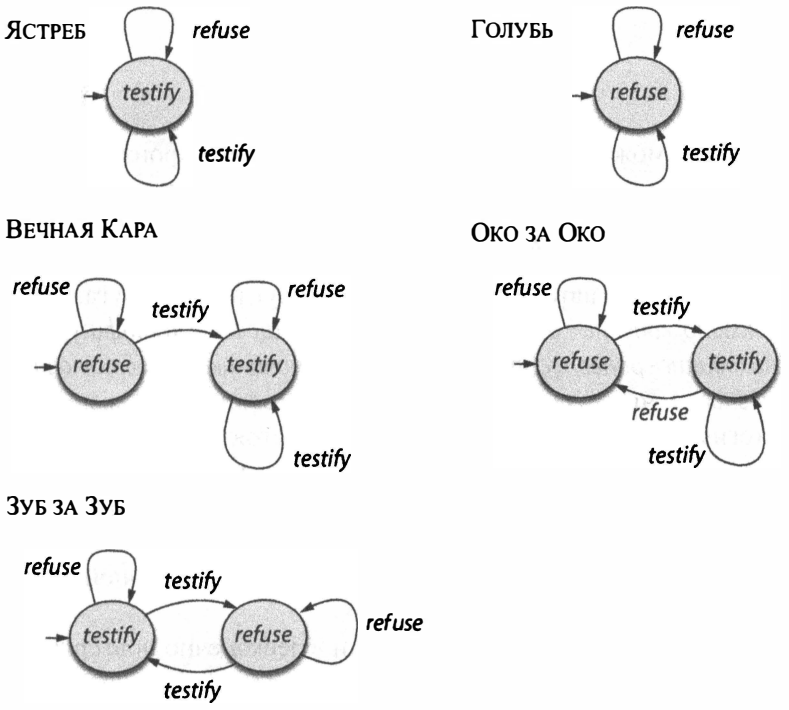


Рис. 18.3. Некоторые общие стратегии конечных автоматов с красочными названиями для бесконечно повторяющейся игры "Дилемма заключенного"

Стратегии Ястреб и Голубь проще: в стратегии Ястреб в каждом раунде выбирается действие *testify*, тогда как в стратегии Голубь в каждом раунде выбирается действие *refuse*. Стратегия Вечная Кара до некоторой степени похожа на стратегию Око за Око, но с одним важным отличием: если противник когда-либо выберет действие *testify*, то она, в сущности, превратится в стратегию Ястреб: в дальнейшем всегда выбирается действие *testify*. Хотя в стратегии Око за Око предусматривается некое прощение — в том смысле, что на действие *refuse* она предусматривает ответное действие *refuse*, в стратегии Вечная Кара пути назад

нет: за первое же выбранное противником действие *testify* он получит наказание (выбор действия *testify*), которое будет продолжаться вечно. (Вы можете описать, как ведет себя стратегия Зуб за Зуб?)

Следующая проблема в случае бесконечно повторяющихся игр — как измерить полезность бесконечной последовательности выплат. Здесь мы сосредоточимся на подходе с определением ▶ **предела средств** (*limit of means*), что, в сущности, означает взятие среднего значения полезности, полученной за бесконечную последовательность ходов. В этом подходе при имеющейся бесконечной последовательности выигрышей (U_0, U_1, U_2,...) определить полезность этой последовательности для соответствующего игрока можно как

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} U_t .$$

Это значение не может гарантированно сходиться для произвольных последовательностей полезности, но будет гарантированно сходиться для таких последовательностей полезности, которые генерируются при использовании стратегий конечных автоматов. Чтобы увидеть это, посмотрим, что будет происходить, если стратегии конечных автоматов будут играть друг против друга. В этом случае ▶ *рано или поздно, конечные автоматы вновь возвращаются в конфигурацию, в которой они уже находились ранее, и с этого момента они начинают просто повторять свое прежнее поведение.* Точнее говоря, любая последовательность полезностей, генерируемая стратегиями конечных автоматов, будет состоять из конечной (возможно, пустой) не повторяющейся последовательности, за которой следует непустая конечная последовательность, повторяющаяся бесконечное количество раз. Чтобы вычислить среднюю полезность, полученную игроком от этой бесконечной последовательности, достаточно просто вычислить среднее для конечной повторяющейся последовательности.

Исходя из этого, будем полагать, что игроки в бесконечно повторяющейся игре просто выбирают некоторый конечный автомат, который будет играть в эту игру от их имени. На эти автоматы не налагается никаких ограничений: они могут быть настолько большими и сложными, насколько этого захотят игроки. Когда все игроки выберут тот конечный автомат, который будет играть от их имени, появится возможность вычислить выплаты каждому игроку, используя подход с пределом средств, как было описано выше. Таким образом, бесконечно повторяющаяся игра сводится к нормальной форме игры, которая, тем не менее, предоставляет каждому игроку бесконечно много возможных стратегий.

Давайте посмотрим, что произойдет, если играть в бесконечно повторяющуюся дилемму заключенного, используя некоторые из стратегий, приведенных на рис. 18.3. Сначала предположим, что и Али, и Бо выбрали стратегию Голубь.

		0	1	2	3	4	5		Полезность
Али:	Голубь	*refuse*	*refuse*	*refuse*	*refuse*	*refuse*	*refuse*	...	–1
Бо:	Голубь	*refuse*	*refuse*	*refuse*	*refuse*	*refuse*	*refuse*	...	–1

Легко увидеть, что эта пара стратегий не формирует равновесие Нэша: любой из игроков сделал бы для себя лучше, изменив свой выбор на стратегию Ястреб. Итак, предположим, что Али переходит на стратегию Ястреб.

		0	1	2	3	4	5		Полезность
Али:	Ястреб	*testify*	*testify*	*testify*	*testify*	*testify*	*testify*	...	0
Бо:	Голубь	*refuse*	*refuse*	*refuse*	*refuse*	*refuse*	*refuse*	...	–10

Этот вариант является худшим из возможных исходов для Бо, к тому же эта пара стратегий вновь не является равновесием Нэша. Бо может сделать себе лучше, если также выберет стратегию Ястреб.

		0	1	2	3	4	5		Полезность
Али:	Ястреб	*testify*	*testify*	*testify*	*testify*	*testify*	*testify*	...	–5
Бо:	Ястреб	*testify*	*testify*	*testify*	*testify*	*testify*	*testify*	...	–5

Эта пара стратегий *образует* равновесие Нэша, но не очень интересное, поскольку в той или иной мере отбрасывает ситуацию назад, когда мы начали обсуждение одноэтапного варианта этой игры и оба игрока приняли решение свидетельствовать друг против друга. Этот наблюдение иллюстрирует ключевое свойство бесконечно повторяющихся игр: ➡ *равновесия Нэша для игры этапа будут поддерживаться как состояния равновесия и в бесконечно повторяемой версии этой игры.*

Однако наша история еще не закончена. Предположим, что Бо переключился на стратегию Вечная Кара.

		0	1	2	3	4	5		Полезность
Али:	Ястреб	*testify*	*testify*	*testify*	*testify*	*testify*	*testify*	...	–5
Бо:	Вечная Кара	*refuse*	*testify*	*testify*	*testify*	*testify*	*testify*	...	–5

В этом случае Бо не сделал себе хуже, чем при выборе стратегии Ястреб: в первом раунде Али выбирает *testify*, тогда как Бо выбирает *refuse*, но выбор Али далее переключает стратегию Бо на постоянный выбор *testify*, а потеря полезности на первом этапе в пределе исчезает. В конечном счете оба эти игрока получают ту же полезность, как если бы они оба следовали стратегии Ястреб. Но здесь есть важный момент: стратегии Ястреб и Вечная Кара не образуют равновесия Нэша, поскольку на этот раз у Али есть возможность сделать благоприятное для него

изменение — переход на стратегию ВЕЧНАЯ КАРА. Если оба игрока выбирают стратегию ВЕЧНАЯ КАРА, то происходит следующее.

		0	1	2	3	4	5		Полезность
Али:	ВЕЧНАЯ КАРА	*refuse*	*refuse*	*refuse*	*refuse*	*refuse*	*refuse*	...	–1
Бо:	ВЕЧНАЯ КАРА	*refuse*	*refuse*	*refuse*	*refuse*	*refuse*	*refuse*	...	–1

Результаты и выигрыши такие же, как если бы оба игрока выбрали стратегию ГОЛУБЬ, но в отличие от этого случая, стратегия ВЕЧНАЯ КАРА в игре против стратегии ВЕЧНАЯ КАРА формирует равновесие Нэша: и Али, и Бо способны рационально достичь результата, который невозможен в одноходовой версии этой игры.

Чтобы понять, что эти стратегии образуют равновесие Нэша, будем рассуждать методом от противного — предположим, что они его не образуют. В этом случае один игрок — без потери общности можно предположить, что это Али — имеет выгодное отклонение в форме стратегии конечного автомата, которая принесет ему более высокую выплату, чем стратегия ВЕЧНАЯ КАРА. Тогда в какой-то момент эта стратегия будет иметь возможность выбрать какое-то другое действие, отличное от действия в стратегии ВЕЧНАЯ КАРА, — в противном случае он получит ту же полезность. Следовательно, в какой-то момент Али придется выбрать действие *testify*. Но тогда Бо, следуя стратегии ВЕЧНАЯ КАРА, переключится в режим наказания, постоянно выбирая в ответ действие *testify*. С этого момента Али будет обречен всегда получать выплату не более чем –5, т.е. хуже, чем –1, которые он получал, придерживаясь стратегии ВЕЧНАЯ КАРА. Следовательно, когда оба игрока выбирают стратегию ВЕЧНАЯ КАРА, в бесконечно повторяемой игре "Дилемма заключенного" формируется равновесие Нэша, давая рационально устойчивый результат, который невозможен в одноходовой версии этой игры.

Это пример общего класса результатов, называемых ▶ **народными теоремами Нэша**, характеризующих исходы, которые могут быть поддержаны равновесиями Нэша в бесконечно повторяющихся играх. Пусть *обеспеченная стоимость* (*security value*) — это лучшая выплата, которую игрок может гарантированно получить. Тогда общую форму народных теорем Нэша можно сформулировать примерно так: ➔ *каждый результат, в котором каждый игрок получает по крайней мере свою обеспеченную стоимость, может быть подтвержден как равновесие Нэша в бесконечно повторяющейся игре.* Стратегия ВЕЧНАЯ КАРА является ключом к народным теоремам: взаимная угроза наказания, если какой-либо из агентов откажется сыграть свою роль в желаемом общем результате, удерживает игроков от отклонения от принятой стратегии. Но это сработает как фактор сдерживания, только если другой игрок уверен, что вы уже приняли эту стратегию или, по крайней мере, могли бы ее принять.

Кроме того, другие решения могут быть получены в результате модификации самих агентов, а не изменения правил их взаимодействия. Предположим, что агенты представляют собой конечные автоматы с n состояниями и играют в игру с общим количество ходов $m > n$. В результате агенты в какой-то момент оказываются неспособными представить целый ряд оставшихся ходов и должны рассматривать их как неизвестные. Это означает, что они не могут выполнять логический вывод по индукции и в повторяющейся игре "Дилемма заключенного" вправе переходить к наиболее благоприятному равновесию (*refuse*, *refuse*). В таком случае неведение идет на пользу — или, скорее, идет на пользу мнение противника о том, что вы *находитесь* в неведении. Ваш успех в таких повторяющихся играх в значительной степени зависит от *представления* о вас другого игрока как о хвастуне или простаке, а не от ваших фактических характеристик.

18.2.4. Последовательные игры: развернутая форма

В общем случае игра состоит из последовательности ходов, которые не обязательно должны быть одинаковыми. Такие игры лучше всего представлять в виде дерева игры, которое теоретики игр называют ►**развернутой формой** (*extensive form*). Такое дерево игры включает в себя всю ту же информацию, которая рассматривалась в разделе 5.1: начальное состояние S_0, функцию PLAYER(s), определяющую игрока, которому принадлежит право совершить ход, функцию ACTIONS(s), которая задает множество возможных действий, функцию RESULT(s, a), определяющую переход в новое состояние, и частичную функцию UTILITY(s,p), которая определена только в терминальных состояниях и представляет выигрыш для каждого игрока. Включить в рассмотрение стохастические игры можно за счет введения необычного игрока *Chance* (случай), способного выполнять случайные действия. "Стратегия" игрока *Chance* является частью определения игры и задается как распределение вероятностей для действий (остальным игрокам предоставляется право выбирать себе собственную стратегию). Для представления игр с недетерминированными действиями, таких как бильярд, каждое такое действие разбивается на две части: действие игрока само по себе, имеющее детерминированный результат, а затем действие игрока *Chance*, которому предоставляется возможность отреагировать на выполняемое действие собственным непостоянным образом.

На данный момент мы сделаем одно упрощающее предположение: будем полагать, что игроки обладают ►**полной информацией**. Грубо говоря, полнота информации означает, что когда игра требует от игрока принять решение, он точно знает, где именно он находятся в дереве игры: нет никакой неопределенности в отношении того, что произошло в данной игре ранее. Это, безусловно, ситуация, характерная для таких игр, как шахматы или го, но невозможная в таких играх, как покер или кригшпиль. В следующем разделе будет показано, как развернутая форма может быть использована в подобных играх для извлечения **неполной информации**, но на данный момент мы будем считать, что игрокам доступна полная информация.

Для игрока в развернутой форме игры с полной информацией стратегией является функция, которая для каждого из его состояний принятия решения s предписывает, какое действие в ACTIONS(s) игрок должен выбрать для выполнения. Если каждый игрок уже выбрал стратегию, то результирующий профиль стратегии будет отмечать пути в дереве игры из начального состояния S_0 в некоторое терминальное состояние, а его функция UTILITY будет определять полезности, которые получит каждый из игроков.

Приняв эти установки, можно непосредственно применить представленный выше аппарат определения равновесий Нэша для анализа игр в развернутой форме. Чтобы вычислить равновесия Нэша, можно использовать прямое обобщение метода минимаксного поиска, обсуждавшегося нами в главе 5. В литературе по играм в развернутой форме этот метод называют обратной индукцией — выше в этой главе обратная индукция уже неформально использовалась при анализе повторяющейся ограниченное число раз игры "Дилемма заключенного". В методе обратной индукции используется динамическое программирование, работающее в обратном направлении, от терминальных состояний к исходному состоянию, последовательно помечая каждое состояние профилем выплат (присвоение выплат игрокам), которые были бы ими получены, если бы игра была оптимально сыграна, начиная с этого момента.

Если говорить подробнее, для каждого нетерминального состояния s, когда все его дочерние узлы уже были помечены профилем выплат, узел s помечается профилем выплат того из его дочерних узлов, который максимизирует выплату игрока, принимающего решение в состоянии s. (Если есть несколько одинаковых максимизирующих профилей выплат, то выбор выполняется произвольно, если есть узлы жеребьевки, вычисляется ожидаемая полезность.) Алгоритм обратной индукции гарантированно завершается, а кроме того, его время выполнения находится в полиномиальной зависимости от размера дерева игры.

В процессе работы алгоритм отслеживает стратегии для каждого игрока. Как было установлено, эти стратегии являются стратегиями равновесия Нэша, а профиль выплат, которым помечено начальное состояние, является профилем тех выплат, которые будут получены при ведении игры в соответствии со стратегиями равновесия Нэша. Таким образом, стратегии равновесия Нэша для игр в развернутой форме могут быть вычислены за полиномиальное время с использованием метода обратной индукции. А поскольку этот алгоритм гарантированно маркирует начальное состояние профилем выплат, то можно сделать вывод, что каждая игра в развернутой форме имеет по крайней мере одно равновесие Нэша в чистых стратегиях.

Это привлекательный результат, но все же следует сделать несколько предостережений. Деревья игры очень быстро становятся чрезвычайно большими, а это значит, что полиномиальность времени выполнения следует понимать именно в этом контексте. Но еще более проблематично то, что равновесие Нэша само по себе имеет некоторые ограничения, когда оно применяется к играм в развернутой форме. Рассмотрим игру, представленную на рис. 18.4. У игрока 1 есть два

возможных хода: *above* (выше) и *below* (ниже). Если он выберет ход *below*, то оба игрока получают выплату 0 (независимо от хода, выбранного игроком 2). Если он выберет ход *above*, то игроку 2, в свою очередь, будет предоставлена возможность выбора хода — *up* (вверх) или *down* (вниз). Если он выберет ход *down*, то оба игрока получат выплату 0, а если он выберет ход *up*, то оба игрока получают выплату 1.

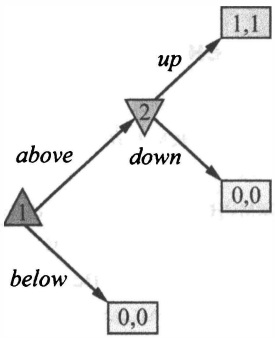


Рис. 18.4. Развернутая форма игры с контринтуитивным равновесием Нэша

Алгоритм обратной индукции непосредственно говорит нам о том, что профиль стратегии (*above*, *up*) является равновесием Нэша и приводит к тому, что оба игрока получают выплату 1. Однако профиль стратегии (*below*, *down*) также является равновесием Нэша, но приводит обоих игроков к получению выплаты 0. Игрок 2 является угрозой для игрока 1, поскольку, получив право хода, может принять решение выбрать ход *down*, в результате чего выплата игроку 1 будет равна 0, и в этом случае у игрока 1 нет лучшей альтернативы, чем выбрать ход *below*. Проблема заключается в том, что угроза со стороны игрока 2 (выбрать ход *down*) не является ▶**реальной угрозой**, поскольку, если игрок 2 на самом деле получит право хода, то он выберет ход *up*.

Уточнение равновесия Нэша, называемое ▶**идеальным равновесием Нэша в подыгре,** устраняет эту проблему. Чтобы сформулировать полное определение, необходимо ввести понятие ▶**подыгры**. Каждое состояние принятия решения в дереве игры (включая исходное состояние) определяет подыгру, например игра, представленная на рис. 18.4, содержит две подыгры: у первой корневым узлом является узел принятия решения игроком 1, а у второй — узел принятия решения игроком 2. ◀ *В игре G профиль стратегии формирует идеальное равновесие Нэша в подыгре в том случае, если равновесие Нэша есть в каждой подыгре из игры G.* Применив это определение к игре на рис. 18.4, находим, что профиль стратегии (*above*, *up*) является идеальной подыгрой, а профиль (*below*, *down*) — нет, поскольку выбор хода *down* не является равновесием Нэша в подыгре с корнем в узле принятия решения игроком 2.

Хотя для определения идеального равновесия Нэша в подыгре нам потребовались некоторые новые термины, нам не требуется никаких новых алгоритмов. Стратегии, рассчитанные с помощью метода обратной индукции, будут являться идеальными равновесиями Нэша в подыгре, а отсюда следует, что каждая игра в развернутой форме с полной информацией имеет идеальное равновесие Нэша в подыгре, которое может быть вычислено за полиномиальное время в зависимости от размера дерева игры.

Игрок Chance и одновременные ходы

Чтобы представить в развернутой форме стохастическую игру, такую как нарды, в нее добавляется игрок под именем *Chance*, выбор которого определяется распределением вероятностей.

Для представления одновременных ходов, как в играх "Дилемма заключенного" или в чет-нечет на двух пальцах, установим для игроков произвольный порядок, но при этом будем полагать, что действие игрока, который на этом этапе игры сделал ход ранее, является ненаблюдаемым для всех последующих игроков, делающих свой ход на этом этапе. Например, Али первым выбирает действие *refuse* или *testify*, а затем действие выбирает Бо, но при этом Бо не знает, какой выбор сделал Али на данном этапе (можно также принять тот факт, что ход Али будет раскрыт позже). Тем не менее предполагается, что игроки всегда помнят все *собственные* предыдущие действия; это предположение называют **идеальной памятью** (*perfect recall*).

Сбор неполной информации

Ключевой особенностью развернутой формы, отличающей ее от игровых деревьев, приведенных в главе 5, является то, что она позволяет отражать частичную наблюдаемость. Теоретики игр используют термин ► **неполная информация** (*imperfect information*) для описания ситуаций, когда игроки не имеют уверенности относительно фактического состояния игры. К сожалению, метод обратной индукции не применим к играми с неполной информацией, и, как правило, они значительно сложнее, чем игры с полной информацией.

В разделе 5.6 было показано, что игрок в частично наблюдаемой игре, такой как кригшпиль, может построить дерево игры в пространстве **доверительных состояний**. Мы видели, что при наличии подобного дерева игрок в некоторых случаях может найти последовательность ходов (стратегию), которая приводит к принудительному мату, независимо от того, в каком фактическом состоянии он начал, и независимо от того, какую стратегию использует противник. Тем не менее методы, обсуждавшиеся в главе 5, не могли указать игроку, что делать, когда нет гарантированного мата. Если лучшая стратегия игрока зависит от стратегии его соперника и наоборот, то минимаксный (или альфа-бета) алгоритм сам по себе не сможет найти решение. Однако развернутая форма *позволит* нам найти требуемое

решение, поскольку она представляет доверительные состояния (теоретики игр называют их ►**информационными множествами** (*information sets*)) сразу для *всех* игроков. В таком представлении можно найти равновесные решения, точно так же, как это делалось в случае игр в нормальной форме.

В качестве простого примера последовательной игры поместим двух агентов в клеточный мир 4×3, представленный на рис. 17.1, и позволим им одновременно совершать перемещения, пока один из них не достигнет конечного состояния и не получит вознаграждение, полагающееся в данном квадрате. Если предположить, что в случае, когда два агента пытаются перейти в один и тот же квадрат, никакого перемещения не происходит (общая проблема для многих уличных перекрестков), то некоторые чистые стратегии могут попасть в тупик. Следовательно, чтобы хорошо играть в этой игре, агентам нужна смешанная стратегия: случайный выбор между движением вперед и пребыванием на месте. Это именно то, что выполняется для разрешения коллизий передаваемых сетевых пакетов в сетях Ethernet.

Давайте рассмотрим очень простой вариант покера. В колоде всего четыре карты, два туза и два короля. Каждому из двух игроков сдается одна карта. Далее первый игрок может играть либо *raise* — повысить ставки в игре с 1 до 2, либо *check* — согласиться открыть карты. Если игрок 1 играет *check*, то игра окончена. Если игрок 1 играет *raise*, то игрок 2 может играть либо *call*, принимая повышение стоимости игры до 2 очков, либо *fold*, сбрасывая свою карту и проигрывая 1 очко. Если игра не заканчивается ходом *fold*, то выигрыш зависит от карт: он равен нулю для обоих игроков, если у них на руках карты одинакового старшинства; в противном случае игрок, имеющий короля, выплачивает установленную ставку игроку, у которого туз.

Развернутая форма дерева этой игры показана на рис. 18.5. Здесь игрок 0 — это игрок *Chance*, а игроки 1 и 2 представлены треугольниками. Каждое действие изображено в виде стрелки с однобуквенной меткой, соответствующей действиям ***raise***, ***check***, ***call*** и ***fold***, а для игрока *Chance* — это четыре возможных варианта раздачи (АК означает, что игрок 1 получает туза, а игрок 2 — короля). Терминальные состояния представлены прямоугольниками, помеченными соответствующими выплатами игроку 1 и игроку 2. Информационные множества (доверительные состояния) показаны в виде пунктирных прямоугольников с соответствующей меткой. Например, $I_{1,1}$ — это информационное множество для случая, когда ходит игрок 1, который уже знает, что у него туз (но не знает, какая карта у игрока 2). В случае информационного множества $I_{2,1}$ ход делает игрок 2 и он знает, что у него туз и что игрок 1 поднял ставку до 2, но не знает, какая у этого игрока карта. (Из-за ограничений, связанных с двухмерностью изображений на бумаге, это информационное множество представлено на рисунке двумя пунктирными прямоугольниками, а не одним.)

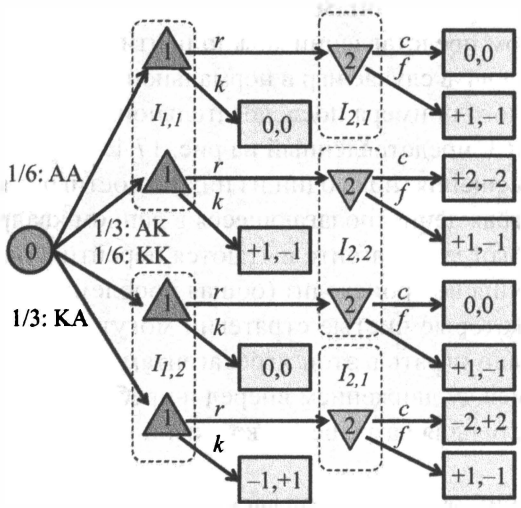


Рис. 18.5. Развернутая форма упрощенной версии покера с двумя игроками и только четырьмя картами. Ходы игроков обозначены как *r* (*raise*), *f* (*fold*), *c* (*call*) и *k* (*check*)

Один из способов решения игры в развернутой форме — преобразовать ее в игру в нормальной форме. Напомним, что нормальной формой игры является матрица, каждая строка которой маркируется чистой стратегией для игрока 1, а каждый столбец — чистой стратегией для игрока 2. В развернутой форме игры чистая стратегия для игрока *i* соответствует действию в каждом информационном множестве, включающем этого игрока. Так, на рис. 18.5 одной из чистых стратегий для игрока 1 является "*raise* в $I_{1,1}$ (т.е. когда у меня туз) и *check* в $I_{1,2}$ (т.е. когда у меня король)". В приведенной ниже матрице выплат эта стратегия называется *rk*. Аналогичным образом стратегия *cf* для игрока 2 означает "*call*, когда у меня туз, и *fold*, когда у меня король". Поскольку это игра с нулевой суммой, в приведенной ниже матрице указан выигрыш только для игрока 1, — у игрока 2 выплата всегда такая же, но противоположная по знаку.

	2: *cc*	2: *cf*	2: *ff*	2: *fc*
1: *rr*	0	−1/6	1	7/6
1: *kr*	−1/3	−1/6	5/6	2/3
1: *rk*	1/3	**0**	1/6	1/2
1: *kk*	0	**0**	0	0

Эта игра настолько проста, что в ней есть два равновесия с чистой стратегией, выделенные жирным шрифтом: *cf* — для игрока 2 и *rk* или *kk* — для игрока 1.

Но в общем случае вполне возможно решать игры в развернутой форме, преобразовав их в нормальную форму, а затем отыскав решение (обычно это смешанная стратегия) с использованием стандартных методов линейного программирования. Все это хорошо работает в теории. Но если у игрока есть I информационных наборов по a действий в каждом из них, то этот игрок будет иметь a^I чистых стратегий. Другими словами, размер матрицы нормальной формы экспоненциально зависит от количества информационных наборов, поэтому на практике такой подход работает только для крошечных деревьев игры — дюжины состояний или около того. Даже такая относительно простая игра как техасский холдем (классический покер) всего для двух игроков имеет около 10^{18} состояний, что делает для нее этот подход абсолютно невыполнимым.

Какие имеются альтернативы? В главе 5 было показано, что поиск альфа-бета может успешно применяться в случае игр с полной информации, обладающими огромными деревьями игры, создавая это дерево шаг за шагом, отсекая некоторые ветви и эвристически оценивания нетерминальные узлы. Но этот подход не слишком хорошо работает для игр с неполной информацией — по двум причинам. Во-первых, в этом случае отсечение затруднительно, поскольку необходимо рассматривать смешанные стратегии, сочетающие в себе несколько ветвей, а не чистую стратегию, в которой всегда выбирается лучшая ветвь. Во-вторых, в данном случае очень усложняется эвристическая оценка нетерминальных узлов, — приходится иметь дело с информационными множествами, а не с отдельными состояниями.

Коллер и соавт. ([1263], 1996) пришел на помощь с альтернативным представлением обширных игр, называемым ▶ **последовательной формой** (*sequence form*), которая линейно пропорциональна размеру дерева, а не экспоненциально. Вместо стратегий в ней представляются пути по дереву, а количество этих путей равно количеству терминальных узлов. К этому представлению также могут применяться стандартные методы линейного программирования. Получившаяся система способна решать варианты покерных задач с 25 тысячами состояний за минуту или две. Это экспоненциальное ускорение относительно подхода с использованием нормальной формы, но, по-прежнему, еще очень далеко даже до решения задачи стандартного покера с двумя игроками, имеющей 10^{18} состояний.

Если мы не можем обработать 10^{18} состояний, то, вероятно, сможем упростить задачу, изменив форму игры на более простую. Например, если я держу в руке туз и рассматриваю возможность того, что следующая карта даст мне пару тузов, то меня не интересует масть этой следующей карты. В соответствии с правилами покера любая масть будет в этом случае одинаково хороша. Это подсказывает, что можно образовать некую **абстракцию** игры, в которой игнорируются масти. Получившееся игровое дерево будет меньше в $4! = 24$ раза. Предположим, что есть возможность решить эту упрощенную игру, тогда как полученное решение будет относиться к оригинальной игре? Если нет игрока, который рассматривает

вариант сбора флеша (единственную комбинацию в игре, в которой масть имеет значение), то решение для абстракции также будет решением для исходной игры. Если же какой-то игрок рассматривает флеш как возможный вариант, то абстракция будет лишь приблизительным решением (но при этом можно вычислить границы ошибки).

Существует много возможностей для абстракции. Например, в тот момент игры, когда у каждого из игроков по две карты, и у меня на руках пара дам, карты у других игроков можно разделить на три категории: *лучше* (только пара королей или пара тузов), *равно* (пара дам) и *хуже* (все остальное). Однако эта абстракция может оказаться слишком грубой. Лучшей абстракцией будет разделить *худшее* на, скажем, *средние пары* (от девяток до валетов), *младшие пары* и *нет пары*. Эти примеры являются абстракциями состояний; но возможно абстрагировать и действия. Например, вместо того, чтобы делать ставки в виде любого целого числа от 1 до 1000, можно ограничиться ставками 10^0, 10^1, 10^2 и 10^3. Или же можно вообще полностью исключить один из раундов ставок. Также допустимо абстрагироваться от узлов жеребьевки, рассматривая только подмножество возможных раздач. Это эквивалентно методу развертывания, используемому в программах игры в го. Объединив все эти абстракции, можно сократить количество состояний в игре в покер с 10^{18} до 10^7 — размера игрового дерева, для которого уже можно найти решение с помощью современных методов.

Как уже говорилось в главе 5, программы игры в покер, такие как Libratus и DeepStack, смогли нанести поражение чемпиону среди игроков-людей в парной (два игрока) игре в техасский холдем. Совсем недавно программа Pluribus смогла победить чемпионов-людей в покере с шестью игроками в двух форматах: пять копий программы за столом с одним человеком и одна копия программы за столом с пятью людьми. В этих случаях следует отметить огромный скачок в сложности игры. При одном противнике существует возможность сдачи $\binom{50}{2}=1225$ вариантов скрытых карт противнику. Но при пяти противниках из 50 *выбирают* $10 \approx 10$ миллиардов возможностей. Программа Pluribus сначала полностью разрабатывает базовую стратегию собственной игры, а затем модифицирует ее в процессе реальной игры, реагируя на конкретную ситуацию. В программе Pluribus используется комбинация нескольких методов, в том числе поиск по дереву методом Монте-Карло, поиск с ограничением глубины и абстракция.

Развернутая форма является универсальным представлением: ее можно использовать в частично наблюдаемой, мультиагентной, стохастической, последовательной среде, причем в режиме реального времени. Этот перечень типов охватывает большинство самых сложных вариантов свойств проблемной среды, обсуждавшихся в разделе 2.3.2. Однако все же существуют два ограничения на использование развернутой формы в частности и в теории игр — в целом. Во-первых, эта форма не очень хорошо справляется с непрерывными состояниями и действиями (хотя были разработаны некоторые расширения для непрерывного

случая, — например, теория ▶**конкуренции Курно** использует теорию игр для решения задачи, в которой две компании устанавливают цены на свою продукцию из непрерывного диапазона. Во-вторых, теория игр предполагает, что игра *известна*. Отдельные элементы игры могут быть определены как ненаблюдаемые для некоторых игроков, но всегда должно быть известно, какие элементы являются ненаблюдаемыми. Но в тех случаях, когда игроки изучают неизвестную структуру игры с течением времени, данная модель оказывается непригодной. Давайте проанализируем каждый источник неопределенности и выясним, можно ли их представить в теории игр.

Действия. Нет простого способа представить игру, в которой игроки должны выяснять, какие действия им доступны. Рассмотрим игру между создателями компьютерных вирусов и экспертами по безопасности. Часть задачи состоит в ожидании того, какие действия авторы вирусов предпримут дальше.

Стратегии. Теория игр очень хороша для представления идеи, что стратегии других игроков изначально неизвестны, — при условии, что все агенты рациональны. Но теория ничего не говорит о том, что делать, если другие игроки не полностью рациональны. Понятие ▶**равновесия Байеса–Нэша** частично решает данную проблему: это равновесие относительно априорного распределения вероятностей игрока по отношению к стратегиям других игроков. Иными словами, оно выражает убеждения игрока относительно вероятных стратегий других игроков.

Жеребьевка. Если игра зависит от броска кубика, то это достаточно просто смоделировать узлом жеребьевки с равномерным распределением по результатам. Но что, если допустить возможность использования поддельных кубиков? Можно представить ситуацию другим узлом жеребьевки, выше по дереву игры, с двумя ветвями: "кубик честный" и "кубик нечестный", построив их таким образом, что соответствующие узлы в каждой из ветвей будут находиться в одном и том же информационном множестве (т.е. игроки не знают, каким является кубик, честным или поддельным). А как поступить, если допустить, что другой игрок это знает? Тогда можно добавить *другой* узел жеребьевки, с одной ветвью, представляющей случай, когда противник знает о поддельном кубике, и другой ветвью, где противник этого не знает.

Полезности. Что делать, если полезности оппонента нам неизвестны? И вновь, эту ситуацию можно смоделировать с помощью узла жеребьевки, такого, чтобы другой агент знал собственные полезности в каждой ветви, но нам они были неизвестны. Но что делать, если мы не знаем *собственных* полезностей? Например, как я узнаю, будет ли рационально заказать салат от шеф-повара, если я не знаю, насколько он мне понравится? И это тоже можно смоделировать с помощью еще одного узла жеребьевки, определяющего ненаблюдаемое "внутреннее качество" салата.

Таким образом, можно прийти к заключению, что теория игр хорошо подходит для большинства источников неопределенности, но за счет удвоения размера дерева игры всякий раз, когда в него добавляется еще один узел. К сожалению,

подобный подход быстро приводит к получению непреодолимо больших деревьев. Из-за этих и других проблем теория игр использовалась главным образом для *анализа* равновесных сред, а не для *управления* агентами внутри среды.

18.2.5. Неопределенные выплаты и игры содействия

В главе 1, раздел 1.1.5, отмечалась важность разработки систем искусственного интеллекта, которые смогут работать в условиях неопределенности относительно истинной цели человека. В главе 16, раздел 16.7.1, была предложена простая модель представления неопределенности о *собственных* предпочтениях на примере мороженого с ароматом дуриана. Простым приемом добавления в модель новой скрытой переменной для представления неизвестных предпочтений, вместе с соответствующей моделью восприятия (например, проверка вкуса небольшого кусочка мороженого), неопределенные предпочтения могут быть обработаны вполне естественным образом.

В главе 16 также рассматривалась **проблема отключения**: было показано, что робот при наличии неопределенности в отношении предпочтений человека будет подчиняться человеку и позволит себе отключиться. В этой задаче робот Робби не имеет полной определенности в отношении предпочтений Гарриет, человека, но мы моделируем решения Гарриет (отключить Робби от решения задачи или не отключать) как простое, детерминированное следствие из ее собственных предпочтений в отношении действия, предложенного Робби. Ниже эта идея будет обобщена до полной игры с двумя игроками под названием **игра содействия** (*assistance game*), в которой Гарриет и Робби будут игроками. Предположим, что Гарриет наблюдает за своими предпочтениями θ и действует в соответствии с ними, в то время как у Робби есть лишь априорная вероятность $P(\theta)$ относительно предпочтений Гарриет. Выплата определяется по θ и одинакова для обоих игроков: как Гарриет, так и Робби стремятся к максимизации выплаты Гарриет. Таким образом, игра содействия обеспечивает более формальную модель идеи доказуемо полезного ИИ, введенной в главе 1.

В дополнение к подчиненному поведению, проявленному Робби в задаче отключения — которая представляет собой ограниченный вариант игры содействия, — другие варианты поведения, возникающие как равновесие стратегий в обобщенной игре содействия, включают в себя действия со стороны Гарриет, которые можно определить как обучение, награждение, командование, исправление, демонстрация или объяснение, а также действия со стороны Робби, которые можно определить как просьба о разрешении, обучение на основе демонстрации, выявление предпочтений и т.д. Ключевым пунктом здесь является то, что эти модели поведения не должны подчиняться какому-либо сценарию: в процессе решения игры Гарриет и Робби вырабатывают у себя способность передавать информацию о предпочтениях от Гарриет к Робби таким образом, чтобы Робби мог стать полезнее для Гарриет. Не следует заранее оговаривать, что Гарриет "награждает" или

что Робби "следует инструкциям", хотя это может быть разумной интерпретацией того, как они будут вести себя в конечном счете.

Для иллюстрации игр содействия мы воспользуемся ▶**игрой канцелярской скрепки**. Это очень простая игра, в которой Гарриет-человек намеревается "сигнализировать" роботу Робби некоторую информацию о своих предпочтениях. Робби способен интерпретировать этот сигнал, потому что он может решить эту игру и, следовательно, способен понять, что могло бы быть правдой в отношении предпочтений Гарриет, чтобы она подала ему сигнал таким образом.

Этапы игры представлены на рис. 18.6, — она касается производства канцелярских скрепок и скоб. Предпочтения Гарриет выражены с помощью функции вознаграждения, зависящей от количества изготовленных скрепок и скоб с определенным "обменным курсом" между ними. Параметр предпочтения Гарриет θ определяет относительную стоимость (в долларах) одной скрепки. Например, она может оценить скрепки по $\theta = 0{,}45$ долл. и это означает, что скобы будут стоить по $1 - \theta = 0{,}55$ долл. Таким образом, если будет произведено p скрепок и s скоб, то вознаграждение Гарриет составит $p\theta + s(1-\theta)$ долл. за все. Для Робби установлено априорное равномерное распределение $P(\theta) = Uniform(\theta; 0, 1)$. В процессе самой игры Гарриет делает ход первой и может сделать две скрепки, две скобы или по одной каждой. Затем Робби может сделать 90 скрепок, 90 скобок или по 50 штук каждой.

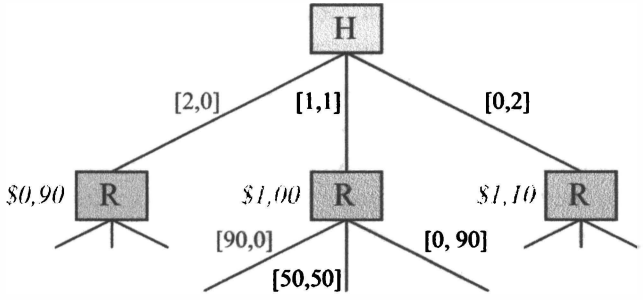


Рис. 18.6. Дерево игры "Канцелярская скрепка". Каждая ветвь помечена значениями $[p, s]$, обозначающими количество скрепок и скоб, которые будут изготовлены при выборе этой ветки. Гарриет — человек — может сделать или две скрепки, или две скобы, или по одной каждой. (Значения, выделенные светлым курсивом, являются стоимостью игры для Гарриет, если игра на этом закончится, при условии, что $\theta = 0{,}45$.) Затем робот Робби делает выбор из трех возможных вариантов — изготовить 90 скрепок, 90 скоб или по 50 штук каждой

Обратите внимание, что если бы Гарриет делала это только для себя, она бы просто сделала две скобы со стоимостью игры 1,10 долл. (Обратите внимание на отметки, приведенные для узлов первого уровня дерева игры на рис. 18.6.) Но Робби наблюдает за ее действиями и учится на ее выборе. Что именно он изучает?

Это зависит от того, как Гарриет делает свой выбор. А как Гарриет сделать свой выбор? Это зависит от того, как Робби собирается его интерпретировать. Можно разорвать этот порочный круг, если отыскать для игры равновесие Нэша. В данном случае оно является уникальным и может быть найдено посредством применения метода близорукого наилучшего ответа: выбрать любую стратегию Гарриет; выбрать лучшую стратегию для Робби, учитывая выбор стратегии Гарриет; выбрать лучшую стратегию для Гарриет, учитывая выбор стратегии Робби; и т.д. Процесс разворачивается следующим образом.

1. Начать игру с выбора жадной стратегии для Гарриет: сделать две скрепки, если она предпочитает скрепки; сделать две скобы, если она предпочитает скобы, и сделать по одной каждой, если она безразлична к выбору.
2. Теперь есть три возможности, которые Робби должен рассмотреть с учетом стратегии, которую выбрала Гарриет.
   а) Если Робби видит, что Гарриет сделала две скрепки, он делает вывод, что она предпочитает скрепки, поскольку в настоящее время он полагает, что ценность скрепки равномерно распределяется между 0,5 и 1,0 со средним значением 0,75. В этом случае для него лучшим планом является сделать 90 скрепок с ожидаемым вознаграждением для Гарриет в 67,50 долл.
   б) Если Робби видит, что Гарриет сделала одну скрепку и одну скобу, он делает вывод, что она оценивает скрепки и скобы одинаково по 0,50, поэтому лучший выбор — сделать по 50 штук каждой.
   в) Если Робби видит, что Гарриет сделала две скобы, то, исходя из той же аргументации, что и в пункте *а*, он должен сделать 90 скоб.
3. С учетом выбора стратегии со стороны Робби лучшая стратегия для Гарриет на этом этапе несколько отличается от жадной стратегии на этапе 1. Если Робби будет реагировать на изготовление ею одной единицы путем изготовления 50 аналогичных единиц, то ей лучше делать по штуке каждого изделия не только тогда, когда она действительно равнодушна к выбору между двумя изделиями, но и если ее предпочтения где-то недалеко от такого равнодушия. В действительности для нее оптимальной политикой в настоящий момент будет делать по одной штуке каждого изделия, если предпочтительность скрепок для нее находится приблизительно между 0,446 и 0,554.
4. С учетом этой новой стратегии Гарриет стратегия Робби остается неизменной. Например, если она решила сделать по одной штуке каждого вида продукции, он делает вывод, что предпочтение в отношении скрепки является равномерно распределенным между 0,446 и 0,554, со средним значением 0,50, так что лучшим выбором для него будет сделать по 50 штук каждого изделия. Поскольку стратегия Робби на этом этапе осталась той же, что и на этапе 2, лучшим ответом для Гарриет будет поступить так же, как на этапе 3, а значит, равновесие уже найдено.

Благодаря своей стратегии Гарриет, по сути, обучает Робби своим предпочтениям, используя простой код — язык, если хотите, — который следует из анализа равновесия. Также обратите внимание, что Робби никогда точно не узнает, в чем же состоят предпочтения Гарриет, но он выучит достаточно, чтобы оптимально действовать от ее имени, т.е. он будет действовать так же, как если бы он точно знал ее предпочтения. Он доказуемо выгоден для Гарриет в соответствии со сделанными предположениями и при допущении, что Гарриет играет в эту игру правильно.

Метод близорукого наилучшего ответа хорошо работает в этом примере и других, ему подобных, но непригоден для более сложных случаев. Можно доказать, что при условии, когда нет ни одной связи, вызывающей проблемы координации, нахождение оптимального профиля стратегии для игры содействия сводится к решению задачи POMDP, пространство состояний которой является основным пространством состояния базовой игры плюс параметры θ предпочтений человека. Задачи POMDP в общем случае решать очень трудно (см. раздел 17.5), но что касается задач POMDP, представляющих игры содействия, они всегда имеют дополнительную структуру, позволяющую применять более эффективные алгоритмы.

Игры содействия можно обобщить на самые разные ситуации, чтобы учесть наличие в игре нескольких участников-людей, нескольких роботов, людей с несовершенной рациональностью, людей, которые не знают своих предпочтений и т.д. Если для пространства действий использовать развернутое или структурированное предоставление — вместо простых атомарных действий, как в игре "Канцелярская скрепка", — то возможности взаимодействия могут быть значительно расширены. Очень немногие из этих вариаций уже были изучены на текущий момент, но можно ожидать, что ключевое свойство игр содействия останется неизменным: более умный робот, лучший результат для человека.

18.3. Теория кооперативных игр

Напомним, что в кооперативных играх используются сценарии принятия решений, в которых агенты могут заключать между собой обязывающие соглашения о сотрудничестве. В результате они могут получить дополнительную выгоду в сравнении с тем, что получили бы, действуя в одиночку.

Мы начнем с введения модели для класса **кооперативных игр**. Формально эти игры называются "кооперативными играми с переносимой полезностью в форме характеристических функций". Идея модели состоит в том, что, когда группа агентов сотрудничает, эта группа как целое получает некоторую полезность, которая затем может быть разделена между членами группы. Наша модель не сообщает, какие действия предпримут агенты, и сама структура игры не определяет, как полученное значение будет разделено (это будет обсуждаться позже).

Формально мы будем использовать формулу $G = (N, \nu)$, чтобы сказать, что кооперативная игра G определяется множеством игроков $N = \{1,\ldots, n\}$ и

►**характеристической функцией** ν, определяющей для каждого подмножества игроков $C \subseteq N$ выигрыш, который может получить эта группа игроков, если они решат действовать вместе.

Как правило, мы будем предполагать, что пустое множество игроков не достигает никакого выигрыша ($\nu(\{\}) = 0$) и что характеристическая функция неотрицательна ($\nu(C) \geq 0$ для всех C). В некоторых играх может быть сделано дополнительное допущение о том, что, работая в одиночку, игроки ничего не достигают: $\nu(\{i\}) = 0$ для всех $i \in N$.

18.3.1. Коалиционные структуры и результаты

В коалиционных играх подмножества игроков принято называть ►**коалициями** и обозначать как C. В повседневной жизни под понятием "коалиция" обычно подразумевается совокупность людей, имеющих общий мотив (например, "Коалиция за прекращение насилия с использованием огнестрельного оружия"), но здесь мы будем называть коалицией *любое* подмножество игроков. Множество всех игроков N принято называть ►**большой коалицией**.

В нашей модели каждый игрок должен сделать выбор и присоединиться только к одной коалиции (которая может быть коалицией, состоящей только из одного игрока). Следовательно, коалиции образуют некое **разделение** всего множества игроков. Мы будем называть подобное разделение ►**коалиционной структурой**. Формально коалиционная структура множества игроков N — это множество коалиций $\{C_1,\ldots, C_k\}$, такое, что

$$\begin{aligned} &C_i \neq \{\} \\ &C_i \subseteq N \\ &C_i \cap C_j = \{\} \quad \text{для всех} \quad i \neq j \in N \\ &C_1 \cup \cdots \cup C_k = N. \end{aligned}$$

Например, если имеется $N = \{1, 2, 3\}$, то существует семь возможных коалиций:

$$\{1\}, \{2\}, \{3\}, \{1, 2\}, \{2, 3\}, \{3, 1\} \text{ и } \{1, 2, 3\}$$

и пять возможных коалиционных структур:

$$\{\{1\}, \{2\}, \{3\}\}, \{\{1\}, \{2, 3\}\}, \{\{2\}, \{1, 3\}\}, \{\{3\}, \{1, 2\}\} \text{ и } \{\{1, 2, 3\}\}.$$

Мы будем использовать обозначение **CS**(N) для представления множества всех коалиционных структур для множества игроков N и обозначение $CS(i)$ для представления коалиции, к которой принадлежит игрок i.

Результат, или **исход** (*outcome*), игры определяется выборами, которые делают игроки в отношении того, какие коалиции сформировать, и выбором того, как разделить между собой выигрыш $\nu(C)$, который получит каждая коалиция. Формально при заданной кооперативной игре, определенной как (N, ν), результатом является пара (CS, **x**), состоящая из коалиционной структуры и ►**вектора выплат**

(*payoff vector*) $\mathbf{x} = (x_1, \ldots, x_n)$, где x_i — выигрыш, получаемый игроком i. Этот вектор должен удовлетворять следующему ограничению: каждая коалиция C распределяет весь свой выигрыш $v(C)$ между ее членами, т.е.

$$\sum_{i \in C} x_i = v(C) \quad \text{для всех} \quad C \in CS.$$

Например, при заданной игре $(\{1, 2, 3\}, v)$, где $v(\{1\}) = 4$ и $v(\{2, 3\}) = 10$, возможным исходом является

$$(\{\{1\}, \{2, 3\}\}, (4, 5, 5)).$$

То есть игрок 1 остается в одиночестве и получает выигрыш 4, тогда как игроки 2 и 3 образуют команду и получают выигрыш 10, который они решили разделить поровну.

Некоторые кооперативные игры имеют такую особенность, что когда две коалиции сливаются воедино, они действуют не хуже, чем если бы они продолжали действовать по отдельности. Это свойство называется ▶**супераддитивностью**. Формально игра является супераддитивной, если ее характеристическая функция удовлетворяет следующему условию:

$$v(C \cup D) \geq v(C) + v(D) \quad \text{для всех} \quad C, D \subseteq N.$$

Если игра супераддитивна, то большая коалиция получает выигрыш, который по крайней мере так же высок или выше, чем общий выигрыш, полученный любой другой коалиционной структурой. Тем не менее, как мы скоро увидим, супераддитивные игры не всегда заканчиваются большой коалицией и чаще всего по той же причине, по которой игроки не всегда приходят к оптимальному по Парето совокупному желаемому результату в игре "Дилемма заключенного".

18.3.2. Стратегия в кооперативных играх

Основным допущением в теории кооперативных игр является то, что игроки будут вырабатывать стратегические решения в отношении того, с кем они будут сотрудничать. Интуитивно понятно, что игроки не заходят сотрудничать с непродуктивными игроками, — они, естественно, будут стараться найти игроков, в совокупности дающих высокий коалиционный выигрыш. Но эти востребованные игроки будут делать собственные стратегические рассуждения. Прежде чем мы сможем описать эти рассуждения, необходимо дать некоторые дополнительные определения.

Для кооперативной игры (N, v) распределением выигрышей или ▶**дележом** (*imputation*) называют вектор выплат, удовлетворяющий следующим двум условиям:

$$\sum_{i=1}^{n} x_i = v(N),$$

$$x_i \geq v(\{i\}) \quad \text{для всех} \quad i \in N.$$

Первое условие говорит о том, что дележ должен распределять общую стоимость большой коалиции; второе условие, известное как ▶**индивидуальная рациональность**, говорит о том, что каждый игрок в коалиции должен получить выигрыш, который по крайней мере не меньше, чем тот, который он получит, работая в одиночку.

При заданном дележе $\mathbf{x} = (x_1, \ldots, x_n)$ и коалиции $C \subseteq N$ мы определяем $x(C)$ как сумму $\sum_{i \in C} x_i$, т.е. как общую сумму, выплаченную коалиции C в дележе $\mathbf{x}$.

Далее определим ▶**ядро** игры (N, v) как множество всех дележей $\mathbf{x}$, удовлетворяющих условию $x(C) \geq v(C)$ для каждой возможной коалиции $C \subset N$. Следовательно, если дележ $\mathbf{x}$ не входит в ядро, то существует некоторая коалиция $C \subset N$, такая, что $v(C) > x(C)$. Игроки в коалиции C отказались бы присоединиться к большой коалиции, потому что им выгоднее находиться в коалиции C.

Следовательно, ядро игры состоит из всех возможных векторов выплат, против которых ни одна коалиция не может возразить на том основании, что они могли бы добиться большего успеха, не вступая в большую коалицию. Таким образом, если ядро пустое, то большая коалиция просто не может сформироваться, потому что независимо от того, как большая коалиция распределит свои выплаты, какая-то меньшая коалиция откажется к ней присоединиться. Основные вычислительные вопросы, касающиеся ядра, связаны с выяснением, является ли оно пустым и находится ли в ядре некоторое конкретное распределение выплат.

Определение ядра естественным образом приводит к системе линейных неравенств, приведенных ниже. (Неизвестными являются переменные $x_1, \ldots, x_n$, а значения $v(C)$ являются константами.)

$$\begin{aligned} x_i &\geq v(\{i\}) \quad \text{для всех} \quad i \subseteq N \\ \sum_{i \in N} x_i &= v(N) \\ \sum_{i \in C} x_i &\geq v(C) \quad \text{для всех} \quad C \subseteq N \end{aligned}$$

Любое решение этих неравенств будет определять дележ в ядре. Эти неравенства можно представить в виде задачи линейного программирования, воспользовавшись фиктивной целевой функцией (например, максимизировать сумму $\sum_{i \in N} x_i$), что позволит выполнять расчет дележей с полиномиальными временными затратами по отношению к количеству неравенств. Трудность состоит в том, что это приводит к экспоненциальному количеству неравенств (по одному для каждой из 2^n возможных коалиций). Таким образом, данный подход дает алгоритм проверки не пустоты ядра, который требует экспоненциальных затрат времени. Можно ли добиться большего, чем этот результат, зависит от изучаемой игры: для многих классов кооперативных игр проблема проверки не пустоты ядра является co-NP-полной. Соответствующий пример будет приведен ниже.

Прежде чем продолжить обсуждение, давайте рассмотрим пример супераддитивной игры с пустым ядром. В игре имеются три игрока, $N = \{1, 2, 3\}$, и характеристическая функция, определяемая следующим образом.

$$v(C)=\begin{cases}1 & \text{если } |C| \geq 2\\ 0 & \text{в противном случае}\end{cases}$$

Теперь рассмотрим некоторый дележ (x_1, x_2, x_3) для этой игры. Поскольку $v(N) = 1$, должен иметь место случай, когда по крайней мере один игрок i имеет $x_i > 0$, а два других получат общую выплату, которая меньше 1. Однако эти два игрока могут улучшить ситуацию, создав коалицию без игрока i и разделив выплату 1 между собой. Но так как это утверждение верно для всех дележей, ядро игры должно быть пустым.

Понятие ядра формализует идею *стабильности* большой коалиции в том смысле, что ни одна коалиция не может нарушить ее с выгодой для себя. Однако ядро может содержать *необоснованные* дележи в том смысле, что один или несколько игроков могут почувствовать, что к ним отнеслись несправедливо. Предположим, что $N = \{1, 2\}$ и дана характеристическая функция v, определенная следующим образом:

$$v(\{1\}) = v(\{2\}) = 5$$
$$v(\{1, 2\}) = 20.$$

Здесь сотрудничество даст игрокам прибавку 10 по сравнению с вариантом, когда они работают по отдельности, поэтому интуитивно понятно, что в этом сценарии сотрудничество будет иметь смысл. Также легко увидеть, что дележ (6, 14) входит в ядро этой игры: ни один из игроков не может отказаться от коалиции, чтобы получить для себя более высокую полезность. Однако с точки зрения игрока 1 такой дележ может показаться необоснованным, потому что 9/10 прибавки от создания коалиции досталось игроку 2. Следовательно, понятие ядра показывает, *когда* большая коалиция может быть сформирована, но ничего не говорит о том, *как* распределить выплаты между игроками.

►**Подход Шепли** — это элегантное предложение, как разделить выигрыш $v(N)$ среди игроков при условии, что большая коалиция N сформирована. Этот метод был сформулирован нобелевским лауреатом Ллойдом Шепли в начале 1950-х годов. Подход Шепли предназначен для получения *справедливой* схемы распределения.

Что значит *справедливый* в этом контексте? Было бы несправедливо распределять выигрыш $v(N)$ в зависимости от цвета глаз игроков, их пола или цвета кожи. Студенты часто предполагают, что выигрыш $v(N)$ должен всегда делиться поровну, что может показаться справедливым, пока мы не примем во внимание, что это означает выдать одинаковое вознаграждение игрокам, которые вносят большой вклад, и игрокам, которые ничего не вносят. Идея Шепли состояла в предположении, что единственный справедливый способ разделить выигрыш $v(N)$ — сделать это в зависимости от того, сколько каждый игрок *внес* в создание этого выигрыша.

Сначала необходимо определить понятие ►**предельного вклада** (*marginal contribution*) игрока. Предельный вклад, который игрок i делает в коалиции C, —

это тот выигрыш, который игрок i сможет добавить (или удалить), если присоединится к коалиции C. Формально предельный вклад, который игрок i делает в коалиции C, обозначается как $mc_i(C)$:

$$mc_i(C) = \nu(C \cup \{i\}) - \nu(C).$$

Теперь первой попыткой определения схемы распределения выигрыша в соответствии с предложением Шепли — о том, что игроки должны быть вознаграждены в зависимости от их вклада — может быть выплата каждому игроку i выигрыша, который он мог бы добавить в коалиции, включающей всех остальных игроков:

$$mc_i(N - \{\mathrm{i}\}).$$

Проблема здесь в том, что такой подход неявно подразумевает, что игрок i является *последним* игроком, вступившим в коалицию. Поэтому, предложил Шепли, необходимо рассмотреть все возможные способы, которыми могла бы сформироваться большая коалиция, т.е. все возможные упорядочения игроков в N, и рассмотреть выигрыш, который игрок i добавляет к предшествующим игрокам в упорядочении. Затем игрок должен быть вознагражден тем, что ему выплачивается *средний предельный вклад, который игрок i вносит во всех возможных упорядочениях игроков, к множеству игроков, предшествующих игроку i в этом упорядочении.*

Пусть $\mathcal{P}$ обозначает все возможные перестановки (т.е. упорядочения) игроков в N; тогда членов $\mathcal{P}$ мы обозначим как p, p',... и т.д. Принимая, что $p \in \mathcal{P}$ и $i \in N$, обозначим как p_i множество игроков, предшествующих игроку i в упорядоченности p. Тогда выигрышем по Шепли для игры G является дележ $\phi(G) = (\phi_1(G),\ldots, \phi_n(G))$, определяемый следующим образом:

$$\varphi_i(G) = \frac{1}{n!} \sum_{p \in \mathcal{P}} mc_i(p_i). \qquad (18.1)$$

Все это должно убедить вас, что распределение выигрыша по Шепли следует считать разумным предложением. Но наиболее примечательным фактом является то, что это *уникальное* решение для набора аксиом, характеризующих "справедливую" схему распределения выплат. Однако прежде, чем привести определения этих аксиом, необходимо дать еще несколько определений.

Определим ▶ **нейтрального игрока** (*dummy player*) как игрока i, который никогда не добавляет никакой выгоды коалиции, т.е. $mc_i(C) = 0$ для всех $C \subseteq N - \{i\}$. Будем говорить, что два игрока, i и j, являются ▶ **симметричными игроками**, если они всегда вносят *одинаковый* вклад в коалиции, т.е. $mc_i(C) = mc_j(C)$ для всех $C \subseteq N - \{i,j\}$. И наконец, если $G = (N, \nu)$ и $G' = (N, \nu')$ — игры с одинаковым набором игроков, то игра $G + G'$ — также игра с тем же набором игроков и характеристической функцией ν'', которая определяется как $\nu''(C) = \nu(C) + \nu'(C)$.

Теперь, имея эти определения, можно дать определение аксиом справедливости, удовлетворяемых распределением выигрыша по Шепли.

- **Эффективность**. $\sum_{i \in N} \phi_i(G) = v(N)$. (Весь выигрыш должен быть распределен.)
- **Нейтральный игрок**. Если игрок i в игре G является нейтральным, то $\phi_i(G) = 0$. (Игрок, который никогда ничего не вносит, не должен ничего получать.)
- **Симметрия.** Если игроки i и j симметричны в игре G, то $\phi_i(G) = \phi_j(G)$. (Игроки, дающие одинаковый вклад, должны получать одинаковые выплаты.)
- **Аддитивность**. Выигрыш является аддитивным относительно игр. Для всех игр $G = (N, v)$ и $G' = (N, v')$ и для всех игроков $i \in N$ имеем $\phi_i(G + G') = \phi_i(G) + \phi_i(G')$.

Аксиома аддитивности по общему признанию является довольно формальной. Однако, если принять ее как требование, можно установить следующее ключевое свойство: ➡ *распределение выигрыша по Шепли является единственным способом распределения коалиционного выигрыша таким образом, чтобы удовлетворить эти аксиомы справедливости.*

18.3.3. Вычисления в кооперативных играх

С теоретической точки зрения у нас теперь есть вполне удовлетворительное решение. Но с точки зрения выполнения расчетов необходимо найти *компактный способ* представления кооперативных игр и методы *эффективного вычисления* основных концепций решения, таких как ядро и распределение по Шепли.

Очевидным представлением для характеристической функции является таблица, содержащая значения $v(C)$ для всех 2^n коалиций. Однако это невозможно для достаточно больших значений n. Был разработан ряд подходов к компактному представлению кооперативных игр, которые можно разделить на два класса в зависимости от того, являются они *полными* или нет. Полная схема представления — это схема, позволяющая представить любую кооперативную игру. Недостаток полных схем представления заключается в том, что всегда найдутся игры, которые нельзя будет представить компактно. Альтернативой является использование схемы представления, которая гарантированно компактна, но не является полной.

Сети предельных вкладов

Рассмотрим одну из таких схем представления, называемую ▶**сетями предельного вклада** (*marginal contribution nets* — MC-сети). Будем использовать слегка упрощенную версию для облегчения представления, — это упрощение делает ее неполной, тогда как исходная версия MC-сетей является полным представлением.

Идея сетей предельного вклада состоит в представлении характеристической функции игры (N, ν) в виде множества правил выигрыша коалиции, представленных в форме (C_i, x_i), где $C_i \subseteq N$ — коалиция, а x_i — числовое значение. Чтобы вычислить выигрыш коалиции C, достаточно просто просуммировать все правила (C_i, x_i), такие, что $C_i \subseteq C$. Следовательно, при известном множестве правил $R = \{(C_1, x_1), \ldots, (C_k, x_k)\}$ соответствующая характеристическая функция имеет вид

$$\nu(C) = \sum \{x_i \mid (C_i, x_i) \in R \text{ и } C_i \subseteq C\}.$$

Предположим, что имеется набор правил R, содержащий следующие три правила:

$$\{(\{1, 2\}, 5), (\{2\}, 2), (\{3\}, 4)\}.$$

Тогда, например, мы имеем следующее:

- $\nu(\{1\}) = 0$ (поскольку правила не применяются),
- $\nu(\{3\}) = 4$ (третье правило),
- $\nu(\{1, 3\}) = 4$ (третье правило),
- $\nu(\{2, 3\}) = 6$ (второе и третье правила) и
- $\nu(\{1, 2, 3\}) = 11$ (первое, второе и третье правила).

С помощью такого представления можно вычислить распределение выигрыша по Шепли за полиномиальное время. Основная идея заключается в том, что каждое правило можно понимать как самостоятельное определение игры, в которой игроки симметричны. С учетом аксиом аддитивности и симметрии Шепли можно записать, что, выигрыш по Шепли $\phi_i(R)$ игрока i в игре, связанной с набором правил R, будет равен просто

$$\phi_i(R) = \sum_{(C,x) \in R} \begin{cases} \frac{x}{|C|} & \text{если } i \in C \\ 0 & \text{в противном случае.} \end{cases}$$

Представленная здесь версия сетей предельных вкладов является *неполной* схемой представления: существуют такие игры, характеристическую функцию которых невозможно представить с использованием наборов правил в форме, описанной выше. Более общий тип сетей предельных вкладов допускает правила в виде (ϕ, x), где ϕ — формула логики высказываний по игрокам N: коалиция C удовлетворяет условию ϕ, если она соответствует удовлетворяющему присваиванию для формулы ϕ.

Такая схема является полным представлением, — в худшем случае нам потребуется правило для каждой возможной коалиции. Тем не менее с помощью этой схемы распределение выигрыша по Шепли также можно вычислить за полиномиальное время, однако это потребует включения деталей, более сложных, чем простейшие правила, описанные выше, хотя основной принцип будет тот же. (Подробности приведены в разделе "Библиографические и исторические заметки" в конце этой главы.)

Коалиционные структуры для достижения максимального общего благополучия

Если предположить, что агенты имеют общую цель, то это приводит к совершенно иному взгляду на кооперативные игры. Например, если рассматривать агентов как работников одной компании, то стратегические соображения, связанные, скажем, с формированием коалиций, соответствующих ядру, утрачивают актуальность. Вместо этого может потребоваться организовать рабочую силу (агентов) в команды таким образом, чтобы максимизировать их общую производительность. При более общем подходе задача будет заключаться в том, чтобы найти такую коалицию, которая максимизирует **общее благополучие** (*social welfare*) в системе, определяемое как сумма выигрышей отдельных коалиций. Если обозначить общее благополучие коалиционной структуры CS как $sw(CS)$, то оно будет определяться следующей формулой:

$$sw(CS) = \sum_{C \in CS} v(C).$$

Тогда оптимальная для всех коалиционная структура CS^* относительно игры G максимизирует эту величину. Нахождение оптимальной для всех коалиционной структуры является вполне очевидной вычислительной задачей, которая изучалась вне сообщества мультиагентных систем: иногда ее называют ▶**задачей разбиения множества**. К сожалению, эта задача является NP-трудной, поскольку число возможных структур коалиции растет в геометрической прогрессии в зависимости от количества игроков.

Поэтому нахождение оптимальной структуры коалиции посредством наивного исчерпывающего поиска вообще невозможно. Пользующийся популярностью подход к формированию оптимальной структуры коалиции основан на идее поиска подпространства в ▶**графе коалиционной структуры**. Объяснять эту идею удобнее всего на конкретном примере.

Предположим, имеется игра с четырьмя агентами, $N = \{1, 2, 3, 4\}$. Для этого набора агентов существует пятнадцать возможных коалиционных структур. Их можно организовать в виде графа коалиционной структуры так, как показано на рис. 18.7, где узлы на уровне ℓ графа соответствуют всем структурам коалиции с точно ℓ коалициями. Восходящее ребро в графе представляет разделение коалиции в нижнем узле на две отдельные коалиции в верхнем узле.

Например, существует ребро от коалиционной структуры $\{\{1\}, \{2, 3, 4\}\}$ к коалиционной структуре $\{\{1\}, \{2\}, \{3, 4\}\}$, потому что эта последняя коалиционная структура получается из первой путем деления коалиция $\{2, 3, 4\}$ на коалиции $\{2\}$ и $\{3, 4\}$.

Оптимальная коалиционная структура CS* находится где-то в пределах графа коалиционной структуры, поэтому, чтобы найти ее, как кажется, следовало бы оценить каждый узел в графе. Но давайте рассмотрим два нижних ряда узлов

графа — уровни 1 и 2. На этих двух уровнях появляется каждая возможная коалиция (исключая пустую коалицию). (Но, безусловно, на этих двух уровнях приведены не все возможные коалиционные структуры.) Теперь давайте предположим, что поиск возможной коалиционной структуры ограничивается *только* двумя этими уровнями, — выше по графу он не поднимется. Пусть CS' будет лучшей коалиционной структурой, которая будет найдена на этих двух уровнях, и пусть CS^* будет оптимальной коалиционной структурой для графа в целом. Далее, пусть C^* — коалиция с наибольшим выигрышем из всех возможных коалиций:

$$C^* \in \arg\max_{C \subseteq N} v(C).$$

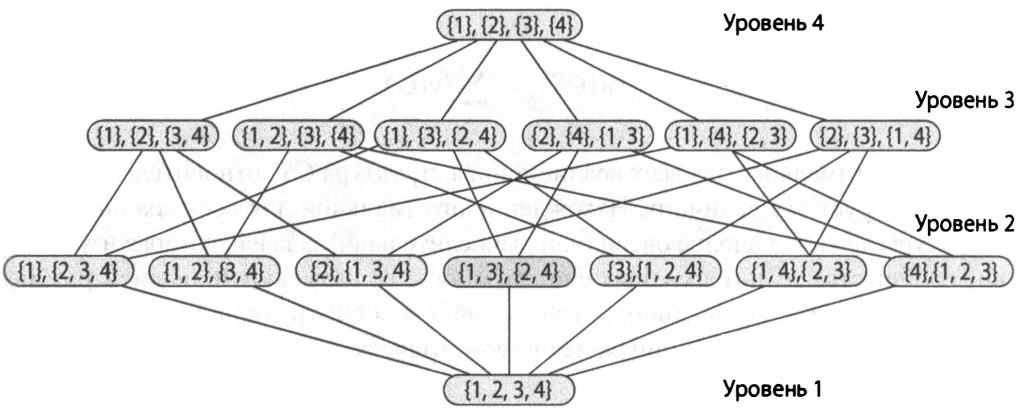


Рис. 18.7. Граф коалиционной структуры для $N = \{1, 2, 3, 4\}$. Уровень 1 включает коалиционную структуру, содержащую единственную коалицию; уровень 2 включает коалиционные структуры, содержащие по две коалиции, и т.д.

Выигрыш лучшей коалиционной структуры, которую можно найти на первых двух уровнях графа, должен быть по крайней мере таким же, как выигрыш лучшей возможной коалиции: $sw(CS') \geq v(C^*)$. Так должно быть потому, что каждая возможная коалиция появляется по меньшей мере в одной коалиционной структуре на первых двух уровнях графа. Поэтому как худший случай предположим, что $sw(CS') = v(C^*)$.

Сравним выигрыши $sw(CS')$ и $sw(CS^*)$. Поскольку $sw(CS')$ является максимально возможным выигрышем для любой структуры коалиции и в игре существует n агентов (на рис. 18.7 $n = 4$), то максимально возможное значение $sw(CS^*)$ будет равно $nv(C^*) = n \cdot sw(CS')$. Другими словами, в наихудшем случае выигрыш наилучшей коалиционной структуры можно найти на первых двух уровнях графа как $\frac{1}{n}$ часть лучшей, где n — это число агентов. Следовательно, хотя поиск на первых двух уровнях графа не гарантирует нахождение оптимальной коалиционной

структуры, он гарантированно дает нам одну, которая будет не хуже $\frac{1}{n}$ части оптимальной. На практике это часто будет намного лучше, чем нахождение оптимума.

18.4. Принятие коллективных решений

Теперь от разработки агентов мы переходим к **разработке механизма** — задаче построения правильной игры, в которую будет играть ряд агентов. Формально **механизм** состоит из следующих элементов.

1. Язык для описания множества допустимых стратегий, которые смогут использовать агенты.
2. Особый агент, называемый ▶**центром**, собирающий сообщения о выборе стратегии от агентов в игре. (Например, аукционист является центром аукциона.)
3. Правило определения результатов, известное всем агентам, которое центр использует для определения выплат каждому агенту с учетом выбранной им стратегии.

В этом разделе обсуждаются некоторые из наиболее важных механизмов.

18.4.1. Распределение задач в контрактной сети

Вероятно, старейшим и наиболее важным методом решения задач в мультиагентной среде, изучавшимся в сообществе ИИ, является ▶**протокол контрактных сетей**. Это протокол высокого уровня, предназначенный для организации процесса совместного решения задач и управления им. Как следует из его названия, метод контрактной сети был вдохновлен тем, как компании используют контракты.

Общий протокол контрактных сетей включает четыре основных этапа, как показано на рис. 18.8. Процесс начинается с того, что агент устанавливает необходимость совместных действий в отношении какой-либо проблемы. Такая необходимость может возникнуть потому, что агент не имеет возможности решить стоящую перед ним задачу самостоятельно либо совместное ее решение может оказаться лучше (быстрее, эффективнее, точнее).

Агент сообщает о возникшей проблеме другим агентам, отправив в сеть сообщение с ▶**объявлением о задаче**, а затем выступает в качестве ▶**менеджера** этой задачи на весь период ее решения. Сообщение с объявлением о задаче должно содержать достаточно информации, чтобы получатели могли судить о том, готовы ли они взяться за нее и могут ли они участвовать в торгах за это задание. Точная информация, включаемая в объявление о задаче, будет зависеть от проблемной области: это может быть некоторый код, который требуется выполнить, или логическое определение той цели, которая должна быть достигнута. Объявление о задаче

также может включать другую информацию, которая может потребоваться получателям, например срок выполнения, требования к качеству исполнения и т.д.

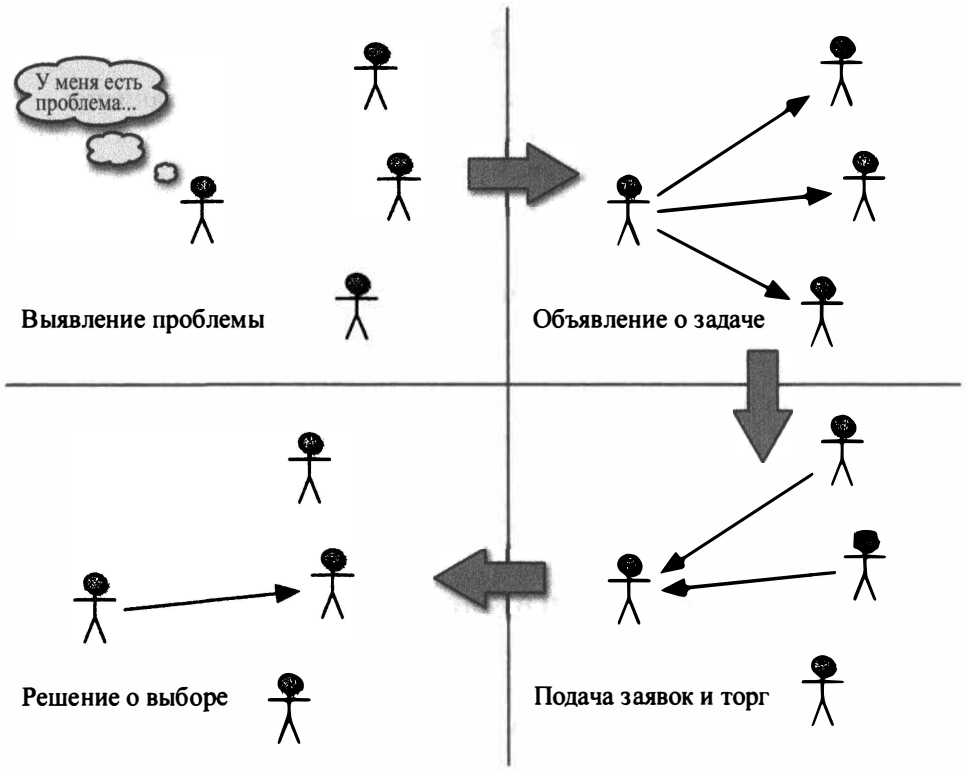


Рис. 18.8. Протокол распределения задач в контрактной сети

Когда агент получает объявление о задаче, он должен оценить ее с точки зрения его собственных возможностей и предпочтений. В частности, каждый агент должен определить, есть ли у него возможность выполнить поставленное задание, а также желает он этого или нет. На этом основании он затем может подать ▶**заявку** на выполнение поставленной задачи. Заявка, как правило, включает сведения о возможностях ее подателя, имеющих отношение к объявленной задаче, а также сроки и любые прочие условия, при которых эта задача будет выполнена.

В общем случае в ответ на одно объявление о задаче менеджеру может поступить сразу несколько заявок. Исходя из содержащейся в заявках информации, менеджер выбирает наиболее подходящего для решения данной задачи агента (или агентов). Выигравшие торг агенты будут уведомлены через сообщения о выигрыше и станут подрядчиками этой задачи, взяв на себя ответственность за нее до тех пор, пока выполнение задачи не будет завершено.

Основные вычислительные задачи, необходимые для реализации протокола контрактных сетей, можно определить следующим образом.

- *Обработка объявления о задаче*. После получения объявления о задаче агент решает, хочет ли он подать заявку на ее выполнение.
- *Обработка заявок*. При получении нескольких заявок менеджер задачи должен принять решение, какому агенту поручить ее выполнение, а затем отправить ему сообщение о выигрыше.
- *Выполнение принятых решений*. Успешные участники торгов (подрядчики) должны попытаться выполнить поставленную задачу, что может означать создание новых подзадач, которые будут предложены в сети с помощью дополнительных объявлений о задачах.

Несмотря на свою простоту (а возможно, из-за нее), контрактная сеть, вероятно, является наиболее широко применяемой и наиболее изученной структурой для совместного решения задач. Она вполне естественным образом может применяться во многих ситуациях, например каждый раз, когда вы заказываете такси в компании Uber.

18.4.2. Распределение ограниченных ресурсов с помощью аукционов

Одной из важнейших проблем в мультиагентных системах является распределение дефицитных ресурсов, но можно назвать это и просто "распределением ресурсов", поскольку на практике самых полезных ресурсов в определенном смысле всегда недостаточно. Наиболее важным подходом в этом случае является ▶**аукцион**. Простейший вариант аукциона — когда имеется единственный ресурс и много возможных **участников торгов** или ▶**претендентов**. Каждый претендент i имеет собственное значение полезности v_i для данного ресурса.

В некоторых случаях каждый претендент имеет **собственную оценку** ценности ресурса. Например, нелепый рождественский свитер может быть очень привлекательным для одного участника и совершенно бесполезным для другого.

В других случаях, как, например, на аукционе по приобретению прав на бурение в нефтеносном регионе, ресурс может иметь **общепринятую оценку** — добыча нефти в регионе позволит получить определенную сумму денег X, и все участники торгов оценивают доллар одинаково, — но существует неопределенность в отношении того, каким фактически будет это значение X на практике. Разные претенденты имеют различную информацию и, следовательно, дают разные оценки истинной стоимости ресурса. В любом случае участники торгов в конечном счете приходят к собственным оценкам v_i. Зная собственную оценку v_i, каждый претендент по ходу аукциона получает шанс в соответствующий момент или несколько моментов сделать ▶**ставку**, т.е. предложить свою цену b_i. Предложивший наивысшую ставку b_{max} выигрывает ресурс, но выплачиваемая

им цена не обязательно должна быть именно b_{max}, это зависит от используемого типа аукциона.

Самым известным типом аукциона является ▶**аукцион с растущей ставкой**,[3] или ▶**английский аукцион**, в котором центр начинает торг, объявляя минимальную (или **резервную**) ставку b_{min}. Если какой-то из претендентов готов заплатить эту сумму, центр увеличивает ставку на некоторое приращение d, предлагая новую цену $b_{min}+d$, и торг продолжается в том же духе. Аукцион заканчивается, когда больше никто не желает делать ставок, и последний претендент выигрывает ресурс, заплатив определяемую его ставкой цену.

Почему следует считать этот тип хорошим методом выбора подрядчика? Одной из целей аукциона является максимизация ожидаемого дохода для продавца. Другой целью является максимизация такого показателя, как глобальная полезность. Эти цели в определенной степени совпадают, поскольку одним из аспектов максимизации глобальной полезности является гарантия того, что аукцион выиграет тот агент, у которого собственная оценка ресурса наивысшая (а значит, он согласен заплатить больше других). Говорят, что аукцион является ▶**эффективным**, если ресурсы поступают к агенту, который ценит их больше всех других агентов. Аукцион с растущей ставкой является, как правило, и эффективным, и максимизирующим доход, но если резервная ставка будет установлена слишком высокой, претендент, который ценит ресурс в наибольшей степени, может так и не сделать ставки, а если резервная ставка будет установлена слишком низкой, то продавец может получить меньший доход.

Вероятно, самыми важными аспектами, которые должен обеспечивать выбранный тип аукциона, является воодушевление достаточного количества претендентов войти в игру, и предотвращение возможности их вхождения в ▶**сговор**. Сговор — это несправедливое или незаконное соглашение между двумя или более претендентами с целью манипулирования ценами. Участники могут вступить в сговор, как заключая секретные закулисные сделки, так и без предварительной договоренности в рамках правил для данного типа аукциона. Например, в 1999 году в Германии было выставлено на продажу с аукциона десять блоков частот электромагнитных волн из диапазона, выделенного для сотовой телефонной связи. Все блоки были представлены на торги одновременно (т.е. ставки принимались одновременно на все десять блоков) с установленным правилом, что любая очередная ставка должна быть не менее чем на 10% выше предыдущей ставки на данный блок. В аукционе участвовали только два надежных, платежеспособных участника торгов, и первый из них, корпорация Mannesman, сделала ставку по 20 млн немецких марок за блок на блоки 1–5 и по 18,18 млн за блок на блоки 6–10. Почему 18,18 млн, в чем здесь смысл? Один из менеджеров второго надежного участника торгов, компании T-Mobile, сказал, что они "интерпретировали первую заявку

[3] Слово "аукцион" происходит от латинского *augeo*, означающего "увеличивать, умножать".

корпорации Mannesman как предложение". Обе стороны понимали, что повышение ставки 18,18 млн на 10% составит 19,99 млн, поэтому ставку корпорации Mannesman можно было понимать как предложение: "Каждый из нас может получить половину всех блоков по цене 20 млн, так давайте не будем портить игру, поднимая цены". И действительно, компания T-Mobile сделала ставку по 20 млн марок на блоки частот 6–10, и на этом торги закончились.

Правительство Германии получило гораздо меньше, чем ожидало, потому что два конкурента смогли использовать механизм торгов, чтобы прийти к негласному соглашению о том, как им не конкурировать между собой. С точки зрения правительства, гораздо лучший результат мог бы быть получен за счет любого из следующих изменений в правилах аукциона: установить более высокую резервную ставку; провести аукцион с той же первичной стоимостью, но с закрытыми ставками, чтобы конкуренты не могли общаться посредством своих ставок; наконец, найти способ привлечь к торгам третьего участника. Также, возможно, само правило 10% было ошибкой в схеме работы аукциона, поскольку оно существенно упрощало точную передачу сигналов от корпорации Mannesman к компании T-Mobile.

В общем случае как функция полезности продавца, так и глобальная функция полезности выигрывают от увеличения количества участников торгов, хотя глобальная полезность может и пострадать, если принять во внимание потерю времени тех участников торгов, которые не имеют никаких шансов на победу. Один из способов привлечения к торгам большего числа участников — упростить для них сам механизм аукциона. В конце концов, если участие в аукционе требует от участников проведения слишком больших исследований или расчетов, они могут решить поискать свои деньги где-то еще.

Поэтому желательно, чтобы участники торгов имели **доминантную стратегию**. Напомним, что "доминантная" означает, что данная стратегия успешно работает против всех других стратегий, а это, в свою очередь, означает, что агент может принять ее без оглядки на все остальные стратегии. Агент с доминантной стратегией может просто делать свои ставки, не теряя времени на анализ возможных стратегий других агентов. Механизм, посредством которого агенты получают доминирующую стратегию, называется механизмом ►**защиты от манипулирования** (*strategy-proof*). Если, как это обычно бывает, эта стратегия предполагает открытие участниками торгов их истинных значений v_i, то ее называют механизмом с ►**прямым раскрытием** (*truth-revealing*) информации или **истинным**, иногда также используется термин **мотивационно совместимый** (*incentive compatible*). ►**Принцип раскрытия** гласит, что любой механизм может быть преобразован в эквивалентный механизм с прямым раскрытием информации, поэтому частью процесса разработки механизма является поиск этих эквивалентных механизмов.

Как было установлено, аукцион с растущей ставкой имеет большинство этих желательных свойств. Участник с высшим значением v_i получает ресурс за цену $b_0 + d$, где b_0 является самой высокой ставкой среди всех других агентов, а d — это

приращение от организатора аукциона.[4] У участников торгов есть простая доминирующая стратегия: продолжай делать ставки до тех пор, пока текущая цена ресурса будет ниже твоей оценки v_i. Механизм не является *истинным* в полном смысле слова, потому что победитель аукциона раскрывает только то, что его $v_i \geq b_0 + d$, т.е. у нас есть лишь нижняя граница v_i, но не точное значение оценки.

Недостатком (с точки зрения продавца) аукциона с растущей ставкой является то, что он может препятствовать конкуренции. Предположим, что в аукционе на диапазон частот сотовой связи участвует одна преуспевающая компания, которая, несомненно, имеет возможность влиять на существующих клиентов и инфраструктуру таким образом, чтобы получить бо́льшую прибыль, чем кто-либо другой. Потенциальные конкуренты могут понять, что у них нет шансов на этом аукционе с восходящей ставкой, поскольку преуспевающая компания, находясь в более выгодном положении, всегда сможет предложить ставку выше. В результате конкуренты могут вообще не входить в торги, и преуспевающая компания в конечном итоге выигрывает аукцион по резервной цене.

Другим отрицательным свойством английского аукциона являются достаточно высокие расходы на коммуникацию. Либо аукцион должен проводиться в одном подходящем помещении, либо все его участники должны иметь высокоскоростные безопасные каналы связи. В любом случае им нужно будет потратить определенное время, чтобы пройти через несколько раундов торгов.

Альтернативным механизмом, предъявляющим значительно меньше требований в отношении коммуникаций, является ►**аукцион с закрытыми предложениями**. Каждый участник делает одну ставку и сообщает об этом аукционисту, при этом другим участникам она остается неизвестной. При таком механизме уже не существует простой доминирующей стратегии. Если ваша оценка равна v_i и вы считаете, что для всех прочих агентов максимальная ставка равна b_o, то вам следует предложить $b_o + \epsilon$ при небольшом ϵ, если эта сумма будет меньше вашей оценки v_i. Следовательно, размер вашей ставки зависит от вашей оценки ставок других агентов, что потребует от вас больше работы. Кроме того, обратите внимание, что агент с наивысшей оценкой v_i не обязательно выиграет такой аукцион. Это компенсируется тем фактом, что аукцион подобного типа будет более конкурентным, снижающим преимущества преуспевающих участников торгов.

Небольшое изменение в механизме аукциона с закрытыми ставками приводит к ►**закрытому аукциону второй цены**, известному также как ►**аукцион Викри**.[5] В таких аукционах победителем является участник с самой высокой ставкой, но

[4] На самом деле существует небольшая вероятность того, что агент с самым высоким v_i не сможет получить ресурс, если $b_0 < v_i < b_0 + d$. Эту вероятность можно сделать сколь угодно малой, уменьшая приращение d.

[5] Этот тип аукционов получил свое название в честь Уильяма Викри (1914–1996), который получил за эту работу Нобелевскую премию 1996 года по экономике и умер от сердечного приступа три дня спустя.

уплатить он должен "*вторую цену*", b_o, т.е. ставку своего ближайшего конкурента. Эта простая модификация полностью устраняет все сложные размышления, неизбежные в стандартных аукционах (или аукционах **первой цены**), поскольку доминирующей стратегией в этом случае будет просто ставка, равная v_i, — этот механизм раскрывает истинное состояние дел. Обратите внимание, что полезность агента i с точки зрения его ставки b_i равна v_i, а самым лучшим предложением со стороны других агентов, b_o, будет

$$U_i = \begin{cases} (v_i - b_o) & \text{если } b_i > b_o, \\ 0 & \text{в противном случае.} \end{cases}$$

Чтобы понять, почему $b_i = v_i$ является доминирующей стратегией, обратите внимание, что если разность $(v_i - b_o)$ положительна, то любая ставка, которая приводит к выигрышу в аукционе, является оптимальной, и, в частности, ставка, равная v_i, также выиграет аукцион. С другой стороны, если разность $(v_i - b_o)$ отрицательна, то оптимальной будет любая ставка, приводящая к проигрышу в аукционе, и, в частности, ставка, равная v_i, также не выиграет аукцион. Поэтому сделать ставку, равную v_i, будет оптимальной стратегией для всех возможных значений b_o, и в действительности v_i является единственной ставкой, обладающей этим свойством. Из-за своей простоты и минимальных требований к вычислениям со стороны как продавца, так и участников торгов, аукцион Викри широко используется в распределенных системах ИИ.

Поисковые системы в Интернете каждый год проводят несколько триллионов аукционов с целью продажи мест для рекламных объявлений, выводимых вместе с результатами поиска, а сайты онлайн-аукционов обрабатывают товаров на сумму 100 млрд долл. в год, причем во всех этих случаях применяются те или иные варианты аукциона Викри. Обратите внимание, что ожидаемый доход продавца равен b_o, что является тем же самым ожидаемым результатом, который в пределе будет получен от английского аукциона, когда приращение d стремится к нулю. В действительности это очень общий результат: ▶**теорема эквивалентности доходов** утверждает, что, с небольшими оговорками, любой механизм аукциона, в котором участники имеют собственные значения v_i, известные только им (но знают распределение вероятностей, на основании которого эти значения были выбраны), будет приносить одинаковый ожидаемый доход. Этот принцип означает, что различные механизмы аукционов конкурируют между собой не в отношении генерации доходов, а скорее в отношении других их качеств.

Хотя аукцион второй цены является открывающим истинные значения v_i, было установлено, что при выставлении на продажу n товаров и выборе для аукциона механизма с $n+1$ ценой, он уже не будет обладать свойством раскрытия истинных значений v_i. Многие поисковые системы Интернета в своих аукционах используют такой механизм, когда на продажу выставляется одновременно n слотов для размещения рекламных объявлений на странице результатов поиска. Претендент,

сделавший самую высокую ставку, выигрывает верхний слот, занявший второе место, получает второй слот и т.д. Каждый победитель платит цену в размере ставки следующего за ним участника и при этом понимает, что выплата будет сделана только в том случае, когда пользователь действительно щелкнет на данном рекламном объявлении. Верхние слоты считаются более ценными, поскольку предполагается, что они с большей вероятностью будут замечены и прочитаны пользователем.

Допустим, есть три участника торгов, b_1, b_2 и b_3, и каждый из них имеет собственную оценку полезности щелчка мышью: $v_1 = 200$, $v_2 = 180$ и $v_3 = 100$. Пусть на торги выставлено $n = 2$ слота и известно, что на верхнем слоте пользователи щелкают в 5% случаев отображения, а на нижнем — в 2%. Если все участники торгов сделают ставки в соответствии со своими истинными оценками, то участник b_1 выиграет верхний слот и заплатит за него 180, при этом его ожидаемая отдача составит $(200 - 180) \times 0{,}05 = 1$. Второй слот получит участник b_2. Однако участник b_1 может заметить, что если он сделает ставку где-то в диапазоне 101–179, то уступит верхний слот участнику b_2, но выиграет второй слот с ожидаемой отдачей $(200 - 100) \times 0{,}02 = 2$. Как видите, в этом случае участник b_1 удваивает ожидаемую отдачу, сделав ставку меньше своей истинной оценки ожидаемой полезности.

В общем случае участники торгов на этом аукционе с $n + 1$ ценой должны потратить много усилий на анализ предложений других участников, чтобы определить свою лучшую стратегию, — в этом случае нет простой доминирующей стратегии.

Аггарвал и соавт. ([18], 2006) показали, что для подобной задачи с несколькими слотами существует уникальный механизм открывающего истину аукциона, в котором выигравший слот j платит цену за слот j только в размере платы те дополнительные щелчки, которые доступны для слота j и недоступны для слота $j + 1$. Выигравший аукцион платит цену за нижний слот для оставшихся щелчков. В нашем примере ставка участника b_1 будет истинной, т.е. 200, и он должен будет уплатить 180 за дополнительные $0{,}05 - 0{,}02 = 0{,}3$ щелчка на верхнем слоте, но заплатит по стоимости нижнего слота, 100, для остальных 0,02 щелчка. Таким образом, общая ожидаемая отдача для участника b_1 будет $(200 - 180) \times 0{,}03 + (200 - 100) \times 0{,}02 = 2{,}6$.

Другой пример того, когда аукционы могут вступить в игру с ИИ, — в случае, когда несколько агентов должны решить, стоит ли им скооперироваться с целью выполнения совместного плана. Хунсбергер и Гросц ([1097], 2000) показали, что подобная цель может быть эффективно достигнута с помощью аукциона, на котором агенты претендуют на роли в совместном плане.

Общее благо

Теперь давайте рассмотрим игру другого типа, в которой страны устанавливают свою политику контроля загрязнения воздуха. У каждой страны есть выбор: можно уменьшить загрязнение со стоимостью –10 пунктов за реализацию

необходимых изменений или же можно сохранить прежнюю ситуацию с загрязнением атмосферы, что даст ей чистую полезность −5 (в виде дополнительных медицинских расходов и т.д.), а также добавит по −1 пункту каждой другой стране (поскольку воздух свободно перемещается между странами). Очевидно, что доминирующей стратегией для каждой страны будет "продолжать загрязнение", но если имеется 100 стран и каждая будет следовать этой политике, то каждая страна получит общую полезность в размере −104 пункта, тогда как, если бы каждая страна снизила у себя загрязнение воздуха, все страны получили бы полезность −10 пунктов. Такая ситуация называется ►**трагедией общего достояния**: если никто не обязан платить за использование общего ресурса, то он может использоваться таким образом, что это приведет к снижению общей полезности для всех агентов. Ситуация похожа на игру "Дилемма заключенного": в игре есть другое решение, которое будет лучше для всех участников, но, похоже, рациональные агенты не смогут прийти к этому решению в рамках текущей игры.

Один из подходов к решению проблемы трагедии общего достояния состоит в замене нынешнего механизма таким, в котором с каждого из агентов будет взиматься плата за использование общего ресурса. В более общем плане необходимо убедиться, что все ►**внешние эффекты** — т.е. такие воздействия на глобальную полезность, которые не распознаются в действиях отдельных агентов — были сделаны явными.

Правильная установка цен — самая сложная часть задачи. В пределе этот подход сводится к созданию механизма, в котором каждый агент будет эффективно привлечен к максимизации глобальной полезности, но сможет достигать этой цели путем принятия локальных решений. Так, налог на выбросы углекислого газа можно рассматривать как пример механизма, стимулирующего использование общих ресурсов таким образом, что при правильном его применении глобальная полезность максимизируется.

Как оказалось, существует такая конструкция механизма, известная как механизм ►**Викри–Кларка–Гровса**, или ►**VCG**, которая обладает двумя необходимыми свойствами. Во-первых, это максимизация полезности, т.е. этот механизм максимизирует глобальную полезность, представленную как сумма полезностей для всех сторон, $\sum_i v_i$. Во-вторых, этот механизм является раскрывающим истинную значимость — доминирующей стратегией для всех агентов является раскрытие их истинной оценки значимости. У них нет необходимости в проведении сложных стратегических расчетов.

Приведем пример использования задачи распределения каких-то общих ресурсов. Предположим, что городские власти пришли к заключению о необходимости установки нескольких бесплатных беспроводных интернет-трансиверов. Однако количество имеющихся в наличии трансиверов меньше числа районов, которые хотели бы установить их у себя. Городские власти, безусловно, стремятся максимизировать общегородскую полезность новых устройств, но если они просто спросят у администрации каждого района "Насколько ценно для вас получение

бесплатного трансивера (кстати, мы отдадим их тем, кто больше всего их ценит)?", то у каждого района будет стимул дать как можно более высокую оценку. Механизм VCG сводит эту уловку на нет и стимулирует районные власти указать их истинную оценку значимости. Этот механизм работает следующим образом.

1. Центр просит каждого агента сообщить его оценку полезности по обсуждаемому вопросу v_i.
2. Центр выделяет имеющиеся ресурсы множеству победителей W, отобранному так, чтобы максимизировать $\sum_{i \in W} v_i$.
3. Для каждого победившего агента центр вычисляет, какой размер потерь, которые понесли проигравшие (каждый из которых получил 0 полезности, а мог бы получить v_j, если бы стал победителем), вызван их личным участием в игре.
4. В завершение каждый выигравший агент платит центру налог, эквивалентный этой потере.

Например, предположим, что у городских властей имеется в наличии 3 трансивера, на которые претендуют 5 городских районов, определивших ценность получения устройства соответственно как 100, 50, 40, 20 и 10. Таким образом, в множество из трех победителей W войдут районы, указавшие значения ценности 100, 50 и 40, а глобальная полезность от распределения устройств таким образом составит 190. Для каждого победителя, в случае, если бы он не участвовал в игре, победителем стал бы район, предложивший ценность 20. Следовательно, каждый победитель платит городским властям налог в размере 20.

Все победители должны быть счастливы, потому что они уплатят налог, который меньше их объявленного значения ценности, а все проигравшие также будут счастливы, поскольку объявленное ими значение ценности меньше, чем взимаемый налог. Вот почему механизм раскрывает истинные значения стоимости. В нашем примере ключевое значение равно 20: было бы иррационально делать ставку выше 20, если ваше истинное значение оценки на самом деле было ниже 20, и наоборот. Поскольку ключевое значение может быть каким угодно (оно зависит от других участников, оценки которых вам неизвестны), то это означает, что в любой ситуации будет иррационально делать любые ставки, кроме той, которая соответствует вашей истинной оценке значимости ресурса.

Механизм VCG является очень общим и может применяться ко всем видам игр, а не только к аукционам, — будет достаточно лишь слегка обобщить описанный выше механизм. Например, в ▶ **комбинаторном аукционе** на торгах доступно несколько разных лотов и каждый претендент имеет право разместить несколько ставок, каждая на некоторое их подмножество. Например, в торгах на участки земли один претендент может хотеть либо участок X, либо участок Y, но не то и другое одновременно; другому могут быть интересны любые три соседних участка и т.д. Механизм VCG может использоваться для нахождения оптимального результата

и в этом случае, хотя и с 2^N подмножествами из N лотов, за которые ведется борьба, — вычисление оптимального результата в этом случае является NP-полной задачей. С несколькими оговорками механизм VCG является уникальным: любой другой оптимальный механизм, по существу, будет ему эквивалентен.

18.4.3. Голосование

Следующим классом механизмов, которые мы рассмотрим, являются процедуры голосования, подобные тем, которые используются при принятии политических решений в демократических обществах. Направление изучения процедур голосования принадлежит к научной области, получившей название ▶**теория общественного выбора**.

Основными установками здесь являются следующие. Как обычно, у нас есть множество $N = \{1, \ldots, n\}$ агентов, которые в этом разделе будут *участниками* голосования (*voters*). Эти участники хотят принимать решения относительно множества $\Omega = \{\omega_1, \omega_2, \ldots\}$ возможных результатов. При политических выборах каждый элемент множества Ω может обозначать отдельного кандидата, победившего на выборах.

Каждый участник голосования будет иметь собственные предпочтения в отношении множества Ω. Эти предпочтения, как правило, выражаются не в количественных значениях полезности, а скорее в виде качественных сравнений, поэтому здесь мы будем использовать запись $\omega \succ_i \omega'$, означающую, что исход ω ранжируется агентом i выше исхода ω'. Так, в случае выборов с тремя кандидатами агент i может иметь предпочтения $\omega_2 \succ_i \omega_3 \succ_i \omega_1$.

Основная задача теории социального выбора состоит в том, чтобы объединить эти предпочтения с использованием ▶**функции общественного блага** (*social welfare*) для вывода ▶**упорядоченности социального предпочтения** (*social preference order*): рейтинга кандидатов от наиболее предпочтительного до наименее предпочтительного. В некоторых случаях нас может интересовать лишь ▶**социальный результат** (*social outcome*) — наиболее предпочтительный результат по группе в целом. Мы будем записывать $\omega \succ^* \omega'$ для обозначения того, что ω ранжируется выше ω' в упорядоченности социального предпочтения.

Более простая установка — это случай, когда нас не интересует получение полного рейтинга кандидатов и требуется лишь выбрать множество победителей. ▶**Функция социального выбора** (*social choice function*) принимает в качестве входных данных порядок предпочтений для каждого участника голосования и выдает в качестве выходных данных множество победителей.

Демократические общества хотят получить социальный результат, отражающий предпочтения участников голосования, т.е. избирателей. К сожалению, это не всегда бывает просто. Рассмотрим ▶**парадокс Кондорсе**, знаменитый пример, представленный в 1785 году маркизом де Кондорсе (1743–1794). Предположим,

у нас есть три результата, $\Omega = \{\omega_a, \omega_b, \omega_c\}$, и три участника голосования $N = \{1, 2, 3\}$, имеющие следующие предпочтения.

$$\begin{aligned} &\omega_a \succ_1 \omega_b \succ_1 \omega_c \\ &\omega_c \succ_2 \omega_a \succ_2 \omega_b \\ &\omega_b \succ_3 \omega_c \succ_3 \omega_a \end{aligned} \tag{18.2}$$

Теперь предположим, что необходимо выбрать одного из этих трех кандидатов на основе приведенных предпочтений. Парадокс в данном случае состоит в том, что:

- 2/3 избирателей предпочитают претендента ω_3 претенденту ω_1,
- 2/3 избирателей предпочитают претендента ω_1 претенденту ω_2,
- 2/3 избирателей предпочитают претендента ω_2 претенденту ω_3!

В результате для каждого возможного победителя можно указать другого кандидата, которого бы предпочли по крайней мере 2/3 участников голосования. Отсюда вполне очевидно, что при демократии не следует надеяться сделать *каждого* участника голосования счастливым. Это свидетельствует о том, что существуют сценарии, в которых ➔ *не зависимо от того, какой из результатов будет выбран, большинство участников голосования предпочло бы иной исход.* Возникает естественный вопрос: существует ли какая-либо "хорошая" процедура социального выбора, которая действительно отражала бы предпочтения участников голосования? Чтобы ответить на этот вопрос, сначала нужно уточнить, что имеется в виду, когда говорится, что процедура "хорошая". Ниже перечислены некоторые свойства, которыми должна была бы обладать хорошая функция общественного блага.

- *Условие Парето*. Условие Парето просто говорит о том, что если каждый участник голосования ставит ω_i выше ω_j, то $\omega_i \succ^* \omega_j$.
- *Условие победителя Кондорсе*. О результате говорят, что он является победителем Кондорсе, если большинство кандидатов предпочитают его в сравнении со всеми иными результатами. Другими словами, победитель Кондорсе — это кандидат, который победит любого другого кандидата в парных выборах. Условие победителя Кондорсе гласит, что если ω_i является победителем Кондорсе, то ω_i должен быть первым в общем рейтинге.
- *Независимость нерелевантных альтернатив* (*Independence of Irrelevant Alternatives* — IIA). Предположим, существует ряд кандидатов, включающих ω_i и ω_j, а предпочтения участников голосования таковы, что $\omega_i \succ^* \omega_j$. Теперь предположим, что один из участников голосования некоторым образом изменил свои предпочтения, но *не* в отношении взаимного рейтинга ω_i и ω_j. Условие IIA гласит, что предпочтение $\omega_i \succ^* \omega_j$ в этом случае не должно измениться.
- *Нет диктатуры*. Не должно быть так, чтобы функция общественного блага просто выводила в результат предпочтения одного избирателя и игнорировала предпочтения всех остальных избирателей.

Эти четыре условия кажутся вполне разумными, но фундаментальная теорема теории социального выбора, называемая ▶**теоремой Эрроу** (по имени ее автора, Кеннета Эрроу), утверждает, что все эти четыре условия выполнить невозможно (для случаев, когда имеется по крайней мере три возможных результата). А это означает, что для любого механизма социального выбора, на котором мы могли бы остановиться, будут иметь место некоторые ситуации (возможно, необычные или патологические), при которых неизбежно получение противоречивых результатов. Однако это вовсе не означает, что демократическое принятие решений является делом безнадежным в большинстве случаев. Мы еще не познакомились ни с одной из фактически существующих процедур голосования, так что давайте сначала рассмотрим некоторые из них.

- При наличии только двух кандидатов предпочтительным механизмом является ▶**простое большинство голосов** (стандартный метод голосования в США и Великобритании). Каждого избирателя тем или иным образом спрашивают, какого из двух кандидатов он предпочитает, и тот, кто набрал наибольшее количество голосов, является победителем.
- При более чем двух возможных результатах общепринятой является система ▶**множественного голосования**. Каждому избирателю тем или иным образом предлагают сделать свой выбор, и избранным считается тот кандидат (или кандидаты) (более чем один в случае равенства голосов), который получит наибольшее количество голосов, даже если никто из кандидатов не получит большинства. Несмотря на то что на практике этот вариант очень распространен, множественное голосование многократно подвергалось критике за непопулярность результатов. Ключевой проблемой здесь является то, что она принимает во внимание только кандидата с наивысшим рейтингом в предпочтениях каждого избирателя.
- ▶**Метод Борда** (назван по имени предложившего его Жана-Шарля де Борда, современника и противника маркиза Кондорсе) — это процедура голосования, учитывающая всю информацию об упорядоченности предпочтений избирателя. Предположим, имеется k кандидатов. Тогда для каждого избирателя i берется его упорядоченность предпочтений $\succ_i$ и возглавляющему ее кандидату присваивается k баллов, следующему за ним кандидату — $k-1$ баллов и так далее, до последнего в упорядоченности кандидата, которому дается один балл. Все набранные кандидатами баллы суммируются, образуя социальный результат $\succ^*$, в котором кандидаты упорядочены по набранному ими суммарному количеству баллов, от большего к меньшему. На практике при использовании этой системы одной из проблем является то, что избирателям предлагается выразить свои предпочтения в отношении *всех* кандидатов, тогда как некоторых из них может интересовать только определенное подмножество кандидатов.

- В системе ▶**голосования по одобрению** избиратель определяет некоторое подмножество из всех кандидатов, которых он одобряет. Победителем (или победителями) становится тот, кто одобрен большинством избирателей. Эта система часто используется, когда задача состоит в том, чтобы выбрать сразу нескольких победителей.
- В системе ▶**мгновенного повторного голосования** избиратели ранжируют всех кандидатов, и если некоторый кандидат получает большинство первых мест в этих списках, он объявляется победителем. Если такого нет, из числа избираемых исключается кандидат с наименьшим количеством первых мест в списках ранжирования избирателей. Этот кандидат будет удален из всех списков предпочтений (в результате место того избирателя, который был на первом месте, теперь займет другой кандидат), и процесс повторится. В конце концов какой-то из кандидатов получит большинство первых мест в списках ранжирования избирателей (если не окажется кандидатов с равным количеством голосов) и будет считаться избранным.
- При ▶**голосовании по правилу истинного большинства** победителем становится кандидат, который побеждает каждого другого кандидата в парном сравнении. Избирателям предлагается предоставить полный рейтинг предпочтений по всем кандидатам. Говорят, что кандидат ω выигрывает у кандидата ω', если для большинства избирателей $\omega \succ \omega'$, а не наоборот, $\omega' \succ \omega$. Эта система имеет хорошее свойство: большинство всегда соглашается с выбором победителя, но в то же время она имеет и нехорошее свойство: не каждые выборы могут состояться: например, в парадоксе Кондорсе ни один из кандидатов не получит большинства.

Стратегические манипуляции

Помимо теоремы Эрроу, еще одним важным негативным результатом в области теории общественного выбора является ▶**теорема Гиббарда–Саттертуейта**. Эта теорема касается обстоятельств, при которых избиратель способен получить выгоду за счет *искажения своих предпочтений*.

Вспомним, что функция социального выбора принимает в качестве входных данных порядок предпочтений для каждого участника голосования и выдает в качестве выходных данных множество кандидатов-победителей. Безусловно, каждый участник голосования имеет собственные истинные предпочтения, но в определении функции социального выбора ничего не говорится о том, что участники голосования обязаны *правдиво* сообщать о своих истинных предпочтениях, — они могут заявить о любых предпочтениях, которых пожелают.

В некоторых случаях для участника голосования может иметь смысл исказить свои предпочтения. Например, при множественном голосовании избиратели, полагающие, что у предпочитаемого ими кандидата нет никаких шансов на победу, могут проголосовать за свой второй выбор вместо первого. А это означает, что,

в сущности, множественное голосование представляет собой игру, в которой участники голосования должны думать стратегически (в отношении других избирателей), и это позволит им максимизировать свою ожидаемую полезность.

Тогда возникает интересный вопрос: можно ли разработать избирательный механизм, который будет обладать иммунитетом к подобным манипуляциям, — механизм, раскрывающий истинное состояние дел? Теорема Гиббарда–Саттертуейта утверждает, что это невозможно: ➨ *любая функция социального выбора, удовлетворяющая условию Парето для проблемной области более чем с двумя результатами, является либо манипулируемой, либо диктатурой.* Это означает, что для любой "разумной" процедуры социального выбора могут иметь место некоторые обстоятельства, при которых участник голосования в принципе может получить выгоду, искажая свои предпочтения. Однако эта теорема ничего не говорит о том, *как* такие манипуляции могут быть выполнены; и она не говорит о том, что подобные манипуляции вероятны *на практике*.

18.4.4. Торг

Торг или переговоры — это еще один механизм, который часто используется в повседневной жизни. Эта тема рассматривалась в теории игр, начиная с 1950-х годов, а в недавнее время торг стал задачей для автоматизированных агентов. Переговоры применяются в тех ситуациях, когда агентам необходимо достичь соглашения по вопросу, представляющему общий интерес. Агенты делают друг другу предложения (также называемые заявками или сделками) в рамках конкретных протоколов, и либо принимают, либо отклоняют каждое поступившее предложение.

Торг по протоколу чередующихся предложений

Одним из распространенных протоколов ведения переговоров является ▶**модель торга с чередующимися предложениями**. Для простоты вновь примем наличие только двух агентов. Торг проходит в виде последовательности раундов. Агент A_1 начинает торг в раунде 0, делая некоторое предложение. Если агент A_2 принимает это предложение, торг завершается и оно реализуется. Если агент A_2 отклоняет предложение, то переговоры переходят к следующему раунду. На этот раз предложение делает агент A_2, а агент A_1 решает, принять его или отклонить, и т.д. Если переговоры никогда не прекращаются (поскольку агенты отклоняют каждое поступающее предложение), то результат определяется как ▶**конфликт** (*conflict deal*). Удобное упрощающее допущение заключается в том, что оба агента предпочитают достичь результата — любого результата — за конечное время, не зацикливаясь на бесконечно длительном конфликте.

Чтобы проиллюстрировать модель переговоров с чередующимися предложениями, воспользуемся сценарием **дележа пирога**. Основная идея состоит в том,

что имеется некоторый ресурс ("пирог") с ценностью 1, который может быть разделен на две части, по одной для каждого агента. Таким образом, предложением в этом сценарии является пара значений $(x, 1-x)$, где x представляет часть пирога, которую получит агент A_1, а $1-x$ — оставшуюся часть пирога, которая достанется агенту A_2. Следовательно, пространством возможных сделок (►**переговорным множеством**) является

$$\{(x, 1-x) : 0 \le x \le 1\}.$$

Итак, как агенты должны вести переговоры в данной ситуации? Чтобы понять ответ на этот вопрос, сначала рассмотрим несколько более простых случаев.

Сначала предположим, что допускается *только один раунд* переговоров. Следовательно, агент A_1 делает предложение, а затем агент A_2 может его либо принять (и сделка будет реализована), либо отклонить (и тогда будет реализован конфликт). Фактически это ►**ультиматум** (*ultimatum game*), поскольку в этой ситуации очевидно, что агент A_1 — **тот, кто ходит первым** — обладает всей властью. Предположим, что агент A_1 предлагает отдать ему весь пирог, т.е. предлагает сделку $(1, 0)$. Если агент A_2 отклонит ее, то будет реализован конфликт. Поскольку по определению агент A_2 предпочел бы получить 0, но не конфликт, ему будет выгоднее принять предложение. Безусловно, агент A_1 не может получить лучшего результата, чем забрать себе весь пирог. Следовательно, эти две стратегии — агент A_1 предлагает отдать ему весь пирог и агент A_2 принимает это предложение — образуют равновесие Нэша.

Следующим рассмотрим случай, когда разрешается ровно *два* раунда переговоров. Теперь власть перешла в другие руки: агент A_2 может просто отклонить первое предложение и тем самым превратить эту игру в игру с одним раундом, в которой уже он, агент A_2, делает первый ход, а следовательно, может получить весь пирог. В общем случае, если количество раундов является фиксированным числом, то тот агент, который делает последний ход, получает весь пирог.

А теперь перейдем к общему случаю, когда количество раундов *не ограничено*. Предположим, что агент A_1 использует следующую стратегию:

Всегда предлагать (1, 0) и всегда отклонять любое встречное предложение.

А что для агента A_2 является лучшим ответом на это предложение? Если агент A_2 будет постоянно его отклонять, то агенты будут вести переговоры вечно, что по определению является худшим результатом для агента A_2 (а также и для агента A_1). Следовательно, для агента A_2 не может быть лучшего варианта, чем принять первое предложение, сделанное агентом A_1. И вновь это будет равновесие по Нэшу. Но что если агент A_1 использует иную стратегию, например такую:

Всегда предлагать (0,8, 0,2) и всегда отклонять любое встречное предложение.

Используя аналогичные аргументы, можно показать, что для этого предложения или ➔ *для любой возможной в переговорном множестве сделки* $(x, 1-x)$, *существует пара равновесных стратегий Нэша, заключающихся в том, что результатом будет соглашение по сделке, предложенной на первом этапе переговоров.*

Нетерпеливые агенты

Проведенный выше анализ говорит о том, что если не накладывать ограничений на количество раундов, то количество равновесий Нэша будет бесконечным. Поэтому давайте добавим следующее допущение:

> Для любого результата x и этапов времени t_1 и t_2, где $t_1 < t_2$, оба агента предпочтут результат x в момент времени t_1 результату x в момент времени t_2.

Другими словами, агенты являются **нетерпеливыми**. Стандартный подход к нетерпению состоит в использовании **коэффициента обесценивания** γ_i (см. раздел 17.1.1) для каждого агента ($0 \leq \gamma_i < 1$). Предположим, что в какой-то момент переговоров агент i предлагает кусок пирога размером x. Ценность куска пирога размером x в момент времени t будет равна $\gamma_i^t x$. Таким образом, на первом этапе переговоров (время 0) ценность будет равна $\gamma_i^0 x = x$ и в любой последующий момент времени ценность того же предложения будет уже меньше. Большее значение коэффициента γ_i (ближе к 1) подразумевает больше терпения, а меньшее его значение означает меньше терпения.

Прежде чем проанализировать общий случай, сначала рассмотрим торг при фиксированном количестве этапов времени, как это было выше. На первом раунде анализ будет таким же, как тот, который был приведен выше: переговоры сводятся к ультиматуму. Однако на *втором* раунде ситуация меняется, поскольку стоимость пирога уменьшается в соответствии с коэффициентом обесценивания γ_i. Допустим, что агент A_2 отклонил начальное предложение агента A_1 и теперь он может получить весь пирог благодаря ультиматуму во втором раунде. Но *ценность* всего пирога будет уже меньше: для агента A_2 она составит только γ_2. Агент A_1 может учесть этот факт, сделав предложение $(1-\gamma_2, \gamma_2)$, и агент A_2 вполне может его принять, поскольку на текущий момент времени A_2 не может получить лучший результат, чем γ_2. (Если вас беспокоит, что произойдет при ничейном результате, то просто сделайте предложение равным $(1-(\gamma_2+\epsilon), \gamma_2+\epsilon)$ для небольшого значения ϵ.)

Можно сделать вывод, что две стратегии — агент A_1 делает предложение $(1-\gamma_2, \gamma_2)$ и агент A_2 его принимает — находятся в равновесии Нэша. В соответствии с этим протоколом терпеливые игроки (те, у которых γ_2 больше) смогут получить более крупные куски пирога и в этом случае терпение действительно является добродетелью.

Теперь рассмотрим общий случай, когда нет ограничений на количество раундов. Как и в случае торгов с единственным раундом, агент A_1 может подготовить

предложение, которое агент A_2 должен будет принять, поскольку оно даст ему максимально возможную сумму с учетом коэффициентов обесценивания. Как было установлено, в такой ситуации агент A_1 получит

$$\frac{1-\gamma_2}{1-\gamma_1\gamma_2},$$

а агент A_2 получит остаток.

Переговоры в отношении распределения заданий

В этом разделе рассматриваются переговоры в ▶ **проблемных областях, предполагающих выполнение заданий**. В таких проблемных областях требуется выполнить некоторое множество заданий, каждое из которых первоначально назначается определенному множеству агентов. Эти агенты могут извлечь выгоду путем проведения переговоров о том, кто будет выполнять какое задание. Например, предположим, что одни задания выполняются на токарном станке, а для выполнения других необходим фрезерный станок, и что любому агенту, работающему на станке, приходится нести значительные расходы, связанные с его настройкой. Тогда одному агенту будет иметь смысл предложить другому следующее: "Мне в любом случае придется выполнить настройку фрезерного станка, тогда, может, я выполню все твои задания, для которых необходим фрезерный станок, а ты выполнишь все мои задания, для которых нужен токарный станок?"

В отличие от сценария с торгом в данном случае переговоры начинаются при уже известном первоначальном распределении заданий, поэтому, если агенты не смогут договориться в отношении каких-либо предложений по перераспределению работ, то они просто будут выполнять задания T_i^0, которые были назначены им изначально.

Чтобы упростить ситуацию, еще раз предположим, что имеется только два агента. Пусть T — это множество всех заданий и пусть (T_1^0, T_2^0) — это начальное распределение заданий между двумя агентами в момент времени 0. Каждое задание из множества T может быть назначено только одному из агентов. Далее предположим, что определена функция затрат c, которая для любого множества заданий T' дает положительное действительное число $c(T')$, определяющее те затраты, которые любой из агентов понесет при выполнении заданий множества T'. (Будем полагать, что размер затрат зависит только от самого задания и не связан с выполняющим его агентом.) Функция затрат является монотонной: увеличение количества заданий никогда не приводит к уменьшению затрат, а затраты при отсутствии заданий — т.е. когда агент ничего не делает — равны нулю: $c(\{\}) = 0$. В качестве примера предположим, что затраты на настройку фрезерного станка равны 10, а выполнение на нем каждого задания требует затрат, равных 1, поэтому затраты на выполнение на фрезерном станке двух заданий составят 12, а на выполнение множества из пяти заданий потребует затрат, равных 15.

Предложение вида (T_1, T_2) означает, что агент i обязуется выполнить множество заданий T_i с затратами $c(T_i)$. Полезность U для агента i — это сумма, которую он получит в случае принятия им предложения, которая вычисляется как разность между затратами на выполнение этого нового множества заданий и затратами на выполнение первоначально назначенного ему множества заданий:

$$U_i((T_1, T_2)) = c(T_i) - c(T_i^0).$$

Предложение (T_1, T_2) будет ▶**индивидуально рациональным**, если $U_i((T_1, T_2)) \geq \geq 0$ для обоих агентов. Если сделка не является индивидуально рациональной, то по крайней мере один агент может добиться большего успеха, просто выполняя задания, которые ему были назначены изначально.

Переговорное множество для проблемных областей, предполагающих выполнение заданий (при условии рациональности агентов), представляет собой множество предложений, которые являются как индивидуально рациональными, так и оптимальными по Парето. В этой ситуации нет смысла делать индивидуально нерациональное предложение, которое будет отклонено, как и делать некое предложение, когда существует лучшее предложение, повышающее полезность одного агента и не ухудшающее полезностей всех остальных агентов.

Монотонный протокол уступок

Протокол переговоров, который будет рассмотрен применительно к проблемным областям, предполагающим выполнение заданий, носит название ▶**монотонный протокол уступок**. Правила этого протокола следующие.

- Переговоры проводятся в виде серии раундов.
- В первом раунде оба агента *одновременно* предлагают сделку, $D_i = (T_1, T_2)$ из переговорного множества. (Это важное отличие от правила чередующихся предложений, использовавшегося выше.)
- Договоренность будет достигнута, если два агента предлагают сделки D_1 и D_2 соответственно, такие, что либо $U_1(D_2) \geq U_1(D_1)$, либо $U_2(D_1) \geq U_2(D_2)$, т.е. если один из агентов установит, что сделка, предложенная другим, по крайней мере так же хороша или даже лучше, чем сделка, которую предложил он. Если соглашение достигнуто, то правило определения договорной сделки будет следующим. Если предложение каждого агента соответствует или превосходит предложение другого агента, то одно из них выбирается случайным образом. Если только одно предложение превосходит предложение другого или совпадает с ним, то это соглашение и будет соглашением сделки.
- Если соглашение не было достигнуто, то переговоры вступают в новый раунд одновременных предложений. В раунде $t + 1$ каждый агент должен либо повторить свое предложение из предыдущего раунда, либо сделать

▶ **уступку** — внести предложение, которое для другого агента будет предпочтительнее (т.е. будет иметь для него более высокую полезность).

- Если ни один из агентов не делает уступок, переговоры завершаются и оба агента реализуют конфликт, выполняя задания, которые были назначены им первоначально.

Поскольку переговорное множество конечно, агенты не могут вести переговоры бесконечно: либо они достигнут соглашения, либо будет иметь место раунд, в котором ни один из них не сделает уступки. Однако этот протокол не гарантирует, что соглашение будет достигнуто *быстро*: поскольку количество возможных сделок определяется как $O(2^{|T|})$, можно предположить, что переговоры будут продолжаться в течение раундов, количество которых экспоненциально зависит от количества назначенных заданий.

Стратегия Жозена

До сих пор еще ничего не было сказано о том, как участники переговоров могли бы или должны вести себя при использовании монотонного протокола уступок в проблемных областях, предполагающих выполнение заданий. В таких случаях одной из возможных стратегий является ▶ **стратегия Жозена**.

Идея стратегии Жозена заключается в измерении *готовности агента рисковать конфликтом*. Интуитивно понятно, что агент будет более склонен к риску конфликта, если разница в полезности между его текущим предложением и конфликтной ситуацией невелика. В этом случае агент мало что потеряет, если переговоры зайдут в тупик и возникнет конфликт, и поэтому готов пойти на риск конфликта в большей степени, чем пойти на уступки. И наоборот, если разница в полезности между текущим предложением агента и конфликтной ситуацией высока, то агент может многое потерять в результате конфликта и, следовательно, будет менее склонен к риску его возникновения, а значит, в большей степени готов делать уступки.

Готовность агента i к риску возникновения конфликта в раунде t можно оценить следующим образом.

$$risk_i^t = \frac{\text{Полезность агента } i\text{, утраченная за счет уступки и принятия предложения агента } j}{\text{Полезность агента } i\text{, утраченная за счет отказа от уступок и возникновения конфликта}}$$

До тех пор пока не будет достигнуто соглашение, значение готовности агента $risk_i^t$ будет находиться в пределах от 0 до 1. Более высокие значения (ближе к 1) показывают, что агент i мало что потеряет в случае конфликта и поэтому более охотно примет риск его возникновения.

Стратегия Жозена утверждает, что первое предложение каждого агента должно представлять собой сделку из переговорного множества, максимизирующую его

собственную полезность (их может быть больше, чем одна). Далее, в раунде t переговоров агентом, который должен сделать уступку, будет тот агент, у которого значение $risk_i^t$ меньше, т.е. агент, который потеряет больше от возникновения конфликта, когда ни один из агентов не сделает уступки.

Следующий вопрос, на который нужно найти ответ, — "Сколько нужно уступить?" Ответ, предоставляемый стратегией Жозена, формулируется так: "Достаточно, чтобы изменить баланс риска конфликта для другого агента". Иначе говоря, агент должен пойти на *минимальную* уступку, которая заставит другого агента уступить в следующем раунде.

В стратегии Жозена есть еще одно, последнее, уточнение. Предположим, что в какой-то момент оба агента имеют *равный* риск. Тогда, согласно стратегии, они оба должны уступить. Но, зная это, один из агентов может потенциально "дезертировать", отказавшись от уступки и тем самым получив преимущество. Всякий раз, когда возникает ситуация одинакового риска, чтобы избежать возможности уступок с обеих сторон, стратегия дополняется предложением агентам "бросить монету", чтобы решить, кто из них должен будет уступить.

При такой стратегии достигнутое соглашение будет оптимальным по Парето и индивидуально рациональным. Однако, поскольку размер переговорного множества экспоненциально зависит от количества заданий, следование этой стратегии может потребовать $O(2^{|T|})$ раз вычислять функцию затрат на каждом этапе переговоров. И в завершение укажем, что стратегия Жозена (дополненная правилом броска монеты) находится в равновесии Нэша.

Резюме

- **Мультиагентное** планирование необходимо, если в проблемной среде присутствуют и другие агенты, с которыми можно сотрудничать или соперничать. Существует возможность построения совместных планов, но они должны быть дополнены некоторой формой координации, если два агента соглашаются на выполнение совместного плана.
- **Теория игр** описывает рациональное поведение агентов в ситуациях, когда несколько агентов взаимодействуют между собой. Теория игр определяет способы принятия мультиагентных решений, а теория принятия решений — способы принятия решений единственным агентом.
- **Концепции решений** в теории игр предназначены для описания рациональных результатов игр — результатов, которые могут быть достигнуты, если каждый агент будет действовать рационально.
- **Теория некооперативных игр** предполагает, что агенты должны принимать свои решения независимо друг от друга. **Равновесие Нэша** является наиболее важной концепцией решения в теории некооперативных игр. Равновесие Нэша — это профиль стратегии, в котором ни один из агентов не

имеет стимула отклониться от указанной ему стратегии. Разработаны методы для работы с повторяющимися и последовательными играми.

- **Теория кооперативных игр** рассматривает ситуации, в которых агенты могут принимать обязывающие соглашения для формирования коалиций с целью сотрудничества. Концепции решения в кооперативной игре направлены на формальное определение, какие из коалиций будут стабильными (**ядро**) и как можно честно разделить выигрыш, полученный коалицией (**подход Шепли**).
- Для некоторых важных классов задач мультиагентного принятия решений были разработаны специализированные методы: контрактная сеть для организации совместного выполнения заданий; аукционы для эффективного распределения ограниченных ресурсов; переговоры и торги для достижения соглашений по вопросам, представляющим общий интерес; процедуры голосования для суммирования и обобщения предпочтений.

Библиографические и исторические заметки

Примечательно, что исследователи в области искусственного интеллекта даже не начинали сколько-нибудь серьезно рассматривать вопросы, связанные с взаимодействием агентов между собой, вплоть до начала 1980-х годов, а теория мультиагентных систем все еще не определилась как отдельная дисциплина в области ИИ и десятилетие спустя. Тем не менее идеи, намекавшие на мультиагентные системы, выдвигались еще в 1970-х годах. Например, в своей весьма влиятельной теории *общества мышления* Марвин Мински ([1583], 1986; [1584], 2007) выдвинул предположение, что сознание человека организовано в виде совокупности агентов. У Дуга Лената были похожие идеи в рамках структуры, которую он назвал BEINGS (Ленат [1383], 1975). В 1970-х годах, опираясь на свою докторскую работу по системе PLANNER, Карл Хьюитт предложил модель вычислений в виде взаимодействующих агентов, получившую название **модель акторов**, — она стала одной из фундаментальных моделей в параллельных вычислениях (Хьюитт [1017], 1977; Агха [19], 1986).

Предыстория области мультиагентных систем тщательно задокументирована в сборнике статей, озаглавленном *Readings in Distributed Artificial Intelligence* (Бонд и Гассер [245], 1988). Этот сборник предваряет подробное изложение ключевых исследовательских задач в мультиагентных системах, которое остается чрезвычайно актуальным и сегодня, спустя более тридцати лет после его публикации. Для ранних исследований в области мультиагентных систем характерна тенденция предполагать, что все агенты в системе действуют в общих интересах при единственном разработчике. В настоящее время этот подход рассматривается как особый случай более общей мультиагентной среды — этот частный случай известен как ▶ **кооперативное распределенное решение задач**. Ключевой системой того

времени была *Distributed Vehicle Monitoring Testbed* (DVMT), разработанная под руководством Виктора Лессера в Университете штата Массачусетс (Лессер и Коркилл [1390], 1988). Система DVMT моделировала сценарий, в котором комплект агентов с географически разнесенными акустическими датчиками совместно занимался отслеживанием движения транспортных средств.

Современная эра исследований в области мультиагентных систем началась в конце 1980-х годов, когда окончательно утвердилось представление, что агенты с разными предпочтениями являются нормой как в ИИ, так и в обществе; с этой точки зрения теория игр начала восприниматься как основная методология изучения таких агентов.

Мультиагентное планирование приобрело широкую популярность лишь в последние годы, хотя в действительности оно имеет довольно долгую историю. Конолиге ([1279], 1982) формализовал мультиагентное планирование средствами логики первого порядка, тогда как Педналт ([1762], 1986) дал ее описание в стиле STRIPS. Понятие совместного намерения, очень важное, если агенты должны выполнять совместный план, появилось в работе по актам коммуникации (Коэн и Перро [460], 1979; Коэн и Левек [458], 1990; Коэн и др. [459], 1990). Бутилье и Брафман ([266], 2001) показали, как адаптировать планирование с частичным упорядочением к многоакторной обстановке. Брафман и Домшлак ([287], 2008) разработали алгоритм многоакторного планирования, сложность которого возрастает лишь линейно в зависимости от количества акторов — при условии, что степень связывания (частично измеряется шириной дерева графа взаимодействий между агентами) является ограниченной.

Мультиагентное планирование труднее всего осуществлять в тех случаях, когда имеются противоборствующие агенты. Как сказал Жан-Поль Сартр ([1979], 1960), "В футбольном матче все осложняется присутствием другой команды". Генерал Дуайт Д. Эйзенхауэр говорил: "При подготовке к битве я всегда обнаруживал, что планы бесполезны, но планирование необходимо"; это означает, что важно иметь условный план или стратегию, а не ожидать, что успех может быть достигнут за счет выполнения безусловного плана.

Тема распределенного и мультиагентного обучения с подкреплением (*reinforcement learning* — RL) не рассматривается в этой главе, но в настоящее время представляет большой интерес. В области распределенного обучения с подкреплением целью является разработка методов, с помощью которых несколько координирующих свои действия агентов учатся оптимизировать общую функцию полезности. Например, можно ли разработать методы, посредством которых отдельные субагенты навигации и обхода препятствий для роботов смогут совместно создать комбинированную систему управления, которая будет глобально оптимальной? Некоторые основные результаты в этом направлении уже были получены (Гестрин и др. [932], 2002; Рассел и Зимдарс [1949], 2003). Основная идея заключается в том, что каждый субагент обучает собственную Q-функцию (своего рода функцию полезности; см. раздел 22.3.3) на основании собственного потока

вознаграждений. Например, компонент навигации робота может получать награды за продвижение к цели, а компонент обхода препятствий, в свою очередь, будет получать отрицательные награды за каждое столкновение. Каждое глобальное решение максимизирует сумму Q-функций, и весь процесс сходится к глобально оптимальным решениям.

Корни теории игр можно проследить в предложениях, сделанных в XVII веке Кристианом Гюйгенсом и Готфридом Лейбницем в отношении научного и математического изучения конкурентных и кооперативных взаимодействий между людьми. На протяжении XIX века несколько ведущих экономистов создали простые математические примеры для анализа определенных вариантов конкурентных ситуаций.

Первые формальные результаты в теории игр были достигнуты Цермелом ([2426], 1913), который за год до этого предложил свою форму минимаксного поиска для игр, хотя и некорректную. Эмиль Борель ([252], 1921) ввел понятие смешанной стратегии. Джон фон Нейман ([2281], 1928) доказал, что каждая игра с нулевой суммой для двух игроков имеет максиминное равновесие в смешанных стратегиях и четко определенную стоимость. Сотрудничество фон Неймана с экономистом Оскаром Моргенштерном привело к публикации в 1944 году книги *Theory of Games and Economic Behavior*, определяющей книги по теории игр. Ее публикация из-за нехватки бумаги в военное время откладывалась до тех пор, пока член семьи Рокфеллеров лично не субсидировал ее издание.

В 1950 году в возрасте 21 года Джон Нэш опубликовал свои идеи в отношении равновесий в общих играх (с ненулевой суммой). Данное им определение равновесного решения, хотя и предвосхищенное в работе Курно ([481], 1838), впоследствии стало известно как равновесие Нэша. В 1994 году, после длительной задержки, вызванной шизофренией, которой он страдал начиная с 1959 года, Нэш был награжден Нобелевской премией по экономике (наряду с Рейнхартом Зельтеном и Джоном Харшаньи). Равновесие Байеса–Нэша было описано Харшаньи ([971], 1967) и дополнительно обсуждалось Кадане и Ларки ([1164], 1982). Некоторые вопросы использования теории игр для управления агентами были освещены Бинмором ([219], 1982). Ауманн и Бранденбургер ([91], 1995) показали, как можно достичь различных равновесий в зависимости от знаний, которые имеет каждый из игроков.

Игра "Дилемма заключенного" была изобретена в качестве упражнения в классе Альбертом В. Такером в 1950 году (на примере Мерилла Флуда и Мелвина Дрешера) и тщательно изучена Аксельродом ([92], 1985) и Поундстоном ([1817], 1993). Повторяющиеся игры были предложены Люсом и Райффой ([1463], 1957), а Абреу и Рубинштейн в работе [9] (1988) обсуждают использование конечных автоматов для повторяющихся игр, точнее — **машин Мура**. Учебник Майлата и Самуэльсона ([1477], 2006) концентрируется на повторяющихся играх.

Игры с неполной информацией в развернутой форме были предложены Куном ([1319], 1953). Последовательная форма для игр с неполной информацией была

изобретена Романовским ([1908], 1962) и независимо Коллером и соавт. ([1263], 1996). В статье Коллера и Пфеффера [1264] (1997) предоставлено более читабельное введение в эту область и описана система представления и решения последовательных игр.

Использование абстракции для уменьшения дерева игры до размеров, которые позволяют найти решение методами Коллера, было предложено Биллингсом и соавт. ([214], 2003). Впоследствии усовершенствованные методы нахождения равновесного решения позволили решать абстракции с 10^{12} состояниями (Гилпин и др. [860], 2008; Зинкевич и др. [2448], 2008). Боулинг и соавт. в работе [273] (2008) показали, как можно использовать выборку по важности для получения лучшей оценки выигрыша для стратегии. Во и соавт. в [2302] (2009) установил, что подход с использованием абстракции является уязвимым в отношении накопления систематических ошибок в равновесных решениях: для одних игр он работает, а для других — нет. Браун и Сендхольм в [321] (2019) показали, что по крайней мере в случае игры в Техасский холдем с несколькими игроками эти уязвимости могут быть преодолены при наличии достаточной вычислительной мощности. Они использовали 64-ядерный сервер, работавший в течение 8 дней, для расчета базовой стратегии в их программе Pluribus. С помощью этой стратегии программа Pluribus смогла победить в турнире с чемпионами среди игроков-людей.

Теория игр и марковский процесс принятия решений (MDP) объединяются в теории марковских игр, иначе называемых стохастическими играми (Литтман [1419], 1994; Ху и Веллман [1082], 1998). Независимо от Беллмана, Шепли в работе [2042] (1953) фактически описал алгоритм итерации по значениям, но его результаты не были широко признаны, возможно потому, что они были представлены в контексте марковских игр. Эволюционная теория игр (Смит [2095], 1982; Вейбулл [2303], 1995) рассматривает дрейф стратегии во времени: если стратегия вашего оппонента меняется, то как вам следует на это реагировать?

К учебникам по теории игр с подходом с экономической точки зрения можно отнести книги Майерсона ([1653], 1991), Фуденберга и Тироля ([796], 1991), Осборна ([1717], 2004) и Осборна и Рубинштейна ([1718], 1994). Как учебники по теории игр с подходом с точки зрения ИИ можно рекомендовать книги Нисана и соавт. ([1693], 2007) и Лейтона-Брауна и Шохама ([1399], 2008). Полезный обзор по теме мультиагентного принятия решений представлен в статье Сендхольма [1972] (1999).

Мультиагентное обучение с подкреплением отличается от распределенного обучения с подкреплением присутствием агентов, которые не могут координировать свои действия (за исключением явных актов коммуникаций) и которые не могут совместно использовать одну и ту же функцию полезности. Следовательно, мультиагентное обучение с подкреплением имеет дело с последовательными задачами теории игр или **марковскими играми**, как это определенно в главе 17. Осложнения вызывает тот факт, что, пока агент обучается тому, как победить стратегию своего противника, противник изменяет свою стратегию, чтобы победить агента. Таким образом, данная среда является **нестационарной** (см. раздел 13.4.2).

Литтман в [1419] (1994) отметил эту трудность при представлении первых алгоритмов обучения с подкреплением для марковских игр с нулевой суммой. Ху и Веллман в [1083] (2003) предложили алгоритм Q-обучения для игр с общей суммой, который сходится, когда равновесие по Нэшу является уникальным. В случае, когда существует множество равновесий, определить понятие сходимости оказывается не так просто (Шохам и др. [2055], 2004).

Игры содействия были впервые введены под названием **кооперативное обратное обучение с подкреплением** Хэдфилдом-Менеллом и соавт. в работе [942] (2017). Малик и соавт. ([1480], 2018) представил эффективный решатель POMDP, разработанный специально для игр содействия. Эти игры связаны с классом ▶**игр "начальник–агент"** в экономике, в которых начальник (например, работодатель) и агент (например, работник) должны найти взаимовыгодное соглашение несмотря на наличие совершенно разных предпочтений. Основные различия между этими классами игр состоят в том, что, во-первых, робот не имеет никаких собственных предпочтений, и во-вторых, робот не имеет уверенности в отношении предпочтений человека, которые необходимо оптимизировать.

Кооперативные игры впервые были изучены фон Нейманом и Моргенштерном ([2282], 1944). Понятие ядра было предложено Дональдом Гиллисом ([858], 1959), а понятие подхода Шепли было введено Ллойдом Шепли в работе [2041] (1953). Хорошее введение в математику кооперативных игр можно найти в статье Пелега и Зюдхолтера ([1767], 2002). Простые игры в целом подробно обсуждаются Тейлором и Цвикером в [2185] (1999). Как введение в вычислительные аспекты теории кооперативных игр можно рекомендовать работу Халкиадакиса и соавт. [385] (2011).

Начиная с работы Дэнга и Пападимитриу [606] (1994), в последние три десятилетия было разработано много схем компактного представления кооперативных игр. Самой влиятельной из этих схем является модель сетей предельного вклада, предложенная Йонгом и Шохэмом в [1112] (2005). Подход к формированию коалиции, который описывался в этой главе, был разработан Сендхольмом и соавт. ([1973], 1999). Обзор состояния дел в этой области предоставлен Рахваном и соавт. ([1845], 2015).

Протокол контрактных сетей был предложен Рейдом Смитом в докторской диссертации, которую он защищал в Стэнфордском университете в конце 1970-х годов (Смит [2098], 1980). Этот протокол, кажется, представляется настолько естественным, что регулярно изобретается заново и до наших дней. Его экономические основы были изучены Сендхольмом ([1974], 1993).

Аукционы и разработка механизмов являются одной из ведущих тем в области компьютерных наук и искусственного интеллекта уже в течение нескольких десятилетий (см. работы Нисана ([1692], 2007) — в качестве господствующей точки зрения в компьютерных науках; Кришны ([1310], 2002) — в качестве введения в теорию аукционов; Крамтона и соавт ([492], 2006) — в качестве сборника важнейших статей по вычислительным аспектам аукционов).

В 2007 году Нобелевская премия по экономике была вручена Гурвицу, Маскину и Майерсону "за достижения в закладывании основ теории разработки механизмов" (Гурвич [1102], 1973). Трагедия общего достояния — мотивирующая проблема в этой области — впервые анализировалась Уильямом Ллойдом ([1437], 1833), но получила свое имя и привлекла общественное внимание благодаря статье Гаретта Хардина ([965], 1968). Рональд Коуз представил теорему о том, что если ресурсы находятся в частной собственности и если транзакционные издержки достаточно низкие, то ресурсы могут управляться эффективно (Коуз [454], 1960). Однако он также указал, что на практике транзакционные издержки высоки, поэтому данная теорема неприменима, и необходимо искать другие решения, помимо приватизации и рынка. В книге *Governing the Commons* Элинор Остром ([1720], 1990) описал решение этой задачи, основанное на передаче управления контролем над ресурсами в руки местных специалистов, имеющих наилучшие знания о текущей ситуации. И Коуз, и Остром получили Нобелевскую премию по экономике за свои работы.

Определение принципа раскрытия принадлежит Майерсону ([1652], 1986), а теорема об эквивалентности доходов была независимо разработана Майерсоном ([1651], 19811981) и Райли и Самуэльсоном ([1881], 1981). Два экономиста, Милгром ([1574], 1997) и Клемперер ([1241], 2002), в своих работах описывают целый спектр многомиллиардных аукционов, в которых они принимали участие.

Проектирование механизмов используется в мультиагентном планировании (Хунсбергер и Грош [1097], 2000; Стоун и соавт. [2139], 2009) и составлении графиков (Рассенти и др. [1857], 1982). Вариан в работе [2264] (1995) дает краткий обзор связей с литературой по компьютерным наукам, а Розеншейн и Злоткин ([1915], 1994) представляют подборку книжного размера материалов о приложениях распределенного ИИ. Близкие работы о распределенном ИИ вышли под разными названиями, включая "Коллективный разум" (Тумер и Волперт [2231], 2000; Сегаран [2020], 2007) и "Рыночный контроль" (Клеарвотер [451], 1996). С 2001 года проводится ежегодный конкурс торговых агентов (*Trading Agents Competition — TAC*), на котором агенты пытаются получить максимальную прибыль на серии аукционов (Веллман и др. [2320], 2001; Арунахалам и Садех [81], 2005).

Литература о социальном выборе огромна и охватывает все от философских рассуждений о природе демократии и до узкоспециализированного анализа конкретных процедур голосования. Работа Кэмпбелла и Келли [360] (2002) представляет собой хорошую отправную точку в этой литературе. Справочник *Handbook of Computational Social Choice* содержит ряд статей, посвященных темам и методам исследований в этой области (Брандт и др. [290], 2016). Теорема Эрроу перечисляет желаемые свойства системы голосования и доказывает, что достичь их всех одновременно невозможно (Эрроу [78], 1951). Дасгупта и Маскин в [529] (2008) показали, что голосование по правилу истинного большинства (не множественное голосование и не голосование по упорядоченности предпочтений) является

наиболее надежной системой голосования. Вычислительная сложность манипулирования выборами впервые была изучена Бартольди и соавт. ([137], 1989).

Авторы с трудом справились с работой по переговорам в мультиагентном планировании. Дурфи и Лессер в работе [664] (1989) обсуждают, как задания могут быть распределены между агентами путем переговоров. Краус и соавт. ([1305], 1991) описывают систему имитации игры "Дипломатия" — настольной игры, требующей ведения переговоров, формирования коалиции и нечестности. Стоун в [2136] (2000) показывает, как агенты могут сотрудничать в команде в конкурентной, динамичной, частично наблюдаемой среде игры роботов в футбол. В более поздней статье [2136] (2003) Стоун анализирует две конкурентные мультиагентные среды — соревнования RoboCup по игре роботов в футбол и TAC, а также соревнования торговых агентов, построенные на аукционах — и приходит к заключению, что вычислительная неподатливость нашего текущего, теоретически обоснованного подхода привела к тому, что многие мультиагентные системы приходится разрабатывать специальными методами. Сарит Краус разработал целый ряд агентов, которые могут вести переговоры с людьми и другими агентами (Краус [1304], 2001). Монотонный протокол уступок для автоматизированного ведения переговоров был предложен Джеффри С. Розеншейном и его студентами (Розеншейн и Злоткин [1915], 1994). Протокол чередующихся предложений был разработан Рубинштейном ([1928], 1982).

Учебники по мультиагентным системам включают книги Вейса ([2305], 2000), Юнга ([2410], 2004), Влассиса ([2279], 2008), Шохама и Лейтон-Брауна ([2054], 2009), а также Уолдриджа ([2377], 2009). Основной конференцией по мультиагентным системам является *International Conference on Autonomous Agents and MultiAgent Systems* (*AAMAS*); также издается журнал с тем же названием. Ассоциация вычислительной техники (*ACM*) регулярно проводит конференции *Conference on Electronic Commerce* (*EC*), а также публикует множество соответствующих статей, в частности в области алгоритмов аукционов. Главным журналом по теории игр является *Games and Economic Behavior*.

Упражнения

18.1. Покажите, что любое равновесие доминантной стратегии представляет собой равновесие Нэша, но обратное утверждение неверно.

18.2. В детской игре "Камень–бумага–ножницы" каждый игрок одновременно раскрывает свой выбор — камень, бумагу или ножницы. Бумага обертывает камень, камень тупит ножницы, а ножницы режут бумагу. В расширенной версии этой игры ("Камень–бумага–ножницы–огонь–вода") приняты дополнительные условия: огонь побеждает камень, бумагу и ножницы; камень, бумага и ножницы побеждают воду; вода побеждает огонь. Запишите матрицу вознаграждений и найдите решение со смешанной стратегией для этой игры.

18.3. Найдите решение для игры в чет-нечет на трех пальцах.

18.4. В игре "Дилемма заключенного" рассмотрите случай, когда после каждого раунда Али и Бо с вероятностью X встретятся вновь. Предположим, что оба игрока выбирают стратегию ВЕЧНАЯ КАРА (когда каждый выбирает действие *refuse*, пока другой игрок не выберет действие *testify*, в ответ на что выбирается также действие *testify*). Предположим, что до этого момента ни один из игроков еще не выбирал действия *testify*. Какова ожидаемая будущая общая выплата за выбор действия *testify* в ответ на действие *refuse*, когда $X = 0{,}2$? Что изменится, если $X = 0{,}05$? Для какого значения X ожидаемая будущая общая выплата будет одной и той же, независимо от выбора этим игроком действия *testify* или *refuse* в текущем раунде?

18.5. Приведенная ниже матрица вознаграждений, впервые проанализированная Бернштейном в [1920] (1996), показывает, какую игру ведут между собой политики (сокращенно — Pol) и руководство Федеральной резервной системы (сокращенно — Fed).

	Fed: сузить	Fed: ничего не делать	Fed: расширить
Pol: сузить	$F=7, P=1$	$F=9, P=4$	$F=6, P=6$
Pol: ничего не делать	$F=8, P=2$	$F=5, P=5$	$F=4, P=9$
Pol: сузить	$F=3, P=3$	$F=2, P=7$	$F=1, P=8$

Конгрессмены могут расширять или сужать налоговую стратегию, а руководители Федеральной резервной системы могут расширять или сужать бюджетную стратегию (и, безусловно, каждая сторона может выбрать вариант ничего не изменять.) Кроме того, каждая из этих сторон имеет определенные предпочтения в отношении того, кто и что должен делать, поскольку ни те, ни другие не хотят вызвать недовольство населения. Показанные выше вознаграждения представляют собой ранги упорядочения от 9 до 1, т.е. от первого варианта до последнего. Найдите равновесие Нэша для этой игры в рамках чистых стратегий. Является ли это решение оптимальным согласно критерию Парето?

18.6. Голландский аукцион похож на английский аукцион, но вместо того, чтобы начинать торги с низкой цены и увеличивать ставки, на голландском аукционе торг начинается с высокой цены и аукционер постепенно снижает цену, пока какой-то участник торгов не захочет ее принять. (Если цена принимается

одновременно несколькими участниками, один из них произвольным образом выбирается в качестве победителя.) Говоря формально, аукционер начинает с цены p и последовательно снижает ее с шагом d до тех пор, пока хотя бы один из участников торгов не примет цену. Предполагая, что все участники торгов действуют рационально, верно ли утверждение, что при сколь угодно малом шаге d голландский аукцион всегда приводит к тому, что участник торгов с наибольшим значением собственной ценности лота получает этот лот? Если это так, математически покажите, почему. Если нет, то объясните, как может случиться, что участник с наивысшей собственной оценкой лота его не получит.

18.7. Представьте механизм аукциона, который ничем не отличается от аукциона с возрастающей ставкой за исключением того, что в конце победитель, предложивший ставку b_{max}, платит только $b_{max}/2$, а не b_{max}. Если предположить, что все агенты рациональны, каков ожидаемый доход аукциониста по этому механизму по сравнению со стандартным аукционом с возрастающей ценой?

18.8. Команды в Национальной хоккейной лиге исторически получали 2 очка за победу в игре и 0 очков — за поражение. Если игра заканчивалась вничью, назначался дополнительный период, и если он также не заканчивался победой одной из команд, то игра считалась сыгранной вничью и каждая команда получала по 1 очку. Но должностные лица лиги чувствовали, что команды слишком консервативно играют в дополнительном периоде (чтобы избежать проигрыша), и игра была бы интереснее, если бы дополнительный период чаще заканчивался победой. Поэтому в 1999 году должностные лица провели эксперимент со следующей схемой механизма: правила были изменены, и теперь команде, которая проигрывала в дополнительном периоде, давалось 1 очко, а не 0, — остальные правила оставались без изменений: 2 очка за победу и 1 очко за ничью.

а) Был ли хоккей игрой с нулевой суммой до изменения правил? А после их изменения?

б) Предположим, что в определенный момент времени t в игре хозяева поля имеют вероятность p победить в основное время, вероятность $0{,}78 - p$ — проиграть и вероятность 0,22 — свести игру к ничьей и получить дополнительный период, где у них будет вероятность q победить, вероятность $0{,}9 - q$ — проиграть и вероятность 0,1 — закончить игру вничью. Приведите уравнения для определения ожидаемого результата игры для хозяев поля и для команды противника.

в) Представим, что для этих двух команд будет допустимо юридически и этично заключить договор, в котором они согласны свести игру к ничьей в основное время, а затем, в дополнительном периоде, обе команды в полную силу будут стараться выиграть. При каких условиях, с точки зрения значений p и q, для обеих команд будет разумно заключить такое соглашение?

г) В отчете за 2005 год сообщалось о том, что с момента изменения правила процент игр с победой в дополнительном периоде увеличился на 18,2%, но при этом процент игр с дополнительным периодом также вырос на 3,6%. Что можно предположить о возможности сговора или консервативной игре после изменения правил?

ПРИЛОЖЕНИЕ A

Математические основы

A.1. Анализ сложности и нотация $O()$

Специалистам в области компьютерных наук часто приходится решать задачу сравнения алгоритмов для определения того, насколько быстро они действуют или сколько памяти для них требуется. Для решения этой задачи предусмотрены два подхода. Первым из них является применение эталонных тестов (►**benchmarking**) — прогон реализующих алгоритмы программ на компьютере и измерение их быстродействия в секундах и потребления памяти в байтах. Безусловно, в конечном итоге нас действительно интересуют именно такие практические характеристики, но эталонное тестирование может оказаться неудовлетворительным просто потому, что оно слишком специфично: в нем измеряется производительность конкретной программы, написанной на конкретном языке, выполняемой на конкретном компьютере с конкретным компилятором и с конкретными входными данными. К тому же на основании единственного результата, полученного с помощью эталонного тестирования, очень трудно предсказать, насколько успешно этот алгоритм будет действовать в случае использования другого компилятора, компьютера или набора данных. Второй подход предполагает математический ►**анализ алгоритмов**, не зависящий от их конкретной реализации и входных данных. Именно этот подход будет обсуждаться ниже.

A.1.1. Асимптотический анализ

Подход, основанный на математическом анализе алгоритмов и не зависящий от их конкретной реализации и входных данных, мы рассмотрим на примере приведенной ниже программы, вычисляющей сумму последовательности чисел.

```
function SUMMATION(последовательность) returns число
  сумма ← 0
  for i = 1 to LENGTH(последовательность) do
     сумма ← сумма + последовательность[i]
  return сумма
```

Первый этап анализа состоит в том, что создается определенное абстрактное представление входных данных, позволяющее найти какой-то параметр или

параметры, характеризующие объем входных данных. В рассматриваемом примере объем входных данных можно охарактеризовать с помощью такого параметра, как длина последовательности, которую мы обозначим как n. На втором этапе необходимо определить абстрактное представление реализации и найти какой-то критерий, отражающий продолжительность выполнения алгоритма, но не привязанный к конкретному компилятору или компьютеру. Применительно к программе SUMMATION этим критерием может служить количество строк выполняемого кода; кроме того, данный критерий может быть более детализированным и измеряющим количество сложений, присваиваний, обращений к элементам массивов, а также ветвлений, выполняемых в этом алгоритме. В любом случае будет получена характеристика общего количества шагов, выполняемых алгоритмом, как функция от объема входных данных. Обозначим эту характеристику как $T(n)$. Если за основу берется количество строк кода, то в данном примере $T(n) = 2n + 2$.

Если бы все программы были такими же простыми, как SUMMATION, то область анализа алгоритмов не заслуживала бы названия научной. Но исследования в этой области существенно усложняются из-за наличия двух проблем. Первая проблема заключается в том, что редко удается найти параметр, подобный $T(n)$, который бы полностью характеризовал количество шагов, выполняемых при прогоне алгоритма. Вместо этого чаще всего можно лишь вычислить данный показатель для наихудшего случая $T_{\text{worst}}(n)$ или для среднего случая $T_{\text{avg}}(n)$. Причем, для вычисления среднего показателя необходимо, чтобы аналитик принял какие-то обоснованные предположения в отношении распределения, характеризующего набор входных данных.

Вторая проблема состоит в том, что алгоритмы обычно не поддаются точному анализу. В этом случае приходится прибегать к аппроксимации. В нашем случае, например, можно было бы сказать, что быстродействие алгоритма SUMMATION характеризуется величиной $O(n)$, имея в виду, что быстродействие этого алгоритма измеряется величиной, пропорциональной n, — возможно, за исключением нескольких небольших значений n. Более формально это определение можно представить с помощью следующей формулы.

$$T(n) \text{ есть } O(f(n)), \text{ если } T(n) \leq kf(n) \text{ для некоторого } k, \text{ для всех } n > n_0$$

Перейдя к использованию нотации $O(\ )$, мы получаем возможность воспользоваться так называемым ►**асимптотическим анализом**. В рамках этого подхода можно, например, безоговорочно утверждать, что если n асимптотически приближается к бесконечности, то алгоритм, характеризующийся показателем $O(n)$, проявляет себя лучше по сравнению с алгоритмом $O(n^2)$, тогда как единственное число, полученное с помощью эталонного тестирования, не может служить обоснованием подобного утверждения.

Нотация $O(\ )$ позволяет создать абстрактное представление, в котором не учитываются постоянные коэффициенты, благодаря чему она становится более простой в использовании, но менее точной, чем нотация $T(\ )$. Например, в конечном

итоге алгоритм $O(n^2)$ всегда будет считаться худшим по сравнению с алгоритмом $O(n)$, но если бы эти два алгоритма характеризовались значениями $T(n^2+1)$ и $T(100n+1000)$, то фактически алгоритм $O(n^2)$ был бы лучше при $n<110$.

Несмотря на этот недостаток, асимптотический анализ представляет собой наиболее широко используемый инструмент анализа алгоритмов. Этот метод можно считать достаточно точным, поскольку в процессе анализа создается абстрактное представление и для точного количества операций (поскольку игнорируется постоянный коэффициент k), и для точного содержимого входных данных (поскольку рассматривается исключительно их объем n); благодаря чему анализ становится вполне осуществимым с использованием математических методов. Система обозначений $O()$ представляет собой хороший компромисс между точностью и простотой анализа.

A.1.2. Изначально сложные и недетерминированные полиномиальные задачи

Анализ алгоритмов и нотация $O()$ позволяют рассуждать об эффективности конкретного алгоритма. Однако эти методы не позволяют определить, может ли существовать лучший алгоритм для рассматриваемой задачи. В области ▶**анализа сложности** исследуются задачи, а не алгоритмы. Первая широкая градация в этой области проводится между задачами, которые могут быть решены за время, измеряемое полиномиальным соотношением, и задачами, которые не могут быть решены за время, измеряемое полиномиальным соотношением, независимо от того, какой алгоритм для этого используется. Класс полиномиальных задач — т.е. задач, которые могут быть решены за время $O(n^k)$ для некоторого k), — обозначается как ▶**P**. Эти задачи иногда называют "простыми", поскольку данный класс содержит задачи, имеющие такую продолжительность выполнения, как $O(\log n)$ и $O(n)$. Но он содержит и задачи с затратами времени $O(n^{1000})$, поэтому определение "простая" не следует понимать слишком буквально.

Другим важным классом является ▶**NP** (Nondeterministic Polynomial) — класс недетерминированных полиномиальных задач. Задача относится к этому классу, если существует алгоритм, позволяющий выдвинуть гипотезу о возможном решении, а затем проверить правильность этой гипотезы с помощью полиномиальных затрат времени. Идея такого подхода состоит в том, что если бы можно было воспользоваться сколь угодно большим количеством процессоров, чтобы проверить одновременно все гипотезы, или оказаться крайне удачливым и всегда с первого раза находить правильную гипотезу, то NP-трудные задачи стали бы P-трудными задачами. Одним из самых важных нерешенных вопросов в компьютерных науках является то, будет ли класс NP эквивалентным классу P, если нельзя воспользоваться бесконечным количеством процессоров или способностью находить правильную гипотезу с первого раза. Большинство специалистов в области компьютерных наук согласны с тем, что $P \neq NP$, иными словами, что NP-задачи

являются изначально трудными и для них не существует алгоритмов с полиномиальными затратами времени. Но это утверждение так и не было доказано.

Ученые, пытающиеся найти ответ на вопрос о том, эквивалентны ли классы P и NP, выделили подкласс класса NP, называемый ►**NP-полными** задачами. В этой формулировке слово "полный" означает "являющийся наиболее ярким представителем" и поэтому указывает на самые трудные задачи из класса NP. Было доказано, что либо все NP-полные задачи принадлежат к классу P, либо ни одна из них к нему не относится. Таким образом, данный класс представляет определенный теоретический интерес, но он важен также с точки зрения практики, поскольку известно, что многие серьезные задачи являются NP-полными. В качестве примера можно указать задачу установления выполнимости: если дано высказывание логики высказываний, то есть ли такой вариант присваивания истинностных значений символам высказывания, при котором оно становится истинным? Если не произойдет чудо и не совпадут друг с другом классы P и NP, то нельзя будет найти алгоритм, который позволяет решить *все* задачи установления выполнимости за полиномиальное время. Но исследователей в области искусственного интеллекта в большей степени интересует то, существуют ли алгоритмы, действующие достаточно эффективно при решении *типичных задач*, выбранных с помощью заранее заданного распределения; как было показано в главе 7, существуют алгоритмы наподобие WalkSAT, которые действуют вполне успешно при решении многих задач.

Класс ►**NP-сложных** задач состоит из тех задач, которые сводятся (за полиномиальное время) ко всем проблемам в NP, поэтому, если вы решили любую NP-сложную задачу, вы сможете решить все проблемы в NP. Все NP-полные задачи являются NP-сложными, но есть некоторые NP-сложные проблемы, которые даже сложнее, чем NP-полные.

Класс ►**co-NP** является комплементарным (дополнительным) по отношению к классу NP в том смысле, что для каждой задачи принятия решения из класса NP существует соответствующая задача в классе co-NP, на которую может быть дан положительный или отрицательный ответ, противоположный ответу на задачу класса NP. Известно, что класс P является подмножеством и NP, и co-NP, кроме того, считается, что к классу co-NP относятся некоторые задачи, не входящие в класс P. Такие задачи называются ►**co-NP-полными** и являются самыми трудными задачами в классе co-NP.

Класс #P (произносится как "шарп P" или "диез P") представляет собой множество задач подсчета количества вариантов, соответствующих задачам принятия решения из класса NP. Задачи принятия решения имеют однозначный (положительный или отрицательный) ответ. Примером задачи такого типа является следующая: "Существует ли решение для данной формулы 3-SAT?" Задачи подсчета количества вариантов имеют целочисленный ответ, например: "Сколько решений существует для данной формулы 3-SAT?" В некоторых случаях задача подсчета количества вариантов намного труднее по сравнению с соответствующей задачей

принятия решения. Например, принятие решения о том, имеет ли двухдольный граф идеальное сочетание пар, может быть выполнено за время $O(VE)$ (где V — количество вершин; E — количество ребер графа), тогда как задача подсчета того, какое количество идеальных сочетаний пар имеется в этом двухдольном графе, является #P-полной. Это означает, что она не менее трудна, чем любая задача из класса #P, и поэтому по меньшей мере столь же трудна, как и любая задача NP.

Другим классом является класс задач PSPACE. К нему относятся задачи, для которых требуется объем пространства, определяемый полиномиальной зависимостью, даже при их прогоне на недетерминированной машине. Считается, что PSPACE-трудные задачи решаются хуже NP-полных задач, но не исключено, что в ходе дальнейших исследований может быть установлено, что класс NP эквивалентен классу PSPACE и класс P эквивалентен классу NP.

А.2. Векторы, матрицы и линейная алгебра

В математике ▶ **вектор** определяется как элемент векторного пространства, но мы будем использовать более конкретное определение: *вектор* — это упорядоченная последовательность значений. Например, в двухмерном пространстве могут быть определены такие векторы, как $\mathbf{x} = \langle 3, 4 \rangle$ и $\mathbf{y} = \langle 0, 2 \rangle$. В этом приложении соблюдаются обычные соглашения об обозначении векторов с помощью полужирных символов, хотя некоторые авторы отмечают имена векторов с помощью стрелок ($\vec{x}$) или знаков надчеркивания ($\overline{y}$). Элементы вектора обозначаются с помощью подстрочных индексов: $\mathbf{z} = \langle z_1, z_2, \ldots, z_n \rangle$. К сожалению, в этой книге синтезированы работы из многих областей исследований, где упорядоченные последовательности элементов могут называться по-разному ("векторы", "списки" или "кортежи"), в соответствии с чем для них используются и разные нотации: $\langle 1, 2 \rangle$, $[1, 2]$ или $(1, 2)$.

Двумя фундаментальными операциями над векторами являются векторное сложение и скалярное умножение. Векторное сложение $\mathbf{x} + \mathbf{y}$ — это поэлементная сумма: $\mathbf{x} + \mathbf{y} = \langle 3 + 0, 4 + 2 \rangle = \langle 3, 6 \rangle$, а скалярное умножение — это операция умножения каждого элемента вектора на некоторую константу: $5\mathbf{x} = \langle 5 \times 3, 5 \times 4 \rangle = \langle 15, 20 \rangle$.

Длина вектора обозначается как $|\mathbf{x}|$ и вычисляется путем извлечения квадратного корня из суммы квадратов его элементов: $|\mathbf{x}| = \sqrt{(3^2 + 4^2)} = 5$. Точечное произведение (называемое также скалярным произведением) двух векторов, $\mathbf{x} \cdot \mathbf{y}$, представляет собой сумму произведений их соответствующих элементов; иными словами, $\mathbf{x} \cdot \mathbf{y} = \sum_i x_i y_i$, или в данном конкретном случае: $\mathbf{x} \cdot \mathbf{y} = 3 \times 0 + 4 \times 2 = 8$.

Векторы часто интерпретируются как направленные отрезки прямых (подобные стрелкам) в n-мерном евклидовом пространстве. В таком случае операция сложения векторов эквивалентна совмещению конца одного вектора с началом другого, а точечное произведение $\mathbf{x} \cdot \mathbf{y}$ эквивалентно выражению $|\mathbf{x}| \cdot |\mathbf{y}| \cdot \cos\theta$, где θ — угол между векторами $\mathbf{x}$ и $\mathbf{y}$.

► **Матрица** представляет собой прямоугольный массив значений, упорядоченных по строкам и столбцам. Ниже показана матрица **A** размером 3×4.

$$\begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \mathbf{A}_{1,3} & \mathbf{A}_{1,4} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \mathbf{A}_{2,3} & \mathbf{A}_{2,4} \\ \mathbf{A}_{3,1} & \mathbf{A}_{3,2} & \mathbf{A}_{3,3} & \mathbf{A}_{3,4} \end{pmatrix}$$

Первый подстрочный индекс в обозначении $\mathbf{A}_{i,j}$ определяет строку, а второй — столбец. В языках программирования $\mathbf{A}_{i,j}$ часто записывается как `A[i,j]` или `A[i][j]`.

Сумма двух матриц определяется путем сложения соответствующих элементов, поэтому $(\mathbf{A} + \mathbf{B})_{i,j} = \mathbf{A}_{i,j} + \mathbf{B}_{i,j}$. (Если матрицы **A** и **B** имеют разные размеры, то их сумма не определена.) Можно также определить операцию умножения матрицы на скаляр: $(c\mathbf{A})_{i,j} = c\mathbf{A}_{i,j}$. Операция умножения матриц (получения произведения двух матриц) является более сложной. Произведение **AB** определено, только если матрица **A** имеет размеры $a \times b$, а матрица **B** имеет размер $b \times c$ (т.е. вторая матрица имеет количество строк, совпадающее с количеством столбцов первой матрицы). Результатом этой операции является матрица с размерами $a \times c$. Если матрицы имеют допустимые размеры, то результат их умножения является следующим:

$$(\mathbf{AB})_{i,k} = \sum_j \mathbf{A}_{i,j}\mathbf{B}_{j,k}.$$

Умножение матриц не является коммутативным даже для квадратных матриц: $\mathbf{AB} \neq \mathbf{BA}$ в любом случае. Однако эта операция является ассоциативной: $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$. Обратите внимание, что точечное произведение векторов может быть выражено в терминах транспонирования и умножения матриц: $\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^{\mathsf{T}}\mathbf{y}$.

► **Единичная матрица I** имеет элементы $\mathbf{I}_{i,j}$, равные 1, если $i = j$, и равные 0 в противном случае. Она обладает таким свойством, что $\mathbf{AI} = \mathbf{A}$ для всех матриц **A**.

► **Транспонирование** — это специфическая для матриц операция (записывается как $\mathbf{A}^{\mathsf{T}}$), заключающаяся в превращении строк матрицы в столбцы и обратно, или, более формально, $\mathbf{A}^{\mathsf{T}}{}_{i,j} = \mathbf{A}_{j,i}$. Если матрица **A** квадратная, то ее ► **обратная матрица** $\mathbf{A}^{-1}$ определяется как такая матрица, для которой $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$. Для ► **вырожденной матрицы** обратной матрицы не существует, а для невырожденной матрицы обратная матрица может быть вычислена за время $O(n^3)$.

Матрицы используются для решения систем линейных уравнений за время $O(n^3)$, которое необходимо для инвертирования матрицы, составленной из коэффициентов уравнений. Рассмотрим следующую систему уравнений, для которой требуется найти решение в терминах переменных x, y и z.

$$\begin{aligned} +2x + y - z &= 8 \\ -3x - y + 2z &= -11 \\ -2x + y + 2z &= -3 \end{aligned}$$

Эту систему уравнений можно представить в виде матричного уравнения $\mathbf{Ax} = \mathbf{b}$, где:

$$\mathbf{A} = \begin{pmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 8 \\ -11 \\ -3 \end{pmatrix}.$$

Чтобы решить уравнение $\mathbf{Ax} = \mathbf{b}$, достаточно умножить обе его стороны на обратную матрицу $\mathbf{A}^{-1}$, в результате чего получим $\mathbf{A}^{-1}\mathbf{Ax} = \mathbf{A}^{-1}\mathbf{b}$, что можно упростить до $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. Проинвертировав матрицу $\mathbf{A}$ и умножив результат на вектор $\mathbf{b}$, получим искомый ответ:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ -1 \end{pmatrix}.$$

Еще несколько замечаний по поводу используемых в книге обозначений.

1. Мы используем запись $\log(x)$ для представления натуральных логарифмов $\log_e(x)$.
2. Мы используем запись $\text{argmax}_x f(x)$ для ссылок на значение x, при котором значение функции $f(x)$ является максимальным.

А.3. Распределения вероятностей

Вероятность — это мера, заданная на множестве событий, которая удовлетворяет трем приведенным ниже аксиомам.

1. Мера каждого события находится в пределах от 0 до 1. Это утверждение записывается как $0 \leq P(X = x_i) \leq 1$, где X — случайная переменная, представляющая событие, а x_i — возможные значения X. Обычно принято обозначать случайные переменные прописными буквами, а их значения — соответствующими строчными.
2. Мера всего множества равна 1, а это означает, что $\sum_{i=1}^{n} P(X = x_i) = 1$.
3. Вероятность объединения взаимоисключающих событий равна сумме вероятностей отдельных событий, т.е. $P(X = x_1 \vee X = x_2) = P(X = x_1) + P(X = x_2)$, гдесобытия x_1 и x_2 являются взаимоисключающими.

Вероятностная модель состоит из пространства выборок взаимоисключающих возможных результатов вместе с вероятностной мерой для каждого результата. Например, в модели погоды на завтра результатами могут быть *sunny* (солнечно), *cloudy* (облачно), *rainy* (дождь) и *snowy* (снег). Подмножество этих результатов представляет собой событие. Например, событие выпадения осадков — это подмножество $\{rainy, snowy\}$.

Мы будем использовать запись $\mathbf{P}(X)$ для обозначения вектора значений $\langle P(X = x_1),\ldots, P(X = x_n)\rangle$. Мы также используем запись $P(x_i)$ в качестве сокращения для $P(X = x_i)$ и $\sum_x P(x)$ в качестве сокращения для $\sum_{i=1}^{n} P(X = x_i)$.

Условная вероятность $P(B|A)$ определяется как $P(B \cap A)/P(A)$. Переменные A и B являются условно независимыми, если $P(B|A) = P(B)$ (или, равным образом, если $P(A|B) = P(A)$).

Непрерывные переменные имеют бесконечное количество значений, и если распределение вероятностей этих значений не характеризуется наличием пиковых значений в отдельных точках, то вероятность любого отдельно взятого значения равна 0. Поэтому имеет смысл говорить о значении вероятности в пределах некоторого диапазона. Это реализуется с помощью ▶**функции плотности вероятности**, которая имеет немного иной смысл по сравнению с дискретной функцией вероятности. Поскольку вероятность $P(X = x)$ — т.е. вероятность, которую X имеет при точном значении x — равна нулю, мы будем использовать другую меру: отношение вероятности того, что X попадает в интервал вокруг x, к ширине этого интервала, измеряемой в пределе приближения ширины интервала к нулю. Итак, функция плотности вероятности $P(X = x)$ определяется как

$$P(x) = \lim_{dx \to 0} P(x \le X \le x + dx)/dx.$$

Функция плотности вероятности должна быть неотрицательной при всех x и соответствовать следующему требованию:

$$\int_{-\infty}^{\infty} P(x)dx = 1.$$

Мы также можем определить ▶**кумулятивное распределение** $F_X(x)$, которое является вероятностью того, что случайная величина будет меньше x. Кумулятивная функция распределения определяется следующим образом:

$$F_X(x) = P(X \le x) = \int_{-\infty}^{x} P(u)du.$$

Следует отметить, что функция плотности вероятности измеряется в определенных единицах, а дискретная функция вероятности является безразмерной. Например, если переменная X измеряется в секундах, то плотность вероятности измеряется в герцах (Гц, т.е. 1/с). Если $\mathbf{X}$ — точка в трехмерном пространстве, измеряемом в метрах, то плотность вероятности будет измеряться в $1/м^3$.

Одним из наиболее важных распределений вероятностей является ▶**распределение Гаусса**, известное также под названием ▶**нормальное распределение**. Для этого распределения мы используем обозначение $\mathcal{N}(x; \mu, \sigma^2)$ — как функции от x со средним значением μ и среднеквадратичным отклонением σ (и, следовательно, с дисперсией σ^2). Нормальное распределение определяется следующей формулой:

$$\mathcal{N}(x;\mu;\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)},$$

где x — непрерывная переменная, изменяющаяся в пределах от $-\infty$ до $+\infty$. Если среднее $\mu = 0$ и дисперсия $\sigma^2 = 1$, то имеет место частный случай нормального распределения, называемый ▶**стандартным нормальным распределением**. Если распределение задано на векторе $\mathbf{x}$ в пространстве с d измерениями, то оно представляет собой ▶**многомерное гауссово распределение**:

$$\mathcal{N}(\mathbf{x};\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}\left((\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right)},$$

где μ — вектор средних, а Σ — ▶**матрица ковариации** этого распределения (см. ниже). Кумулятивное распределение для одномерного нормального распределения определяется как

$$F(x) = \int_{-\infty}^{x} \mathcal{N}(z;\mu,\sigma^2)dz = \frac{1}{2}(1 + \mathrm{erf}(\tfrac{x-\mu}{\sigma\sqrt{2}})),$$

где $\mathrm{erf}(x)$ — так называемая ▶**функция ошибок**, не имеющая представления в замкнутой форме.

В ▶**центральной предельной теореме** утверждается, что среднее n случайных переменных приближается к нормальному распределению по мере того, как n стремится к бесконечности. Такому свойству подчиняется почти любая коллекция случайных переменных, при том условии, что значение дисперсии любого конечного подмножества переменных не доминирует над значениями дисперсии других конечных подмножеств.

▶**Математическое ожидание** случайной величины, $M(X)$, представляет собой среднее значение, взвешенное по вероятности каждого значения. Для дискретной переменной это

$$M(X) = \sum_x x_i P(X = x_i).$$

Для непрерывной переменной суммирование следует заменить интегралом и использовать функцию плотности вероятности, $P(x)$:

$$M(X) = \int_{-\infty}^{\infty} xP(x)dx.$$

Для любой функции f мы также имеем

$$M(f(X)) = \int_{-\infty}^{\infty} f(x)P(x)dx.$$

И наконец, при необходимости математическое ожидание можно определить через распределение случайной величины:

$$M_{X\sim Q(x)}(g(X)) = \int_{-\infty}^{\infty} g(x)Q(x)dx.$$

Помимо математического ожидания, другие важные статистические характеристики распределения включают ▶**дисперсию** $D[X]$, которая является математическим ожиданием квадрата отклонения случайной величины от ее математического ожидания:

$$D[X] = M((X - M(X))^2)$$

и среднеквадратическое или ▶**стандартное отклонение** σ, которое является квадратным корнем из дисперсии (по этой причине математическое ожидание в статистике часто обозначают как σ^2).

▶**Среднее квадратическое** s для набора значений (чаще всего это выборка значений случайной величины) представляет собой квадратный корень из среднего арифметического квадратов значений в наборе:

$$s(x_1, \ldots, x_n) = \sqrt{\frac{x_1^2 + \ldots + x_n^2}{n}}.$$

▶**Ковариация** двух случайных величин определяется как математическое ожидание произведения разностей их значений и соответствующих средних значений:

$$\text{cov}(X, Y) = M((X - \mu X)(Y - \mu Y)).$$

▶**Ковариационная матрица**, часто обозначаемая как S, является матрицей, составленной из попарных ковариаций между элементами вектора случайных величин. Принимая, что вектор представлен как $\mathbf{X} = \langle X_1, \ldots X_n \rangle^\top$, элементами ковариационной матрицы будут:

$$S_{i,j} = \text{cov}(X_i, X_j) = M((X_i - \mu_i)(X_j - \mu_j)).$$

▶**Выборка** — это часть общей совокупности возможных элементов случайной величины, полученная в результате некоторого эксперимента. Мы не знаем, каково будет распределение вероятности для любой конкретной выборки, но в пределе достаточно большой набор данных выборок будет приближаться к той же функции распределения плотности вероятности, которая свойственна общей совокупности элементов этой случайной величины. ▶**Равномерное распределение** характеризуется тем, что каждый элемент в общей совокупности в равной степени (равномерно) вероятен. Поэтому, если мы говорим, что "случайная величина имеет равномерное распределение в диапазоне целых чисел от 0 до 99", то это означает, что при выборке с одинаковой вероятностью может быть получено любое целое число из этого диапазона.

Библиографические и исторические заметки

Система обозначений $O(\ )$, которая широко используется в компьютерных науках в наши дни, была впервые определена в контексте теории чисел немецким математиком П.Г.Н. Бахманом (1894). Понятие NP-полноты было предложено Куком (1971), а современный метод определения способа сведения одной проблемы к другой предложен Карпом (1972). И Кук, и Карп получили за свою работу премию Тьюринга — высшую награду в компьютерных науках.

Из лучших учебников по анализу и разработке алгоритмов следует упомянуть книги Седвига и Уэйна (2011), а также Кормена, Лейзерсона, Ривеста и Штейна (2009). В этих изданиях особое внимание уделяется разработке и анализу алгоритмов для решения поддающихся решению задач. Теория NP-полноты и других форм неразрешимости представлена в книгах Гэри и Джонсона (1979) и Пападимитриу (1994). Хорошими учебниками по теории вероятности являются книги Чунга (1979), Росса (2015), а также Бертсекаса и Цициклиса (2008).

ПРИЛОЖЕНИЕ Б

Сведения о языках и алгоритмах, используемых в книге

Б.1. Определение языков с помощью формы Бэкуса–Наура

В этой книге дается определение нескольким языкам, в том числе языку логики высказываний (раздел 7.4), языку логики первого порядка (раздел 8.2) и подмножеству английского языка (раздел 23.4). Формальный язык определяется как множество строк, в котором каждая строка представляет собой последовательность символов. Все языки, которые были необходимы нам в этой книге, состоят из бесконечного множества строк, поэтому нужен простой способ, позволяющий охарактеризовать это множество. Для достижения данной цели удобнее всего воспользоваться **грамматикой**. Конкретный тип грамматики, который мы выбрали, носит название ▶**контекстно свободная грамматика**, поскольку каждое выражение в этом случае будет иметь одну и ту же форму в любом контексте. В качестве способа оформления грамматик мы приняли формальную систему под названием ▶**форма Бэкуса–Наура** или **БНФ** (*Backus-Naur form — BNF*). В любой БНФ-грамматике присутствуют четыре компонента.

- Множество ▶**терминальных символов**. Это символы или слова, из которых состоят строки языка. В качестве таких символов могут использоваться буквы (**A**, **B**, **C**, ...), слова (**a**, **aardvark**, **abacus**, ...) или любые символы, являющиеся допустимыми для области определения.
- Множество ▶**нетерминальных символов**, которые обозначают компоненты предложений языка. Например, нетерминальный символ *NounPhrase* (именное словосочетание) в английском языке обозначает бесконечное множество строк, включая такие, как *you* или *the big slobbery dog*.
- ▶**Начальный символ**, который представляет собой нетерминальный символ, обозначающий законченный набор из строк языка. В грамматике английского языка таковым является символ *Sentence*; в грамматике

арифметических выражений для этой цели может быть определен символ *Expr*, а в грамматике языка программирования это может быть *Program*.

- Множество ▶ **правил подстановки** в форме $LHS \rightarrow RHS$, где *LHS* — нетерминальный символ, а *RHS* — последовательность, в которую входят нуль и больше символов. Это могут быть как терминальные, так и нетерминальные символы, а также символ ϵ, который мы будем использовать для обозначения пустой строки.

Правило подстановки вида

Sentence → *NounPhrase VerbPhrase*

означает, что при наличии двух строк, обозначенных как *NounPhrase* (именное словосочетание) и *VerbPhrase* (глагольное словосочетание), их можно соединить друг с другом и классифицировать результат как *Sentence*. Определение этого правила также можно записать в сокращенном виде: если есть два правила ($S \rightarrow A$) и ($S \rightarrow B$), то можно записать ($S \rightarrow A \mid B$). Чтобы проиллюстрировать эту концепцию, ниже приведена БНФ-грамматика для простых арифметических выражений.

Expr	→	*Expr Operator Expr* | (*Expr*) | *Number*
Number	→	*Digit* | *Number Digit*
Digit	→	**0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**
Operator	→	+ | − | ÷ | ×

Более подробные сведения о языках и грамматиках приведены в главе 23. Следует отметить, что в других книгах в системе обозначений БНФ могут использоваться немного другие правила, например нетерминальные символы могут обозначаться ⟨*Digit*⟩, а не *Digit*, терминальные символы — 'word', а не **word**, а в правилах может использоваться символ ::= вместо →.

Б.2. Описание алгоритмов с помощью псевдокода

В этой книге все упоминаемые алгоритмы представлены на ▶ **псевдокоде**. Основные конструкции этого псевдокода должны быть понятны пользователям таких языков, как Java, C++ и в особенности Python. В некоторых местах для описания вычислений в псевдокоде используются математические формулы или текст на естественном языке, поскольку в противном случае пришлось бы применять более громоздкие конструкции. Также следует отметить, что в алгоритмах используются приведенные ниже соглашения.

- **Сохраняемые переменные**. Ключевое слово **persistent** (сохраняемая) используется как указание на то, что переменной присваивается начальное значение при первом вызове содержащей ее функции, после чего это

значение (или значение, присвоенное переменной с помощью выполненных в дальнейшем операторов присваивания) сохраняется при всех последующих вызовах функции. Таким образом, сохраняемые переменные подобны глобальным переменным в том, что они сохраняют свое значение после каждого вызова содержащей их функции, но при этом доступны только в данной функции. В программах агентов, приведенных в данной книге, сохраняемые переменные используются в качестве "памяти". В объектно-ориентированных языках, таких как C++, Java, Python и Smalltalk, программы с сохраняемыми переменными могут быть реализованы в виде *объектов*. В функциональных языках они могут быть реализованы с помощью *функциональных замыканий* в той среде, в которой определены требуемые переменные.

- **Функции как значения.** В нашем псевдокоде имена функций и процедур начинаются с прописных букв, а имена переменных состоят из строчных букв и выделяются курсивом. Поэтому в большинстве случаев вызов функции выглядит наподобие FN(x). Однако также допускается, чтобы значением переменной была функция, например если значением переменной f является функция вычисления квадратного корня, то выражение $f(9)$ возвращает 3.
- **Значимость отступов**. Наличие отступов при записи псевдокода имеет большое значение — по аналогии с языками Python и CoffeeScript, но в отличие от языков Java, C++ и Go (в которых используются скобки) или языков Lua и Ruby (в которых используется оператор **end**), в нашем псевдокоде они определяют область действия цикла или условного выражения.
- **Деструктуризация наборов данных**. Нотация "$x, y \leftarrow$ *пара*" означает, что правая часть выражения после вычисления должна представлять собой двухэлементную коллекцию, первый элемент которой присваивается переменной x, а второй — y. Та же самая идея может быть представлена записью "**for** x, y **in** *пара* **do**". Эта нотация также может использоваться для обмена значениями между двумя переменными: "$x, y \leftarrow y, x$".
- **Значения по умолчанию для параметров**. Нотация "**function** F(x, y = 0) **returns** число" означает, что в этой функции y является дополнительным аргументом со значением по умолчанию, равным 0. Иначе говоря, вызовы функции F(3, 0) и $F(3)$ являются эквивалентными.
- **Ключевое слово yield**. Функция, содержащая ключевое слово **yield**, представляет собой ▶**генератор**, генерирующий *последовательность* значений, — по одному каждый раз, когда встречается выражение, содержащее это ключевое слово. После генерации очередного значения функция продолжает свое выполнение со следующего оператора. В языках Python, Ruby, C# и JavaScript (ECMAScript) присутствует подобная функциональная возможность.

- **Циклы**. В псевдокоде допускается четыре типа циклов.
  - "**for** *x* **in** *c* **do**" — выполняется цикл с управляющей переменной *x*, возможные значения которой указаны как последовательность элементов в коллекции *c*.
  - "**for** *i* = 1 **to** *n* **do**" — выполняется цикл с управляющей переменной *i*, значения которой представляют собой последовательность целых чисел от 1 до *n* включительно.
  - "**while** *условие* **do**" — в этом цикле выполнение *условия* проверяется перед началом очередной его итерации, и если условие ложно, цикл завершается.
  - "**repeat** ... **until** *условие*" — этот цикл безусловно выполняется первый раз, а затем проверяется *условие*. Если оно истинно, цикл завершается, иначе он выполняется вновь (и в конце опять проверяется *условие*).
- **Списки.** Нотация [*x*, *y*, *z*] определяет список из трех элементов. Оператор "+" может употребляться для выполнения конкатенации списков: [1, 2] + [3, 4] = [1, 2, 3, 4]. Список может использоваться и как стек: функция Pop удаляет и возвращает последний элемент списка, а функция Top возвращает его последний элемент.
- **Множества.** Нотация {*x*, *y*, *z*} определяет множество (набор) из трех элементов. Запись {*x*: *p*(*x*)} определяет множество из всех элементов *x*, для которых *p*(*x*) истинно.
- **Индексация в массивах начинается с 1.** Первый элемент массива всегда имеет индекс 1, как это принято в обычной математической записи (и в языках R и Julia), а не 0 (как это принято в языках Python, Java и C).

Б.3. Дополнительный материал в Интернете

Для этой книги имеется собственный веб-сайт, содержащий дополнительные материалы и инструкции по отправке предложений, а также предоставляющий посетителям возможность присоединиться к дискуссионным группам.

- `aima.cs.berkeley.edu`

Все приведенные в этой книге алгоритмы и несколько дополнительных упражнений по программированию были реализованы на языках Python и Java (а некоторые и на других языках). Код этих реализаций находится в интернет-хранилище и доступен на веб-сайте, который в настоящее время размещается по адресу

- `github.com/aimacode`

Предметный указатель

Б

Н

О

П

Т

У

Ф

Х

Ц

Ч

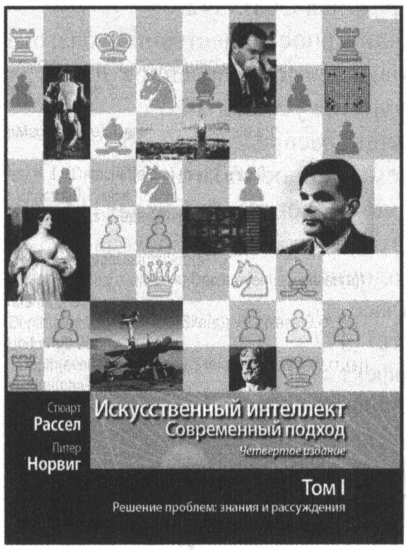
Э

Я

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ СОВРЕМЕННЫЙ ПОДХОД

4-Е ИЗДАНИЕ.
ТОМ I. РЕШЕНИЕ ПРОБЛЕМ: ЗНАНИЯ И РАССУЖДЕНИЯ

Стюарт Рассел
Питер Норвиг



www.dialektika.com

В этом обновленном и пересмотренном издании читатель найдет самое полное и актуальное введение в теорию и практику искусственного интеллекта. В книге изложены идеи, которые были сформулированы в исследованиях, проводившихся в этой области в течение последних пятидесяти лет, а также представлены современные достижения и новейшие технологии и концепции в таких областях, как машинное обучение, многоагентные системы, робототехника, обработка естественного языка, вероятностное программирование, а также конфиденциальность, беспристрастность и безопасность ИИ. Написанная без излишнего академизма (но достаточно строго) книга будет полезна широкому кругу читателей: студентам, аспирантам и преподавателям высших учебных заведений, инженерам, разработчикам ПО и т.д.

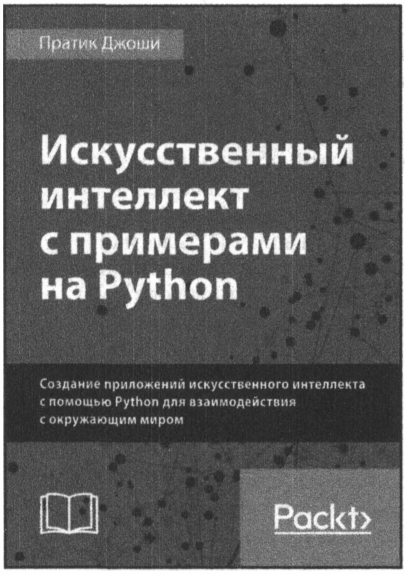
В связи с исключительно большим объемом этого энциклопедического издания его русскоязычный вариант представлен в виде трехтомника. В первом томе излагаются основы основ ИИ — общая характеристика этой области как науки об интеллектуальных агентах, действующих в различных средах и решающих различные задачи на основании знаний, результатов восприятия, рассуждений и планирования.

ISBN 978-5-907365-25-4

в продаже

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ С ПРИМЕРАМИ НА PYTHON

Пратик Джоши



www.dialektika.com

В этой книге исследуются различные сценарии применения искусственного интеллекта. Вначале рассматриваются общие концепции искусственного интеллекта, после чего обсуждаются более сложные темы, такие как предельно случайные леса, скрытые марковские модели, генетические алгоритмы, сверточные нейронные сети и др. Вы узнаете о том, как принимать обоснованные решения при выборе необходимых алгоритмов, а также о том, как реализовывать эти алгоритмы на языке Python для достижения наилучших результатов.

Основные темы книги:

- различные методы классификации и регрессии данных;
- создание интеллектуальных рекомендательных систем;
- логическое программирование и способы его применения;
- построение автоматизированных систем распознавания речи;
- основы эвристического поиска и генетического программирования;
- разработка игр с использованием искусственного интеллекта;
- обучение с подкреплением;
- алгоритмы глубокого обучения и создание приложений на их основе.

ISBN: 978-5-907114-41-8

в продаже

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ: СТРАТЕГИИ И МЕТОДЫ РЕШЕНИЯ СЛОЖНЫХ ПРОБЛЕМ

4-Е ИЗДАНИЕ

Джордж Люгер

www.williamspublishing.com

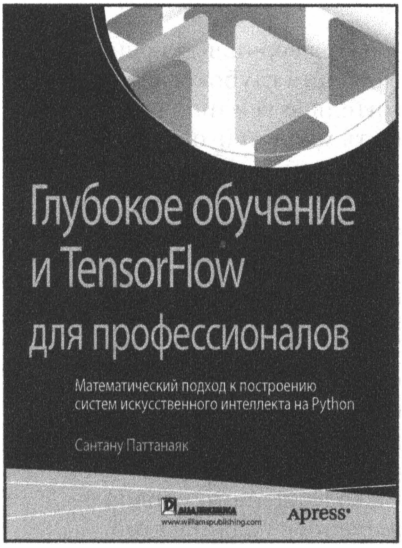
Данная книга посвящена одной из наиболее перспективных и привлекательных областей развития научного знания — методологии искусственного интеллекта (ИИ). В ней детально описываются как теоретические основы искусственного интеллекта, так и примеры построения конкретных прикладных систем. Книга дает полное представление о современном состоянии развития этой области науки. Подробно рассматриваются вопросы представления знаний при решении задач ИИ, логика решения этих задач, алгоритмы поиска, продукционные системы и машинное обучение. Эти вопросы остаются центральными в области искусственного интеллекта. В книге также представлены результаты новейших исследований, связанных с вопросами понимания естественного языка, обучения с подкреплением, рассуждения в условиях неопределенности, эмерджентных вычислений, автоматического доказательства теорем и решения задач ИИ на основе моделей. Большое внимание уделяется описанию реальных прикладных систем, построенных на принципах ИИ, и современных областей приложения этой области знаний.

ISBN 978-5-8459-0437-9

в продаже

ГЛУБОКОЕ ОБУЧЕНИЕ И TENSORFLOW
ДЛЯ ПРОФЕССИОНАЛОВ

Сантану Паттанаяк



www.williamspublishing.com

Данная книга представляет собой углубленное практическое руководство, которое позволит читателям освоить методы глубокого обучения на уровне, достаточном для развертывания готовых решений. Прочитав книгу, вы сможете быстро приступить к работе с библиотекой TensorFlow и заняться оптимизацией архитектур глубокого обучения. Весь программный код доступен в виде блокнотов iPython и сценариев, позволяющих с легкостью воспроизводить примеры и экспериментировать с ними.

Благодаря этой книге вы:

- овладеете полным стеком технологий глубокого обучения с использованием TensorFlow и получите необходимую для этого математическую подготовку;
- научитесь развертывать сложные приложения глубокого обучения в производственной среде с помощью TensorFlow;
- сможете проводить исследования в области глубокого обучения и выполнять самостоятельные эксперименты в TensorFlow.

ISBN: 978-5-907144-25-5

в продаже

ПРИКЛАДНОЕ МАШИННОЕ ОБУЧЕНИЕ С ПОМОЩЬЮ SCIKIT-LEARN, KERAS И TENSORFLOW

2-е ПОЛНОЦВЕТНОЕ ИЗДАНИЕ

Орельен Жерон

www.williamspublishing.com

Благодаря серии выдающихся достижений глубокое обучение значительно усилило всю область машинного обучения. В наше время даже программисты, почти ничего не знающие об этой технологии, могут использовать простые и эффективные инструменты для реализации программ, которые способны обучаться на основе данных.

За счет применения конкретных примеров, минимума теории и фреймворков Python производственного уровня обновленное издание этой ставшей бестселлером книги поможет вам получить интуитивное представление о концепциях и инструментах, предназначенных для построения интеллектуальных систем.

ISBN 978-5-907203-33-4 **в продаже**

Самое полное и актуальное введение в теорию и практику искусственного интеллекта!

В четвертом, обновленном, пересмотренном и дополненном издании этой книги область искусственного интеллекта (ИИ) исследуется и анализируется во всей ее обширности и глубине. Здесь представлены все современные достижения и изложены идеи, которые были сформулированы в исследованиях, проводившихся в течение последних пятидесяти лет, а также собраны на протяжении двух тысячелетий в областях знаний, ставших стимулом к развитию ИИ как науки. Предыдущие издания этой книги стали классическими образцами литературы по ИИ и приняты в качестве учебного пособия более чем в 1400 университетах 128 стран мира, где были высоко оценены как убедительный итог обобщения результатов, достигнутых в этой области науки.

Книга дополнена обширным набором интернет-ресурсов, включая упражнения, программные проекты и исследовательские проекты, реализации алгоритмов, дополнительные материалы и ссылки для студентов и преподавателей

Что нового в четвертом издании

В четвертом издании читатель познакомится с новейшими технологиями и концепциями, представленными в более унифицированном виде с новым или расширенным охватом таких тем, как машинное обучение, глубокое обучение, трансферное обучение, многоагентные системы, робототехника, обработка естественного языка, проблема причинности, вероятностное программирование, а также конфиденциальность, беспристрастность и безопасность ИИ.

Что представлено в этом томе

В связи с исключительно большим объемом этого энциклопедического издания его русскоязычный вариант представлен в виде трехтомника. В этом томе, получившем подзаголовок *Знания и рассуждения в условиях неопределенности*, обсуждаются различные аспекты подхода к реализации агентов, действующих в подобных средах: вероятностные рассуждения, в том числе о времени, построение планов и принятие решений (простых и сложных), в том числе в многоагентной среде.

Что в других томах

В предыдущем томе I обсуждались основы теории ИИ: интеллектуальные агенты и возможный подход к решению проблем: знания, рассуждения и планирование. В следующем томе III обсуждаются методы и возможности машинного обучения, совершенствования восприятия и реализации автономных устройств — роботов и роботизированных систем, включая такие важные социальные аспекты, как справедливость, доверие, общественное благо и безопасность.

ОБ АВТОРАХ

Стюарт Рассел, профессор компьютерных наук Калифорнийского университета в г. Беркли, адъюнкт-профессор неврологической хирургии Калифорнийского университета в г. Сан-Франциско, основатель и директор центра Center for Human-Compatible AI и заведующий кафедрой инженерного искусства Смита–Задэ в Калифорнийском университете в г. Беркли.

Питер Норвиг, директор по исследованиям в компании Google, Inc., прежде — глава ее группы разработки основных алгоритмов веб-поиска. До этого руководил отделом компьютерных наук в Исследовательском центре Эймса, НАСА, был профессором в Университете Южной Калифорнии и членом совета факультета в Беркли и Станфорде.

Авторы этой книги разделили первую награду AAAI/EAAI "Выдающийся педагог" в 2016 году.

На сайте издательства "Диалектика" можно скачать перечень литературы, ссылки на которую приведены в этой книге:
http://www.williamspublishing.com/Books/978-5-907365-26-1.html

ISBN 978-5-907365-26-1

ДИАЛЕКТИКА
www.dialektika.com

www.pearson.com