В этой книге представлено общее введение в вероятностные графовые модели (ВГМ) с инженерной точки зрения. В книге подробно рассматриваются теоретические основы для каждого из основных классов ВГМ, включая принципы и методы представления, логического вывода и обучения, а также обзоры реальных практических приложений для каждого типа модели. Примеры приложений взяты из самых разнообразных предметных областей и наглядно демонстрируют множество вариантов применения байесовских классификаторов, скрытых марковских моделей, байесовских сетей, динамических и временны́х байесовских сетей, марковских случайных полей, диаграмм влияния и марковских процессов принятия решений.

Особенности книги:

- представлена универсальная рабочая среда, включающая все основные классы ВГМ;
- освещается практическое применение разнообразных методик;
- рассматриваются все самые последние разработки в области ВГМ, включая многомерные байесовские классификаторы, реляционные графовые модели и причинно-следственные модели;
- в конце каждой главы предлагаются задания и упражнения для самостоятельного выполнения, а также направления и идеи для исследовательских или программных проектов.

Интернет-магазин:
www.dmkpress.com

Оптовая продажа:
КТК «Галактика»
books@alians-kniga.ru

Springer

ДМК ПРЕСС ИЗДАТЕЛЬСТВО
www.дмк.рф

ISBN 978-5-97060-874-6

Вероятностные графовые модели

Луис Энрике Сукар

Вероятностные графовые модели

Принципы и приложения

Springer

ДМК ПРЕСС ИЗДАТЕЛЬСТВО

Луис Энрике Сукар

Вероятностные графовые модели

Принципы и приложения

Luis Enrique Sucar

Probabilistic Graphical Models

Principles and Applications

Springer

Луис Энрике Сукар

Вероятностные графовые модели

Принципы и приложения

ДМК Пресс
издательство
Москва, 2021

УДК 004.021
ББК 32.973
С89

С89 Луис Энрике Сукар

Вероятностные графовые модели. Принципы и приложения / пер. с англ. А. В. Снастина. – М.: ДМК Пресс, 2021. – 338 с.: ил.

ISBN 978-5-97060-874-6

В настоящее время вероятностные графовые модели широко распространены как мощная и вполне сформировавшаяся методика для выполнения умозаключений и выводов в условиях неопределенности. В отличие от некоторых узкоспециализированных методик, применявшихся в ранних экспертных системах, вероятностные графовые модели основаны на строгих математических принципах теории графов и теории вероятностей. Эта книга на современном уровне охватывает более широкий диапазон вероятностных графовых моделей, чем другие недавно опубликованные материалы в данной области: разнообразные классификаторы, скрытые марковские модели, марковские случайные поля, байесовские сети и их динамические, временные и причинно-следственные варианты, реляционные вероятностные графовые модели, графы решений и марковский процесс принятия решений. В книге представлены эти вероятностные графовые модели и соответствующие методы логического вывода и обучения в ясном и доступном стиле. Автор делится своим богатым опытом, накопленным в процессе активной практической работы в области использования вероятностных графовых моделей, и демонстрирует примеры их применения в разнообразных областях реальной деятельности: от биоинформатики до задач наблюдения за загрязнением воздуха и распознавания объектов.

Книга предназначена для студентов старших курсов и аспирантов, а также для ученых-исследователей и инженеров-практиков, работающих в других отраслях и интересующихся применением вероятностных моделей.

ISBN 978-1-4471-6698-6 (англ.)
ISBN 978-5-97060-874-6 (рус.)

Посвящается моей семье – Дорис, Эдгару и Диане – за их безграничную любовь и поддержку

Оглавление

Предисловие от издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Скачивание исходного кода примеров

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

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Springer очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Вступительное слово

Вероятностные графовые модели и методы их использования для разумных рассуждений и выводов в условиях неопределенности появились в 1980-х гг. в сообществах, занимающихся проблемами статистических выводов и искусственного интеллекта. Конференция по неопределенности в области искусственного интеллекта (UAI – Uncertainty in Artificial Intelligence) стала самым первым форумом, посвященным этой быстро развивающейся области исследований. На этой конференции UAI в 1992 году в Сан Хосе (San Jose) я впервые встретила Энрике Сукара (Enrique Sucar) – тогда мы оба были аспирантами, – он представлял свою работу по реляционным и временны́м (темпоральным) моделям для систем высокоуровневой обработки компьютерного зрения. За прошедшие с тех пор 25 лет Энрике внес впечатляющий вклад в эту область исследований: от фундаментальной работы по объективной вероятности до разработки усовершенствованных форм вероятностных графовых моделей, таких как временны́е байесовские сети и байесовские сети для вычисления вероятности событий, а также методики обучения вероятностных графовых моделей, например самая последняя работа Энрике по классификаторам типа байесовская цепь для многомерной классификации.

В настоящее время вероятностные графовые модели широко распространены как мощная и вполне сформировавшаяся методика для выполнения умозаключений и выводов в условиях неопределенности. В отличие от некоторых узкоспециализированных методик, применявшихся в ранних экспертных системах, вероятностные графовые модели основаны на строгих математических принципах теории графов и теории вероятностей. Их можно использовать для широкого диапазона задач, связанных с выводом и обоснованием результатов, в том числе задач прогнозирования, мониторинга, диагностики, оценки рисков и принятия решений. Существует множество эффективных алгоритмов как для логического вывода, так и для обучения, доступных в виде программного обеспечения с открытым исходным кодом и коммерческих программных продуктов. Более того, мощь и эффективность этих алгоритмов уже подтверждена их успешным практическим применением в огромном диапазоне областей задач реального окружающего нас мира. Энрике Сукар стал главным специалистом по практическому внедрению и утверждению вероятностных графовых моделей как эффективной и полезной технологии с учетом его работы в широком диапазоне прикладных областей. Это медицина, реабилитация и здравоохранение в целом, робототехника и машинное зрение, образование, анализ надежности и разнообразные промышленные приложения – от нефтедобычи до энергетики.

Первыми авторами, которые начали исследования байесовских сетей и с большим мастерством убедительно описали их в книгах, стали Джуда Перл (Judea Pearl) с книгой «Probabilistic Reasoning in Intelligent Systems» и Рич Неаполитан (Rich Neapolitan) с книгой «Probabilistic Reasoning in Expert Systems». Предлагаемая читателям монография Энрике Сукара представляет собой своевременное дополнение к комплекту литературы, изданной после книг Перла и Неполитана. Эта книга на современном уровне охватывает более широкий диапазон вероятностных графовых моделей, чем другие недавно опубликованные материалы в этой области: разнообразные классификаторы, скрытые марковские модели, марковские случайные поля, байесовские сети и их динамические, временные и причинно-следственные варианты, реляционные вероятностные графовые модели, графы решений и марковский процесс принятия решений. В книге представлены эти вероятностные графовые модели и соответствующие методы логического вывода и обучения в ясном и доступном стиле. Поэтому издание подходит для студентов старших курсов и аспирантов, а также для ученых-исследователей и инженеров-практиков, работающих в других отраслях и интересующихся применением вероятностных моделей. В этой книге Энрике делится своим богатым опытом, накопленным в процессе активной практической работы в области использования вероятностных графовых моделей, и демонстрирует примеры их применения в разнообразных областях реальной деятельности: от биоинформатики до задач наблюдения за загрязнением воздуха и распознавания объектов. Я искренне поздравляю Энрике с выходом этой книги и настоятельно рекомендую ее будущим читателям.

Энн Е. Николсон,
Мельбурн, Австралия,
май 2015 года

Предисловие

О чем эта книга

Вероятностные графовые модели уже превратились в мощный набор методик, используемых в нескольких предметных областях. В этой книге представлен общий вводный курс по вероятностным графовым моделям (probabilistic graphical models – PGM) с учетом их применения в инженерных дисциплинах. Здесь подробно рассматриваются главные принципы основных классов вероятностных графовых моделей: байесовские классификаторы, скрытые марковские модели, байесовские сети, динамические и временны́е байесовские сети, марковские случайные поля, диаграммы влияния и марковские процессы принятия решений. Кроме того, описано представление, процесс логического вывода и принципы обучения для всех этих методик. В книге также рассматриваются реальные приложения для каждого типа модели.

Некоторые характерные особенности этой книги:

- основные классы вероятностных графовых моделей представлены в одной монографии в единой универсальной рабочей среде;
- книга охватывает основополагающие принципы: представление, процесс логического вывода и процесс обучения для всех методов;
- демонстрируются практические приложения различных методик для решения реальных задач, что весьма важно для студентов, аспирантов и инженеров-практиков;
- в книгу включены некоторые самые последние разработки в этой области, такие как многомерные байесовские классификаторы, реляционные графовые модели и причинно-следственные модели;
- к каждой главе прилагается ряд упражнений и заданий, которые могут послужить отправным пунктом для исследовательских и программных проектов.

Одна из целей этой книги – стимулирование практического приложения вероятностных графовых моделей к реальным задачам. Такой подход требует не только теоретических знаний о различных моделях и методиках, но также некоторого практического опыта и знания предметной области. Чтобы помочь профессионалам в различных областях получить некоторое углубленное представление об использовании вероятностных графовых моделей для решения практических задач, в книгу включено множество примеров применения разнообразных типов моделей для широкого диапазона предметных областей, в том числе:

- компьютерное зрение;
- биомедицинские приложения;
- промышленные приложения;
- извлечение информации;
- интеллектуальные обучающие системы;
- биоинформатика;
- приложения в области охраны окружающей среды;
- робототехника;
- взаимодействие человек–компьютер;
- проверка достоверности информации;
- уход за пациентами (медицина).

Для кого предназначена эта книга

Эта книга может использоваться как учебное пособие для студентов последнего года обучения и аспирантов по курсу вероятностных графовых моделей для изучающих информатику, инженерное дело, физику и т. д. Книга также может служить справочником для профессионалов, которым необходимо применять вероятностные графовые модели в различных областях, и для всех, кто хочет овладеть основами применения этих методик.

Для чтения данной книги не требуется какой-либо особенной предварительной подготовки, тем не менее желательно знание основ теории вероятностей и статистики. Предполагается, что читатель обладает базовыми знаниями математики на уровне средней школы, а также имеет определенный уровень подготовки в области информатики и программирования. Упражнения и задания по программированию требуют некоторых знаний и практического опыта использования какого-либо языка программирования, например C, C++, Java, MATLAB и т. д.

Упражнения и задания

К каждой главе (за исключением введения) прилагается ряд упражнений и заданий. Некоторые из них являются вопросами и задачами, способствующими более глубокому пониманию концепций и методик, изложенных в соответствующей главе. Кроме того, в каждой главе предлагается несколько заданий, представляющих собой отправные пункты для исследовательских и программных проектов (такие задания отмечены тремя звездочками «***»), которые могут использоваться как учебные курсовые проекты.

Обзор содержимого книги

Книга разделена на четыре части. В первой части представлено общее введение и обоснование использования вероятностных графовых мо-

делей, приведены необходимые для чтения основы теории вероятностей и теории графов. Во второй части описываются модели, которые не считаются решениями или утилитами: байесовские классификаторы, скрытые марковские модели, марковские случайные поля, байесовские сети, динамические и временны́е байесовские сети. Третья часть начинается с краткого введения в теорию решений, затем рассматриваются модели с поддержкой принятия решений, включая деревья решений, диаграммы влияния и марковские процессы принятия решений. В четвертой части представлены два расширения стандартных вероятностных графовых моделей: реляционные вероятностные графовые модели и причинно-следственные вероятностные графовые модели.

Зависимости между содержимым глав показаны на рис. 1. Дуга от главы X к главе Y, т. е. $X \rightarrow Y$, означает, что изучение содержимого главы X требуется (или, по меньшей мере, рекомендуется) для понимания главы Y. Такое графическое представление структуры книги дает немалый объем информации в аналоговом представлении о графовых моделях, которые будут рассматриваться в этой книге.

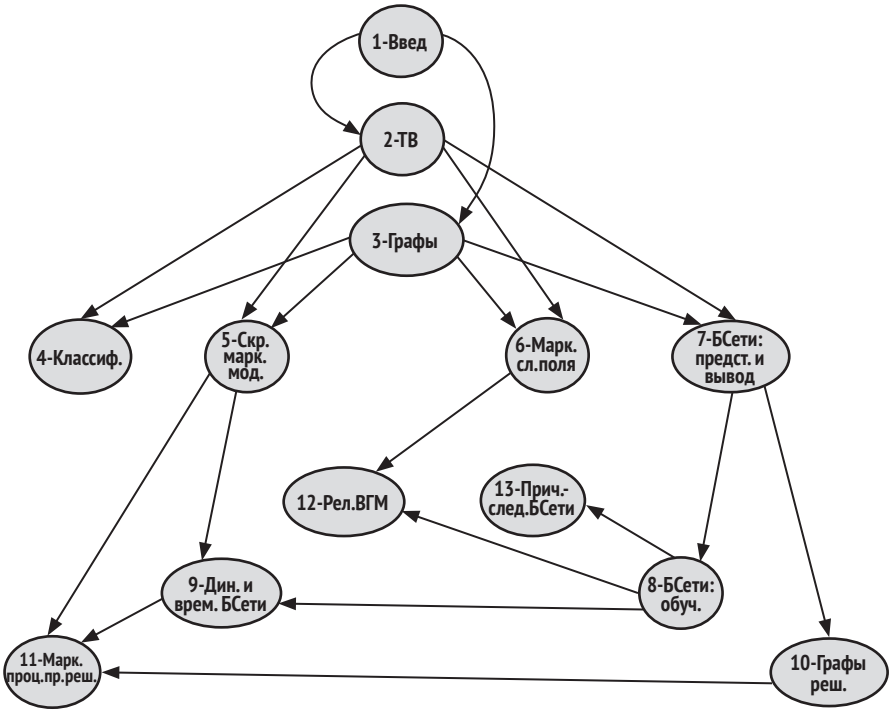


Рис. 1. Структура книги в виде направленного ациклического графа, показывающего, какие главы требуется изучить для перехода к следующим главам

По рис. 1 можно определить несколько различных способов чтения данной книги. В первую очередь рекомендуется прочитать введение и главы о математических основах 2 и 3. Затем более или менее незави-

симо друг от друга можно изучать различные модели в части II: классификацию (глава 4), скрытые марковские модели (глава 5), марковские случайные поля (глава 6) и байесовские сети (главы 7–9). Перед чтением главы об обучении байесовских сетей (глава 8) необходимо прочитать главу 7 – представление и логический вывод, кроме того, изучение обеих этих глав требуется перед переходом к динамическим и временны́м байесовским сетям.

Темы в частях III и IV требуют изучения некоторых глав из части II. Для чтения главы 10, в которой рассматриваются деревья решений и диаграммы влияния, вы должны как минимум прочитать первую главу по байесовским сетям. Перед чтением главы 11 о последовательном принятии решений рекомендуется изучить скрытые марковские модели, а также динамические и временны́е байесовские сети. Реляционные вероятностные графовые модели (глава 12) основаны на марковских случайных полях и байесовских сетях, поэтому требуется предварительное изучение глав 6 и 8. Наконец, причинно-следственные модели в главе 13 также основаны на байесовских сетях, включая методики их обучения.

Если в учебном курсе недостаточно времени для полного охвата материала всей книги, то можно предложить несколько альтернативных вариантов. Например, сосредоточиться на вероятностных моделях без рассмотрения теории принятия решений или более продвинутых расширений, т. е. изучать части I и II. Другой вариант – все внимание уделить моделям принятия решений, включая часть I, необходимые подготовительные главы из части II и часть III. Кроме того, вы можете запланировать курс по своему усмотрению, но непременно с учетом зависимостей, показанных на графе на рис. 1. Однако если вы располагаете достаточным временем и имеете серьезные намерения, то я предлагаю прочесть всю книгу в порядке глав. Желаю успеха.

Луис Энрике Сукар (Luis Enrique Sucar),
Пуэбла, Мексика,
февраль 2015 г.

Благодарности

Эта книга появилась на основе курса, который я читал в течение нескольких лет аспирантам. Сначала это был курс «Заключения и выводы в условиях неопределенности» (Uncertain Reasoning) в Монтеррейском технологическом институте (Tec de Monterrey) в г. Куэрнавака (Cuernavaca), затем он превратился в курс по вероятностным графовым моделям, когда я перешел в Национальный институт астрофизики, оптики и электроники (INAOE) в г. Пуэбла (Puebla) в 2006 году. В течение всех этих лет мои студенты являлись главным стимулом и источником вдохновения для написания этой книги. Я хочу поблагодарить всех своих студентов за их заинтересованность, за их вопросы и многочисленные поправки к моим заметкам. Книга посвящается всем моим студентам – прошлым, настоящим и будущим.

Я благодарен тем студентам, с которыми сотрудничал особенно тесно, обычно в течение небольшого интервала времени, при подготовке диссертаций для получения степени бакалавра, магистра или доктора. Некоторые из новейших аспектов этой книги и большинство примеров практических приложений взяты из их работ. Я благодарю всех этих студентов и ниже назову лишь тех, чьи работы оказали наибольшее влияние при создании рукописи данной книги: Густаво Арройо (Gustavo Arroyo), Эктор Уго Авилес (Héctor Hugo Avilés), Леонардо Чанг (Leonardo Chang), Рикардо Омар Чавес (Ricardo Omar Chávez), Франсиско Элисалде (Francisco Elizalde), Уго Хаир Эскаланте (Hugo Jair Escalante), Линдси Фидлер (Lindsey Fiedler), Джовани Гомес (Giovani Gómez), Карлос Эрнандес (Carlos Hernández), Пабло Эрнандес (Pablo Hernández), Ясмин Эрнандес (Yasmín Hernández), Пабло Ибаргуэнгойтиа (Pablo Ibargüengoytia), Рохер Луис-Веласкес (Roger Luis-Velásquez), Мириам Мартинес (Miriam Martínez), Хосе Антонио Монтеро (José Antonio Montero), Хулиета Ногес (Julieta Noguez), Аннетте Моралес (Annette Morales), Хоакин Перес-Брито (Joaquín Pérez-Brito), Мигель Паласиос (Miguel Palacios), Мальинали Рамирес (Mallinali Ramírez), Альберто Рейес (Alberto Reyes), Андрес Родригес (Andrés Rodríguez), Элиас Руис (Elías Ruiz), Херардо Торрес-Толедано (Gerardo Torres-Toledano), и Хулио Сарагоса (Julio Zaragoza). Особая благодарность Линдси Фидлер (Lindsey Fiedler), которая помогла мне создать все иллюстрации к этой книге и проверила мой английский.

Я также благодарен моим сотрудникам, совместная исследовательская работа и технические дискуссии с которыми обогатили мои знания по многим темам и помогли написать эту книгу. Хотелось бы особенно отметить следующих коллег и друзей: Хуан Мануэль Ауацин (Juan Manuel Ahuactzin), Оливер Айкард (Olivier Aycard), Конча Биелса

(Concha Bielza), Роберто Лей Боррас (Roberto Ley Borrás), Кристина Конати (Cristina Conati), Хавьер Диэс (Javier Díez), Уго Хаир Эскаланте (Hugo Jair Escalante), Данкан Жилль (Duncan Gillies), Хесус Гонсалес (Jesús González), Эдель Гарсиа (Edel García), Хессе Оэ (Jesse Hoey), Пабло Ибаргуэнгойтиа (Pablo Ibargüengoytia), Педро Ларраньяга (Pedro Larrañaga), Рон Ледер (Ron Leder), Джим Литтл (Jim Little), Хосе Луис Маррокин (José Luis Marroquín), Оскар Майора (Oscar Mayora), Мануэль Монтес (Manuel Montes), Эдуардо Моралес (Eduardo Morales), Энрике Муньос де Коте (Enrique Muñoz de Cote), Хулиета Ногэс (Julieta Noguez), Фелипе Ориуэла (Felipe Orihuela), Луис Пинеда (Luis Pineda), Дэвид Пул (David Poole), Альберто Рейес (Alberto Reyes), Карлос Руис (Carlos Ruiz), Сунил Вадера (Sunil Vadera) и Луис Вильясеньор (Luis Villaseñor). Я благодарю Эдель, Фелипе и Пабло за комментарии к некоторым первоначальным версиям книги. Спасибо Энн Николсон (Ann Nicholson) за превосходное вступительное слово.

Я благодарю за поддержку Национальный институт астрофизики, оптики и электроники (INAOE), предоставивший мне великолепную рабочую среду для исследовательской и преподавательской работы, а также обеспечивший возможность выделить часть моего времени для написания этой книги.

В последнюю по порядку, но не по важности очередь благодарю мою семью. Мои родители Фуэд (Fuhed†) и Аида (Aida) поощряли мое желание учиться и упорно работать, поддерживали мои научные занятия. Особенно мой отец, который написал несколько замечательных книг и вдохновил на писательскую деятельность (возможно, подействовали гены). Мой брат Рикардо (Ricardo) и сестры Шафиа (Shafia) и Беатрис (Beatriz) всегда поддерживали и поощряли мои стремления и мечты. Особенная благодарность моей жене Дорис (Doris) и моим детям Эдгару (Edgar) и Диане (Diana), которые скучали в течение тех долгих часов, которые я посвятил книге, а не им. Их любовь и поддержка – это то, что позволяет мне двигаться дальше.

Список сокращений, принятых в книге

ADD	Algebraic Decision Diagram	АДР	алгебраическая диаграмма решений
AI	Artificial Intelligence	ИИ	искусственный интеллект
BAN	Bayesian network Augmented Naive Bayes classifier	НБКБС	наивный байесовский классификатор, дополненный байесовской сетью
BCC	Bayesian Chain Classifier	БЦ	байесовская цепь (классификатор)
BCCD	Bayesian Constraint-based Causal Discovery	БПСА	байесовский причинно-следственный анализ с учетом ограничений
BN	Bayesian Network	БС	байесовская сеть
CBN	Causal Bayesian Network	ПСБС	причинно-следственная байесовская сеть
CMI	Conditional Mutual Information	УВИ	условная взаимная информация
CPT	Conditional Probability Table	ТУВ	таблица условных вероятностей
CRF	Conditional Random Field	УСП	условное случайное поле
DAG	Directed Acyclic Graph	НАГ	направленный ациклический граф
DBN	Dynamic Bayesian Network	ДБС	динамическая байесовская сеть
DBNC	Dynamic Bayesian Network Classifier	КДБС	классификатор на основе ДБС
DD	Decision Diagram	ДГР	диаграмма решений
DDN	Dynamic Decision Network	ДСР	динамическая сеть принятия решений
DT	Decision Tree	ДР	дерево решений
EC	Expected Cost	ОЗ	ожидаемые затраты
EM	Expectation	Maximization	EM-алгоритм
FN	False Negative	ЛО	ложноотрицательный
FP	False Positive	ЛП	ложноположительный
GRM	Gibbs Random Field	СПГ	случайное поле Гиббса

HMM	Hidden Markov Model	СММ	скрытая марковская модель
ICM	Iterative Conditional Modes	ИУМ	итеративные условные методы (алгоритм)
ID	Influence Diagram	ДВ	диаграмма влияния
ILP	Inductive Logic Programming	ИЛП	индуктивное логическое программирование
KB	Knowledge Base	БЗ	база знаний
LIMID	Limited Memory Influence Diagram	ДВОП	диаграмма влияния с ограниченной памятью
MAG	Maximal Ancestral Graph	МНГ	максимальный наследственный граф
MAP	Maximum a Posteriori	ОАМ	оценка апостериорного максимума
MB	Markov Blanket	МО	марковское ограждение
MBC	Multidimensional Bayesian network Classifier	МБС	многомерная байесовская сеть (классификатор)
MC	Markov Chain	МЦ	марковская цепь
MDL	Minimum Description Length	ПМДО	принцип минимальной длины описания
MDP	Markov Decision Process	МППР	марковский процесс принятия решений
MLN	Markov Logic Network	МЛС	марковская логическая сеть
MN	Markov Network	МС	марковская сеть
MPE	Most Probable Explanation	НВО	наиболее вероятное объяснение (обоснование)
MRF	Markov Random Field	МСП	марковское случайное поле
NBC	Naïve Bayes Classifier	НБК	наивный байесовский классификатор
PAG	Parental Ancestral Graph	РНГ	родительский наследственный граф
PGM	Probabilistic Graphical Model	ВГМ	вероятностная графовая модель
PL	Pseudolikelihood	ПП	псевдоправдоподобие

POMDP	Partially Observable Markov Decision Process	ЧНМПР	частично наблюдаемый марковский процесс принятия решений
PRM	Probabilistic Relational Model	ВРМ	вероятностная реляционная модель
RPGM	Relational Probabilistic Graphical Model	РВГМ	реляционная вероятностная графовая модель
SNBC	Semi-Naïve Bayesian Classifier	ЧНБК	частично наивный байесовский классификатор
TAN	Tree Augmented Naive Bayes classifier	НБКД	наивный байесовский классификатор, дополненный деревом
TEN	Temporal Event Network	ВСС	временна́я сеть событий
TN	Temporal Node	ВУ	временной узел (сети)
TNBN	Temporal Nodes Bayesian Network	БСВУ	байесовская сеть с временны́ми узлами

Условные математические обозначения, используемые в книге

T	True (Истина)
F	False (Ложь)
$A, B, C, \ldots$	Утверждения (бинарные/двоичные переменные)
$\neg A$	Не A (отрицание)
$A \wedge B$	A и B (конъюнкция)
$A \vee B$	A или B (дизъюнкция)
$A \rightarrow B$	B если A (импликация)
$A \leftrightarrow B$	A, если B, и B, если A (двойная импликация)
$X \in A$	X является элементом A
$\forall(X)$	Квантор всеобщности: для всех X
$\exists(X)$	Квантор существования: существует X
$C \cup D$	Объединение двух множеств
$C \cap D$	Пересечение двух множеств
Ω	Пространство выборки
X	Случайная переменная
x	Конкретное значение случайной переменной, $X = x$
$\mathbf{X}$	Вектор случайных переменных, $\mathbf{X} = X_1, X_2, \ldots, X_N$
$\mathbf{x}$	Конкретная реализация вектора $\mathbf{X}$, $\mathbf{x} = x_1, x_2, \ldots, x_N$
$X_{1:T}$	Вектор переменной X от $t = 1$ до $t = T$, $X_{1:T} = X_1, X_2, \ldots, X_T$
$P(X = x)$	Вероятность переменной X, находящейся в состоянии x; сокращенно $P(x)$
$P(\mathbf{X} = \mathbf{x})$	Вероятность вектора $\mathbf{X}$, находящегося в состоянии $\mathbf{x}$; сокращенно $P(\mathbf{x})$
$P(x, y)$	Вероятность x и y
$P(x \vee y)$	Вероятность x или y
$P(x \mid y)$	Условная вероятность x при заданном y
$P(x) \sim y$	Вероятность x *пропорциональна* y, т. е. $P(x) = k \times y$
$\mathbf{P}(\mathbf{X})$	Кумулятивная функция распределения дискретной переменной X
$P(X)$	Функция вероятности дискретной переменной X
$F(X)$	Кумулятивная функция распределения непрерывной переменной X

$f(X)$	Функция плотности (распределения) вероятности непрерывной переменной X
$I(X, Y, Z)$	Переменная X независима от Z при заданной переменной Y
$G(V, E)$	Граф G с множеством вершин V и множеством ребер E
$Pa(X)$	Родители (предки) узла X в направленном (ориентированном) графе
$Nei(X)$	Соседи узла X в графе
$n!$	Факториал числа n; $n! = n \times (n-1) \times (n-2) \times \ldots 1$
$\binom{n}{r}$	Число сочетаний из r элементов множества n; $\binom{n}{r} = \frac{n!}{r!(n-r)!}$
$exp(x)$	Экспоненциальная функция от x; $exp(x) = e^x$
$\|X\|$	Размерность, или число состояний дискретной переменной X
μ	Меана (среднее значение)
σ^2	Дисперсия случайной величины
σ	Среднеквадратическое (стандартное) отклонение
$N(\mu, \sigma^2)$	Нормальное распределение со средним значением μ и среднеквадратическим отклонением σ
$I(m)$	Информация
$H(M)$	Энтропия
$E(X)$	Ожидаемое значение случайной переменной X
$ArgMax_x F(X)$	Значение переменной X, при котором функция F достигает максимума

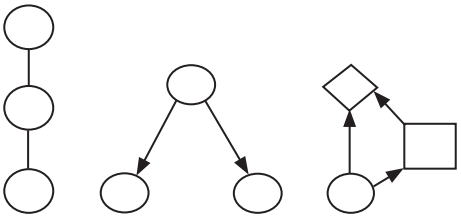
Часть I

Теоретические основы

Первые главы книги содержат общую вводную информацию по вероятностным графовым моделям и излагают теоретические основы, необходимые для чтения остальной части книги, – теорию вероятности и теорию графов.

Глава 1

Введение



1.1 Неопределенность

Для достижения своих целей интеллектуальные агенты (сущности), естественные или искусственные, должны выбирать образ действий из многих возможных вариантов. То есть они должны принимать решение на основе информации, получаемой из окружающей среды, накопленных ранее знаний и конкретных целей (задач). Во многих случаях информация и знания неполные или ненадежные, поэтому результаты этих решений являются неопределенными, т. е. агенты должны принимать решения в условиях неопределенности. Например, врач скорой помощи должен действовать очень быстро, даже если располагает недостаточной информацией о состоянии пациента. Самоуправляемое транспортное средство, определяющее возможные препятствия на своем пути, должно принимать решение о повороте или остановке, не зная точно об удаленности, размере и скорости объекта, препятствующего движению. Финансист должен выбрать наилучший вариант инвестирования в соответствии со своими приблизительными прогнозами ожидаемого оборота средств и дохода при различных альтернативных вариантах и в соответствии с требованиями своих клиентов.

Одна из целей искусственного интеллекта – разработка систем, которые способны обосновывать и принимать решения в условиях неопределенности. Обоснование в условиях неопределенности представляло трудную задачу для первых интеллектуальных систем, так как привычные парадигмы и принципы не очень хорошо подходили для управления неопределенностью.

1.1.1 Воздействие неопределенности

Первые системы искусственного интеллекта были основаны на классической логике, в которой знания можно представить как набор логи-

ческих высказываний или правил. Такие системы обладали двумя важными свойствами: *модульностью* и *монотонностью*, которые помогают упростить получение знаний и логические выводы из них.

Система является *модульной*, если каждый элемент знаний может использоваться независимо для получения логических заключений (выводов). Таким образом, если предпосылки любого логического высказывания (условия) или правила являются истинными, то мы можем получить (вывести) соответствующее заключение без необходимости рассмотрения других элементов базы знаний. Например, если существует правило $\forall X$, *паралич*(X) → *повреждена_рука*(X), тогда если известно, что у *Мери* был паралич, то известно также, что у нее повреждена рука.

Система является *монолитной*, если ее знания всегда расширяются монотонно, т. е. любой дедуктивно выводимый факт или заключение сохраняет свою силу, даже если системе становятся известны новые факты. Например, если в системе существует правило $\forall X$, *птица*(X) → *летает*(X), тогда если *Твити* – птица, то можно сделать вывод, что она летает.

Но если имеется неопределенность этих двух свойств, то в общем случае они не являются истинными. В медицинских системах обычно существует неопределенность диагноза для пациента, так что даже если пациент перенес паралич, возможно, его рука не повреждена. Это зависит от конкретного участка мозга, пораженного параличом. Не все птицы летают, поэтому если в дальнейшем мы узнаем, что *Твити* – пингвин, то необходимо *отменить* заключение о том, что он(а) летает.

Неточность описанных выше пар свойств усложняет систему, которая должна обосновывать выводы и заключения в условиях неопределенности. Теоретически такая система должна учитывать все доступные знания и факты при выводе заключения, а кроме того, непременно должна обладать способностью изменять свои выводы и заключения при получении новых данных.

1.2 Краткая история

В истории искусственного интеллекта можно выделить следующие стадии развития методик управления неопределенностью:

- *начальная стадия* (1950-е и 1960-е гг.) – исследователи искусственного интеллекта (ИИ) сосредоточены на решении таких задач, как доказательство теорем, игра в шахматы и подобные игры, а также область задач планирования «мир многогранников», в которые не включалась неопределенность. Таким образом, не было необходимости в разработке методик управления неопределенно-

стью. В начальной стадии развития ИИ преобладающей являлась символическая парадигма;

- *ad-hoc-методики* (1970-е гг.) – разработка экспертных систем для реальных приложений, например для медицины и горнодобывающей промышленности, потребовала развития методик управления неопределенностью. Новейшие на тот момент ad-hoc-методики разрабатывались для специализированных экспертных систем, например фактор определенности MYCIN [15] и псевдовероятности Prospector [3]. Позже выяснилось, что эти методики содержали набор неявных предположений, которые ограничивали их применяемость [5]. Кроме того, в этот период предлагались альтернативные теории управления неопределенностью в экспертных системах, в том числе нечеткая логика (fuzzy logic) [17] и теория Демпстера–Шафера [14];
- *возрождение вероятности* (1980-е гг.) – теория вероятностей использовалась в некоторых ранних экспертных системах, но несколько позже от нее отказались, так как применение теории вероятностей в ее простейших формах приводило к высокой сложности вычислений (см. раздел 1.3). Новые разработки, в особенности байесовские сети [11], предоставляют возможность создавать сложные вероятностные системы эффективным способом. Таким образом, на этом этапе начинается новая эра в управлении неопределенностью в области ИИ;
- *разнотипный (теоретический) формализм* (1990-е гг.) – продолжается развитие байесовских сетей, кроме того, они объединяются с разработками эффективных алгоритмов логического вывода и обучения. В то же время и другие методики, такие как нечеткая логика и немонотонная логика, рассматриваются как альтернативные варианты обоснования выводов и заключений в условиях неопределенности;
- *вероятностные графовые модели* (2000-е гг.) – несколько методик, основанных на теории вероятностей и представлениях в форме графов, объединяются, образуя мощные методы представления, обоснования и принятия решений в условиях неопределенности. В их числе байесовские сети, марковские сети, диаграммы влияния, марковские процессы принятия решения и другие.

1.3 Основные вероятностные модели

Теория вероятностей предоставляет прочную основу для управления неопределенностью, следовательно, вполне естественно воспользоваться теорией вероятностей для обоснования выводов и заключений в

условиях неопределенности. Но если применять теорию вероятностей в ее простейшей форме для решения сложных задач, то очень скоро нас остановит проблема сложности вычислений.

В этом разделе показано, как можно смоделировать задачу, используя простейший вероятностный метод, основанный на равномерном представлении, а также как можно воспользоваться этим представлением для ответа на некоторые вероятностные запросы. Это поможет лучше понять ограничения такого простейшего метода и объяснит причины, стимулирующие разработку вероятностных графовых моделей[1].

Многие задачи можно сформулировать в виде набора (множества) переменных $X_1, X_2, \ldots, X_n$, при этом мы знаем значения некоторых переменных, тогда как другие неизвестны. Например, в медицинской диагностике переменные могут представлять определенные симптомы и соответствующие болезни. Обычно нам известны конкретные симптомы, и необходимо определить (найти) наиболее вероятную болезнь (или болезни). Другим примером может послужить разработка в финансовой сфере системы, которая помогает принять решение о сумме кредита, выдаваемого конкретному клиенту. В этом случае значимыми переменными являются атрибуты (характеристики) этого клиента, т. е. возраст, доходы, ранее выданные кредиты и т. д., а также переменная, содержащая сумму выдаваемого кредита. На основе атрибутов (характеристик) конкретного клиента необходимо определить, например, максимальную сумму кредита, которую можно предоставить этому клиенту без особого риска. Вообще говоря, существует несколько типов задач, которые можно смоделировать таким способом, – это задачи диагностики, классификации, распознавания и некоторые другие.

В вероятностной рабочей среде можно считать, что каждый атрибут задачи представляет собой случайную переменную, так что есть возможность получить определенное значение из набора (множества) значений[2]. Будем считать, что это множество возможных значений является конечным, например $X = \{x_1, x_2, \ldots, x_m\}$ может представлять m возможных заболеваний в области медицинской диагностики. Каждое значение случайной переменной будет иметь определенную вероятность, связанную с внутренним контекстом. В рассматриваемом здесь примере случайной переменной X это может быть вероятность возникновения каждого заболевания в определенной группе населения (это называют частотой заболеваний или просто заболеваемостью), т. е. $P(X = x_1)$, $P(X = x_2)$, …, или в сокращенной форме $P(x_1)$, $P(x_2)$, ….

Если рассматриваются две случайные переменные X и Y, то можно вычислить вероятность получения определенного значения переменной X и вероятность получения определенного значения переменной Y,

[1] В этом и в следующих разделах предполагается, что читатель знаком с некоторыми основными концепциями теории вероятностей. Обзор этих и других концепций приведен в главе 2.

[2] Формальное определение случайной переменной будет дано несколько позже.

т. е. $P(X = x_i \wedge Y = y_j)$ или просто $P(x_i, y_j)$. Это называют совместной вероятностью переменных X и Y. Данный принцип можно обобщить для n случайных переменных, где совместная вероятность обозначается как $P(X_1, X_2, \ldots, X_n)$. Можно интерпретировать выражение $P(X_1, X_2, \ldots, X_n)$ как функцию, которая присваивает значение вероятности всем возможным сочетаниям значений переменных $X_1, X_2, \ldots, X_n$.

Таким образом, можно представить любую предметную область как:

1) множество случайных переменных $X_1, X_2, \ldots, X_n$;
2) совместное распределение вероятностей, связанное с этими переменными, $P(X_1, X_2, \ldots, X_n)$.

Приняв такое представление, можно ответить на некоторые запросы с учетом значений конкретных переменных в рассматриваемой предметной области, например:

- *частное (или маргинальное) распределение вероятностей* – вероятность одной из переменных принимает конкретное значение. Это можно вычислить, суммируя значения всех остальных переменных из совместного распределения вероятностей. Другими словами, $P(X_i) = \sum_{\forall X \neq X_i} P(X_1, X_2, \ldots, X_n)$. Это называют маргинализацией. Маргинализацию можно обобщить для получения частного распределения вероятностей некоторого подмножества переменных путем суммирования значений всех остальных переменных;
- *условные вероятности* – по определению условная вероятность X_i с учетом того, что нам известно значение X_j, равна $P(X_i \mid X_j) = P(X_i, X_j) / P(X_j)$, $P(X_j) \neq 0$. Значения $P(X_i, X_j)$ и $P(X_j)$ можно получить с помощью маргинализации, и с помощью этих значений вычисляются условные вероятности;
- *полная абдукция* – средняя ошибка в процентах (MPE) – если подмножество (E) переменных известно, то абдукция заключается в поиске значений остальных переменных (J), которые максимизируют условную вероятность, давая обоснование, $\max P(J \mid E)$. Таким образом, $ArgMax_J [P(X_1, X_2, \ldots, X_n) / P(E)]$;
- *частичная абдукция* – оценка апостериорного максимума (MAP) – в этом случае существуют три подмножества переменных: обоснование E, запрашиваемые переменные, которые необходимо максимизировать, J и остальные переменные K, так что необходимо максимизировать $P(J \mid E)$. Результат получается путем маргинализации на подмножестве K и максимизации на подмножестве J, т. е. $ArgMax_J [\sum_{X \in K} P(X_1, X_2, \ldots, X_n) / P(E)]$.

Кроме того, если имеются данные из интересующей нас предметной области, то можно получить модель по этим данным, т. е. оценку совместного распределения вероятностей для значимых переменных.

В следующем разделе рассматривается простой пример применения базовой методики.

1.3.1 Пример

Воспользуемся обычным примером определения возможности игры в гольф для демонстрации простой методики. В этой задаче имеется пять переменных: прогноз погоды, температура, влажность, ветер, игра. В табл. 1.1 показаны некоторые данные для этого примера. Все переменные дискретные, так что они могут содержать значения из конечного множества возможных значений, например прогноз может быть одним из следующих: солнечно, облачно или дождь. Теперь рассмотрим, как можно выполнить описанные выше различные вероятностные запросы для этого примера.

Таблица 1.1. Выборка данных для примера определения возможности игры в гольф

Прогноз	Температура	Влажность	Ветер	Игра
Солнечно	Высокая	Высокая	Ложь	Нет
Солнечно	Высокая	Высокая	Истина	Нет
Облачно	Высокая	Высокая	Ложь	Да
Дождь	Средняя	Высокая	Ложь	Да
Дождь	Низкая	Нормальная	Ложь	Да
Дождь	Низкая	Нормальная	Истина	Нет
Облачно	Низкая	Нормальная	Истина	Да
Солнечно	Средняя	Высокая	Ложь	Нет
Солнечно	Низкая	Нормальная	Ложь	Да
Дождь	Средняя	Нормальная	Ложь	Да
Солнечно	Средняя	Нормальная	Истина	Да
Облачно	Средняя	Высокая	Истина	Да
Облачно	Высокая	Нормальная	Ложь	Да
Дождь	Средняя	Высокая	Истина	Нет

Сначала упростим пример, используя только две переменные – Прогноз и Температура. По данным из табл. 1.1 можно вычислить совместное распределение вероятностей для Прогноза и Температуры, как показа-

но в табл. 1.2. Каждая запись (строка) табл. 1.2 соответствует совместному распределению вероятностей P(*Прогноз*, *Температура*), например P(*Прогноз* = *С*, *Температура* = *В*) = 0.143.

Таблица 1.2. Совместное распределение вероятностей для переменных Прогноз и Температура

Прогноз	Температура		
	В	С	Н
С	0.143	0.143	0.071
О	0.143	0.071	0.071
Д	0	0.214	0.143

Сначала получим частное (маргинальное) распределение вероятностей для этих двух переменных. Если выполнить суммирование по строкам (маргинализация переменной Температура), то получим частное распределение вероятностей для переменной Прогноз P(*Прогноз*) = [0.357, 0.286, 0.357]. Если просуммировать столбцы, то получим частное распределение вероятностей для переменной Температура P(*Температура*) = [0.286, 0.428, 0.286]. Из вычисленных распределений получаем, что наиболее вероятной температурой является *С* (средняя), а для прогноза наиболее вероятны значения *С* (солнечно) и *Д* (дождь).

Теперь можно вычислить условные вероятности для переменной Прогноз при заданной температуре и для переменной Температура при заданном прогнозе. Например:

$$P(\textit{Температура} \mid \textit{Прогноз} = Д) = P(\textit{Температура} \wedge \textit{Прогноз} = Д) / P(\textit{Прогноз} = Д) = [0, 0.6, 0.4];$$

$$P(\textit{Прогноз} \mid \textit{Температура} = Н) = P(\textit{Прогноз} \wedge \textit{Температура} = Н) / P(\textit{Температура} = Н) = [0.25, 0.25, 0.5].$$

С учетом вычисленных распределений при прогнозе дождливой погоды наиболее вероятной является средняя температура, а при низкой температуре наиболее вероятно прогнозирование дождя.

Наконец, наиболее вероятным сочетанием значений переменных Прогноз и Температура является {*Дождь*, *Средняя*}. Это сочетание в данном примере можно получить непосредственно из таблицы совместного распределения вероятностей.

Несмотря на возможность выполнения разнообразных вероятностных запросов для этого небольшого примера, описанная выше методика становится неприемлемой для сложных задач с многочисленными переменными, так как размер таблицы и объем прямых вычислений

частных и условных вероятностей возрастает экспоненциально при увеличении числа переменных в модели.

Еще один недостаток этой простейшей методики состоит в том, что для получения правильных оценок совместного распределения вероятностей по данным потребуется очень большая база данных, если количество переменных в модели велико. Существует простое практическое правило: число экземпляров (записей) как минимум в 10 раз больше числа возможных сочетаний значений переменных в модели, так что если рассматривается 50 бинарных переменных, то потребуется не менее 10×2^{50} экземпляров (записей в базе данных).

Кроме того, таблица совместного распределения вероятностей дает слишком мало информации о задаче человеку, т. е. эта методика имеет еще и ограничения по восприятию и пониманию (когнитивности).

Перечисленные выше недостатки рассматриваемой в этом разделе простейшей методики являются одним из стимулов развития вероятностных графовых моделей.

1.4 Вероятностные графовые модели

Вероятностные графовые модели (ВГМ) предоставляют рабочую среду для управления неопределенностью на основе теории вероятностей с выполнением вычислений наиболее эффективным способом. Основная идея состоит в рассмотрении только тех отношений независимости, которые являются действительными и обоснованными для конкретной задачи, и включении только этих отношений в вероятностную модель для уменьшения сложности по критериям требований к объему памяти и времени вычислений. Естественным способом представления отношений зависимости и независимости в множестве переменных является использование графов. Переменные, которые связаны прямой зависимостью, соединяются ребрами (дугами), а отношения независимости неявно присутствуют в таком графе зависимостей.

Вероятностная графовая модель – это компактное представление совместного распределения вероятностей, из которого можно получить частное распределение вероятностей и условные вероятности. Вероятностная графовая модель обладает следующими преимуществами по сравнению с простым традиционным представлением:

- в общем гораздо более компактная модель (по объему памяти);
- в общем гораздо более эффективная модель (по времени вычислений);
- проще для понимания и получения информации;
- проще для обучения на различных формах данных или для создания на основе экспертных знаний.

Вероятностная графовая модель определяется двумя компонентами:

1) граф $G(V, E)$, который определяет структуру модели;
2) набор локальных функций $f(Y_i)$, которые определяют параметры, где Y_i – подмножество X.

Совместное распределение вероятностей вычисляется как произведение локальных функций:

$$P(X_1, X_2, \ldots, X_N) = K \prod_{i=1}^{M} f(Y_i), \tag{1.1}$$

где K – константа нормализации (приводит сумму вероятностей к единице).

Такое представление в форме графа и набор локальных функций (называемых потенциальными функциями) являются основой для логического вывода и обучения в ВГМ.

Логический вывод (inference): получение частного распределения вероятностей или условных вероятностей на любом подмножестве переменных Z с учетом любого другого подмножества Y.

Обучение: при заданном наборе данных – значений переменной X (набор может быть неполным) определяется оценка структуры (графа) и параметров (локальных функций) модели.

Вероятностные графовые модели можно классифицировать по следующим трем характеристикам:

1) направленные или ненаправленные (ориентированные или неориентированные);
2) статические или динамические;
3) вероятностные или предназначенные для принятия решений.

Первая характеристика определяет тип графа, используемого для представления отношений зависимости. Ненаправленные (неориентированные) графы представляют симметричные отношения, тогда как направленные (ориентированные) графы представляют отношения, в которых направление важно. В заданном множестве случайных переменных с соответствующими условными отношениями зависимостей невозможно представить все отношения с помощью одного типа графа [11], следовательно, необходимы оба типа моделей.

Вторая характеристика определяет, представляет ли модель множество переменных в конкретный момент времени (статическая модель) или в течение различных интервалов времени (динамическая модель). Вероятностные модели включают только случайные переменные, а модели для принятия решений также содержат параметры принятия решения и переменные полезности.

Самые распространенные классы ВГМ и их типы в соответствии с приведенной выше классификацией перечислены в табл. 1.3.

Таблица 1.3. Основные типы вероятностных графовых моделей

Тип модели	Направленный / Ненаправленный граф	Статическая / Динамическая	Вероятностная / Для принятия решений
Байесовские классификаторы	Напр. / Ненапр.	Статическая	Вероятностная
Марковские цепи	Напр.	Динамическая	Вероятностная
Скрытые марковские модели	Напр.	Динамическая	Вероятностная
Марковские случайные поля	Ненапр.	Статическая	Вероятностная
Байесовские сети	Напр.	Статическая	Вероятностная
Динамические байесовские сети	Напр.	Динамическая	Вероятностная
Диаграммы влияния	Напр.	Статическая	Для принятия решений
Марковские процессы принятия решений (МППР)	Напр.	Динамическая	Для принятия решений
Частично наблюдаемые МППР	Напр.	Динамическая	Для принятия решений

Все эти типы моделей будут подробно рассматриваться в следующих главах. Кроме того, будут также описаны некоторые расширения, которые считаются более выразительными моделями (реляционные вероятностные графовые модели) или представляют причинно-следственные отношения (причинно-следственные байесовские сети).

1.5 Представление, логический вывод и обучение

Для каждого класса вероятностной графовой модели существуют три аспекта: представление, логический вывод и обучение.

Представление (representation) – это основное свойство каждой модели, которое определяет, какие сущности (объекты) образуют модель и как эти сущности связаны между собой. Например, все вероятностные графовые модели могут быть представлены как графы, которые определяют структуру модели, и локальные функции, описывающие

параметры модели. Но тип графа и локальные функции различны для разных типов моделей.

Логический вывод (inference) заключается в ответах на разнообразные вероятностные запросы и основан на самой модели и некотором доказательстве (подтверждении). Например, получение апостериорного распределения вероятностей переменной или набора переменных, при условии что другие переменные в этой модели известны. Главная трудность – как сделать это эффективно.

Для создания таких моделей существует два основных способа: создание «вручную» с помощью экспертов в рассматриваемой предметной области или вывод модели методом индукции из данных. В последние годы особое внимание уделялось методике вывода модели методом индукции на основе методик машинного обучения, поскольку создание модели с помощью экспертов представляет собой сложную и дорогостоящую задачу. В частности, определение параметров для моделей обычно выполняется на основе данных, так как люди склонны давать неудачные критерии оценки вероятностей.

Важным свойством этих методик с практической прикладной точки зрения является возможность отделения процессов логического вывода и методик обучения от модели. Таким образом, как и в других представлениях методик искусственного интеллекта, таких как логика и правила вывода, механизмы обоснования (доказательства) являются общими и могут применяться к различным моделям. В результате методики, разработанные для вероятностного логического вывода в каждом классе вероятностных графовых моделей, можно напрямую применять для различных моделей в разнообразных приложениях.

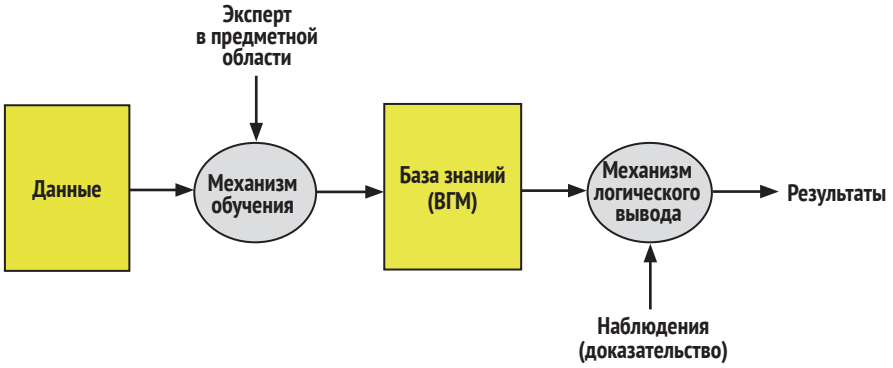


Рис. 1.1. Схематическое представление общего принципа, которому подчиняются различные классы ВГМ: четкое разделение обобщенных механизмов обучения и логического вывода и базы знаний, зависящей от конкретного приложения

Этот главный теоретический принцип схематически показан на рис. 1.1. На основе данных, или знаний эксперта, или сочетания обо-

их источников база знаний – в данном случае вероятностная графовая модель создается с использованием механизма обучения. После создания модели можно применять ее для выполнения вероятностного обоснования с помощью механизма логического вывода. Механизм логического вывода на основе наблюдений и самой модели извлекает результаты. Механизмы обучения и логического вывода являются общими для любого класса вероятностной графовой модели, поэтому могут применяться для моделирования и обоснования в различных предметных областях.

Для каждого типа вероятностной графовой модели, представленной в этой книге, сначала будет описано ее представление, затем будут рассматриваться некоторые из наиболее часто используемых методик логического вывода и обучения.

1.6 Приложения

В большинстве реальных задач подразумевается работа в условиях неопределенности, и в таких задачах обычно существует большое количество факторов или переменных, которые необходимо учитывать при решении. Вероятностные графовые модели формируют идеальную рабочую среду для решения сложных задач в условиях неопределенности, поэтому могут применяться в широком диапазоне предметных областей, включая следующие:

- медицинская диагностика и принятие решений;
- определение местонахождения, навигация и планирование действий мобильного роботизированного устройства;
- диагностика сложного промышленного оборудования, например турбин и оборудования электростанций;
- моделирование пользователем адаптивных интерфейсов и интеллектуальных обучающих средств;
- распознавание речи и обработка естественных языков;
- моделирование и прогнозирование загрязнения окружающей среды;
- анализ надежности и устойчивости сложных процессов;
- моделирование эволюционирования вирусов;
- распознавание объектов в системах компьютерного зрения;
- извлечение информации;
- рынки энергоресурсов.

Для различных приложений в большей или меньшей степени подходят те или иные типы вероятностных графовых моделей, как будет показано в следующих главах, где представлены примеры приложений для каждого класса вероятностных графовых моделей.

1.7 Обзор содержимого книги

Книга разделена на четыре части.

В части I изложены математические основы для изучения и понимания моделей и методик, представленных в последующих главах. Глава 2 содержит обзор некоторых основных концепций теории вероятностей и теории информации, наиболее важных для понимания вероятностных графовых моделей. В главе 3 приводится общий обзор теории графов с выделением некоторых аспектов, которые важны для представления и логического вывода в вероятностных графовых моделях, – рассматриваются клики, хордальные (триангулированные) графы, совершенная упорядоченность и т. п.

В части II описаны различные типы вероятностных моделей, которые содержат только случайные переменные, но в этих моделях не рассматривается принятие решений или полезность. Это самая большая часть книги, которая включает описание следующих типов вероятностных графовых моделей:

- байесовские классификаторы;
- марковские цепи и скрытые марковские модели;
- марковские случайные поля;
- байесовские сети;
- динамические байесовские сети и временны́е сети.

Каждый тип модели рассматривается в отдельной главе (за исключением байесовских сетей, описание которых разделено на две главы), включая представление, логический вывод и обучение, а также примеры практических приложений.

В части III представлены модели, реализующие принятие решений и оценивающие полезность, поэтому основное внимание уделяется помощи механизма принятия решений для определения оптимальных действий в условиях неопределенности. Эта часть содержит две главы. В первой главе рассматриваются методики моделирования для выбора одного или нескольких решений, включая деревья решений и диаграммы влияния. Вторая глава посвящена последовательному принятию решений, в особенности марковским процессам принятия решений.

В части IV рассматриваются другие теоретические принципы, которые можно считать расширениями обычных вероятностных графовых моделей. В этой части также две главы. В первой главе рассматриваются реляционные вероятностные модели, которые увеличивают мощность представления стандартных вероятностных графовых моделей, объединяя выразительную мощь логики первого порядка с возможностями объяснения и обоснования неопределенности вероятностных моделей. Во второй главе представлены причинно-следственные графовые мо-

дели, которые существенно усиливают представление вероятностных зависимостей для выражения отношений причин и следствий (результатов).

1.8 Материалы для дополнительного чтения

В этой книге тема вероятностных графовых моделей рассматривается в широком аспекте. Некоторые другие книги не менее подробно освещают эту тему. Коллер (Koller) и Фридман (Friedman) [7] представляют модели различной структуры, уделяя меньше внимания приложениям. В книге Лауритцена (Lauritzen) [8] в большей степени рассматривается статистический аспект. Байесовское программирование [1] предоставляет другой подход к реализации графовых моделей на основе парадигмы программирования.

Кроме того, существует несколько книг, в которых более подробно рассматривается один или несколько типов моделей: байесовские сети [2, 10, 11], графы решений [6], марковские случайные поля [9], марковские процессы принятия решений [13], реляционные вероятностные модели [4] и причинно-следственные модели [12, 16].

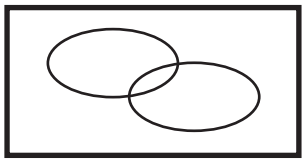
Ссылки на источники

1. Bessiere, P., Mazer, E., Ahuactzin, J. M., Mekhnacha, K. Bayesian Programming. CRC Press, Boca Raton (2014).
2. Darwiche, A. Modeling and Reasoning with Bayesian Networks. Cambridge University Press, New York (2009).
3. Duda, R. O., Hart, P. A., Nilsson, N. L. Subjective Bayesian methods for rule-based inference systems. In: Proceeding of the National Computer Conference, vol. 45, p. 1075–1082 (1976).
4. Getoor, L., Taskar, B. Introduction to Statistical Relational Learning. MIT Press, Cambridge (2007).
5. Heckerman, D. Probabilistic interpretations for MYCIN's certainty factors. In: Proceedings of the First Conference on Uncertainty in Artificial Intelligence (UAI), p. 9–20 (1985).
6. Jensen, F. V. Bayesian Networks and Decision Graphs. Springer, New York (2001).
7. Koller, D., Friedman, N. Probabilistic Graphical Models: Principles and Techniques. MIT Press, Cambridge (2009).
8. Lauritzen, S. L. Graphical Models. Oxford University Press, Oxford (1996).
9. Li, S. Z. Markov Random Field Modeling in Image Analysis. Springer, London (2009).

10. Neapolitan, R. E. Probabilistic Reasoning in Expert Systems. Wiley, New York (1990).
11. Pearl, J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988).
12. Pearl, J. Causality. Models, Reasoning and Inference. Cambridge University Press, New York (2009).
13. Puterman, M. L. Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (1994).
14. Shafer, G. A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976).
15. Shortliffe, E. H., Buchanan, B. G. A model of inexact reasoning in medicine.Math. Biosci. 23, 351–379 (1975).
16. Spirtes, P., Glymour, C., Scheines, R. Causation, Prediction, and Search. MIT Press, New York (2000).
17. Zadeh, L. A. Knowledge Representation in Fuzzy Logic. IEEE Trans. Knowl. Data Eng. 1 (1), 89–100 (1989).

Глава 2

Теория вероятностей



2.1 Введение

Теория вероятностей была создана на основе азартных игр со случайным исходом и имеет долгую и весьма интересную историю. Впоследствии теория вероятностей была преобразована в математический язык для количественной оценки неопределенности.

Рассмотрим конкретный эксперимент, например бросок игральной кости, при котором можно получить различные результаты. Каждый результат будем называть исходом (outcome), или элементарным событием. В примере с бросанием игральной кости возможны следующие исходы, или элементарные события: {1, 2, 3, 4, 5, 6}. Множество всех возможных исходов эксперимента называется пространством элементарных событий Ω. Событие (event) – это множество элементарных событий, или подмножество пространства элементарных событий Ω. Продолжая рассматривать пример с броском игральной кости, отметим, что одним из событий может быть выпадение четного числа, т. е. {2, 4, 6}.

Прежде чем дать математическое определение вероятности, следует обсудить общий смысл или интерпретацию вероятности. Предлагалось несколько определений, или интерпретаций вероятности, начиная с классического определения Лапласа (Laplace) и включая предельную (граничную) частоту, субъективную, логическую и предопределенную интерпретации [1]:

- классическое определение: вероятность следует определять с помощью равновероятных событий. Если некоторый конкретный эксперимент имеет N возможных исходов, то вероятность каждого исхода равна $1/N$;

- логическая интерпретация: вероятность – это мера правильного представления, то есть при наличии доступного подтверждения правильно (рационально) мыслящая личность получает конкретное представление о некотором событии, которое определяет его вероятность;
- субъективная интерпретация: вероятность – это мера личной уверенности (представления) в наступлении определенного события. Эта характеристика может измеряться коэффициентом ставок – вероятность наступления конкретного события для конкретного лица связана с тем, сколько это лицо готово поставить на это событие;
- частотная интерпретация: вероятность – это мера, выраженная числом возникновения некоторого события в конкретном эксперименте, когда количество повторений эксперимента стремится к бесконечности;
- предопределенность: вероятность – это мера, выраженная числом возникновения некоторого события при повторно воспроизводимых условиях, даже если эксперимент выполнялся всего лишь один раз.

Эти интерпретации можно сгруппировать по двум основным подходам, применяемым в теории вероятностей и статистике:

- объективный подход (классическое определение, предельная частота, предопределенность): вероятности существуют в реальном мире и могут быть измерены (вычислены);
- эпистемологический, или гносеологический, подход (логическая и субъективная интерпретации): вероятности должны определяться с помощью человеческих знаний, эти вероятности являются мерами персонального представления (уверенности).

Оба подхода основаны на одних и тех же математических аксиомах, определения которых приведены ниже. Но существуют различия в способе применения вероятности, в особенности в статистическом выводе. Эти различия стали отравным пунктом создания двух основных направлений (школ) в статистике: использование частотной вероятности и байесовская статистика. В области исследований искусственного интеллекта, в особенности в экспертных системах, более предпочтительным является эпистемологический, или субъективный, подход, но объективный подход также применяется [4].

Мы будем рассматривать логический, или нормативный, подход и определение вероятности в терминах степени правдоподобия конкретного утверждения с учетом доступного доказательства (подтвержде-

ния) [2]. На основе работы Кокса (Cox) Джейнс (Janes) устанавливает некоторые основные рекомендации, которые должны соблюдаться при определении степени правдоподобия [2]:

- представление в форме действительных чисел;
- качественное соответствие здравому смыслу;
- логическая целостность (непротиворечивость).

На основе этих трех интуитивных принципов можно вывести следующие три аксиомы вероятности:

1) $P(A)$ – непрерывная монотонная функция в интервале [0,1];
2) $P(A, B \mid C) = P(A \mid C)P(B \mid A, C)$ (правило умножения);
3) $P(A \mid B) + P(\neg A \mid B) = 1$ (правило сложения).

Здесь A, B, C – утверждения (бинарные переменные), а $P(A)$ – вероятность истинности утверждения A. $P(A \mid C)$ – вероятность истинности утверждения A, при условии что C известно. Это называют условной вероятностью. $P(A, B \mid C)$ – это вероятность истинности A И B при условии C (логическая конъюнкция), а $P(\neg A \mid C)$ – вероятность НЕ A (логическое отрицание) при условии C. Эти правила равнозначны весьма часто используемым аксиомам Колмогорова. Из этих аксиом можно вывести все стандартные положения теории вероятностей.

2.2 Основные правила

Вероятность дизъюнкции (логической суммы) двух утверждений определяется правилом сложения: $P(A + B \mid C) = P(A \mid C) + P(B \mid C) - P(A, B \mid C)$. Если утверждения A и B являются взаимоисключающими при заданном C, то правило сложения можно упростить: $P(A + B \mid C) = P(A \mid C) + P(B \mid C)$. Это правило можно обобщить для N взаимоисключающих утверждений:

$$P(A_1 + A_2 + \ldots + A_N \mid C) = P(A_1 \mid C) + P(A_2 \mid C) + \ldots + P(A_N \mid C). \qquad (2.1)$$

Если существует N взаимоисключающих и исчерпывающих гипотез $H_1, H_2, \ldots, H_N$ и если доказательство B не указывает на предпочтение одной из этих гипотез, то в соответствии с принципом индифферентности: $P(H_i \mid B) = 1/N$.

В соответствии с логической интерпретацией *абсолютных* вероятностей не существует, все вероятности условные при наличии некоторой предпосылочной информации[3]. Вероятность $P(H \mid B)$, рассматриваемая

[3] Общепринятой является запись $P(H)$ без явного указания условной информации. В этом случае предполагается, что существует еще и некоторый контекст, в котором рассматриваются вероятности, даже если этот контекст не записан в явной форме.

только вместе с предпосылочной информацией B, называется *априорной* вероятностью. Если в рассмотрение включается некоторая дополнительная информация D, то это называют *апостериорной* вероятностью $P(H \mid D, B)$. По правилу умножения получаем:

$$P(D, H \mid B) = P(D \mid H, B)P(H \mid B) = P(H \mid D, B)P(D \mid B). \quad (2.2)$$

Из этого выражения получаем:

$$P(H \mid D, B) = \frac{P(H \mid B)P(D \mid H, B)}{P(D \mid B)}. \quad (2.3)$$

Последнее равенство (2.3) известно как правило Байеса, а член этой формулы $P(D \mid H, B)$ называют *правдоподобием* (likelihood) $L(H)$.

В некоторых случаях на вероятность гипотезы H не влияет знание дополнительной информации D, поэтому можно считать, что H и D *независимы* при наличии некоторой предпосылки B, следовательно, $P(H \mid D, B) = P(H \mid B)$. В случае когда A и B независимы, правило умножения можно упростить следующим образом: $P(A, B \mid C) = P(A \mid C)P(B \mid C)$. Это упрощенное правило можно обобщить для N взаимно независимых утверждений:

$$P(A_1, A_2, \ldots, A_N \mid B) = P(A_1 \mid B)P(A_2 \mid B) \ldots P(A_N \mid B). \quad (2.4)$$

Если два утверждения независимы с учетом только предпосылочной информации, то они *маргинально* независимы. Но если эти два утверждения независимы с учетом некоторого дополнительного подтверждения E, то они *условно* независимы: $P(H, D \mid B, E) = P(H \mid B, E)$. Например, пусть A представляет утверждение «*поливка сада*», B – *прогноз погоды*, а C – *дождь*. Изначально поливка сада не является независимой от прогноза погоды, но после того, как наблюдается дождь, эти утверждения становятся независимыми. Таким образом (опуская член-предпосылку), $P(A, B \mid C) = P(A \mid C)$.

Вероятностные графовые модели основаны на этих условиях маргинальной и условной независимости.

Вероятность конъюнкции N утверждений, то есть $P(A_1, A_2, \ldots, A_N \mid B)$, обычно называют *совместной* вероятностью (joint probability). Если обобщить правило умножения для N утверждений, то получим известное *цепное* правило (chain rule):

$$P(A_1, A_2, \ldots, A_N \mid B) = P(A_1 \mid A_2, A_3, \ldots, A_N, B)P(A_2 \mid A_3, A_4, \ldots, A_N, B) \ldots P(A_N \mid B). \quad (2.5)$$

Таким образом, совместная вероятность N утверждений может быть определена по этому правилу. Отношения условной независимости

между этими утверждениями могут быть использованы для упрощения приведенного выше произведения. Например, если A_1 и A_2 независимы с учетом $A_3, \ldots, A_N, B$, то первый член в формуле 2.5 можно упростить: $P(A_1 \mid A_3, \ldots, A_N, B)$.

Другим важным отношением является правило (формула) *полной* вероятности (total probability). Рассмотрим множество $B = \{B_1, B_2, \ldots, B_n\}$ в пространстве элементарных событий Ω, такое, что $\Omega = B_1 \cup B_2 \cup \ldots \cup B_n$ и $B_i \cap B_j = \emptyset$. Таким образом, B – это множество взаимоисключающих событий, которые покрывают все пространство элементарных событий. Рассмотрим другое событие A, равное объединению всех его пересечений с каждым событием множества: $A = (B_1 \cap A) \cup (B_2 \cap A) \cup \ldots \cup (B_n \cap A)$. Затем на основании аксиом вероятности и определения условной вероятности можно вывести правило (формулу) полной вероятности:

$$P(A) = \sum_i P(A \mid B_i)P(B_i). \tag{2.6}$$

С учетом формулы полной вероятности можно переписать формулу правила Байеса в следующем виде (опуская член-предпосылку):

$$P(B \mid A) = \frac{P(B)P(B \mid A)}{\sum_i P(A \mid B_i)P(B_i)}. \tag{2.7}$$

Формула 2.7 общеизвестна как теорема Байеса.

2.3 Случайные переменные

Если рассматривать конечное множество исчерпывающих и взаимоисключающих утверждений[4], то дискретная переменная X может представлять это множество утверждений, такое, что каждое значение x_i этой переменной X соответствует одному утверждению. Если присвоить числовое значение каждому утверждению x_i, то X – это дискретная случайная переменная. Например, исход броска игральной кости представляет собой дискретную случайную переменную с шестью возможными значениями 1, 2, ..., 6. Вероятности для всех возможных значений X, то есть $P(X)$ – это распределение вероятностей X. Продолжим рассмотрение примера с игральной костью, тогда для идеальной («честной») игральной кости распределение вероятностей будет следующим:

x	1	2	3	4	5	6
$P(x)$	1/6	1/6	1/6	1/6	1/6	1/6

[4] Это означает, что одно и только одно из этих утверждений имеет значение ИСТИНА (TRUE).

Это пример равномерного распределения вероятностей. Существуют и определения некоторых других распределений вероятностей. Широко известно биномиальное распределение. Предположим, что имеется урна с N шарами красного и черного цвета, из которых M красные, то есть доля красных шаров $\pi = M/N$. Случайным образом извлекается шар, записывается его цвет, после чего шар возвращается в урну. Шары снова перемешиваются (поэтому теоретически каждая операция извлечения шара не зависит от предыдущей операции извлечения). Вероятность получения r красных шаров за n извлечений равна:

$$P(r \mid n, \pi) = \binom{n}{r} \pi^r (1 - \pi)^{n-r}, \tag{2.8}$$

где $\binom{n}{r} = \frac{n!}{r!(n-r)!}$.

Это пример биномиального распределения, которое применяется, если существует n независимых испытаний, при этом каждое испытание имеет два возможных исхода (успех или промах), а вероятность успеха постоянна во всех испытаниях. Существует много других распределений. Заинтересованным читателям рекомендуем ознакомиться с материалами для дополнительного чтения в конце главы.

Существуют две важные числовые величины, которые в целом помогают охарактеризовать распределение вероятности. Ожидаемое значение, или *ожидание* (expectation), дискретной случайной переменной – это среднее арифметическое возможных значений, взвешенных в соответствии с их вероятностями:

$$E(X \mid B) = \sum_{i=1}^{N} P(x_i \mid B) x_i. \tag{2.9}$$

Дисперсия (variance) определяется как ожидаемое значение квадрата разности значения переменной и ее ожидания:

$$Var(X \mid B) = \sum_{i=1}^{N} P(x_i \mid B)(x_i - E(X))^2. \tag{2.10}$$

В общем, дисперсия предоставляет меру *ширины* или *узости* интервала распределения вероятностей для конкретной случайной переменной. Квадратный корень из дисперсии называется стандартным отклонением. Обычно стандартное отклонение более понятно, так как его единицы измерения те же, что и для исследуемой переменной.

До настоящего момента рассматривались дискретные переменные, но правила определения вероятностей распространяются и на непре-

рывные переменные. Если имеется непрерывная переменная X, то можно разделить ее значения на множество взаимно исключающих полных интервалов, таких, что $P = (a < X \leqslant b)$ – утверждение, следовательно, выведенные выше правила можно применять и в этом случае. *Непрерывная случайная переменная* может быть определена в терминах *функции плотности вероятности* (или просто плотности вероятности – probability density) $f(X \mid B)$ следующим образом:

$$P(a < X \leqslant b) = \int_a^b f(X \mid B)dx. \tag{2.11}$$

Функция плотности вероятности непременно должна соответствовать следующему условию:

$$\int_{-\infty}^{\infty} f(X \mid B)dx = 1.$$

Примером непрерывного распределения вероятности является нормальное распределение, или распределение Гаусса. Это распределение играет важную роль во многих приложениях теории вероятностей и статистики, поскольку многие явления природы приблизительно соответствуют нормальному распределению. Этот тип распределения предпочтителен и в вероятностных графовых моделях из-за своих математических свойств.

Нормальное распределение обозначается как $N(\mu, \sigma^2)$, где μ – среднее значение (меана, центральное значение), а σ – *стандартное отклонение*; определяется следующей формулой:

$$f(X \mid B) = \frac{1}{\sigma\sqrt{2\pi}} exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}. \tag{2.12}$$

Функция плотности вероятности распределения Гаусса показана на рис. 2.1.

Еще одним важным непрерывным распределением является экспоненциальное распределение. Например, время наработки конкретного элемента оборудования до критического отказа обычно моделируется экспоненциальным распределением. Экспоненциальное распределение обозначается как $Exp(\beta)$ и имеет единственный параметр $\beta > 0$; определяется следующей формулой:

$$f(X \mid B) = \frac{1}{\beta} e^{-x/\beta}, x > 0. \tag{2.13}$$

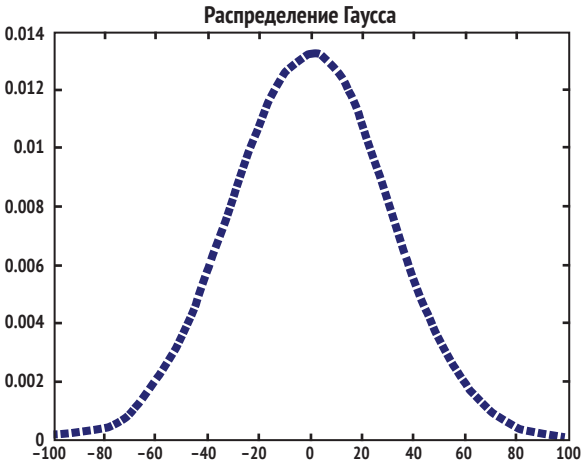


Рис. 2.1. Функция плотности вероятности распределения Гаусса

Пример функции плотности вероятности экспоненциального распределения показан на рис. 2.2.

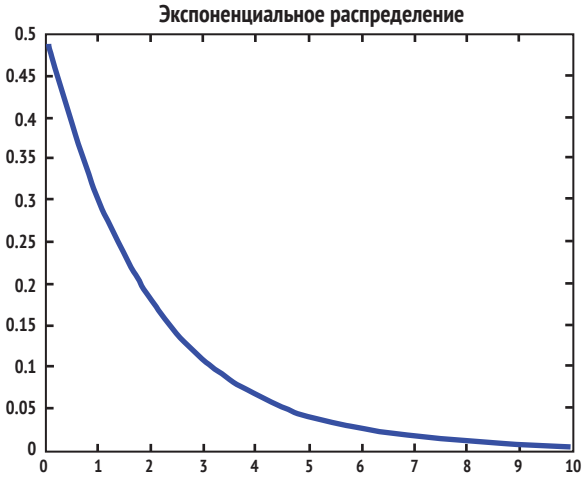


Рис. 2.2. Функция плотности вероятности экспоненциального распределения

Общепринятым считается представление распределений вероятности, в особенности для непрерывных переменных, с использованием (интегральной) функции распределения (cumulative distribution function) F. Функция распределения для случайной переменной X определяет вероятность, такую, что $X \leqslant x$. Для непрерывной переменной функция распределения определяется через функцию плотности вероятности:

$$F(x) = \int_{-\infty}^{x} f(X)dx. \tag{2.14}$$

Ниже перечислены некоторые свойства (интегральных) функций распределения:

- значения расположены в интервале [0, 1]: $0 \leqslant F(X) \leqslant 1$;
- неубывающие функции: $F(X_1) < F(X_2)$, если $X_1 < X_2$;
- пределы: $lim_{x \to -\infty} = 0$ и $lim_{x \to \infty} = 1$.

Для дискретных переменных интегральная (суммарная) вероятность $P(X \leqslant x)$ определяется по следующей формуле:

$$\mathbf{P}(x) = \sum_{X=-\infty}^{X=x} P(X). \tag{2.15}$$

Свойства аналогичны свойствам (интегральной) функции распределения.

2.3.1 Двумерные случайные переменные

Концепцию случайной переменной можно расширить до двух и более измерений. Если даны две случайные переменные X и Y, то их совместное распределение вероятностей определяется как $P(x, y) = P(X = x \wedge Y = y)$. Например, X может представлять количество продукции, выпущенное за один день на конвейере один, а Y – количество продукции, выпущенное за один день на конвейере два. Тогда $P(x, y)$ соответствует вероятности выпуска x единиц продукции на конвейере один и выпуска y единиц продукции на конвейере два. Вероятность $P(X, Y)$ обязательно должна подчиняться аксиомам теории вероятностей, таким образом: $0 \leqslant P(x, y) \leqslant 1$ и $\sum_x \sum_y P(X, Y) = 1$.

Распределение для двумерных дискретных случайных переменных (известное как двумерное распределение – bivariate distribution) может быть представлено в табличной форме. Например, рассмотрим тот же пример с двумя промышленными конвейерами и предположим, что первый конвейер (X) может выпускать 1, 2 или 3 единицы продукции в день, а второй конвейер (Y) – 1 или 2 единицы продукции. Тогда возможное совместное распределение $P(X, Y)$ показано в табл. 2.1.

Таблица 2.1. Пример двумерного дискретного распределения вероятностей

	$X = 1$	$X = 2$	$X = 3$
$Y = 1$	0.1	0.3	0.3
$Y = 2$	0.2	0.1	0

Принимая во внимание совместное распределение вероятностей $P(X, Y)$, можно получить распределение для каждой отдельной случай-

ной переменной, которую называют частной (маргинальной) вероятностью:

$$P(x) = \sum_{y} P(X, Y); \; P(y) = \sum_{x} P(X, Y). \tag{2.16}$$

Например, если рассматривать совместное распределение в табл. 2.1, то можно получить совместные распределения для X и Y. В рассматриваемом здесь примере $P(X = 2) = 0.3 + 0.1 = 0.4$ и $P(Y = 1) = 0.1 + 0.3 + 0.3 = 0.7$.

Также можно вычислить условные вероятности для X с учетом Y и для Y с учетом X:

$$P(x \mid y) = P(x, y)/P(y); \; P(y \mid x) = P(x, y)/P(x). \tag{2.17}$$

Снова вернемся к примеру в табл. 2.1:

$$P(X = 3 \mid Y = 1) = P(X = 3, Y = 1)/P(Y = 1) = 0.3/0.7 = 0.4286.$$

Принцип независимости можно применить и к двумерным случайным переменным. Две случайные переменные X, Y независимы, если их совместное распределение вероятностей равно произведению их частных распределений (для всех значений X и Y):

$$P(X, Y) = P(X)P(Y) \rightarrow Independent(X, Y). \tag{2.18}$$

Еще одна полезная числовая характеристика называется *корреляцией* (correlation) – это количественная мера степени линейной зависимости между двумя случайными переменными X, Y, которая определяется следующей формулой:

$$\rho(X, Y) = E\{[X - E(X)][Y - E(Y)]\} / (\sigma_x \sigma_y), \tag{2.19}$$

где $E(X)$ – ожидаемое значение X, а σ_x – стандартное отклонение для этой переменной. Значения корреляции находятся в интервале $[-1, 1]$. Положительная корреляция означает, что если X возрастает, то Y стремится к возрастанию. Отрицательная корреляция означает, что если X возрастает, то Y стремится к убыванию.

Следует отметить, что корреляция, равная нулю, не обязательно подразумевает независимость, так как корреляция служит мерой только линейной зависимости. Поэтому возможна ситуация, когда переменные X и Y имеют нулевую корреляцию, но связаны через функцию более высокого порядка, следовательно, не могут считаться независимыми.

2.4 Теория информации

Теория информации возникла в области обмена данными (с использованием средств коммуникации), тем не менее она связана со многи-

ми другими дисциплинами. В сфере вероятностных графовых моделей теория информации в основном применяется для обучения. В этом разделе будут рассматриваться основополагающие концепции теории информации.

Предположим, что мы передаем данные о возникновении некоторого конкретного события. Теоретически можно считать, что количество *информации* при передаче данных о событии есть обратная величина по отношению к вероятности этого события. Например, будем считать, что передается сообщение, информирующее об одном из следующих событий:

1) в Нью-Йорке дождь;
2) в Нью-Йорке произошло землетрясение;
3) на Нью-Йорк Сити упал метеорит.

Вероятность первого события больше вероятности второго, а вероятность второго события больше вероятности третьего. Таким образом, сообщение о событии 1 содержит наименьшее количество информации, а сообщение о событии 3 предоставляет наибольшее количество информации.

Теперь рассмотрим, как можно формализовать такую концепцию информации. Предположим, что имеется некоторый источник информации, который может передавать q возможных сообщений $m_1, m_2, \ldots, m_q$. При этом каждое сообщение соответствует событию с вероятностью $P_1, P_2, \ldots, P_q$. Необходимо найти функцию $I(m)$ на основе вероятности m. Эта функция обязательно должна соответствовать следующим свойствам (условиям):

- информация имеет диапазон от нуля до бесконечности: $I(m) \geqslant 0$;
- количество информации увеличивается при уменьшении вероятности: $I(m_i) > I(m_j)$, если $P(m_i) < P(m_j)$;
- количество информации стремится к бесконечности, если вероятность стремится к нулю: $I(m) \to \infty$, если $P(m) \to 0$;
- количество информации в двух сообщениях равно сумме количеств информации в каждом отдельном сообщении, если сообщения (и соответствующие события) являются независимыми: $I(m_i + m_j) = I(m_i) + I(m_j)$, если m_i независимо от m_j.

Функция, удовлетворяющая все перечисленные выше условия (свойства), – это логарифм величины, обратной вероятности, то есть:

$$I(m_k) = log(1/P(m_k)). \tag{2.20}$$

Чаще всего используют логарифмы по основанию два, а информацию измеряют в битах (bit – binary digit):

$$I(m_k) = log_2(1/P(m_k)). \tag{2.21}$$

Например, если предположить, что вероятность сообщения m_r «в Нью-Йорке дождь» равна $P(m_r) = 0.25$, то соответствующее количество информации равно $I(m_r) = log_2(1/0.25) = 2$.

После определения количества информации в конкретном сообщении необходимо рассмотреть еще одну важную концепцию – среднее количество информации в q сообщениях, то есть ожидаемое значение количества информации, также называемой *энтропией* (entropy). В соответствии с определением ожидаемого значения среднее количество информации в q сообщениях, или энтропия, определяется следующей формулой:

$$H(m) = E(I(m)) = \sum_{i=1}^{i=q} P(m_i) log_2(1/P(m_i)). \tag{2.22}$$

Это можно интерпретировать следующим образом: в среднем будет передано H бит информации.

Возникает резонный вопрос: когда H будет иметь максимальное и минимальное значения? Рассмотрим бинарный источник, такой, что существуют только два сообщения m_1 и m_2 с вероятностями $P(m_1) = p_1$ и $P(m_2) = p_2$. Принимая во внимание, что существуют лишь два возможных сообщения, $p_2 = 1 - p_1$, так что H зависит только от одного параметра p_1 (или просто p). На рис. 2.3 показан график H в зависимости от значений p. На графике можно видеть, что H достигает максимума при $p = 0.5$, а минимальное значение H (ноль) соответствует значениям $p = 0$ и $p = 1$. В общем случае энтропия достигает максимума, когда существует равномерное распределение вероятностей для соответствующих событий. Энтропия минимальна, когда только один элемент имеет вероятность, равную единице, а все прочие элементы имеют нулевую вероятность.

Если рассматривать условные вероятности, то можно расширить концепцию энтропии *до условной энтропии* (conditional entropy):

$$H(X \mid y) = \sum_{i=1}^{i=q} P(X_i \mid y) log_2[1/P(X_i \mid y)]. \tag{2.23}$$

Еще одно расширение концепции энтропии называется *перекрестной энтропией* (cross entropy):

$$H(X, Y) = \sum_X \sum_Y P(X, Y) log_2[P(X, Y)/P(X)P(Y)]. \tag{2.24}$$

Перекрестная энтропия предоставляет меру взаимной информации (зависимости) между двумя случайными переменными. Перекрестная энтропия равна нулю, если эти две переменные независимы.

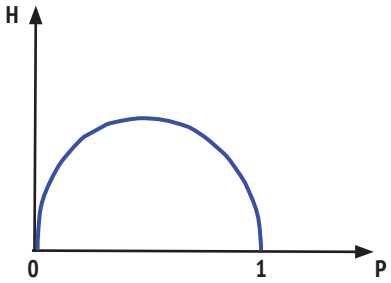


Рис. 2.3. Связь энтропии и вероятности для бинарного источника. Энтропия достигает максимума при вероятности 0.5. Энтропия минимальна при значениях вероятности ноль и единица

2.5 Материалы для дополнительного чтения

Доналд Джиллис (Donald Gillies) [1] предоставляет самое полное описание разнообразных методов и подходов к определению и использованию вероятности. Превосходная книга по теории вероятностей с точки зрения логики [2]. Вассерман (Wasserman) [5] подготовил краткий курс по теории вероятностей и статистике для обучающихся по специальностям информатика и компьютерная инженерия. Предлагается также несколько книг по теории информации, в одной из них устанавливается связь теории информации с машинным обучением и логическим выводом [3].

2.6 Задания и упражнения

1. Какова вероятность того, что при повторных бросках двух игральных костей исходы в точности равняются семи? Семь или больше?
2. Если предположить, что рост группы студентов соответствует нормальному распределению со средним значением (меаной) 1.7 м, а стандартное отклонение равно 0.1 м, то какова вероятность того, что существует студент с ростом выше 1.9 м?
3. Продемонстрируйте применение математической индукции для вывода цепного правила.
4. Предположим, что у некоторого пациента возможно одно из двух заболеваний: гепатит (H) или брюшной тиф (T). Существуют два симптома, характерных для этих заболеваний: головная боль (D) и высокая температура (лихорадочное состояние) (F), для которых возможно одно из значений TRUE или FALSE. Заданы следующие вероятности: $P(T) = 0.5$, $P(D \mid T) = 0.7$, $P(D \mid \neg T) = 0.4$, $P(F \mid T) = 0.9$, $P(F \mid \neg T) = 0.5$. Необходимо описать пространство элементарных событий и заполнить таблицы частичной (неполной) вероятности.

5. Используя данные из предыдущего задания и предполагая, что указанные симптомы болезни являются независимыми, определить вероятность того, что пациент болен гепатитом с учетом того, что головная боль отсутствует, но наблюдается высокая температура и лихорадочное состояние.
6. Принимая во внимание двумерное распределение вероятностей в табл. 2.2, определить:

   - $P(X_1)$;
   - $P(Y_2)$;
   - $P(X_1 \mid Y_1)$.

Таблица 2.2

	Y_1	Y_2	Y_3
X_1	0.1	0.2	0.1
X_2	0.3	0.1	0.2

7. Являются ли в предыдущем задании переменные X и Y независимыми?
8. В некоторой местности статистика сообщает, что в течение года наблюдаются следующие погодные условия. Из 365 дней 200 солнечных, 60 облачных, 40 дождливых, 20 дней идет снег, 20 дней – ливень с грозой, 10 дней – град, 10 ветреных дней и 5 дней – мелкий дождь (изморось). Если каждый день отправляется сообщение о погоде, то какое количество информации о каждом типе погоды содержится в таком сообщении?
9. Если учесть информацию о каждом типе погоды в предыдущем задании, то какое среднее количество информации (энтропия) содержится в ежедневном сообщении о погоде?
10. ***Внимательно изучить различные «философские» интерпретации понятия вероятности и обсудить преимущества и ограничения (недостатки) каждой из этих интерпретаций. Какую интерпретацию вы считаете наиболее удачной? Почему?

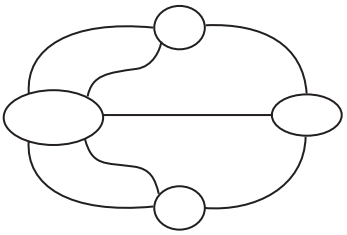
Ссылки на источники

1. Gillies, D. Philosophical Theories of Probability. Routledge, London (2000).
2. Jaynes, E. T. Probability Theory: The Logic of Science. CambridgeUniversity Press, Cambridge (2003).

3. MacKay, D. J. Information Theory, Inference and Learning Algorithms. Cambridge University Press, Cambridge (2004).
4. Sucar, L. E., Gillies, D. F., Gillies, D. A. Objective Probabilities in Expert Systems. Artif. Intell. 61, 187–208 (1993).
5. Wasserman, L. All of Statistics: A Concise Course in Statistical Inference. Springer, New York (2004).

Глава 3

Теория графов



3.1 Определения

Граф (graph) предоставляет компактный и удобный способ представления бинарных отношений в множестве объектов. Например, рассмотрим множество городов в некоторой местности и дороги, связывающие эти города. Тогда карта данной местности по существу является графом, в котором объекты – это города, а дороги, напрямую связывающие пары городов, – это отношения. Графы обычно представлены в графическом виде. Объекты изображаются в форме кружков или овалов, а отношения – линиями или стрелками (см. рис. 3.1). Существует два основных типа графов: *неориентированный граф* (undirected graph) и *ориентированный граф* (directed graph). Ниже будут приведены формализованные определения неориентированного и ориентированного графов.

Пусть дано непустое множество V, а бинарное отношение $E \subseteq V \times V$ в множестве V есть набор упорядоченных пар (V_j, V_k), таких, что $V_j \in V$ и V_k содержится в множестве V. *Ориентированный граф*, или *орграф* (digraph), – это упорядоченная пара $G = (V, E)$, где V – множество вершин или узлов, а E – множество дуг, представляющих бинарные отношения в множестве V (см. рис. 3.1б). Ориентированные графы представляют асимметричные отношения между объектами, например отношение «родитель–потомок».

Неориентированный граф – это упорядоченная пара $G = (V, E)$, где V – множество вершин или узлов, а E – множество ребер, представляющих симметричные бинарные отношения: $(V_j, V_k) \in E \rightarrow (V_k, V_j) \in E$ (см. рис. 3.1а). Ненаправленные графы представляют симметричные отношения между объектами, например отношение «брат–брат».

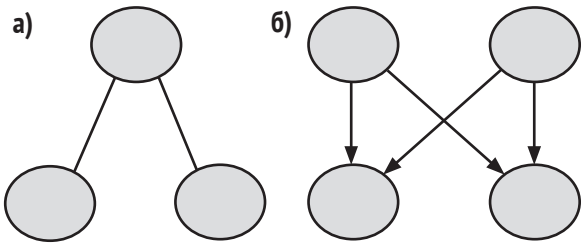


Рис. 3.1. Графы: а) неориентированный, б) ориентированный

Если существует ребро $E_i(V_j, V_k)$ между узлами j и k, то V_j является соседней (иногда используется термин «смежной») с вершиной V_k. *Степень* (degree) узла (вершины) – это количество инцидентных ему ребер. На рис. 3.1а верхний узел имеет степень 2, а оба нижних узла имеют степень 1.

Два ребра, связанных с одной и той же парой вершин, называются *параллельными ребрами*. Ребро, инцидентное единственной вершине, называется циклом. Вершина, которая не является концом ни для одного ребра, называется *изолированной вершиной* и имеет степень 0. Все перечисленные здесь случаи показаны на рис. 3.2.

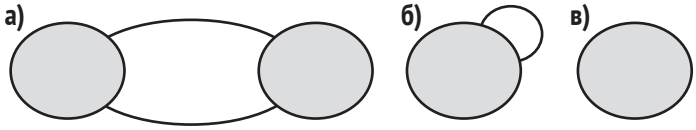


Рис. 3.2. Примеры: а) параллельные ребра, б) цикл, в) изолированная вершина

В ориентированном графе количество дуг, направленных в некоторый узел, является *степенью входа* (indegree) этого узла, а количество дуг, исходящих из узла, называется *степенью выхода* (outdegree) этого узла. На рис. 3.1б два верхних узла имеют степень входа, равную нулю, а их степень выхода равна двум. Для двух нижних узлов степень входа равна двум, а степень выхода нулевая.

Пусть дан граф $G = (V, E)$ и подграф этого графа $G' = (V', E')$, такой, что $V' \subseteq V$ и $E' \subseteq E$. В этом подграфе каждое ребро множества E' инцидентно вершинам множества V'. Например, если убрать направления ребер в графе на рис. 3.1б (то есть сделать его неориентированным графом), то граф на рис. 3.1а будет представлять собой подграф измененного графа на рис. 3.1б.

3.2 Типы графов

Кроме двух основных типов графов – ориентированного и неориентированного – существуют и другие типы графов, перечисленные ниже:

- смешанный граф – граф, который имеет как ориентированные, так и неориентированные ребра;

- простой граф – граф, который не содержит циклов и параллельных дуг;
- мультиграф – граф с несколькими компонентами (подграфами), такой, что каждая компонента не имеет ребер, связывающих ее с другими компонентами, т. е. компоненты являются несвязными (изолированными);
- полный граф – граф, который содержит ребра между каждой парой его вершин;
- двудольный граф – граф, в котором вершины разделены на два подмножества G_1 и G_2, такие, что все ребра соединяют вершины из множества G_1 с вершинами из множества G_2, т. е. в таком графе нет ребер между узлами внутри каждого подмножества;
- взвешенный граф – граф, в котором каждому ребру и/или вершине поставлено в соответствие некоторое значение – вес.

Примеры перечисленных выше типов графов показаны на рис. 3.3.

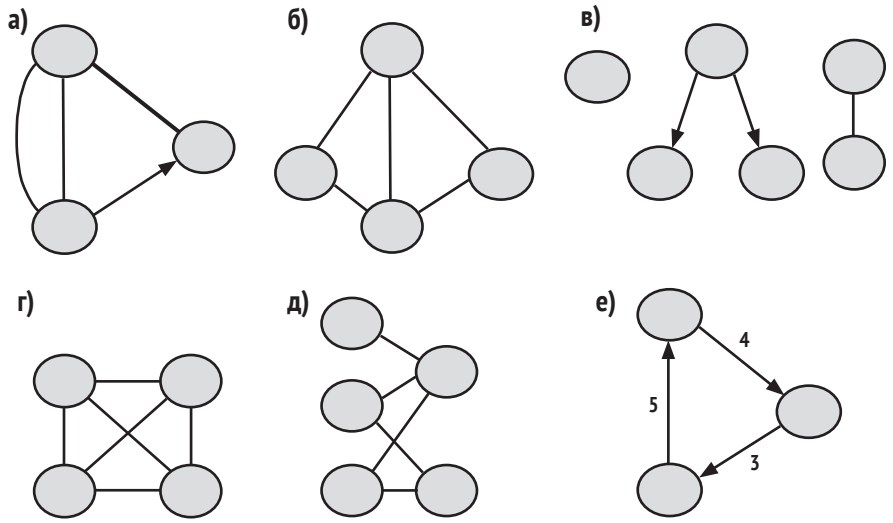


Рис. 3.3. Типы графов: а) смешанный граф, б) простой граф, в) мультиграф, г) полный граф, д) двудольный граф, е) взвешенный граф

3.3 Пути и циклы

Путь (trajectory) – это последовательность ребер $E_1, E_2, \ldots, E_n$, такая, что конечная вершина каждого ребра совпадает с начальной вершиной следующего ребра в этой последовательности (за исключением самой последней вершины), т. е. $E_i(V_j, V_k), E_{i+1}(V_k, V_l)$ от $i = 1$ до $i = n - 1$. Простой путь не включает одно и то же ребро два и более раз. *Элементарный*

путь не проходит через одну вершину более одного раза. Примеры различных путей показаны на рис. 3.4.

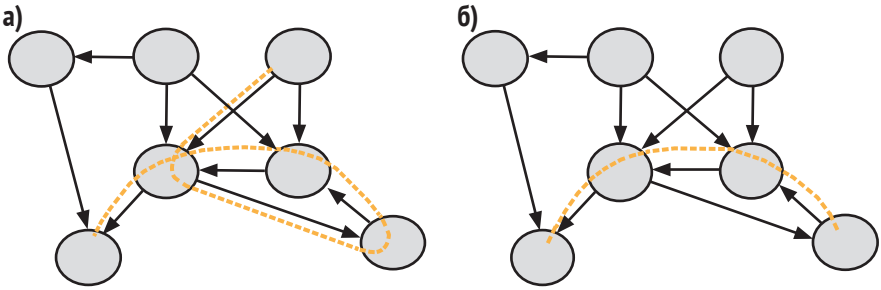


Рис. 3.4. Примеры путей: а) простой, но не элементарный путь, б) простой и элементарный путь

Граф *G* является *связным* (connected), если существует путь между каждой парой различных вершин в этом графе *G*. Если граф *G* не является связным, то каждая его несвязанная часть называется *компонентой* связности графа *G*.

Цикл (circuit или cycle) – это путь, такой, что его конечная вершина совпадает с начальной, т. е. это «замкнутый путь». По аналогии с терминологией путей можно определить простой и элементарный цикл. На рис. 3.5 показан пример цикла.

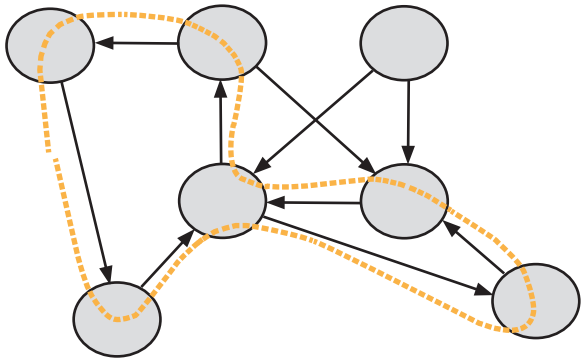


Рис. 3.5. Пример цикла, который является простым, но не элементарным

Важным типом графа для вероятностных графовых моделей является *направленный ациклический граф* (directed acyclic graph – DAG). Направленный ациклический граф – это ориентированный граф, который не содержит направленных циклов (в орграфе цикл называется контуром; направленный контур – это контур, в котором все ребра в последовательности имеют направления, указанные стрелками). Например, показанный на рис. 3.1б орграф является направленным ациклическим графом, а граф на рис. 3.3е не является направленным ациклическим графом.

В некоторых классических задачах теории графов используются пути и циклы (контуры):

- поиск пути, содержащего все ребра графа только один раз (эйлеров путь, или эйлеров граф);
- поиск цикла, содержащего все ребра графа только один раз (эйлеров цикл);
- поиск пути, содержащего все вершины графа только один раз (путь Гамильтона);
- поиск цикла, содержащего все вершины графа только один раз (цикл Гамильтона);
- поиск цикла Гамильтона во взвешенном графе с минимальной стоимостью (задача коммивояжера)[5].

Решение этих задач не относится к теме данной книги, поэтому заинтересованным читателям можно порекомендовать книгу [2].

3.4 Изоморфизм графов

Два графа называются изоморфными, если существует взаимно однозначное соответствие (типа один-к-одному) между их вершинами и ребрами, которое сохраняет их смежность и инцидентность. Для двух графов G_1 и G_2 существует три основных типа изоморфизма:

1) *изоморфизм графов* – графы G_1 и G_2 являются изоморфными;
2) *изоморфизм подграфов* – граф G_1 изоморфен подграфу графа G_2 (и наоборот);
3) *двойной изоморфизм подграфов* – подграф графа G_1 изоморфен подграфу графа G_2.

На рис. 3.6 показан пример двух изоморфных графов.

Определение изоморфизма двух графов (по типу 1) является NP-задачей. Определения изоморфизма и двойного изоморфизма подграфов (типов 2 и 3) являются NP-полными задачами. Подробнее см. [1].

3.5 Деревья

Деревья (trees) – этот тип графа, чрезвычайно важный в информатике в целом, но в особенности для вероятностных графовых моделей. Мы будем рассматривать два типа деревьев: неориентированные и ориентированные (ненаправленные и направленные).

[5] В этой задаче узлы представляют города, а ребра – дороги с обозначением соответствующего расстояния или времени, поэтому решением является «наилучший» маршрут коммивояжера (минимальный по расстоянию или по времени) с посещением всех городов.

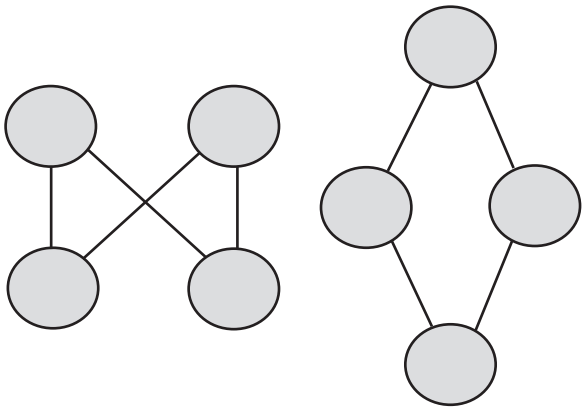


Рис. 3.6. Эти два графа изоморфны

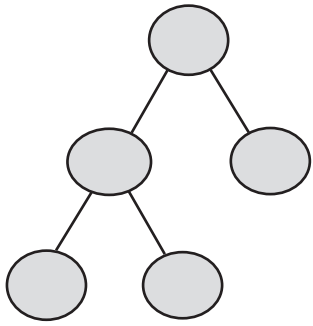


Рис. 3.7. Неориентированное дерево. Это дерево содержит пять узлов, из которых три являются листьями и два – внутренними узлами

Неориентированное дерево – это связный граф, который не содержит простых циклов. На рис. 3.7 показан пример неориентированного дерева. В неориентированном дереве существуют два класса вершин, или узлов:

- лист, или конечный узел со степенью, равной единице;
- внутренний узел со степенью больше единицы.

Ниже перечислены некоторые основные свойства дерева:

- между каждой парой вершин существует простой путь;
- количество вершин $|V|$ равно количеству ребер $|E|$ плюс один: $|V| = |E| + 1$;
- дерево с двумя и более вершинами имеет как минимум два листа (конечных узла).

Ориентированное дерево – это связный ориентированный граф, такой, что существует единственный ориентированный путь между каждой парой узлов (такой граф также называют односвязным ориентированным графом). Существует два типа ориентированных деревьев:

- корневое дерево (rooted tree), или просто дерево;
- полидерево (polytree).

Корневое дерево содержит единственный узел с нулевой степенью входа (корневой узел), а остальные узлы имеют степень входа, равную единице. Полидерево может иметь более одного узла с нулевой степенью входа (более одного корня) и некоторые узлы (ноль и более) со сте-

пенью входа, большей единицы (узлы с несколькими родителями). Если в полидереве удалить ориентацию (направления) ребер, то оно преобразуется в неориентированное дерево. Можно считать дерево особым случаем полидерева. Примеры корневого дерева и полидерева показаны на рис. 3.8.

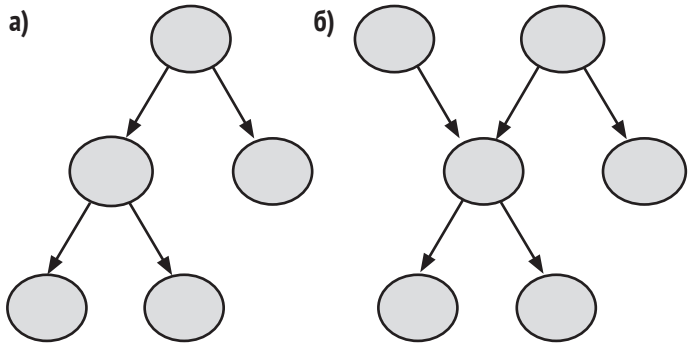


Рис. 3.8. а) Корневое дерево, б) полидерево

Ниже приведен краткий обзор некоторых терминов, связанных с ориентированными деревьями:

- корень – узел с нулевой степенью входа;
- лист – узел с нулевой степенью выхода;
- внутренний узел – узел со степенью выхода, большей нуля;
- родитель/потомок – если существует направленная дуга от *A* к *B*, то *A* – родитель *B*, а *B* – потомок *A*;
- братья – два и более узлов являются братьями, если имеют одного и того же родителя;
- предок/(непрямой) потомок – если существует направленный путь от *A* к *B*, то *A* называется предком *B*, а *B* – (непрямым) потомком *A*;
- поддерево с корнем *A* – поддерево, в котором вершина *A* является корнем;
- поддерево *A* – поддерево с потомком вершины *A*, который (потомок) является корнем;
- *K*-арное дерево – дерево, в котором каждый внутренний узел имеет не более *K* потомков. Это регулярное дерево, если каждый внутренний узел имеет *K* потомков;
- бинарное (двоичное) дерево – дерево, в котором каждый внутренний узел не более двух потомков.

Например, в дереве, показанном на рис. 3.9:

- *A* – корневой узел (корень);
- *C*, *D*, *F*, *G* – листья;
- *B*, *E* – внутренние узлы;
- *A* – родитель *B*, а *B* – потомок *A*;
- *B* и *C* – братья;
- *A* – предок *F*, а *F* – (непрямой) потомок *A*;
- поддерево *B*, *D*, *E*, *F*, *G* является поддеревом с корнем *B*;
- поддерево *E*, *F*, *G* является поддеревом *B*.

Дерево на рис. 3.9 – нерегулярное бинарное дерево.

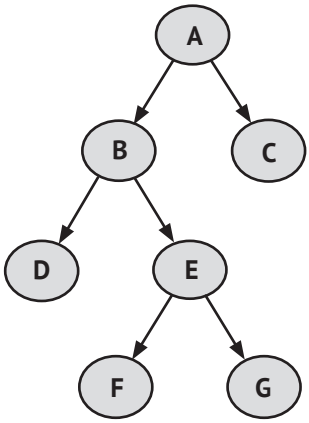


Рис. 3.9. Пример ориентированного дерева для демонстрации терминологии

3.6 Клики

Полный граф – это граф G_c, в котором каждая пара узлов является смежной (соседней), т. е. между каждой парой узлов существует ребро. На рис. 3.3г показан пример полного графа. *Полное множество* (complete set) W_c – это подмножество графа *G*, которое выводит путем индукции полный подграф графа *G*. Это подмножество вершин графа *G*, такое, что каждая пара узлов в этом подграфе является смежной (соседней). Например, на рис. 3.3г каждое подмножество любых трех вершин в этом графе является полным множеством.

Клика (clique) *C* – это подмножество графа *G*, такое, что представляет собой максимальное полное множество, т. е. в графе *G* не существует другого полного множества, которое содержит *C*. Подмножества любых трех узлов на рис. 3.3г не являются кликами, так как они не максимальны, они содержатся в графе, который является полным.

Граф на рис. 3.10 содержит пять клик: одна с четырьмя узлами, одна с тремя узлами и три с двумя узлами. Следует отметить, что каждый узел в графе является частью как минимум одной клики. Таким образом, множество клик в графе всегда покрывает множество вершин *V*.

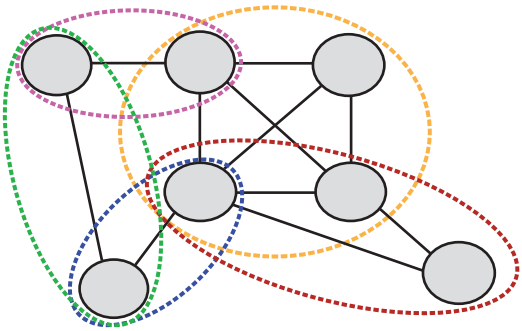


Рис. 3.10. Клики: пять клик в графе выделены штриховыми линиями разного цвета

В следующих разделах рассматриваются некоторые более развитые концепции теории графов, поскольку они используются некоторыми алгоритмами логического вывода для вероятностных графовых моделей.

3.7 Полное упорядочивание

Упорядочивание (ordering) узлов в графе заключается в присваивании целочисленных значений каждой вершине. Рассмотрим граф $G(V, E)$ с n вершинами, тогда $\alpha = [V_1, V_2, \ldots, V_n]$ – это упорядочивание графа G. Вершина V_i по порядку расположена перед вершиной V_j, если $i < j$.

Упорядочивание α графа $G = (V, E)$ является полным упорядочиванием (perfect ordering), если для каждой вершины V_i все соседние (смежные) вершины, расположенные перед ней по порядку, являются полностью связанными, т. е. для каждого значения i $Adj(V_i) \cap \{V_1, V_2, \ldots, V_{i-1}\}$ является полным подграфом графа G. $Adj(V_i)$ – это подмножество узлов в графе G, которые являются соседними (смежными) с вершиной V_i. На рис. 3.11 показан пример полного упорядочивания.

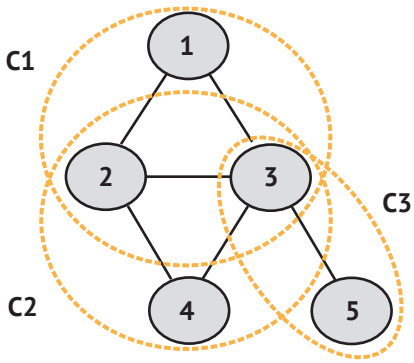


Рис. 3.11. Пример упорядочивания узлов и клик в графе. В этом случае узлы полностью упорядочены, а упорядочивание клик соответствует свойству древесной декомпозиции (running intersection property)

Рассмотрим множество клик $C_1, C_2, \ldots, C_p$ в неориентированном связном графе G. По аналогии с упорядочиванием узлов можно определить упорядочивание клик $\beta = [C_1, C_2, \ldots, C_p]$. Упорядочивание β этих клик обладает *свойством древесной декомпозиции* (running intersection property), если все узлы каждой клики C_i, являющиеся общими с предыдущими по указанному порядку кликами, содержатся в клике C_j. Клика C_j является родителем клики C_i. Другими словами, для каждой клики $i > 1$ существует клика $j < i$, такая, что $C_i \cap \{C_1, C_2, \ldots, C_{i-1}\} \subseteq C_j$. Возможно, что клика имеет более одного родителя.

Клики C_1, C_2 и C_3 на рис. 3.11 являются полностью упорядоченными. В этом примере клика C_1 является родителем клики C_2, в то же время клики C_1 и C_2 являются родителями клики C_3.

Граф G является *триангулированным* (triangulated), если каждый простой контур (цикл) с длиной, большей трех, в графе G содержит хорду. *Хорда* (chord) – это ребро, которое соединяет две вершины в этом контуре (цикле), но не является частью этого контура (цикла). Например, на рис. 3.11 контур (цикл), сформированный вершинами 1, 2, 4, 3, 1, содержит хорду, соединяющую узлы 2 и 3. Граф на рис. 3.11 является триангулированным. Пример графа, не являющегося триангулированным, показан на рис. 3.12. Несмотря на то что внешне этот граф может показаться триангулированным, в нем существует контур (цикл) 1, 2, 5, 4, 1, который не содержит ни одной хорды.

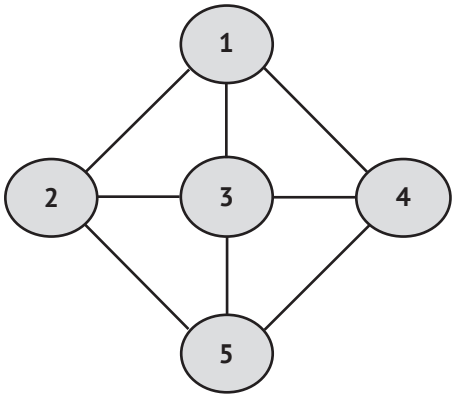


Рис. 3.12. Пример графа, не являющегося триангулированным. Контур (цикл) 1, 2, 5, 4, 1 не содержит хорду

Условием достижения полного упорядочивания вершин и упорядочивания тех клик, которые соответствуют *свойству древесной декомпозиции* (running intersection property), является триангулированность графа. В следующем разделе будут рассматриваться следующие алгоритмы:

- упорядочивание узлов графа для достижения полного упорядочивания;

- нумерация клик для обеспечения свойства древесной декомпозиции при полном упорядочивании;
- триангуляция графа, если он не является триангулированным.

3.8 Алгоритмы упорядочивания и триангуляции

3.8.1 Поиск паросочетания максимальной мощности

Если граф является триангулированным, то следующий алгоритм, известный как *поиск паросочетания максимальной мощности* (maximum cardinality search), обеспечивает полное упорядочивание. Пусть дан неориентированный граф $G = (V, E)$ с n вершинами:

Алгоритм 3.1. Поиск паросочетания максимальной мощности

```
Выбрать любую вершину из множества V и присвоить ей номер 1.
while Не все вершины в графе G пронумерованы do
  Из всех непомеченных вершин выбрать ту, для которой соседние
  (смежные) вершины имеют наибольший номер, и присвоить выбранной
  вершине следующий номер. Связи разрывать в произвольном порядке.
end while
```

После полного упорядочивания всех вершин легко пронумеровать клики так, чтобы их порядок соответствовал свойству древесной декомпозиции. Для этого клики нумеруются в обратном порядке. Пусть существует p клик, тогда клике, содержащей узел с наибольшим номером, присваивается номер p. Клика, включающая узел со следующим наибольшим номером, получает номер $p - 1$ и т. д. Этот метод можно проиллюстрировать с помощью примера на рис. 3.11. Наибольший номер узла равен 5, поэтому клика, его содержащая, обозначается как C_3. Следующий наибольший номер узла 4, а включающая его клика C_2. Последняя клика получает обозначение C_1.

Далее будет рассматриваться дополнение графа, чтобы сделать его триангулированным.

3.8.2 Дополнение графа

Дополнение графа состоит в добавлении дуг в исходный граф G, чтобы получить новый (дополненный) граф G_t, который является триангулированным. Пусть дан неориентированный граф $G = (V, E)$ с n узлами,

тогда приведенный ниже алгоритм дополняет этот граф до триангулированного.

Алгоритм 3.2. Дополнение графа

```
Упорядочить вершины V графа с помощью алгоритма поиска паросочетаний
максимальной мощности: V1, V2, …, Vn.
for i = n to i = 1 do
  Для узла Vi выбрать все смежные с ним узлы Vj, такие, что j > i.
  Обозначить это множество выбранных узлов Aj.
  Добавить дугу от вершины Vi к вершине Vk, если k > i и Vk ∉ Aj.
end for
```

Например, рассмотрим граф на рис. 3.12, не являющийся триангулированным. Если применить к нему алгоритм 3.2, то сначала упорядочиваются узлы и генерируются числовые метки, показанные на рис. 3.12. Затем узлы обрабатываются в обратном порядке и формируются множества A для каждого узла:

A_5: Ø
A_4: 5
A_3: 4, 5
A_2: 3, 5. Добавляется дуга от вершины 2 к вершине 4.
A_1: 2, 3, 4. Добавляется дуга от вершины 1 к вершине 5.

Полученный в итоге (дополненный) граф содержит две дополнительные дуги 2–4 и 1–5, и мы можем проверить и убедиться в том, что новый граф является триангулированным.

Алгоритм 3.2 дополнения графа гарантирует, что полученный в результате граф является триангулированным, но в общем случае этот алгоритм неоптимален с точки зрения возможности добавления минимального количества дополнительных дуг.

3.9 Материалы для дополнительного чтения

В нескольких книгах по теории графов более подробно рассматривается большинство концепций, представленных в этой главе, в том числе [2–4]. Неаполитан (Neapolitan) [5] в главе 3 подробно описывает основы теории графов, требуемые для байесовских сетей, включая и более продвинутые концепции. Некоторые методы теории графов с алгоритмической точки зрения описаны в [1], в том числе и изоморфизм графов.

3.10 Задания и упражнения

1. В XVIII веке город Кенигсберг (в Пруссии; в настоящее время – город Калининград и Калининградская область входят в состав России) был разделен на четыре части, соединенные семью мостами. Говорят, что горожане пытались найти такой путь через весь город, чтобы проход по каждому мосту выполнялся только один раз. Эйлер преобразовал эту задачу в граф (изображенный в самом начале данной главы) и определил в связном графе следующее условие для контура, который проходит через каждое ребро в точности один раз: все вершины в этом графе обязательно должны иметь четную степень. Определить, была ли у жителей Кенигсберга возможность найти контур (маршрут) Эйлера.
2. Доказать условие, определенное Эйлером: граф G содержит контур Эйлера, если и только если все вершины в графе G имеют четную степень.
3. Какое условие должно выполняться для графа, чтобы он содержал путь Эйлера?
4. Пусть дан граф, изображенный на рис. 3.10. Определить, содержит ли этот граф:
   а) контур Эйлера,
   б) путь Эйлера,
   в) гамильтонов контур,
   г) гамильтонов путь.
5. Пусть дан граф, изображенный на рис. 3.10. Является ли этот граф триангулированным? Если это не триангулированный граф, то преобразовать его в триангулированный, применив алгоритм дополнения графа.
6. Доказать, что количество вершин нечетной степени в графе является четным числом.
7. Пусть дан граф, изображенный на рис. 3.13. Преобразовать его в неориентированный граф и упорядочить вершины, применив алгоритм поиска паросочетаний максимальной мощности.
8. Выполнить триангуляцию графа, указанного в задании 7.
9. Для графа, полученного в результате выполнения задания 8:
   а) найти его клики,
   б) упорядочить клики в соответствии с упорядочиванием узлов этого графа и проверить, обладает ли этот граф свойством древесной декомпозиции,

в) показать полученное в результате дерево (или деревья) клик.

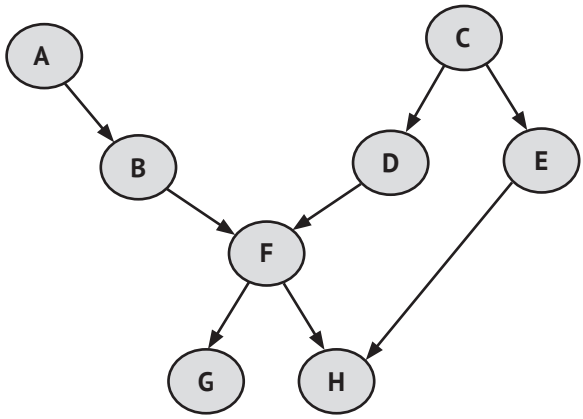


Рис. 3.13. Граф, используемый в некоторых заданиях

10. *** Разработать программу генерации дерева клик в заданном неориентированном графе. При этом условии:
    а) упорядочить узлы по алгоритму поиска паросочетаний максимальной мощности,
    б) выполнить триангуляцию графа, используя алгоритм заполнения графа,
    в) найти клики в триангулированном графе и пронумеровать их.

    Попробуйте найти наиболее подходящую структуру данных для представления графа, учитывая реализацию описанных выше алгоритмов.

11. *** Включить в программу из задания 10 некоторые эвристики для выбора упорядочиваемых узлов, такие, что размер наибольшей клики в этом графе был бы минимальным (это важно для реализации алгоритма для дерева сочленений при логическом выводе для байесовских сетей).

Ссылки на источники

1. Aho, A. V., Hopcroft, J. E., Ullman, J. D. The Design and Analysis of Computer Algorithms. Addison-Wesley, Boston (1974).
2. Golumbic, M. C. Algorithmic Graph Theory and Perfect Graphs. Elsevier, The Netherlands (1994).
3. Gould, R. Graph Theory. Benjamin/Cummings, Menlo Park (1988).

4. Gross, J. L., Yellen, J. Graph Theory and its Applications. CRC Press, Boca Raton (2005).
5. Neapolitan, R. Probabilistic Reasoning in Expert Systems: Theory and Algorithms.Wiley, New York (1990).

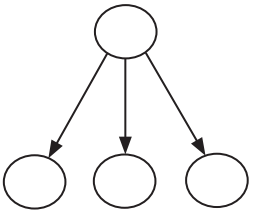
Часть II

Вероятностные модели

В этой части книги представлены основные типы вероятностных графовых моделей, в том числе байесовские классификаторы, скрытые марковские модели, марковские случайные поля, байесовские сети, а также динамические и временны́е байесовские сети. Каждый тип модели рассматривается в отдельной главе, включая представление, логический вывод и обучение, а кроме того, примеры практических приложений. К моделям, описанным в части II, не относятся модели принятия решений, которые будут рассматриваться в части III.

Глава 4

Байесовские классификаторы



4.1 Введение

Классификация – это присваивание классов или меток объектам. Существуют два основных типа задач классификации:

- без учителя (unsupervised) – в этом случае классы неизвестны, поэтому задача заключается в разделении множества объектов на *n* групп или кластеров таким способом, что класс присваивается каждой отдельной группе. Эту методику также называют *кластеризацией* (clustering);
- с учителем (supervised) – возможные классы или метки известны заранее. Задача заключается в нахождении функции или правила, по которым каждый объект причисляется к одному из известных классов.

В обоих случаях объекты обычно описываются набором свойств или атрибутов. В этой главе основное внимание будет уделено классификации с учителем (supervised classification).

С вероятностной точки зрения задача классификации с учителем состоит в присваивании конкретному объекту, описываемому атрибутами $A_1, A_2, \ldots, A_n$, одного из m классов $C = \{c_1, c_2, \ldots, c_m\}$ таким образом, чтобы вероятность назначаемого класса с учетом атрибутов объекта была максимальной, т. е.

$$Arg_C \left[Max\ P(C \mid A_1, A_2, \ldots, A_n) \right]. \tag{4.1}$$

Если обозначить множество атрибутов как $\mathbf{A} = \{A_1, A_2, \ldots, A_n\}$, то формулу 4.1 можно записать следующим образом: $Arg_C [Max\ P(C \mid \mathbf{A})]$.

Существует много способов создания классификатора, в том числе деревья решений, правила, нейронные сети, метод опорных векторов

и многие другие[6]. В этой книге будут рассматриваться модели классификации на основе вероятностных графовых моделей, в частности байесовские классификаторы. Прежде чем перейти к рассмотрению байесовского классификатора, необходимо кратко описать, как оцениваются классификаторы.

4.1.1 Оценки классификатора

Для сравнения различных методик классификации можно определять числовые оценки классификатора с различных точек зрения. Выбор оцениваемой характеристики и степень важности, назначаемая каждой характеристике, зависит от конечного приложения классификаторов. Рассматриваются и оцениваются следующие основные характеристики:

- точность – насколько правильно классификатор прогнозирует корректный класс для ранее не наблюдаемых экземпляров (эта оценка не предназначается для обучения классификатора);
- время классификации – как долго продолжается процесс классификации для прогнозирования класса после тренировочного обучения классификатора;
- время тренировочного обучения – сколько времени требуется для обучения классификатора по имеющимся данным;
- требования к памяти – какой объем памяти требуется для хранения параметров классификатора;
- ясность – может ли человек без затруднений понять результаты работы классификатора.

Обычно самой важной характеристикой является точность (accuracy). Ее можно оценить в числовом выражении, прогнозируя N ранее не наблюдаемых элементов выборки данных и определяя процент правильных прогнозов. Таким образом, точность в процентах равна:

$$Acc = (N_C/N) \times 100, \tag{4.2}$$

где N_C – число правильных прогнозов.

При сравнении классификаторов в общем случае необходимо максимизировать точность классификации, но это *оптимальный* подход только в том случае, если стоимость неправильной классификации одинакова для всех классов. Рассмотрим классификатор, используемый для прогнозирования рака груди. Если классификатор прогнозирует, что пациент болен раком, но это неправильно (ложноположительный слу-

[6] Заинтересованный читатель может найти вводную информацию и сравнение различных типов классификаторов в [10].

чай), то может быть назначено лечение, которое не требуется. С другой стороны, если классификатор прогнозирует отсутствие ракового заболевания, а в действительности у пациента рак груди (ложноотрицательный случай), это может привести к задержке необходимого лечения и даже к смерти. Вполне очевидно, что последствия различных типов ошибок неодинаковы.

Когда существует дисбаланс в стоимостях неправильной классификации, необходимо обязательно минимизировать ожидаемую стоимость (expected cost – EC). Для двух классов ожидаемую стоимость можно вычислить по следующей формуле:

$$EC = FN \times P(-)C(- \mid +) + FP \times P(+)C(+ \mid -), \tag{4.3}$$

где FN – коэффициент ложноотрицательных случаев, FP – коэффициент ложноположительных случаев, $P(+)$ – вероятность положительного результата, $P(-)$ – вероятность отрицательного результата, $C(- \mid +)$ – стоимость классификации положительного случая как отрицательного, $C(+ \mid -)$ – стоимость классификации отрицательного случая как положительного. В некоторых приложениях определение этих стоимостей может оказаться трудной задачей.

4.2 Байесовский классификатор

Определение (формулировка) байесовского классификатора основано на применении правила Байеса для оценки вероятности каждого класса с учетом атрибутов:

$$P(C \mid A_1, A_2, \ldots, A_n) = P(C)P(A_1, A_2, \ldots, A_n \mid C) / P(A_1, A_2, \ldots, A_n). \tag{4.4}$$

Это правило можно записать в более компактной форме:

$$P(C \mid \mathbf{A}) = P(C)P(\mathbf{A} \mid C) / P(\mathbf{A}). \tag{4.5}$$

Задача классификации на основе формулы 4.5 может быть сформулирована следующим образом:

$$Arg_C[Max[P(C \mid \mathbf{A}) = P(C)P(\mathbf{A} \mid C) / P(\mathbf{A})]]. \tag{4.6}$$

Формулу 4.6 можно записать в равнозначной форме, используя любую функцию, которая изменяется монотонно по отношению к вероятности $P(C \mid \mathbf{A})$, например:

- $Arg_C\left[Max[P(C)P(\mathbf{A} \mid C)]\right]$;
- $Arg_C\left[Max[log(P(C)P(\mathbf{A} \mid C))]\right]$;
- $Arg_C\left[Max[(log\, P(C) + log\, P(\mathbf{A} \mid C))]\right]$.

Следует отметить, что вероятность атрибутов $P(\mathbf{A})$ не изменяется по отношению к классу, поэтому ее можно считать постоянной величиной в процессе максимизации.

На основе приведенных выше равнозначных вариантов записи формулы для решения задачи классификации потребуется оценка вероятности $P(C)$, известная как априорная вероятность классов, и оценка вероятности $P(\mathbf{A} \mid C)$, называемая правдоподобием (likelihood), а также $P(C \mid \mathbf{A})$ – апостериорная вероятность. Таким образом, для получения апостериорной вероятности каждого класса необходимо всего лишь умножить априорную вероятность конкретного класса на правдоподобие, которое зависит от значений атрибутов[7].

Прямое применение правила Байеса приводит к задаче с большими вычислительными затратами, это было отмечено в главе 1. Причина в том, что количество параметров в правдоподобии $P(A_1, A_2, \ldots, A_n \mid C)$ возрастает экспоненциально при увеличении количества атрибутов. Это потребует не только огромного объема памяти для хранения всех параметров, но кроме того, будет очень трудно дать оценку всех вероятностей по имеющимся данным или с помощью эксперта в соответствующей предметной области. Таким образом, практическое применение байесовского классификатора возможно только для относительно малых задач с небольшим количеством атрибутов. Другой подход заключается в рассмотрении некоторых независимых свойств как в графовых моделях, в частности что все атрибуты независимы по отношению к конкретному классу. Результатом такого подхода является наивный байесовский классификатор.

4.2.1 Наивный байесовский классификатор

Наивный, или простейший, байесовский классификатор (Naive Bayesian classifier – NBC) основан на предположении о том, что все атрибуты являются независимыми по отношению к переменной класса, т. е. каждый атрибут A_i является условно независимым от всех прочих атрибутов относительно класса: $P(A_i \mid A_j, C) = P(A_i \mid C), \forall\, j \neq i$. С учетом этого предположения формулу 4.4 можно записать так:

$$P(C \mid A_1, A_2, \ldots, A_n) = P(C)P(A_1 \mid C)P(A_2 \mid C) \ldots P(A_n \mid C) / P(A), \qquad (4.7)$$

где $P(A)$, как уже отмечено выше, может рассматриваться как постоянная нормализации.

Формула наивной байесовской классификации резко снижает сложность байесовского классификатора, так как в этом случае требуется

[7] На апостериорные вероятности классов будет воздействовать постоянная величина, так как в формуле 4.6 мы не учитываем знаменатель, т. е. в сумме апостериорные вероятности не будут давать единицу. Но апостериорные вероятности легко нормализовать, разделив каждое значение на сумму для всех классов.

только априорная вероятность (одномерный вектор) класса и n условных вероятностей каждого атрибута относительно класса (двумерные матрицы) как параметры модели. Таким образом, требование к объему памяти снижается: экспоненциальная зависимость заменяется линейной в соответствии с количеством атрибутов. Кроме того, значительно упрощается вычисление апостериорной вероятности, поскольку для ее (ненормализованной) оценки требуется только n операций умножения.

Графическое представление наивной байесовской классификации показано на рис. 4.1. Эта структура, напоминающая по форме звезду, отображает свойство условной независимости между всеми атрибутами относительно класса – между узлами атрибутов нет дуг.

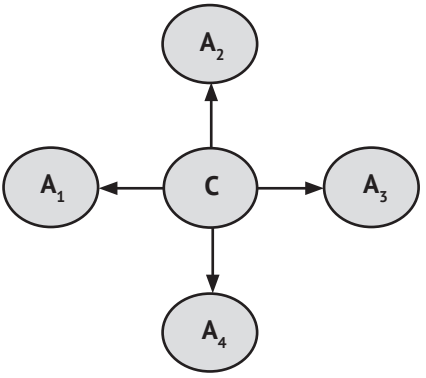


Рис. 4.1. Пример наивного байесовского классификатора с переменной класса C и четырьмя атрибутами A_1, A_2, A_3, A_4

Обучение наивного байесовского классификатора заключается в определении оценки априорной вероятности класса $P(C)$ и условной вероятности каждого атрибута относительно класса $P(A_i \mid C)$. Эти значения можно получить с помощью субъективных оценок эксперта или по данным с использованием максимального правдоподобия[8].

Вероятности могут оцениваться по имеющимся данным, например с использованием оценки максимального правдоподобия. Априорные вероятности переменной класса C вычисляются по следующей формуле:

$$P(c_i) \sim N_i/N, \tag{4.8}$$

где N_i – количество присваиваний (появлений) класса c_i в выборке из N элементов.

Условные вероятности каждого атрибута A_j могут оцениваться по следующей формуле:

$$P(A_{jk} \mid c_i) \sim N_{jki}/N_i, \tag{4.9}$$

[8] Более подробно оценка параметров будет рассматриваться в главе по байесовским сетям.

где N_{jki} – количество случаев, когда атрибут A_j принимает значение k, которое взято из класса i, а N_i – количество элементов класса c_i в рассматриваемом наборе данных.

После вычисления оценок параметров апостериорную вероятность можно получить простым умножением априорной вероятности на правдоподобие каждого атрибута. Таким образом, с учетом значений для m атрибутов $a_1, a_2, \ldots, a_m$ для каждого класса c_i апостериорная вероятность пропорциональна следующему произведению:

$$P(c_i \mid a_1, a_2, \ldots, a_m) \sim P(c_i)P(a_1 \mid c_i) \ldots P(a_m \mid c_i). \tag{4.10}$$

Будет выбран класс c_k, который максимизирует значение этого выражения[9].

Таблица 4.1. Набор данных для примера определения возможности игры в гольф по прогнозу погоды

Прогноз	Температура	Влажность	Ветер	Игра
Солнечно	Высокая	Высокая	Ложь	Нет
Солнечно	Высокая	Высокая	Истина	Нет
Облачно	Высокая	Высокая	Ложь	Да
Дождь	Средняя	Высокая	Ложь	Да
Дождь	Низкая	Нормальная	Ложь	Да
Дождь	Низкая	Нормальная	Истина	Нет
Облачно	Низкая	Нормальная	Истина	Да
Солнечно	Средняя	Высокая	Ложь	Нет
Солнечно	Низкая	Нормальная	Ложь	Да
Дождь	Средняя	Нормальная	Ложь	Да
Солнечно	Средняя	Нормальная	Истина	Да
Облачно	Средняя	Высокая	Истина	Да
Облачно	Высокая	Нормальная	Ложь	Да
Дождь	Средняя	Высокая	Истина	Нет

Снова обратимся к примеру определения возможности игры в гольф по прогнозу погоды. В табл. 4.1 содержится по 14 записей для 5 переменных: четыре атрибута (Прогноз, Температура, Влажность, Ветер) и один класс (Игра). Наивный байесовский классификатор для этого при-

[9] При этом предполагается, что стоимость неправильной классификации одинакова для всех классов. Если стоимости не одинаковы, то должен быть выбран класс, который минимизирует стоимость неправильной классификации.

мера показан на рис. 4.2 вместе с некоторыми необходимыми таблицами вероятностей.

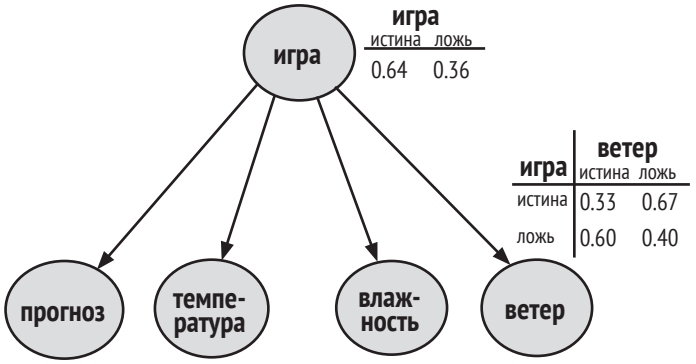


Рис. 4.2. Наивный байесовский классификатор для примера определения возможности игры в гольф. Также показаны некоторые параметры модели: *P*(игра), *P*(ветер | игра)

Ниже кратко описаны основные преимущества наивного байесовского классификатора:

- небольшое количество требуемых параметров. Это снижает уровень требований к объему памяти и облегчает обучение по имеющимся данным;
- низкая стоимость вычислений для логического вывода (оценки апостериорных вероятностей) и для обучения;
- относительно высокая производительность (точность классификации) во многих предметных областях;
- простая и понятная модель.

Но наивный байесовский классификатор имеет и недостатки, самые значительные из которых перечислены ниже:

- в некоторых предметных областях производительность снижается из-за того, что предположение об условной независимости является неверным;
- если имеются непрерывные атрибуты, то необходимо их дискретизировать (или рассмотреть варианты с использованием других моделей, таких как линейная дискриминативная модель классификатора).

В следующих разделах будут подробно рассматриваться альтернативные варианты решения, устраняющие первый недостаток. Сначала будут описаны другие модели, которые учитывают конкретные зависи-

мости между атрибутами. Затем будут представлены методики, которые могут исключить или объединить атрибуты. Это еще один способ устранения некорректности предположения об условной независимости.

4.3 Другие модели: TAN, BAN

Обобщенный байесовский классификатор и наивный байесовский классификатор – это две предельно противоположные структуры зависимостей из множества возможных структур для байесовских классификаторов. Обобщенный байесовский классификатор представляет самую сложную структуру без предположений о независимости. Наивный байесовский классификатор – эта самая простая структура, для которой предполагается, что все атрибуты независимы относительно класса. Между этими двумя предельными вариантами находится широкий спектр возможных разнообразных моделей различной сложности. Интересными вариантами являются классификаторы TAN и BAN.

TAN (Tree augmented Bayesian classifier – *байесовский классификатор, дополненный деревом*) включает некоторые зависимости между атрибутами, создавая ориентированное дерево на переменных атрибутов. Таким образом, *n* атрибутов формируют граф, ограниченный ориентированным деревом, которое представляет отношения зависимости между этими атрибутами. Кроме того, здесь существуют дуги между переменными класса и каждым атрибутом. Структура классификатора TAN показана на рис. 4.3.

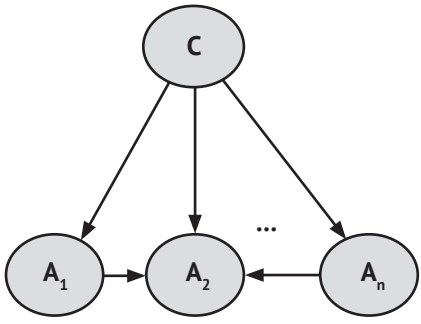


Рис. 4.3. Пример классификатора TAN

Если не ограничиваться структурой дерева, определяющей связи между атрибутами, то получим BAN (Bayesian Network augmented Bayesian classifier – *байесовский классификатор, дополненный байесовской сетью*). В классификаторе BAN структура зависимостей между атрибутами образует направленный ациклический граф (directed acyclic graph). Как и в классификаторе TAN, здесь существуют дуги между узлом класса и каждым атрибутом. Структура классификатора BAN показана на рис. 4.4.

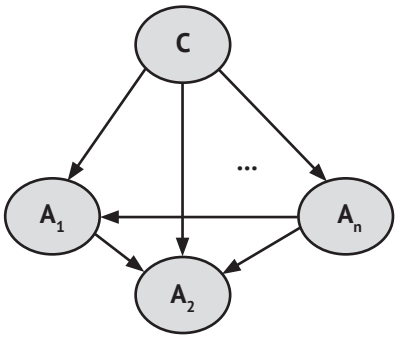


Рис. 4.4. Пример классификатора BAN

Апостериорную вероятность для переменной класса с учетом атрибутов можно получить тем же способом, что и для наивного байесовского классификатора. Но теперь каждый атрибут зависит не только от класса, но и от других атрибутов в соответствии со структурой графа. Таким образом, необходимо учитывать условную вероятность каждого атрибута по отношению к классу и к своим *родительским* атрибутам:

$$P(C \mid A_1, A_2, \ldots, A_n) = P(C)P(A_1 \mid C, Pa(A_1))P(A_2 \mid C, Pa(A_2))\ldots P(A_n \mid C, Pa(A_n)) / P(\mathbf{A}), \quad (4.11)$$

где $Pa(A_i)$ – множество родительских атрибутов для атрибута A_i в соответствии со структурой зависимости атрибутов в классификаторе TAN или BAN.

Классификаторы TAN и BAN могут рассматриваться как частные случаи более общей модели, а именно байесовских сетей, которые подробно описываются в главе 7. В главах 7 и 8 будут рассматриваться различные методики логического вывода и обучения байесовских сетей, которые можно применять для определения апостериорных вероятностей (логический вывод) и для создания модели (обучение) для классификаторов TAN и BAN.

4.4 Частично наивные байесовские классификаторы

Другой вариант обработки зависимых атрибутов – преобразование основной структуры наивного байесовского классификатора с сохранением сети в форме звезды или древовидной структуры. Преимущество такого подхода заключается в сохранении эффективности и простоты наивного байесовского классификатора, но в то же время улучшается производительность в тех случаях, когда атрибуты не являются независимыми. Такие типы байесовских классификаторов называют *частично наивными байесовскими классификаторами* (Semi-Naïve Bayesian Classi-

fiers – SNBC). Некоторые авторы предлагают различные варианты частично наивных байесовских классификаторов [11, 16].

Основная идея частично наивных байесовских классификаторов заключается в исключении или *объединении* атрибутов, зависимых по отношению к классу. Это позволяет улучшить производительность классификатора. Такая методика аналогична *отбору признаков* (feature selection) в машинном обучении, и здесь существуют два типа подходов:

- *фильтр* (filter) – атрибуты отбираются в соответствии с локальной мерой, например взаимной информацией между атрибутом и классом;
- *обертка* (wrapper) – атрибуты отбираются на основе глобальной меры, обычно посредством сравнения производительности классификатора с конкретным атрибутом и без этого атрибута.

Кроме того, обучающий алгоритм может начать работу с пустой структуры и добавлять (или объединять) атрибуты или начать работу с заполненной структурой со всеми атрибутами и постепенно исключать (или объединять) атрибуты.

На рис. 4.5 показаны две различные операции, изменяющие структуру наивного байесовского классификатора: рис. 4.5б – исключение узла, рис. 4.5в – объединение узлов. Предполагается, что работа была начата с заполненной структурой (рис. 4.5а).

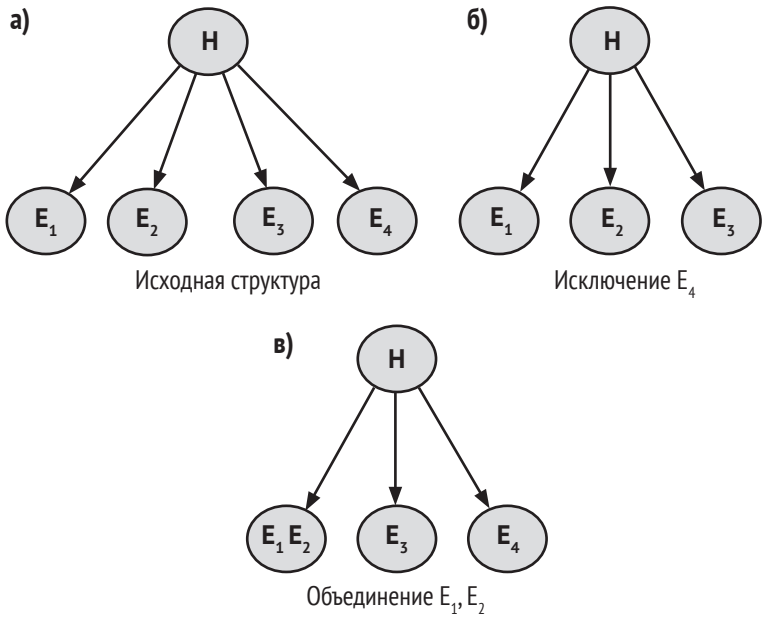


Рис. 4.5. Улучшение структуры: а) исходная структура, б) один атрибут исключен, в) два атрибута объединены в одной переменной

Исключение узла – это просто удаление атрибута A_i из модели, возможно, потому, что он не связан с конкретным классом (A_i и C независимы), или потому, что атрибут A_i и другой атрибут A_j не являются независимыми по отношению к классу (это основное предположение для наивного байесовского классификатора). Обоснование исключения одного из зависимых атрибутов: если атрибут не является независимым по отношению к классу, то один из таких атрибутов избыточен и может быть исключен.

Объединение узлов – это слияние двух атрибутов A_i и A_j в новый атрибут A_k, такой, что для A_k множеством возможных значений является векторное произведение A_i и A_j (предполагается, что атрибуты дискретны). Например, если $A_i = a, b, c$ и $A_j = 1, 2$, то $A_k = a1, a2, b1, b2, c1, c2$. Это альтернативное решение, когда два атрибута не являются независимыми по отношению к классу. При объединении таких атрибутов в единый комбинированный атрибут условие независимости больше не является значимым.

Таким образом, если два атрибута не являются независимыми по отношению к классу, то существуют два различных варианта: исключение одного из атрибутов или объединение этих атрибутов в одну переменную. Теоретически мы должны выбрать тот вариант, который дает больший прирост производительности классификатора, но на практике оценку производительности иногда трудно определить.

Как уже отмечено выше, существует несколько различных вариантов обучения частично наивного байесовского классификатора. Простая «жадная» схема показана в алгоритме 4.1, где обработка начинается с заполненного наивного байесовского классификатора со всеми атрибутами [9].

Этот процесс повторяется до тех пор, пока не останется избыточных или зависимых атрибутов.

По ссылкам [8, 16] можно найти описание еще одной операции изменения структуры наивного байесовского классификатора – добавление нового атрибута, который делает два зависимых атрибута независимыми (см. рис. 4.6). Этот новый атрибут представляет собой тип виртуального или скрытого узла в модели, для которого нет каких-либо данных для обучения его параметров. Альтернативный метод оценки параметров скрытых переменных в байесовских сетях, как и в описанном здесь случае, основан на EM-алгоритме (Expectation-Maximization), который описан в главе 8.

Во всех описанных выше классификаторах предполагалось, что существует единственная переменная класса, т. е. каждый экземпляр (элемент выборки данных) принадлежит одному и только одному классу. Далее мы будем считать, что любой экземпляр (элемент данных) может принадлежать нескольким классам. Эту методику называют многомерной классификацией.

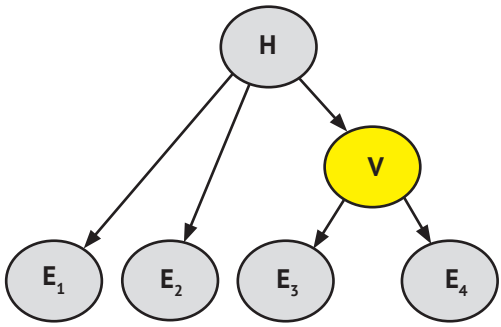


Рис. 4.6. Пример создания узла, позволяющего сделать два зависимых атрибута независимыми. Узел *V* вставляется в модель с учетом того, что узлы E_3 и E_4 не являются условно независимыми по отношению к узлу *H*

Алгоритм 4.1. Алгоритм улучшения структуры

```
Требуемое условие: A – атрибуты
Требуемое условие: C – класс
/* Зависимость между каждым атрибутом и классом оценивается
 * (например, с использованием взаимной информации (MI)).
 */
for all a ∈ A do
  /* Эти атрибуты, которые не превышают пороговое значение, исключаются. */
  if MI(a, C) < ε then
    Исключить a
  end if
end for
/* Оставшиеся атрибуты проверяются на независимость по отношению
 * к классу, например с использованием условной взаимной информации
 * (conditional mutual information - CMI).
 */
for all a ∈ A do
  for all b ∈ A - a do
    /* Эти атрибуты, находящиеся выше порогового значения, исключаются
     * или объединяются.При этом выбирается вариант, обеспечивающий
     * наилучшую производительность классификации.
     */
    if CMI(a, b | C) > ω then
      Исключить или Объединить a и b
    end if
  end for
end for
```

4.5 Многомерные байесовские классификаторы

Некоторые важные задачи требуют одновременного прогнозирования нескольких классов. Например, классификация текстов, при которой документу может быть назначено несколько тем, классификация генов, так как ген может выполнять различные функции, автоматическая аннотация изображений, поскольку изображение может содержать несколько объектов, и т. д. Это примеры *многомерной классификации*, при которой для объекта может быть назначено более одного класса. Формально задача *многомерной классификации* соответствует задаче поиска функции h, которая назначает для каждого экземпляра (элемента данных), представленного вектором m признаков $\mathbf{X} = (X_1, X_2, \ldots, X_m)$, вектор, состоящий из d значений класса $C = (C_1, C_2 \ldots, C_d)$. Функция h должна присваивать каждому экземпляру X наиболее вероятное сочетание классов, т. е.

$$ArgMax_{C1,C2\ldots,Cd} P(C_1 = c_1, \ldots, C_d = c_d \mid \mathbf{X}). \quad (4.12)$$

Классификация с несколькими метками (multi-label classification) – это частный случай многомерной классификации, при которой все переменные классов являются бинарными. При классификации с несколькими метками существует две основные методики: *бинарная релевантность* (binary relevance) и *повышение степени множества (булеана) меток* (label power-set) [18]. Методики типа бинарная релевантность выполняют преобразование задачи классификации с несколькими метками в d независимых задачах бинарной классификации, по одной задаче для каждой переменной класса $C_1, \ldots, C_d$. Производится независимое обучение каждого класса, а результаты объединяются для определения множества прогнозируемых классов. При этом зависимости между классами не учитываются. Методика повышения степени множества меток выполняет преобразование задачи классификации с несколькими метками в вариант с одним классом, определяя новую составную переменную классов, для которой возможными значениями являются все возможные сочетания значений из исходных классов. В этом случае неявно учитываются взаимодействия между классами. Эта методика может быть эффективной для предметных областей с небольшим количеством переменных классов, но при многочисленных классах такой подход неэффективен с практической точки зрения. По существу, методика бинарной релевантности может быть эффективной, когда классы относительно независимы, а методика повышения степени множества меток – когда количество переменных класса невелико.

В общем рабочем комплекте байесовских классификаторов можно выделить две методики, являющиеся альтернативами для основных

методик. Одна из них основана на байесовских сетях, где зависимости между переменными классов (и между атрибутами) учитываются явно. Другая методика неявно включает зависимости между классами, добавляя дополнительные атрибуты в каждый независимый классификатор. Обе эти методики рассматриваются в следующих разделах.

4.5.1 Многомерные классификаторы на основе байесовских сетей

Многомерный классификатор на основе байесовских сетей (multidimensional Bayesian network classifier – MBC) на множестве $\mathbf{V} = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$, $n \geqslant 1$, дискретных случайных переменных – это байесовская сеть с особой структурой[10]. Множество переменных $\mathbf{V}$ разделяется на два множества: $\mathbf{V}_C = \{\mathbf{C}_1, \ldots, \mathbf{C}_d\}$, $d \geqslant 1$, – переменные классов и $\mathbf{V}_X = \{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$, $m \geqslant 1$, – переменные признаков (при этом $d + m = n$). Множество дуг $\mathbf{A}$ разделяется на три множества $\mathbf{A}_C$, $\mathbf{A}_X$, $\mathbf{A}_{CX}$, таких, что $\mathbf{A}_C \subseteq \mathbf{V}_C \times \mathbf{V}_C$ сформировано из дуг между переменными классов, $\mathbf{A}_X \subseteq \mathbf{V}_X \times \mathbf{V}_X$ сформировано из дуг между переменными признаков, а $\mathbf{A}_{CX} \subseteq \mathbf{V}_C \times \mathbf{V}_X$ сформировано из дуг от переменных классов к переменным признаков. Соответствующими подграфами являются $\mathbf{G}_C = (\mathbf{V}_C, \mathbf{A}_C)$ – подграф *классов*, $\mathbf{G}_X = (\mathbf{V}_X, \mathbf{A}_X)$ – подграф *признаков* и $\mathbf{G}_{CX} = (\mathbf{V}, \mathbf{A}_{CX})$ – подграф *мостов* (см. рис. 4.7).

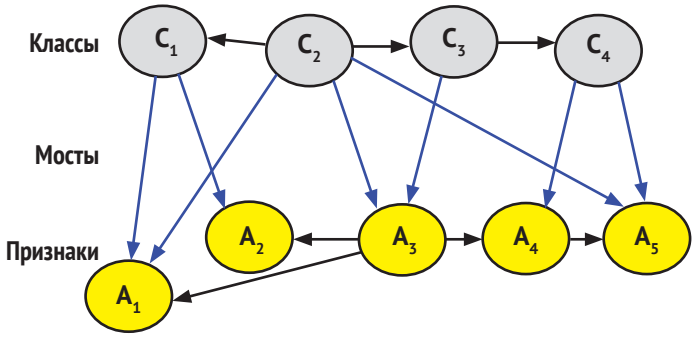


Рис. 4.7. Многомерный классификатор на основе байесовских сетей, в котором выделены три подграфа: классы, признаки и мосты

Различные структуры графов для подграфов классов и признаков могут приводить к созданию разнообразных семейств многомерных классификаторов на основе байесовских сетей [19]. Например, можно было бы ввести ограничение, по которому подграф классов должен иметь только структуру дерева, и предположить, что атрибуты независимы по отношению к переменным класса. В других вариантах можно было бы определить одинаковую структуру для обоих подграфов, например дерево-дерево, полидерево-полидерево, НАГ-НАГ (НАГ – направленный ациклический граф). При увеличении сложности структур, прини-

[10] Байесовские сети рассматриваются в главе 7.

маемых для каждого подграфа, возрастает и сложность обучения этих структур.

Задача классификации экземпляра (элемента) данных с помощью многомерного классификатора на основе байесовских сетей, т. е. наиболее правдоподобное сочетание классов, соответствует задаче наиболее вероятного объяснения (MPE – most probable explanation) или задаче *абдукции* (abduction) для байесовских сетей. Другими словами, необходимо определить наиболее вероятные значения для переменных класса $\mathbf{V} = \{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$ с учетом признаков. Это сложная задача с высокой стоимостью вычислений. Существует несколько методов снижения сложности по времени [2], но даже в этом случае такая методика применима только к задачам с ограниченным количеством классов.

Обучение байесовских сетей (многомерные классификаторы на основе байесовских сетей представляют собой один из типов байесовских сетей) и вычисление наиболее вероятного объяснения (MPE) будут рассматриваться в главах 7 и 8.

4.5.2 Байесовские классификаторы на основе цепи

Классификаторы на основе цепи (chain classifiers) представляют альтернативный метод классификации с несколькими метками, которая включает зависимости классов при сохранении эффективности вычислений, свойственной методике бинарной релевантности [12]. Классификатор на основе цепи состоит из d базовых бинарных классификаторов, объединенных в цепь так, что каждый классификатор содержит классы, спрогнозированные предыдущими классификаторами как дополнительные атрибуты. Таким образом, вектор признаков для каждого бинарного классификатора L_i дополняется метками (0/1) из всех классификаторов, предшествующих ему в цепи. Каждый классификатор в цепи тренируется для обучения присваиванию метки l_i с учетом признаков, дополняемых всеми предыдущими метками классов в цепи $L_1, L_2, \ldots, L_{i-1}$. Классификация начинается с L_1, затем спрогнозированные классы распространяются далее по цепи так, что для $L_i \in \mathcal{L}$ (где $\mathcal{L} = \{L_1, L_2, \ldots, L_d\}$) прогнозируется вероятность $P(L_i \mid \mathbf{X}, L_1, L_2, \ldots, L_{i-1})$. Как и в методе бинарной релевантности, вектор классов определяется объединением выходных данных всех бинарных классификаторов в цепи.

Байесовские классификаторы на основе цепи (Bayesian chain classifiers) представляют собой один из типов классификатора на основе цепи в рабочем пространстве вероятностных методов. Если применить цепочное правило теории вероятностей, то формулу 4.12 можно записать в следующем виде:

$$ArgMax_{C1,\ldots,Cd} P(C_1 \mid C_2, \ldots, C_d, \mathbf{X}) P(C_2 \mid C_3, \ldots, C_d, \mathbf{X}) \ldots P(C_d \mid \mathbf{X}). \qquad (4.13)$$

Если учитывать отношения зависимостей между переменными классов и представить эти отношения в виде направленного ациклического графа (directed acyclic graph – DAG), то можно упростить формулу 4.13, рассматривая зависимости, отображенные в графе, так, чтобы только родители каждой переменной класса включались в цепь, а все прочие предшествующие по порядку в цепи классы исключались. Тогда можно записать формулу 4.13 в следующем виде:

$$ArgMax_{C1,\ldots,Cd} \prod_{i=1}^{d} P(C_i \mid \mathbf{Pa}(C_i), \mathbf{X}), \tag{4.14}$$

где $\mathbf{Pa}(C_i)$ – родители класса i в направленном ациклическом графе, который представляет зависимости между переменными классов.

Возможно и дальнейшее упрощение, если предположить, что наиболее вероятное совместное сочетание классов можно аппроксимировать простым соединением отдельных наиболее вероятных классов. Таким образом, необходимо решить следующую систему уравнений как аппроксимацию формулы 4.14:

$$\begin{aligned} &ArgMax_{C_1} P(C_1 \mid \mathbf{Pa}(C_1), \mathbf{X}) \\ &ArgMax_{C_2} P(C_2 \mid \mathbf{Pa}(C_2), \mathbf{X}) \\ &\qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &ArgMax_{C_d} P(C_d \mid \mathbf{Pa}(C_d), \mathbf{X}) \end{aligned}$$

Эта последняя аппроксимация соответствует байесовскому классификатору на основе цепи. Таким образом, для байесовского классификатора на основе цепи принимаются два основных предположения:

1) структура зависимости классов с учетом признаков может быть представлена в виде направленного ациклического графа;
2) процедура присваивания наиболее вероятного совместного сочетания классов (полная абдукция) аппроксимируется простым соединением наиболее вероятных отдельных классов.

Первое предположение является обоснованным, если имеется достаточный объем данных для правильной аппроксимации структуры зависимостей классов и верно предположение о том, что эта аппроксимация выполнена с учетом соответствующих признаков. Относительно второго предположения хорошо известно, что полная абдукция или наиболее вероятное объяснение не всегда равнозначно максимизации отдельных классов. Но это предположение менее строгое, чем аналогичное предположение в методе бинарной релевантности. Байесовские классификаторы на основе цепи предоставляют неплохой альтернативный

вариант для многомерной классификации, поскольку включают в той или иной степени зависимости между переменными классов, сохраняя при этом эффективность метода бинарной релевантности.

Для *базового* (base) классификатора, принадлежащего каждому классу, можно использовать любой из байесовских классификаторов, представленных в предыдущих разделах, например наивный байесовский классификатор. Предполагая, что имеется структура зависимостей классов, представленная в виде направленного ациклического графа (эту структуру можно обучить по имеющимся данным, см. главу 8), каждый классификатор может быть обучен тем же способом, что и наивный байесовский классификатор. Для этого достаточно просто включить как дополнительные атрибуты переменные классов в соответствии со структурой зависимостей классов. Проще всего включить только родительские узлы каждого класса в соответствии с этим графом зависимостей. Общий принцип построения байесовского классификатора на основе цепи показан на рис. 4.8.

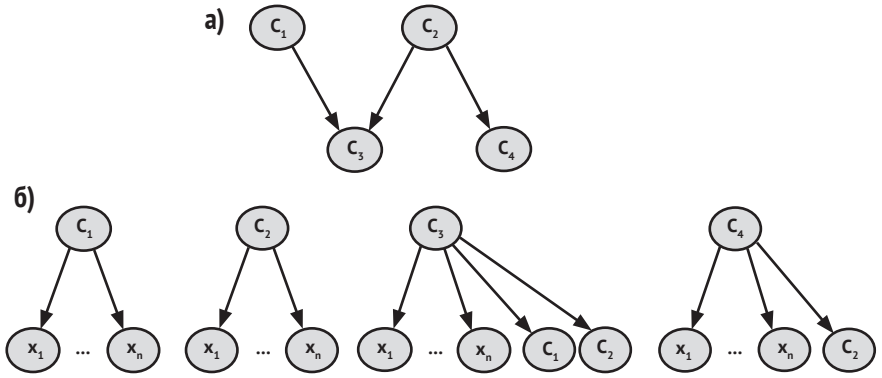


Рис. 4.8. Пример байесовского классификатора на основе цепи: а) байесовская сеть представляет структуру зависимости классов, б) наивные байесовские классификаторы – по одному для каждого класса. Каждый базовый классификатор, определенный для C_j, включает множество атрибутов $X_1, \ldots, X_n$ плюс родители в структуре зависимостей как дополнительный атрибут

Для классификации экземпляра (элемента данных) все классификаторы применяются одновременно и все классы, апостериорная вероятность которых превышает пороговое значение, возвращаются как выходные данные.

4.6 Иерархическая классификация

Иерархическая классификация представляет собой тип многомерной классификации, при которой все классы упорядочены в предварительно определенной структуре, обычно в дереве, или в общем случае в на-

правленном ациклическом графе (directed acyclic graph – DAG). С учетом иерархической организации классов можно улучшить производительность классификации. В иерархической классификации экземпляр, принадлежащий некоторому классу, автоматически принадлежит всем суперклассам, т. е. родительским классам для этого класса. Это называют *ограничением иерархии* (hierarchy constraint). Иерархическая классификация применяется в нескольких областях, таких как категоризация текстов, предсказание функции белка и распознавание образов.

Как и при многомерной классификации, существуют две основные методики иерархической классификации: глобальные классификаторы и локальные классификаторы. При глобальном подходе создается классификатор для прогнозирования всех классов сразу. Для крупных иерархий при этом существенно возрастает сложность вычислений. Схемы с локальными классификаторами тренируют несколько классификаторов и объединяют их выходные данные. Для схем с локальными классификаторами существуют три основные методики. Локальный классификатор на каждый уровень иерархии тренирует один мультиклассовый классификатор для каждого уровня иерархии классов. Локальный бинарный классификатор для каждого узла – здесь создается бинарный классификатор для каждого узла (класса) в иерархии, за исключением корневого узла. Локальный классификатор для каждого родительского узла (Local Classifier per Parent Node – LCPN) – мультиклассовый классификатор тренируется для прогнозирования своих узлов-потомков.

Локальные методики широко используют нисходящий (top-down) способ классификации [14]. Классификатор на самом верхнем уровне выбирает определенный класс, а все прочие классы исключаются. Затем анализируются потомки выбранного класса и т. д. При таком подходе возможна следующая проблема: если ошибка возникает на верхних уровнях иерархии, то ее невозможно устранить, и она распространяется на более низкие уровни. Альтернативным подходом является анализ путей в иерархии и выбор пути, в наибольшей степени соответствующего результатам локальных классификаторов. Метод, основанный на этом принципе, описывается ниже.

4.6.1 Оценка цепного пути

Метод оценки цепного пути (Chained Path Evaluation – CPE) [13] анализирует каждый путь от корня до листа в иерархии, принимая во внимание уровень прогнозируемых меток для получения оценки каждого пути, и в конечном итоге возвращает путь с наилучшей оценкой. Кроме того, этот метод учитывает отношения каждого узла со своими предками в иерархии на основе цепных классификаторов. Метод оценки цепного пути (CPE) состоит из двух частей: тренировочное обучение и классификация.

4.6.1.1 Тренировочное обучение

Локальный классификатор тренируется для каждого узла C_i в иерархии, исключая узлы-листья, для классификации соответствующих узлов-потомков, т. е. используется схема LCPN (см. рис. 4.9). Классификатор для каждого узла C_i, например наивный байесовский классификатор, тренируется с учетом экземпляров из всех своих узлов-потомков, а также с учетом некоторых экземпляров из соседних узлов (узлов-братьев) на том же уровне иерархии. Например, классификатор C_2 на рис. 4.9 будет тренироваться для классификации C_5, C_6, C_7, а дополнительные экземпляры будут взяты для рассмотрения из классов C_3 и C_4, которые представляют неизвестный класс для C_1.

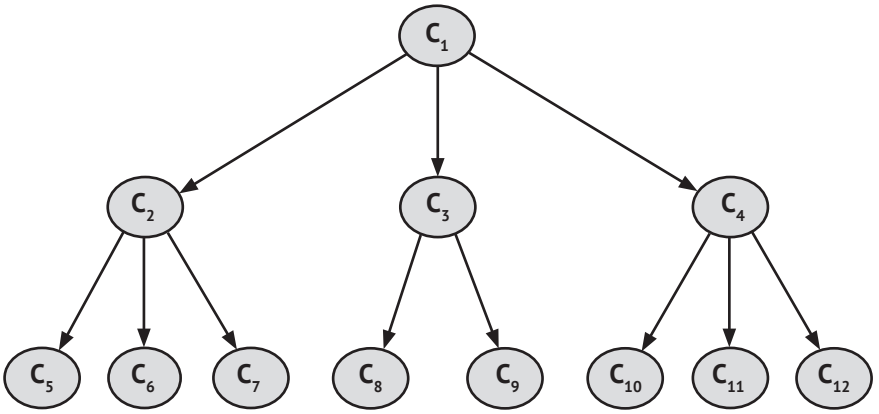


Рис. 4.9. Пример иерархической структуры (дерева). Для каждого узла, не являющегося листом, локальный классификатор тренируется для прогнозирования своих узлов-потомков: C_1 классифицирует C_2, C_3, C_4; C_2 классифицирует C_5, C_6, C_7; C_3 классифицирует C_8, C_9; C_4 классифицирует C_{10}, C_{11}, C_{12}

Чтобы учесть отношения с другими узлами в этой иерархии, класс, прогнозируемый родителем (в структуре дерева) или несколькими родителями (в направленном ациклическом графе), включается как дополнительные атрибуты в каждый локальный классификатор на основании принципа, вводимого байесовскими цепными классификаторами. Например, классификатор C_2 на рис. 4.9 в качестве дополнительного атрибута будет содержать класс, прогнозируемый его родителем C_1.

4.6.1.2 Классификация

На этапе классификации вероятности для каждого класса во всех локальных классификаторах вычисляются на основе входных данных (признаков каждого экземпляра). После вычисления вероятностей для каждого узла в иерархии, не являющегося листом, эти узлы объединяются для получения оценки для каждого пути.

Оценка (score) для каждого пути в иерархии вычисляется как взвешенная сумма логарифмов вероятностей всех локальных классификаторов в этом пути:

$$score = \sum_{i=0}^{n} w_{Ci} \times \log(P(C_i \mid X_i, pa(C_i))), \tag{4.15}$$

где C_i – классы для каждого локального классификатора для каждого родительского узла (LCPN), X_i – вектор атрибутов, $pa(C_i)$ – прогнозируемый родителем класс, w_{Ci} – весовой коэффициент. Цель назначения весовых коэффициентов – придать бóльшую важность более высоким уровням иерархии, так как ошибки на более высоких уровнях иерархии (которые соответствуют более обобщенным концепциям) обходятся дороже, чем ошибки на более низких уровнях (соответствующие более специализированным концепциям) [13]. Включение суммы логарифмов используется для обеспечения стабильности вычислений при определении вероятностей для длинных путей.

После вычисления оценок для всех путей путь с наивысшей оценкой будет выбран как множество классов, соответствующих конкретному экземпляру. Для примера на рис. 4.9 оценка будет вычисляться для каждого пути от корня до каждого узла-листа: Путь 1: C_1–C_2–C_5, Путь 2: C_1–C_2–C_6 и т. д. В этом примере существует восемь путей. Предположим, что наивысшую оценку имеет Путь 4: C_1–C_3–C_8, тогда эти три класса будут возвращены как выходные данные классификатора.

4.7 Приложения

В этом разделе рассматриваются приложения двух типов байесовских классификаторов для решения двух практических задач. Сначала демонстрируется использование частично наивного классификатора для присваивания пикселам в изображении (фотографии) меток «кожа» и «не кожа». Затем рассматривается использование многомерных цепных классификаторов для выбора лекарственных средств для лечения вируса иммунодефицита человека (ВИЧ – HIV).

4.7.1 Визуальное определение кожи человека на изображениях

Определение (участков) кожи человека – это полезный этап предварительной обработки для многих приложений в области компьютерного зрения, таких как идентификация личности, распознавание жестов и многие другие. Поэтому весьма важно применять для этого чрезвычайно точный и эффективный классификатор. Простой и очень быстрый способ получения приблизительной классификации пикселов в изображении как классов «кожа» и «не кожа» основан на цветовых атрибутах

каждого пиксела. Обычно пикселы в цифровом изображении представлены в виде сочетания трех основных (простых) цветов: красного – Red (R), зеленого – Green (G) и синего – Blue (B). Это сочетание известно как *RGB*-модель. Компонент каждого цвета может принимать числовые значения в определенном интервале, а именно от 0 до 255. Существуют и другие цветовые модели, например HSV, YIQ и т. д.

Для классификации пикселов как классов «кожа» и «не кожа» может быть создан наивный байесовский классификатор с использованием трех цветовых значений RGB в качестве атрибутов. Но возможно также применение другой цветовой модели для более эффективной классификации. В другом варианте можно объединить несколько цветовых моделей в одном классификаторе, используя все атрибуты из различных моделей. Последний вариант перспективен с точки зрения получения преимуществ использования информации, предоставляемой различными моделями. Но при использовании наивного байесовского классификатора нарушается предположение о независимости – различные модели не являются независимыми, так как одну модель можно вывести из другой.

Альтернативный вариант – применение частично наивного байесовского классификатора и выбор наилучших атрибутов из различных цветовых моделей для классификации изображения кожи посредством исключения или объединения атрибутов. Для этого воспользуемся тремя различными цветовыми моделями: RGB, HSV и YIQ, – так что всего будет использоваться девять атрибутов. Эти атрибуты (цветовые компоненты) были предварительно дискретизированы в ограниченном количестве интервалов значений. Затем исходный наивный байесовский классификатор был обучен на имеющихся данных, т. е. на экземплярах пикселов классов «кожа» и «не кожа», взятых из нескольких изображений. Этот исходный классификатор достиг точности 94 %, когда применялся к другим (тестовым) изображениям.

Затем классификатор был *оптимизирован* с использованием метода, описанного в разделе 4.4. Начиная с полного наивного байесовского классификатора с девятью атрибутами, этот метод поэтапно применяет процедуры исключения и объединения переменных до тех пор, пока не будет получен наипростейший классификатор с максимальной точностью. Последовательность операций и конечная структура показаны на рис. 4.10. Здесь можно видеть, что изначально алгоритм исключает несколько несущественных или избыточных атрибутов, затем объединяет два зависимых атрибута, далее исключает еще два атрибута и продолжает работу до тех пор, пока не получит окончательную структуру с тремя атрибутами: *RG*, *Y*, *I* (первый атрибут является объединением двух исходных атрибутов). Полученная конечная модель была оценена на тех же тестовых изображениях, и точность была улучшена до 98 %.

Пример изображения с пикселами, определенными этим классификатором как класс «кожа», показан на рис. 4.11.

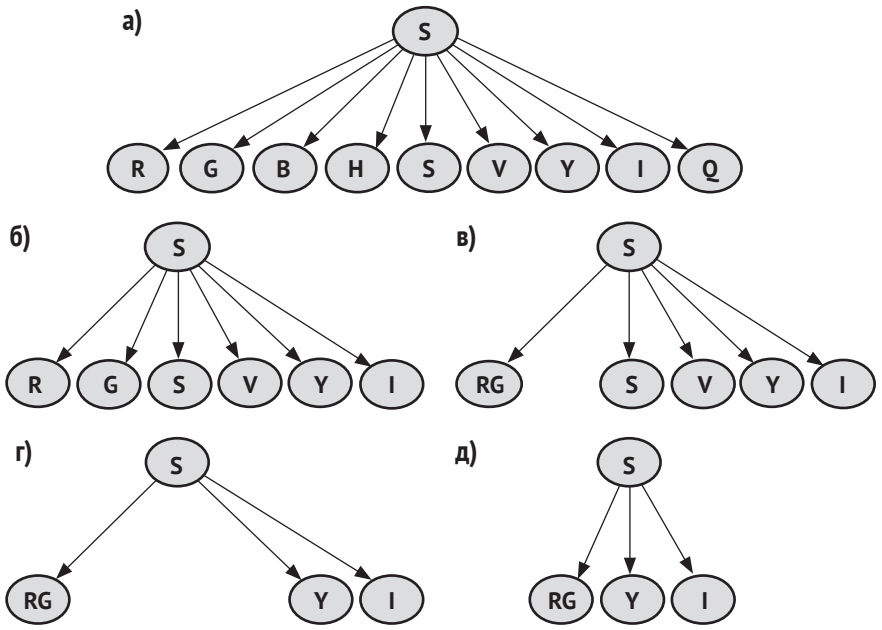


Рис. 4.10. Здесь показан процесс оптимизации частично наивного байесовского классификатора для визуального определения человеческой кожи на изображениях – от первоначальной модели с девятью атрибутами (а) до конечной модели с тремя атрибутами (д): а) первоначальная структура с цветовыми моделями RGB, HSV и YIQ, б) исключение атрибутов *B*, *Q* и *H*; в) объединение атрибутов *R* и *G*, г) исключение атрибутов *V* и *S*, д) конечная модель

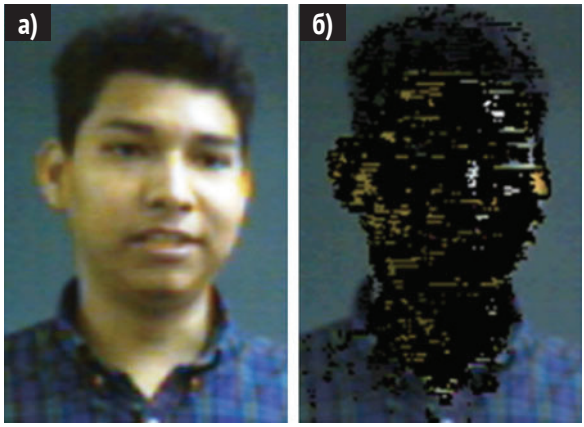


Рис. 4.11. Пример изображения, в котором пикселы классифицированы как «кожа» и «не кожа»: а) исходное изображение, б) изображение с обнаруженными пикселами класса «кожа» (обозначены черным цветом)

В рассматриваемом здесь примере частично наивный байесовский классификатор получает значительное преимущество по сравнению с обычным наивным байесовским классификатором, но в то же время создает более простую модель (в плане количества переменных и требуемых параметров) [9].

4.7.2 Выбор лекарственных средств для лечения вируса иммунодефицита человека

Вирус иммунодефицита человека (ВИЧ; Human Immunodeficiency Virus – HIV) – это (вирусный) фактор, являющийся причиной СПИДа (AIDS), условие развития поражения иммунной системы, которое способствует возникновению инфекционных заболеваний, опасных для жизни человека. Для борьбы с ВИЧ-инфекцией были разработаны некоторые антиретровирусные лекарственные средства, принадлежащие к различным классам медицинских препаратов, воздействующих на специфические фазы цикла воспроизводства вируса. В антиретровирусной терапии в основном применяется сочетание трех или четырех антиретровирусных лекарственных средств. Выбор сочетания этих средств зависит от состояния пациента, которое может быть охарактеризовано в соответствии с мутациями вируса, присутствующего в организме пациента. Таким образом, важно выбрать наилучшее сочетание лекарств в соответствии с мутациями вируса в организме пациента.

Выбор наиболее эффективной группы антиретровирусных лекарственных средств для пациента может рассматриваться как частный случай задачи классификации с несколькими метками, в которой классами являются различные типы антиретровирусных лекарств, а атрибуты – это мутации вируса. Поскольку классификация с несколькими метками представляет собой частный случай многомерной классификации, эта конкретная задача может быть точно смоделирована с использованием многомерной байесовской сети (классификатора) (MBC). Применяя алгоритм обучения, можно выявить связи, существующие между антиретровирусными средствами и мутациями, а кроме того, получить модель с высокой эффективностью прогнозов.

Альтернативой обучению многомерной байесовской сети (классификатора) является алгоритм MB-MBC [3]. Этот специализированный алгоритм использует марковское ограждение (Markov blanket – MB) переменной каждого класса для снижения сложности вычислений при обучении многомерной байесовской сети (классификатора) (MBC), отфильтровывая (исключая) те переменные, которые не улучшают классификацию. Марковское ограждение переменной *C*, обозначаемое как *MB*(*C*), – это минимальное множество переменных, таких,

что $I(C, \mathbf{S} \mid MB(C))$ истинно для каждого подмножества переменных **S**, где **S** не содержит в качестве своих элементов ни одну из переменных, принадлежащих объединению $MB(C) \cup C$. Другими словами, марковское ограждение переменной C – это минимальное множество переменных, при которых C условно независима от всех остальных переменных.

Для прогнозирования наиболее подходящей группы антиретровирусных лекарственных средств для пациента с учетом присутствующих в его организме мутаций вируса выполняется обучение марковского ограждения для каждого антиретровирусного лекарства. Например, если рассматривается группа ингибиторов обратной транскриптазы (группа антиретровирусных лекарственных средств, которые подавляют активность обратной транскриптазы в фазе репликации жизненного цикла ВИЧ) как переменные классов, а группа мутаций как атрибуты, то марковское ограждение для полного множества ингибиторов обратной транскриптазы обучается определению существующих связей типа антиретровирус–антиретровирус и антиретровирус–мутация. Обучение марковского ограждения переменной каждого класса соответствует обучению неориентированной структуры для многомерной байесовской сети (классификатора), т. е. трех подграфов. Наконец, ориентация для всех трех этих подграфов определяется на последнем шаге выполнения алгоритма MB-MBC. На рис. 4.12 показан полученный в результате итоговый многомерный байесовский классификатор (сеть).

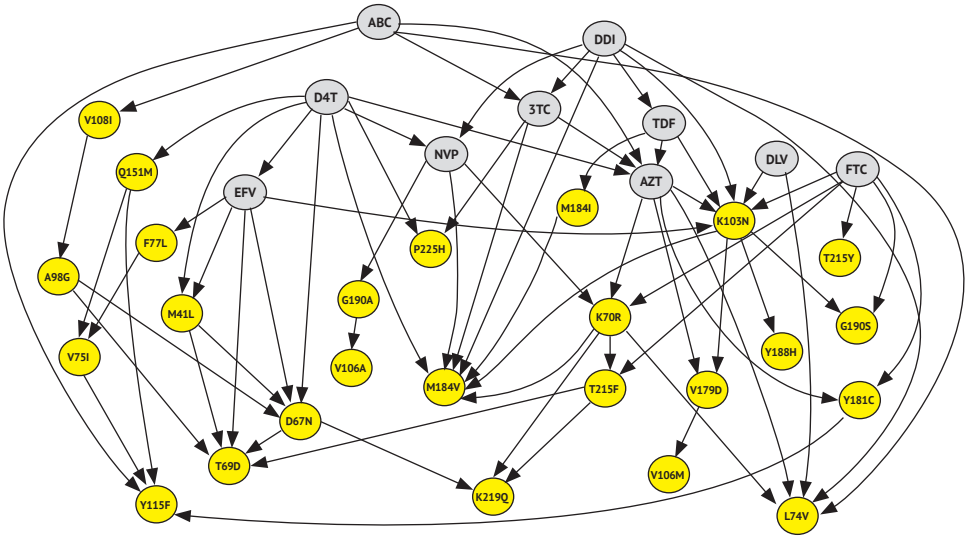


Рис. 4.12. Графовая структура для многомерной байесовской сети (классификатора), обучаемой с помощью алгоритма MB-MBC, для множества ингибиторов обратной транскриптазы (показаны зеленым цветом) и множества мутаций (показаны желтым цветом) (иллюстрация взята из [3])

4.8 Материалы для дополнительного чтения

Общее введение и сравнение различных методик классификации см. в [10]. Определение стоимости классификации описано в [5]. Классификаторы TAN и BAN рассматриваются в [6]. Сравнение различных классификаторов на основе байесовских сетей представлено в [4]. Введение в частично наивную методику см. в [16], более подробное описание в [11]. По ссылке [18] представлен общий обзор многомерных классификаторов. Различные альтернативные варианты для многомерной байесовской сети (классификатора) см. в [2]. Цепные классификаторы представлены в [12], байесовские цепные классификаторы в [17]. Обзор различных методик иерархической классификации и их практических приложений см. в [15].

4.9 Задания и упражнения

1. На основе данных для примера *golf* в табл. 4.1 завершить создание таблиц условных вероятностей для наивного байесовского классификатора с использованием оценки максимального правдоподобия.
2. Определить класс с максимальной вероятностью для наивного байесовского классификатора в примере *golf* из задания 1 с учетом всех сочетаний значений атрибутов.
3. На основе результатов выполнения задания 2 сформировать набор правил классификации, равнозначных наивному байесовскому классификатору, для определения результатов «игра возможна» / «игра невозможна» на основе значений атрибутов.
4. Предположим, что выполняется преобразование наивного байесовского классификатора для примера *golf* в модель TAN со следующей структурой зависимостей атрибутов: *прогноз погоды* → *температура*, *прогноз погоды* → *влажность*, *температура* → *ветер*. Используя тот же набор данных, оценить таблицы условных вероятностей для этой модели TAN.
5. На основе набора данных для примера *golf* определить оценку взаимной информации между классом и каждым атрибутом. Создать частично наивный байесовский классификатор, исключив те атрибуты, которые имеют низкий уровень взаимной информации с классом (необходимо предварительно определить пороговое значение).
6. Расширить задачу 5: теперь необходимо оценить взаимную информацию между каждой парой атрибутов с учетом переменной класса. Исключить или объединить те атрибуты, которые не являются

условно независимыми с учетом класса в соответствии с предварительно определенным пороговым значением. Показать структуру и параметры созданных в результате классификаторов.

7. Предположим, что пример *golf* преобразовывается в задачу многомерной классификации с двумя классами: *игра* и *прогноз погоды* – и с тремя атрибутами: *температура*, *влажность* и *ветер*. Также предположим, что создается многомерный классификатор на основе бинарной релевантности – независимый классификатор для каждой переменной класса. Приняв, что каждый такой классификатор – это наивный байесовский классификатор, определить, какой будет структура итогового классификатора. Получить параметры для этого классификатора на основе данных в табл. 4.1.
8. Для задания 7 предположить, что теперь создается наивный байесовский классификатор на основе методики мощности множества. Какой теперь будет структура и параметры итоговой модели? Использовать тот же набор данных.
9. *** Сравнить структуру, сложность (по количеству параметров) и точность классификации различных байесовских классификаторов – наивного БК, TAN, BAN, – используя несколько наборов данных (например, WEKA [7] реализацию байесовских классификаторов, а также некоторые наборы данных из репозитория UCI [1]). Всегда ли модели TAN и BAN превосходят по эффективности наивный байесовский классификатор? Почему?
10. *** Иерархический классификатор – это особый частный тип многомерного классификатора, в котором классы организованы в иерархию. Например, иерархия классов животных. Ограничение иерархии состоит в том, что экземпляр (элемент данных), принадлежащий конкретному классу, обязательно должен принадлежать и всем соответствующим суперклассам в этой иерархии (иерархическое ограничение). Как можно спроектировать многомерный классификатор, чтобы обеспечить соблюдение этого иерархического ограничения? Расширить байесовский цепной классификатор для иерархической классификации.

Ссылки на источники

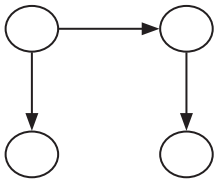
1. Bache, K., Lichman, M. UCI machine learning repository. University of California, School of Information and Computer Science. Irvine. http://archive.ics.uci.edu/ml. Accessed 22 Sept 2014 (2013).
2. Bielza, C., Li, G., Larrañaga, P. Multi-dimensional classification with bayesian networks. Int. J. Approx. Reason. 52, 705–727 (2011).

3. Borchani, H., Bielza, C., Toro, C., Larrañaga, P. Predicting human immunodeficiency virus inhibitors using multi-dimensional Bayesian network classifiers. Artif. Intell. Med. 57, 219–229 (2013).
4. Cheng, J., Greiner, R. Comparing Bayesian network classifiers. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, p. 101–108 (1999).
5. Drummond, C., Holte, R. C. Explicitly representing expected cost: an alternative to the ROC representation. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p. 198–207 (2000).
6. Friedman, N., Geiger, D., Goldszmidt, M. Bayesian network classifiers. Mach. Learn. 29, 131–163 (1997).
7. Hall, M., Frank, E., Holmes, G., Pfahringer, B. and Reutemann, P., Witten, I. H. The WEKA datamining software: an update. In: ACMSIGKDD Explorations Newsletter. ACM, p. 10–18 (2009).
8. Kwoh, C. K., Gillies, D. F. Using hidden nodes in Bayesian networks. Artificial Intelligence, vol. 88, p. 1–38. Elsevier, Essex (1996).
9. Martinez, M., Sucar, L. E. Learning an optimal naive Bayes classifier. In: International Conference on Pattern Recognition (ICPR), vol. 3, p. 1236–1239 (2006).
10. Michie, D., Spiegelhalter, D. J., Taylor, C. C. Machine Learning, Neural and Statistical Classification. Ellis Howard, England (2004).
11. Pazzani, M. J. Searching for Dependencies in Bayesian Classifiers. Artificial Intelligence and Statistics IV. Lecture Notes in Statistics, Springer-Verlag, New York (1997).
12. Read, J., Pfahringer, B., Holmes, G., Frank, E. Classifier chains for multi-label classification. In: Proceedings ECML/PKDD, p. 254–269 (2009).
13. Ramírez, M., Sucar, L. E., Morales, E. Path evaluation for hierarchicalmulti-label classification. In: Proceedings of the Twenty-Seventh International Florida Artificial Intelligence Research Society Conference (FLAIRS), p. 502–507 (2014).
14. Silla Jr., C. N., Freitas, A. A. Novel top-down approaches for hierarchical classification and their application to automatic music genre classification. In: IEEE International Conference on Systems, Man, and Cybernetics, p. 3499–3504. October 2009.
15. Silla Jr., C. N., Freitas, A. A. A survey of hierarchical classification across different application domains. Data Min. Knowl. Discov. 22 (1–2), 31–72 (2011).
16. Sucar, L. E., Gillies, D. F., Gillies, D. A. Objective probabilities in expert systems. Artif. Intell. 61, 187–208 (1993).

17. Sucar, L. E., Bielza, C., Morales, E., Hernandez, P., Zaragoza, J., Larrañaga, P. Multi-label classification with Bayesian network-based chain classifiers. Pattern Recognit. Lett. 41, 14–22 (2014).
18. Tsoumakas, G., Katakis, I. Multi-label classification: an overview. Int. J. DataWareh. Min. 3, 1–13 (2007).
19. van der Gaag L. C., de Waal, P. R. Multi-dimensional Bayesian network classifiers. In: Third European Conference on Probabilistic Graphic Models, p. 107–114. Prague, Czech Republic (2006).

Глава 5

Скрытые марковские модели



5.1 Введение

Марковские цепи (Markov chains) – это еще один класс вероятностных графовых моделей, который представляет динамические процессы, в частности изменение состояния процесса во времени. Например, предположим, что моделируется изменение погоды в конкретной местности в течение некоторого времени. В весьма упрощенной модели предполагается, что погода не изменяется на протяжении дня и может иметь три возможных состояния: *солнечно*, *облачно*, *дождь*. Кроме того, предполагается, что погода в конкретный день зависит только от погоды в предыдущий день. Таким образом, можно считать эту упрощенную модель погоды марковской цепью, в которой существует переменная состояния для каждого отдельного дня с тремя возможными значениями. Эти переменные связываются в цепь (chain) с ориентированной (направленной) дугой от одного дня к следующему (см. рис. 5.1). При этом подразумевается то, что называют *марковским свойством* (Markov property), – состояние погоды в следующий день S_{t+1} не зависит от всех предыдущих дней при рассмотрении погоды в текущий день S_t, т. е. $P(S_{t+1} \mid S_t, S_{t-1}, \ldots) = P(S_{t+1} \mid S_t)$. Таким образом, в марковской цепи основным требуемым параметром является вероятность состояния с учетом вероятности предыдущего состояния.

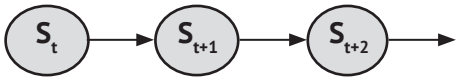


Рис. 5.1. Марковская цепь, в которой каждый узел представляет состояние в определенный момент времени

Описанная выше модель предполагает, что можно точно оценить («измерить») погоду каждый день, т. е. состояние является *наблюдае-*

мым (observable). Но это предположение не всегда истинно. Во многих приложениях нет возможности непосредственного наблюдения за состоянием процесса, поэтому необходимо перейти к *скрытой марковской модели* (Hidden Markov Model), в которой состояние скрыто. В этом случае в дополнение к вероятности следующего состояния с учетом текущего состояния добавляется еще один параметр, моделирующий неопределенность текущего состояния, представленный как вероятность *наблюдения* (observation) с учетом текущего состояния $P(O_t \mid S_t)$. Этот тип модели обладает большей мощью, чем простая марковская цепь, и имеет множество практических приложений, например при распознавании речи и жестов.

После краткого ведения в марковские цепи в следующем разделе будут подробно рассматриваться скрытые марковские модели, в том числе методы выполнения вычислений интереса (значимости) для этого типа моделей. Затем будут описаны некоторые расширения стандартных скрытых марковских моделей, а завершается глава двумя примерами практических приложений.

5.2 Марковские цепи

Марковская цепь (Markov chain – MC) – это *конечный автомат* (finite-state machine), имеющий дискретное (конечное) число состояний $q_1, q_2, \ldots, q_n$, а переходы между состояниями являются недетерминированными, т. е. существует вероятность перехода из состояния q_i в другое состояние q_j: $P(S_t = q_j \mid S_{t-1} = q_i)$. Время также дискретно, так что цепь может находиться в определенном состоянии q_i в любой момент (интервал) времени t. Это соответствует марковскому свойству, т. е. вероятность следующего состояния зависит только от текущего состояния.

Формально марковская цепь определяется в следующем виде:

- множество состояний: $Q = \{q_1, q_2, \ldots, q_n\}$;
- вектор априорных вероятностей: $\Pi = \{\pi_1, \pi_2, .., \pi_n\}$, где $\pi_i = P(S_0 = q_i)$;
- матрица вероятностей переходов: $A = \{a_{ij}\}$, $i = [1..n]$, $j = [1..n]$,
  где $a_{ij} = P(S_t = q_j \mid S_{t-1} = q_i)$,
  где n – количество состояний, а S_0 – начальное состояние. В компактной форме марковскую цепь можно представить так: $\lambda = \{A, \Pi\}$.

Марковская цепь (первого порядка) обладает следующими свойствами:

1) аксиомы вероятности: $\sum_i \pi_i = 1$ и $\sum_j a_{ij} = 1$;
2) марковское свойство: $P(S_t = q_j \mid S_{t-1} = q_i, S_{t-2} = q_k, \ldots) = P(S_t = q_j \mid S_{t-1} = q_i)$.

Например, рассмотрим предложенную выше простую модель погоды с тремя состояниями: $q_1 =$ *солнечно*, $q_2 =$ *облачно*, $q_3 =$ *дождь*. В этом случае

для определения марковской цепи потребуется вектор, состоящий из трех априорных вероятностей (см. табл. 5.1), и матрица 3×3 вероятностей переходов (см. табл. 5.2).

Таблица 5.1. Априорные вероятности для примера прогнозирования погоды

Солнечно	Облачно	Дождь
0.2	0.5	0.3

Таблица 5.2. Вероятности переходов для примера прогнозирования погоды

	Солнечно	Облачно	Дождь
Солнечно	0.8	0.1	0.1
Облачно	0.2	0.6	0.2
Дождь	0.3	0.3	0.4

Матрица переходов может быть представлена графически с помощью так называемой *диаграммы переходов между состояниями*, или просто *диаграммы состояний* (state diagram). Эта диаграмма представляет собой ориентированный граф, в котором каждый узел – это состояние, а дуги представляют возможные переходы между состояниями. Если дуга между состояниями q_i и q_j не отображена на диаграмме, это означает, что вероятность соответствующего перехода равна нулю. Пример диаграммы состояний для примера прогнозирования погоды показан на рис. 5.2[11].

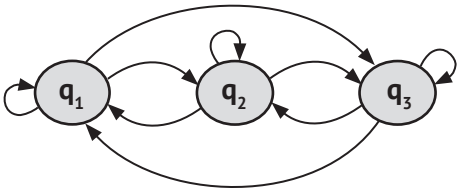


Рис. 5.2. Диаграмма переходов между состояниями для примера прогнозирования погоды

При рассмотрении модели марковской цепи естественным образом возникают три основных вопроса:

- какова вероятность некоторой конкретной последовательности состояний?
- какова вероятность того, что цепь остается в определенном состоянии в течение некоторого интервала времени?

[11] Не следует путать диаграмму состояний, в которой узлы представляют каждое состояние – конкретное значение случайной переменной, – а дуги представляют переходы между состояниями, с диаграммой графовой модели, в которой узел представляет случайную переменную, а дуги представляют вероятностные зависимости.

- каково ожидаемое время, в течение которого цепь будет оставаться в определенном состоянии?

Немного позже мы узнаем, как можно ответить на эти вопросы, и рассмотрим объяснение ответов на примере прогнозирования погоды.

Вероятность последовательности состояний с учетом выбранной модели в общем случае равна произведению вероятностей переходов в этой последовательности состояний:

$$P(q_i, q_j, q_k, \ldots) = a_{0i} a_{ij} a_{jk} \ldots, \tag{5.1}$$

где a_{0i} – переход в начальное состояние рассматриваемой последовательности, который мог бы быть его априорной вероятностью (π_i), или переход из предыдущего состояния (если оно известно).

Например, в модели прогнозирования погоды может потребоваться знание вероятности следующей последовательности состояний: Q = *солнечно*, *солнечно*, *дождь*, *дождь*, *солнечно*, *облачно*, *солнечно*. Предположим, что *солнечно* – это начальное состояние в марковской цепи, тогда:

$$P(Q) = \pi_1 a_{11} a_{13} a_{33} a_{31} a_{12} a_{21} = (0.2)(0.8)(0.1)(0.4)(0.3)(0.1)(0.2)$$
$$= 3.84 \times 10^{-5}.$$

Вероятность того, что модель в течение d интервалов времени остается в некотором конкретном состоянии q_i, равна вероятности для последовательности в этом состоянии в течение $d - 1$ интервалов времени с последующим переходом в некоторое другое состояние. Таким образом:

$$P(d_i) = a_{ii}^{d-1}(1 - a_{ii}). \tag{5.2}$$

Если рассматривать модель прогнозирования погоды, то какова вероятность трех облачных дней (подряд)? Эту вероятность можно вычислить следующим образом:

$$P(d_2 = 3) = a_{22}^2(1 - a_{22}) = 0.6^2(1 - 0.6) = 0.144.$$

Средняя продолжительность нахождения последовательности в определенном состоянии – это ожидаемое значение количества фаз в этом состоянии, т. е. $E(D) = \sum_i d_i P(d_i)$. Подставив формулу 5.2, получим:

$$E(d_i) = \sum_i d_i a_{ii}^{d-1}(1 - a_{ii}). \tag{5.3}$$

Эту формулу можно записать в компактной форме следующим образом:

$$E(d_i) = 1 / (1 - a_{ii}). \tag{5.4}$$

Например, сколько ожидается дней, в течение которых погода будет оставаться облачной? Воспользуемся формулой 5.4:

$$E(d_2) = 1 / (1 - a_{22}) = 1 / (1 - 0.6) = 2.5.$$

5.2.1 Оценка параметров

Еще один важный вопрос – как определить параметры модели – известен как задача *оценки параметров* (parameter estimation). Для марковской цепи параметры могут оцениваться простым подсчетом количества нахождений рассматриваемой последовательности в определенном состоянии i и количества переходов из состояния i в состояние j. Предположим, что существует N последовательностей по наблюдениям. γ_{0i} – количество случаев, когда состояние i является начальным состоянием в последовательности, γ_i – количество наблюдаемых состояний i, γ_{ij} – количество наблюдаемых переходов из состояния i в состояние j. Параметры могут быть оценены по следующим формулам.

Исходные вероятности:

$$\pi_i = \gamma_{0i} / N. \tag{5.5}$$

Вероятности переходов (между состояниями):

$$a_{ij} = \gamma_{ij} / \gamma_i. \tag{5.6}$$

Следует отметить, что для самого последнего состояния в последовательности не наблюдается следующее состояние, поэтому самое последнее состояние для всех последовательностей не включается в расчеты.

Например, пусть в примере прогнозирования погоды имеются следующие четыре наблюдаемые последовательности (q_1 = *солнечно*, q_2 = *облачно*, q_3 = *дождь*):

$$q_2, q_2, q_3, q_3, q_3, q_3, q_1$$
$$q_1, q_3, q_2, q_3, q_3, q_3, q_3$$
$$q_3, q_3, q_2, q_2$$
$$q_2, q_1, q_2, q_2, q_1, q_3, q_1$$

С учетом этих четырех последовательностей соответствующие параметры можно оценить так, как показано в табл. 5.3 и 5.4.

Таблица 5.3. Вычисленные априорные вероятности для примера прогнозирования погоды

Солнечно	Облачно	Дождь
0.25	0.5	0.25

Таблица 5.4. Вычисленные вероятности переходов для примера прогнозирования погоды

	Солнечно	**Облачно**	**Дождь**
Солнечно	0	0.33	0.67
Облачно	0.285	0.43	0.285
Дождь	0.18	0.18	0.64

5.2.2 Сходимость

Возникает еще один важный вопрос: если в последовательности наблюдается большое количество переходов M из одного состояния в другое, то какова вероятность в предельном случае (когда $M \to \infty$) для каждого состояния q_i?

С учетом начального вектора вероятностей Π и матрицы переходов A вероятность каждого состояния $P = \{p_1, p_2, \ldots, p_n\}$ после M *итераций* равна:

$$P = \pi A^M. \tag{5.7}$$

Что происходит, когда $M \to \infty$? Ответ (решение) дает теорема Фробениуса–Перрона (Perron-Frobenius), которая утверждает, что при выполнении следующих двух условий:

1) несократимость: для каждого состояния i существует вероятность $a_{ij} > 0$ перехода в другое состояние j;
2) непериодичность: в цепи не образуются циклы (подмножество состояний, в которых цепь остается непосредственно после перехода в одно из этих состояний).

Тогда при $M \to \infty$ цепь сходится к инвариантному распределению P, такому, что $P \times A = P$, где A – матрица вероятности переходов. Коэффициент сходимости определяется по второму *собственному значению* (eigenvalue) матрицы A.

Например, рассмотрим марковскую цепь с тремя состояниями и следующей матрицей вероятностей переходов между этими состояниями:

$$A = \begin{matrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{matrix}$$

Можно показать, что в этом случае устойчивые вероятности сходятся к $P = \{0.625, 0.3125, 0.0625\}$.

Полезное приложение этого свойства сходимости марковских цепей для классификации веб-страниц описано ниже, в разделе о приложениях. В следующем разделе будут рассматриваться скрытые марковские модели.

5.3 Скрытые марковские модели

Скрытая марковская модель (Hidden Markov model – HMM) – это марковская цепь, в которой состояния не являются непосредственно наблюдаемыми. Если снова обратиться к примеру прогнозирования погоды, то погоду невозможно «измерить» напрямую, в действительности погода оценивается на основе последовательности сенсорных (воспринимаемых) показателей – температуры, давления, скорости ветра и т. п. Поэтому, как и во многих других явлениях, когда состояния не являются непосредственно наблюдаемыми, скрытые марковские модели предоставляют более подходящий и более мощный инструмент моделирования. Кроме того, можно объяснить скрытую марковскую модель как двойной стохастический (случайный) процесс: (i) скрытый стохастический процесс, который мы не можем наблюдать напрямую, (ii) второй стохастический процесс, который создает последовательность наблюдений с учетом первого процесса.

Например, предположим, что имеются две «неправильные» или несимметричные монеты M_1 и M_2. Для монеты M_1 более высока вероятность выпадения *орлов* (heads), а для монеты M_2 более высока вероятность выпадения *решек* (tails). Некто последовательно подбрасывает эти две монеты, но неизвестно, какая именно монета выбрана в каждом конкретном случае. Можно наблюдать только результаты бросков – орлы или решки:

$$O, P, P, O, P, O, O, O, P, O, P, O, P, P, P, O, P, O, O, \ldots$$

Предположим, что игрок, бросающий монеты, сам выбирает первую монету в последовательности (априорные вероятности), а следующую монету для броска выбирает с учетом предыдущего результата (вероятности переходов) с равной вероятностью. Кроме априорных вероятностей и вероятностей переходов для состояний (как и в марковской цепи), в скрытой марковской модели необходимо определить *вероятности наблюдений* (observation probabilities), в рассматриваемом здесь случае это вероятности выпадения орлов и решек с учетом каждой монеты (конкретного состояния). Предположим, что для монеты M_1 вероятность выпадения орлов равна 80 %, а для монеты M_2 вероятность выпадения решек равна 80 %. Далее для этого простого примера определены все требуемые параметры, объединенные в табл. 5.5, 5.6 и 5.7.

Таблица 5.5. Априорные вероятности (*П*)
для примера с несимметричными монетами

$П =$

M_1	M_2
0.5	0.5

Таблица 5.6. Вероятности переходов (*A*)
для примера с несимметричными монетами

$A =$

	M_1	M_2
M_1	0.5	0.5
M_2	0.5	0.5

Таблица 5.7. Вероятности наблюдений (*B*)
для примера с несимметричными монетами

$B =$

	M_1	M_2
M_1	0.8	0.2
M_2	0.2	0.8

Диаграмма состояний для примера с несимметричными монетами изображена на рис. 5.3 с двумя переменными состояния и двумя возможными наблюдениями, которые зависят от состояния.

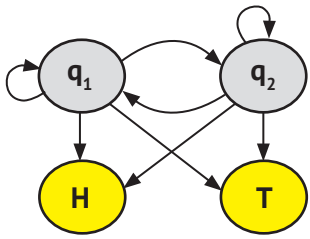


Рис. 5.3. Диаграмма состояния для скрытой марковской модели в примере с несимметричными монетами. Показаны два состояния q_1 и q_2 и два наблюдения *H* (орел) и *T* (решка) с дугами, представляющими переходы и вероятности наблюдений

Формально скрытая марковская модель определяется следующим образом:

- множество состояний: $Q = \{q_1, q_2, \ldots, q_n\}$;
- множество наблюдений: $O = \{o_1, o_2, \ldots, o_m\}$;
- вектор априорных вероятностей: $П = \{\pi_1, \pi_2, \ldots, \pi_n\}$, где $\pi_i = P(S_0 = q_i)$;
- матрица вероятностей переходов: $A = \{a_{ij}\}$, $i = [1..n]$, $j = [1..n]$, где $a_{ij} = P(S_t = q_j \mid S_{t-1} = q_i)$;

- матрица вероятностей наблюдений: $B = \{b_{ij}\}$, $i = [1..n]$, $j = [1..m]$, где $b_{ik} = P(O_t = o_k \mid S_t = q_i)$.

Здесь n – количество состояний, m – количество наблюдений, S_0 – начальное состояние.

Скрытую марковскую модель можно представить в компактной форме: $\lambda = \{A, B, \Pi\}$.

Скрытая марковская модель (первого порядка) обладает следующими свойствами:

- марковское свойство:
  $P(S_t = q_j \mid S_{t-1} = q_i, S_{t-2} = q_k, \ldots) = P(S_t = q_j \mid S_{t-1} = q_i)$;
- стационарный (устойчивый) процесс:
  $P(S_{t-1} = q_j \mid S_{t-2} = q_i) = P(S_t = q_j \mid S_{t-1} = q_i)$ и
  $P(O_{t-1} = o_k \mid S_{t-1} = q_j) = P(O_t = o_k \mid S_t = q_i), \forall(t)$;
- независимость от наблюдений:
  $P(O_t = o_k \mid S_t = q_i, S_{t-1} = q_j, \ldots) = P(O_t = o_k \mid S_t = q_i)$.

Как и для марковской цепи, марковское свойство подразумевает, что вероятность текущего состояния зависит только от предыдущего состояния и не зависит от остальной (предыдущей) хронологии состояний. Второе свойство означает, что вероятности переходов и наблюдений не изменяются со временем, т. е. процесс стационарный (устойчивый). Третье свойство определяет, что наблюдения зависят только от текущего состояния. Существуют расширения базовой скрытой марковской модели, которые смягчают некоторые из этих положений. Некоторые расширения скрытой марковской модели будут рассматриваться в следующем разделе и в следующих главах.

С учетом описанных выше свойств графовая скрытая марковская модель показана на рис. 5.4. Эта модель содержит две последовательности случайных переменных: состояние в момент времени t S_t и наблюдение в момент времени t Q_t.

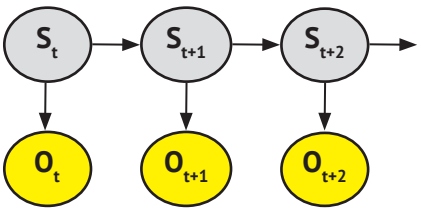


Рис. 5.4. Скрытая марковская модель, представленная в виде графа

С учетом представления конкретной предметной области в форме скрытой марковской модели возникают три основных вопроса (задачи), которые важны в большинстве приложений [7]:

1) *вычисление оценки*: в рассматриваемой модели необходимо определить оценку вероятности последовательности наблюдений;
2) *оптимальная последовательность*: с учетом модели и конкретной последовательности наблюдений необходимо определить оценку наиболее вероятной последовательности состояний, которая создает эти наблюдения;
3) *обучение параметров*: с учетом количества последовательностей наблюдений необходимо отрегулировать (обучить) параметры модели.

Алгоритмы решения этих вопросов (задач) с предположением, что используется стандартная скрытая марковская модель с конечным числом состояний и наблюдений, рассматриваются в следующих разделах.

5.3.1 Вычисление оценки

Вычисление оценки заключается в определении вероятности последовательности наблюдений $O = \{o_1, o_2, o_3, ...\}$ с учетом модели λ, т. е. в оценке вероятности $P(O \mid \lambda)$. Мы рассматриваем два метода. Сначала будет представлен прямой метод, *простейший* (naive) алгоритм, который стал причиной разработки более эффективного метода, который будет описан позже.

5.3.1.1 Прямой метод

Последовательность наблюдений $O = \{o_1, o_2, o_3, ..., o_T\}$ можно сгенерировать по различным последовательностям состояний Q_i, так как состояния неизвестны для скрытых марковских моделей. Таким образом, для вычисления вероятности последовательности наблюдений можно вычислить ее оценку для конкретной последовательности состояний, а затем прибавить вероятности для всех возможных последовательностей состояний:

$$P(O \mid \lambda) = \sum_i P(O, Q_i \mid \lambda). \tag{5.8}$$

Чтобы получить $P(O, Q_i \mid \lambda)$, нужно просто умножить вероятность начального состояния q_1 на вероятности переходов для последовательности состояний $q_1, q_2, ...$ и вероятности наблюдений для последовательности наблюдений $o_1, o_2, ...$:

$$P(O, Q_i \mid \lambda) = \pi_1 b_1(o_1) a_{12} b_2(o_2) ... a_{(T-1)T} b_T(o_T). \tag{5.9}$$

Таким образом, вероятность O с учетом суммирования всех возможных последовательностей состояний Q равна:

$$P(O \mid \lambda) = \sum_{Q} \pi_1 b_1(o_1) a_{12} b_2(o_2) \ldots a_{(T-1)T} b_T(o_T). \quad (5.10)$$

Для модели с N состояниями и длиной (последовательности) наблюдений T существует N^T возможных последовательностей состояний. Каждый член в выражении суммирования требует выполнения $2T$ операций. В результате для вычисления оценки требуется количество операций порядка $2T \times N^T$.

Например, если рассматривается модель с пятью состояниями $N = 5$ и с длиной последовательности наблюдений $T = 100$ – это обычные параметры для приложений скрытых марковских моделей, – то количество требуемых операций будет иметь порядок 10^{72}. Очевидно, что необходим более эффективный метод.

5.3.1.2 Итеративный метод

Основная идея итеративного метода, также известного как алгоритм *Forward*, заключается в определении оценки вероятностей состояний/наблюдений для каждого интервала времени. То есть вычисляется вероятность частичной последовательности наблюдений до времени t (начиная с момента времени $t = 1$), и на основе этого неполного (промежуточного) результата вычисляется вероятность последовательности наблюдений для времени $t + 1$ и т. д.

Сначала необходимо определить вспомогательную переменную, обозначенную как *forward*:

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, S_t = q_i \mid \lambda). \quad (5.11)$$

Это вероятность частичной последовательности наблюдений до момента времени t, находящаяся в состоянии q_i в момент времени t.

Итеративный алгоритм состоит из трех основных частей: инициализация, индукция и завершение. На этапе инициализации определяются переменные α для всех состояний в начальный момент времени. На этапе индукции вычисляются значения $\alpha_{t+1}(i)$ по значениям $\alpha_t(i)$. Эти вычисления повторяются от $t = 2$ до $t = T$. Наконец, на этапе завершения вычисляется значение $P(O \mid \lambda)$ путем суммирования всех значений α_T. Полная процедура показана в алгоритме 5.1.

Алгоритм 5.1. Алгоритм Forward

Требования: Скрытая марковская модель λ; Последовательность наблюдений O; Количество состояний N; Количество наблюдений T

```
for i = 1 to N do
  α₁(i) = P(o₁, S₁ = q_i) = π_i b_i(O₁)    (Инициализация)
end for
for t = 2 to T do
```

```
  for j = 1 to N do
    α_t(j) = [∑_i α_{t-1}(i)a_ij]b_j(O_t)   (Индукция)
  end for
end for
P(O) = ∑_i α_T(i)   (Завершение)
return P(O)
```

Теперь проанализируем сложность по времени этого итеративного метода. На каждой итерации требуется (приблизительно) N операций умножения и N операций сложения, так что для T итераций количество операций будет иметь порядок $N^2 \times T$. Таким образом, сложность по времени от экспоненциальной по T для прямого метода снизилась до линейной по T и квадратичной по N для итеративного метода. Это существенное уменьшение сложности. Следует отметить, что в большинстве приложений $T >> N$.

Если вернуться к примеру с $N = 5$ и $T = 100$, то теперь количество операций приблизительно равно 2500, т. е. эти операции можно выполнить за несколько миллисекунд на стандартном персональном компьютере.

Описанная выше итеративная процедура является основой для решения двух других задач, существующих в скрытых марковских моделях. Эти решения будут рассмотрены в следующих разделах.

5.3.2 Оценка состояния

Поиск наиболее вероятной последовательности состояний для некоторой последовательности наблюдений $O = \{o_1, o_2, o_3,...\}$ можно интерпретировать двумя способами:

- получение наиболее вероятного состояния S_t в каждом интервале времени t;
- получение наиболее вероятной последовательности состояний s_0, s_1, …, s_T.

Следует отметить, что объединение наиболее вероятных состояний для каждого момента (интервала) времени при $t = 1...T$ не обязательно совпадает с наиболее вероятной последовательностью состояний[12]. Сначала решается задача поиска наиболее вероятного или оптимального состояния для конкретного момента времени t, затем задача поиска оптимальной последовательности состояний.

В первую очередь необходимо определить несколько дополнительных вспомогательных переменных. Переменная *backward* аналогична переменной *forward*, но в данном случае мы начинаем с конца последо-

[12] Это частный случай задачи MPE (Most Probable Explanation – наиболее вероятное объяснение), которая будет рассматриваться в главе 7.

вательности, т. е.

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T, S_t = q_i \mid \lambda). \quad (5.12)$$

Это вероятность частичной последовательности наблюдений от $t + 1$ до T, при нахождении в состоянии q_i в момент времени t. Тем же способом, что и для α, можно выполнить итеративные вычисления, но теперь в обратном направлении:

$$\beta_t(i) = \sum_j \beta_{t+1}(j) a_{ij} b_j(o_t). \quad (5.13)$$

Переменные β для момента времени T определяются как $\beta_T(j) = 1$.

Таким образом можно решить и задачу вычисления оценки из предыдущего раздела, используя β вместо α, начиная с конца последовательности наблюдений и выполняя итерации обратно по времени. Или можно объединить обе переменные и выполнять итерации в прямом и обратном направлениях, встретившись в некоторый промежуточный момент времени t, т. е.:

$$P(O, s_t = q_i \mid \lambda) = \alpha_t(i)\beta_t(i). \quad (5.14)$$

Тогда:

$$P(O \mid \lambda) = \sum_i \alpha_t(i)\beta_t(i). \quad (5.15)$$

Теперь определим дополнительную переменную γ – это условная вероятность нахождения в определенном состоянии q_i с учетом последовательности наблюдений:

$$\gamma_t(i) = P(s_t = q_i \mid O, \lambda) = P(s_t = q_i, O \mid \lambda) / P(O). \quad (5.16)$$

Эту формулу можно записать с использованием переменных α и β в следующем виде:

$$\gamma_t(i) = \alpha_t(i)\beta_t(i) / \sum_i \alpha_t(i)\beta_t(i). \quad (5.17)$$

Эта переменная γ дает решение первой подзадачи о нахождении наиболее вероятного состояния (MPS – most probable state) в момент времени t: необходимо просто определить, для какого состояния эта переменная принимает максимальное значение, т. е.

$$MPS(t) = ArgMax_i \gamma_t(i). \quad (5.18)$$

Теперь решим вторую подзадачу – нахождение наиболее вероятной последовательности состояний Q с учетом последовательности наблю-

дений O, такой, что требуется максимизация $P(Q \mid O, \lambda)$. По правилу Байеса: $P(Q \mid O, \lambda) = P(Q, O \mid \lambda) / P(O)$. Учитывая, что $P(O)$ не зависит от Q, это равнозначно максимизации $P(Q, O \mid \lambda)$.

Метод получения оптимальной последовательности состояний известен как алгоритм *Витерби* (Viterbi), который, как и алгоритм *Forward*, решает задачу итеративно. Прежде чем подробно рассматривать сам алгоритм, необходимо определить дополнительную переменную δ. Эта переменная дает максимальное значение вероятности частичной последовательности состояний и наблюдений до момента времени t при нахождении в состоянии q_i в момент времени t, т. е.

$$\delta_t(i) = MAX\left[P(s_1, s_2, \ldots, s_t = q_i, o_1, o_2, \ldots, o_t \mid \lambda)\right]. \tag{5.19}$$

Это значение также можно получить итеративным способом:

$$\delta_{t+1}(i) = \left[MAX\ \delta_t(i)a_{ij}\right]b_j(o_{t+1}). \tag{5.20}$$

Алгоритм Витерби требует выполнения четырех этапов: инициализация, рекурсия, завершение и обратный проход (поиск с возвратом). Для алгоритма Витерби требуется дополнительная переменная $\psi_t(i)$, в которой для каждого состояния i в каждый момент времени t хранится предыдущее состояние, для которого была определена максимальная вероятность. Эта переменная используется для восстановления последовательности при обратном проходе после этапа завершения. Полная процедура показана в алгоритме 5.2.

С помощью алгоритма Витерби можно получить наиболее вероятную последовательность состояний, даже если они невидимы для скрытых марковских моделей.

Алгоритм 5.2. Алгоритм Витерби

Требования: Скрытая марковская модель λ; Последовательность наблюдений O; Количество состояний N; Количество наблюдений T

for i = 1 **to** N **do**
  (Инициализация)
  δ_1(i) = $\pi_i b_i(O_1)$
  ψ_1(i) = 0
end for
for t = 2 **to** T **do**
  for j = 1 **to** N **do**
    (Рекурсия)
    δ_t(j) = $MAX_i[\delta_{t-1}(i)a_{ij}]b_j(O_t)$
    ψ_t(j) = $ARGMAX_i[\psi_{t-1}(i)a_{ij}]$
  end for
end for
(Завершение)

```
P* = MAX_i[δ_T(i)]
q_T* = ARGMAX_i[δ_T(i)]
for t = T to 2 do
  (Обратный проход)
  q*_t-1 = ψ_t(q_t*)
end for
```

5.3.3 Обучение

В этом разделе мы рассмотрим, как можно обучить скрытую марковскую модель по имеющимся данным с использованием алгоритма Баума–Велша (Baum-Welch). Необходимое предварительное замечание: этот метод предполагает, что структура модели известна: предварительно определено количество состояний и наблюдений, следовательно, метод оценивает только параметры. Обычно наблюдения определяются предметной областью приложения, но количество состояний, которые скрыты, не так-то просто определить. Иногда количество скрытых состояний можно определить на основе знаний предметной области, иногда это делается экспериментально, методом проб и ошибок: тестируется производительность модели с различными количествами состояний (2, 3, ...), и выбирается то количество, при котором получены наилучшие результаты. Следует отметить, что при таком выборе неизбежен определенный компромисс, так как увеличение количества состояний дает более точные результаты, но увеличивает сложность вычислений.

Алгоритм Баума–Велша определяет параметры скрытой марковской модели $\lambda = A, B, \Pi$ с учетом количества последовательностей наблюдений $\mathbf{O} = O_1, O_2, \ldots, O_K$. Для этого алгоритм максимизирует вероятность модели с учетом наблюдений: $P(\mathbf{O} \mid \lambda)$. Для скрытой марковской модели с N состояниями и M наблюдениями необходимо оценить $N + N^2 + N \times M$ параметров для Π, A и B соответственно.

Нужно определить еще одну вспомогательную переменную ξ – вероятность перехода из состояния i в момент времени t в состояние j в момент времени $t + 1$ с учетом последовательности наблюдений O:

$$\xi_t(i,j) = P(s_t = q_i, s_{t+1} = q_j \mid O, \lambda) = P(s_t = q_i, s_{t+1} = q_j, O \mid \lambda) / P(O). \qquad (5.21)$$

Эту формулу можно записать с использованием переменных α и β:

$$\xi_t(i,j) = \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) / P(O). \qquad (5.22)$$

Выражение $P(O)$ также можно записать с использованием переменных α и β:

$$\xi_t(i,j) = \alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j) \,/\, \sum_i \sum_j \alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j). \quad (5.23)$$

Формулу для переменной γ еще можно записать с применением переменной ξ: $\gamma_t(i) = \sum_j \xi_t(i,j)$.

Добавляя $\gamma_t(i)$ для всех моментов (интервалов) времени, т. е. $\sum_t \gamma_t(i)$, получаем оценку количества случаев, когда цепь находилась в состоянии i, а при накопительном суммировании $\xi_t(i, j)$ по времени t, т. е. $\sum_t \xi_t(i,j)$, оценивается количество переходов из состояния i в состояние j.

На основании всех приведенных выше определений процедура Баума–Велша для оценки параметров скрытых марковских моделей обобщена в алгоритме 5.3.

Алгоритм 5.3. Алгоритм Баума–Велша

1. Оценка априорных вероятностей – количество случаев нахождения в состоянии i в момент времени t.

$$\pi_i = \gamma_1(i)$$

2. Оценка вероятностей переходов – количество переходов из состояния i в состояние j по отношению к количеству случаев нахождения в состоянии i.

$$a_{ij} = \sum_t \xi_t(i, j) \,/\, \sum_t \gamma_t(j)$$

3. Оценка вероятностей наблюдений – количество случаев нахождения в состоянии j при количестве наблюдений k по отношению к количеству случаев нахождения в состоянии j.

$$b_{jk} = \sum_{t,O=k} \gamma_t(j) \,/\, \sum_t \gamma_t(j)$$

Следует отметить, что вычисление переменных γ и ξ выполняется с использованием переменных α и β, для которых требуются параметры скрытой марковской модели Π, A, B. Здесь возникает проблема «курицы и яйца» – необходимы параметры модели для алгоритма Баума–Велша, который оценивает параметры этой модели. Решение данной проблемы основано на принципе EM (expectation-maximization – EM-алгоритм).

Принцип (алгоритм) EM состоит в том, чтобы начать с некоторых исходных параметров для модели (E-этап), т. е. $\lambda = \{A, B, \Pi\}$, которые можно инициализировать случайным образом или на основе каких-либо знаний предметной области. Затем по алгоритму Баума–Велша эти параметры переоцениваются (M-этап). Цикл повторяется, пока не будет обеспечена сходимость, т. е. до тех пор, пока различия между пара-

метрами модели на последней и предшествующей итерациях не станут меньше определенного порогового значения.

EM-алгоритм предоставляет так называемый механизм оценки *максимального правдоподобия* (maximum-likelihood estimator), который не гарантирует получения оптимального решения – это зависит от начальных условий. Тем не менее механизм оценки максимального правдоподобия на практике обычно работает вполне приемлемо[13].

5.3.4 Расширения

Несколько расширений стандартных скрытых марковских моделей были предложены для устранения конкретных проблем в отдельных приложениях [2]. В этом разделе будут кратко описаны некоторые из этих расширений, графовые модели которых изображены на рис. 5.5.

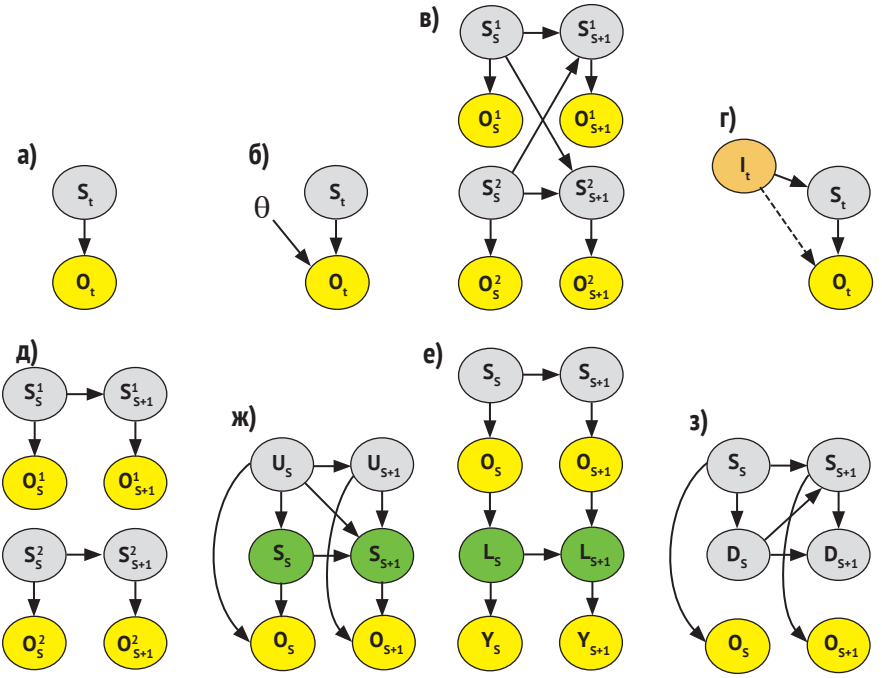


Рис. 5.5. Представление графовых моделей для стандартной скрытой марковской модели (СММ) и нескольких ее расширений: а) стандартная (базовая) модель, б) параметрические СММ, в) связные СММ, г) СММ ввода-вывода, д) параллельные СММ, е) иерархические СММ, ж) динамические байесовские сети со смешанным состоянием, з) скрытые полумарковские модели

Параметрические скрытые марковские модели (ПСММ) представляют предметные области, в которых предполагается использование несколь-

[13] При наличии некоторых знаний предметной области механизм оценки максимального правдоподобия может обеспечить правильную инициализацию параметров, в противном случае можно установить для параметров равномерное распределение вероятностей.

ких вариаций скрытых марковских моделей. В параметрических скрытых марковских моделях переменные наблюдений обусловлены переменной состояния и одним или несколькими параметрами, значимыми для таких переменных (см. рис. 5.5б). Значения параметров известны и постоянны на этапе тренировочного обучения. На этапе тестирования значения, которые максимизируют правдоподобие ПСММ, восстанавливаются (корректируются) с помощью адаптированного EM-алгоритма.

Связные скрытые марковские модели (ССММ) объединяют скрытые марковские модели, вводя условные зависимости между переменными состояния (см. рис. 5.5в). Эти модели вполне подходят для представления взаимовлияний между подпроцессами, которые протекают параллельно.

Скрытые марковские модели ввода-вывода (СММВВ) рассматривают специальный добавочный входной параметр, который воздействует на состояния марковской цепи и дополнительно (но не обязательно) на переменные наблюдения. Эти типы моделей показаны на рис. 5.5г. Входная переменная соответствует наблюдениям. Выходным сигналом СММВВ является класс моделей (например, фонема при распознавании речи или конкретное движение при распознавании жестов), которые должны выполняться. Отдельная СММВВ может описывать полное множество классов.

Параллельные скрытые марковские модели (ПарСММ) требуют меньше СММ, чем связные СММ для сложных составных процессов с предположением о взаимной независимости между СММ (см. рис. 5.5д). Основной принцип заключается в создании независимых скрытых марковских моделей для двух (и более) независимых параллельных процессов (например, для возможных движений каждой руки при распознавании жестов) и объединении их с помощью умножения их отдельных собственных правдоподобий. Параллельные скрытые марковские модели с наиболее вероятным совместным правдоподобием определяют требуемый класс.

Иерархические скрытые марковские модели (ИСММ) создают многослойную структуру скрытых марковских моделей на различных уровнях абстракции (см. рис. 5.5е). В двухуровневых иерархических скрытых марковских моделях нижний уровень – это множество скрытых марковских моделей, которое представляет последовательность подмоделей. Верхний уровень – это марковская цепь, которая управляет динамикой этих подмоделей. Разделение на уровни позволяет повторно использовать стандартные скрытые марковские модели, просто заменяя верхние уровни.

Динамические байесовские сети со смешанным состоянием (ДБССС)[14] объединяют дискретные и непрерывные пространства состояний в

[14] Скрытые марковские модели, включая их расширения, являются частными типами динамических байесовских сетей, более обобщенная модель которых описана в главе 9.

двухуровневую структуру. Динамические байесовские сети со смешанным состоянием состоят из скрытой марковской модели на верхнем уровне и линейной динамической системы (ЛДС) на нижнем уровне. Линейная динамическая система используется для моделирования переходов между действительно-значными (или вещественно-значными) состояниями. Выходные значения скрытой марковской модели управляют этой линейной системой. Графовое представление ДБССС показано на рис. 5.5ж. В ДБССС скрытые марковские модели могут описывать дискретные концепции высокого уровня, такие как грамматика, тогда как линейная динамическая система описывает входные сигналы в пространстве непрерывных состояний.

Скрытые полумарковские модели (СПММ) используют временные знания, относящиеся к текущему процессу, определяя явную продолжительность каждого состояния (см. рис. 5.5з). Скрытые полумарковские модели вполне пригодны для того, чтобы избежать экспоненциального убывания вероятностей наблюдений при моделировании больших последовательностей наблюдений.

5.4 Приложения

В этом разделе рассматриваются приложения марковских цепей и скрытых марковских моделей в двух предметных областях. Сначала описывается использование марковских цепей для упорядочения веб-страниц с помощью алгоритма PageRank. Затем будет представлено применение скрытой марковской модели для распознавания жестов.

5.4.1 Алгоритм PageRank

Всемирную сеть WWW (World Wide Web) можно мысленно представить как весьма большую марковскую цепь, такую, что каждая веб-страница является состоянием, а гиперссылки между веб-страницами соответствуют переходам между этими состояниями. Предположим, что существует N веб-страниц. Любая конкретная веб-страница w_i имеет m исходящих гиперссылок. Если некоторый пользователь находится на веб-странице w_i, то может выбрать здесь любую гиперссылку для перехода на другую веб-страницу. Обоснованное предположение: каждая исходящая гиперссылка может быть выбрана с равной вероятностью. Таким образом, вероятность перехода с веб-страницы w_i на любую веб-страницу, на которую указывает одна из существующих гиперссылок, w_j равна $A_{ij} = 1/m$. Для других веб-страниц, на которые нет гиперссылок с текущей веб-страницы, вероятность перехода равна нулю. При таком подходе в соответствии со структурой WWW можно сформи-

ровать матрицу вероятностей переходов A для соответствующей марковской цепи. Диаграмма состояний для небольшого примера с тремя веб-страницами показана на рис. 5.6.

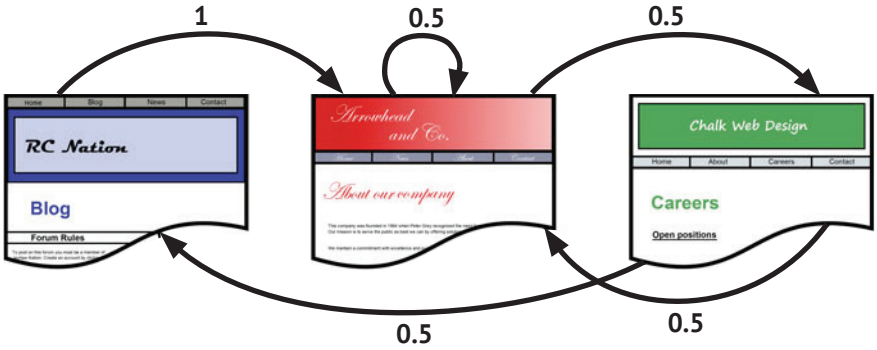


Рис. 5.6. Небольшой пример для WWW с тремя страницами

С учетом матрицы вероятностей переходов для WWW можно достичь сходимости вероятностей для каждого состояния (веб-страницы) в соответствии с теоремой Фробениуса–Перрона (Perron-Frobenius) (см. раздел 5.2). Сходимость вероятности для конкретной веб-страницы можно считать равнозначной вероятности посещения этой веб-страницы некоторым пользователем, произвольно перемещающимся по WWW. Теоретически можно предположить, что веб-страницы с бóльшим количеством входящих гиперссылок с веб-страниц, также содержащих большее количество входящих гиперссылок, будут иметь более высокую вероятность их посещения.

На основе изложенных выше принципов Л. Пэйдж (L. Page) и др. разработали алгоритм *PageRank*, представляющий собой основу для упорядочения (ранжирования) веб-страниц при выполнении поиска с помощью сервиса Google [6]. Веб-страницы, извлекаемые с помощью алгоритма поиска, предъявляются пользователю в соответствии со сходимостью их вероятностей. Здесь основная идея состоит в следующем: более важные веб-страницы, как правило, обладают более высокой сходимостью их вероятности.

5.4.2 Распознавание жестов

Язык жестов весьма важен для общения человека с человеком, поэтому жесты не менее важны для взаимодействия человека с компьютером. Например, жесты можно использовать для подачи команд обслуживающему роботу. Мы сосредоточимся на динамических жестах кистью руки, которые представляют собой движения кисти и руки (в целом) человека. Например, на рис. 5.7 показаны некоторые отдельные кадры с изображением человека, выполняющего жест «стоп».

Рис. 5.7. Видеопоследовательность, изображающая человека, выполняющего жест «стоп» правой рукой. Показаны только некоторые ключевые кадры

Для распознавания жестов весьма эффективным вариантом является скрытая марковская модель [1, 9]. Скрытые марковские модели хорошо подходят для моделирования последовательных процессов, а кроме того, надежны и устойчивы к временны́м вариациям при выполнении динамических жестов. Чтобы появилась возможность применения скрытых марковских моделей для моделирования и распознавания жестов, необходимо предварительно обработать изображения в видеопоследовательности и извлечь из них набор (множество) признаков. Эти признаки сформируют множество наблюдений для скрытой марковской модели.

Обработка изображений состоит в обнаружении фигуры человека в изображении, определении положения его руки (кисти руки) и последующем отслеживании положения кисти руки в последовательности изображений. В рассматриваемой последовательности изображений положение кисти руки (определяемое координатами *XYZ*) извлекается из каждого изображения. Кроме того, могут быть определены и некоторые другие части тела, такие как голова и торс, которые используются для получения признаков общего положения тела, как описано ниже.

Другие варианты описания жестов могут быть разделены на следующие группы: а) признаки движения; б) признаки положения тела (позы); в) признаки положения тела и движения. Признаки движения описывают движение кисти руки человека в декартовой системе координат *XYZ*. Признаки положения тела (позы) представляют положение кисти руки по отношению к другим частям тела, таким как голова и/или торс. Эти признаки движения и положения тела обычно кодируются с помощью конечного числа кодовых слов, которые предоставляют наблюдения для скрытых марковских моделей. Например, если рассматриваются три значения для описания каждой координаты движения и два бинарных признака положения тела (например, кисть руки над головой и т. п.), то в итоге получится $m = 3 \times 3 \times 3 \times 2 \times 2 = 108$ возможных наблюдений. Эти наблюдения определяются для каждого кадра (или для каждых n кадров) в видеопоследовательности, фиксирующей конкретный жест.

Для распознавания N различных жестов необходимо выполнить тренировочное обучение N скрытых марковских моделей, по одной для

каждого жеста. Первый параметр, который необходимо определить, – количество скрытых состояний для каждой модели. Как уже было отмечено ранее, это может быть множество на основе знаний предметной области или множество, полученное посредством кросс-валидации по экспериментально определенной оценке различных количеств состояний. В варианте с динамическими жестами можно интерпретировать состояния как представление различных стадий перемещения кисти руки. Например, жест «стоп» можно определить как состоящий из трех фаз: перемещение всей руки вверх и вперед, раскрытие ладони и движение всей руки вниз, т. е. подразумеваются три скрытых состояния. Экспериментальным путем было определено, что использование трех–пяти состояний обычно дает хорошие результаты.

После определения количества состояний для каждой модели жеста (количество состояний может быть различным для каждой модели) определяются параметры с использованием алгоритма Баума–Велша. Для этого требуется множество M тренировочных последовательностей для каждого класса жестов, и чем больше элементов выборки данных, тем лучше. Таким образом, создается N скрытых марковских моделей, по одной для каждого типа жестов.

Для распознавания признаки извлекаются из видеопоследовательности. Это наблюдения O для N скрытых марковских моделей λ_i, по одной последовательности наблюдений для каждого типа жестов. Вероятность каждой модели с учетом соответствующей последовательности наблюдений $P(O \mid \lambda_i)$ вычисляется с использованием алгоритма *Forward*. Модель с наибольшей вероятностью λ_k^* выбирается как распознанный жест. На рис. 5.8 показан процесс распознавания жеста, рассматривающий пять классов жестов.

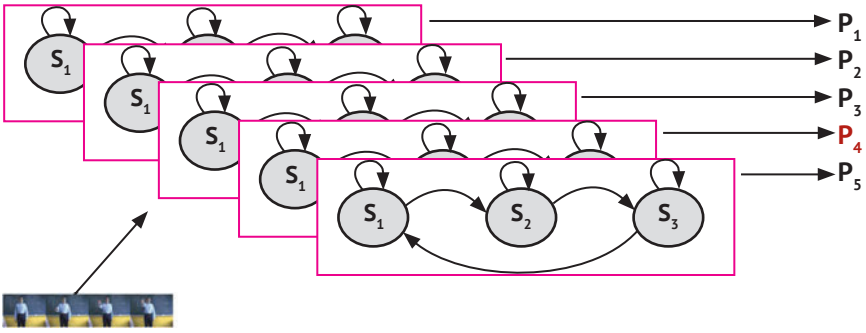


Рис. 5.8. Распознавание жеста с использованием скрытых марковских моделей. Признаки, извлекаемые из видеопоследовательности, передаются как наблюдения в каждую скрытую марковскую модель, по одной последовательности наблюдений для каждого класса жестов, и вычисляется вероятность каждой модели. Модель с наибольшей вероятностью определяется как распознанный жест

5.5 Материалы для дополнительного чтения

Общее введение в марковские цепи представлено в [4]. Рабинер (Rabiner) [7] предлагает превосходное введение в скрытые марковские модели и их приложения для распознавания речи. Более общий обзор технологии распознавания речи можно найти в [8]. Обзор нескольких расширений скрытых марковских моделей в приложении к распознаванию жестов см. в [2]. По ссылке [5] предлагается анализ механизмов поиска и алгоритма PageRank. Практическое применение скрытых марковских моделей для распознавания жестов описано в [1, 9]. Программное обеспечение с открытым исходным кодом для скрытых марковских моделей доступно по ссылке [3].

5.6 Задания и упражнения

1. Для марковской цепи моделирования погоды: а) определить вероятность последовательности состояний: *облачно*, *дождь*, *солнечно*, *солнечно*, *солнечно*, *облачно*, *дождь*, *дождь*; б) определить вероятность последовательности из четырех дождливых дней подряд; в) определить ожидаемое количество дней, в течение которых будет непрерывно идти дождь.
2. Для примера с небольшим количеством веб-страниц на рис. 5.6 определить: а) выполнено ли условие сходимости, и если выполнено, то б) определить порядок, в котором эти три веб-страницы будут представлены пользователю.
3. Рассмотрим пример с несимметричной монетой. С учетом параметров в табл. 5.5, 5.6, 5.7 определить вероятность следующей последовательности наблюдений: *OOPP*, используя а) прямой метод; б) алгоритм *forward*.
4. Для задания 3 определить количество операций для каждого из указанных двух методов.
5. Для задания 3 определить наиболее вероятную последовательность состояний, используя алгоритм Витерби.
6. Назвать три основных предположения в стандартных скрытых марковских моделях. Выразить их в математической форме.
7. Предположим, что имеются две скрытые марковские модели, которые представляют два явления: *ph1* и *ph2*. Для каждой модели существуют два состояния и два наблюдения со следующими параметрами:

Модель 1: Π = [0.5, 0.5], A = [0.5, 0.5 | 0.5, 0.5], B = [0.8, 0.2 | 0.2, 0.8];
Модель 2: Π = [0.5, 0.5], A = [0.5, 0.5 | 0.5, 0.5], B = [0.2, 0.8 | 0.8, 0.2].

С учетом следующей последовательности наблюдений: «o1, o1, o2, o2» – определить наиболее вероятное явление.

8. Необходимо разработать систему распознавания жестов (движений) головой. Имеется система компьютерного зрения, способная распознавать следующие движения головы: (1) вверх, (2) вниз, (3) влево, (4) вправо, (5) стабильное (неподвижное) положение. Система компьютерного зрения выдает соответствующие числовые значения для каждого типа движений (1–5) каждую секунду. Эти числа являются входными данными (наблюдениями) для системы распознавания жестов. Система распознавания жестов должна распознавать четыре класса жестов головой: а) утвердительный жест, б) отрицательный жест, в) поворот вправо, г) поворот влево. 1) Определить модель, соответствующую этой задаче распознавания, в том числе структуру модели и требуемые параметры. 2) Определить, какие алгоритмы подходят для обучения параметров этой модели и для распознавания жестов.
9. *** Разработать программу для решения задания 8.
10. *** Для скрытых марковских моделей остается нерешенной задача установления оптимального количества состояний для каждой модели. Разработать стратегию поиска для определения оптимального количества состояний для каждой модели, используемой в системе распознавания жестов головой, при котором максимизируется коэффициент (правильных) распознаваний. Следует учесть, что набор данных (экземпляров каждого класса жестов) разделяется на три подмножества: а) тренировочный набор – для оценки параметров модели; б) проверочный (валидационный) набор – для сравнения различных моделей; в) тестовый набор – для тестирования конечных моделей.

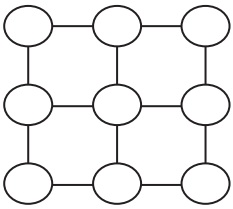
Ссылки на источники

1. Aviles, H., Sucar, L. E., Mendoza C. E. Visual recognition of similar gestures. In: 18th International Conference on Pattern Recognition, p. 1100–1103 (2006).
2. Aviles, H., Sucar, L. E., Mendoza, C. E., Pineda, L. A. A Comparison of dynamic naive Bayesian classifiers and hidden Markov models for gesture recognition. J. Appl. Res. Technol. 9 (1), 81–102 (2011).

3. Kanungo, T. UMDHMM: Hidden Markov Model Toolkit. In: Kornai, A. (ed.) Extended Finite State Models of Language. Cambridge University Press (1999). http://www.kanungo.com/software/software.html.
4. Kemeny, J. K., Snell, L. Finite Markov Chains. Van Nostrand, Princeton (1965).
5. Langville, N., Carl, D., Meyer, C. D. Google's PageRank and Beyond: The Science of Search Engine Rankings. Princeton University Press, Princeton (2012).
6. Page, L., Brin, S., Motwani, R.,Winograd, T. The PageRank Citation Ranking: Bringing Order to the Web. Stanford Digital Libraries Working Paper (1998).
7. Rabiner, L. E.: A tutorial on hidden Markov models and selected applications in speech recognition. In: Waibel, A., Lee, K. (eds.) Readings in Speech Recognition, p. 267–296. Morgan Kaufmann, San Francisco (1990).
8. Rabiner, L., Juang, B. H. Fundamentals on Speech Recognition. Prentice-Hall Signal Processing Series, New Jersey (1993).
9. Wilson, A., Bobick, A. Using hidden Markov models to model and recognize gesture under variation. Int. J. Pattern Recognit. Artif. Intell., Spec. Issue Hidden Markov Models Comput. Vis. 15 (1), 123–160 (2000).

Глава 6

Марковские случайные поля



6.1 Введение

Некоторые процессы, такие как ферромагнитное вещество в магнитном поле или изображение, можно смоделировать как последовательность состояний в цепи или в регулярной сетке (решетке). Каждое состояние может принимать различные значения и с той или иной вероятностью подвергаться воздействию состояний своих соседей. Такие модели известны как *марковские случайные поля* (Markov random fields – MRF).

Марковские случайные поля были введены при моделировании ферромагнитных веществ по методике, известной как *модель Изинга* (Ising model) [2]. В модели Изинга существуют последовательности случайных переменных по линии решетки. Каждая случайная переменная представляет диполь, который может находиться в одном из двух возможных состояний: *up*(+) или *down*(−). Состояние каждого диполя зависит от внешнего поля и от состояния соседних диполей в линии решетки. Простой пример с четырьмя переменными показан на рис. 6.1. *Конфигурация* марковского случайного поля – это присваивание конкретных значений каждой переменной в модели. В примере модели на рис. 6.1 существует 16 возможных конфигураций: + + + +, + + + −, + + − +, ..., − − − −.

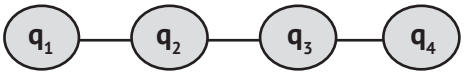


Рис. 6.1. Пример модели Изинга (марковского случайного поля) с четырьмя переменными

Марковское случайное поле представлено в форме неориентированной графовой модели, как в приведенном выше примере. Важное

свойство марковского случайного поля – состояние любой переменной независимо от всех прочих переменных в модели с учетом соседей этой переменной в графе. В примере на рис. 6.1 переменная q_1 не зависит от q_3 и q_4 с учетом q_2. Таким образом, $P(q_1 \mid q_2, q_3, q_4) = P(q_1 \mid q_2)$.

Главная задача в марковском случайном поле – поиск конфигурации с максимальной вероятностью. Обычно вероятность конфигурации зависит от совокупности *внешних* воздействий (например, от воздействия магнитного поля в модели Изинга) и *внутренних* воздействий со стороны соседних узлов. В более общем смысле апостериорная вероятность конфигурации зависит от предварительных знаний или от контекста, а также от данных или от правдоподобия.

Если воспользоваться физической аналогией, то марковское случайное поле можно мысленно представить как последовательность колец на стержнях, при этом каждое кольцо представляет случайную переменную, а высота кольца на стержне соответствует его состоянию. Кольца размещены вдоль прямой линии, как показано на рис. 6.2. Каждое кольцо соединено со своими соседями пружиной, представляющей внутренние воздействия, а также соединено с основанием своего стержня другой пружиной, представляющей внешнее воздействие. Отношение между постоянными коэффициентами (упругости) пружин определяет относительный весовой коэффициент между внутренними и внешними воздействиями. Если кольца оставить в свободном состоянии, то их положение стабилизируется в конфигурации с минимальной энергией, что соответствует конфигурации с максимальной вероятностью в марковском случайном поле.

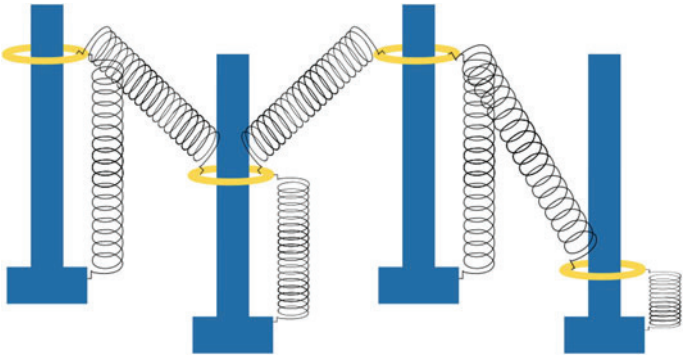


Рис. 6.2. Физическая аналогия марковского случайного поля. Кольца стремятся к конфигурации с минимальной энергией в соответствии с воздействием на них пружин, соединяющих кольца с основаниями стержней (внешнее воздействие) и с соседними кольцами (внутреннее воздействие)

Марковские случайные поля, также известные как *марковские сети*, формально определяются в следующем разделе.

6.2 Марковские сети

Случайное поле (random field) – это набор S случайных переменных $\mathbf{F} = F_1, \ldots, F_S$, проиндексированных по позициям узлов. Случайные переменные могут быть дискретными или непрерывными. В дискретном случайном поле случайная переменная может принимать значение f_i из множества m возможных значений или меток $L = \{l_1, l_2, .., l_m\}$. В непрерывном случайном поле случайная переменная может принимать значения из множества действительных чисел R или из некоторого интервала множества действительных чисел.

Марковское случайное поле, или марковская сеть (Markov network), – это случайное поле, обладающее свойством локальности, или просто локальным свойством: переменная F_i независима от всех прочих переменных в этом поле с учетом соседей этой переменной $Nei(F_i)$. Таким образом:

$$P(F_i \mid \mathbf{F}_\mathbf{C}) = P(F_i \mid Nei(F_i)), \tag{6.1}$$

где $\mathbf{F}_\mathbf{C}$ – множество всех случайных переменных в данном поле, за исключением переменной F_i.

В графическом виде марковская сеть представлена в виде неориентированной графовой модели, состоящей из множества случайных переменных **V** и множества неориентированных ребер **E**. Эти множества формируют неориентированный граф, представляющий отношения независимости между случайными переменными в соответствии со следующим критерием. Подмножество переменных **A** независимо от подмножества переменных **C** с учетом подмножества **B**, если переменные подмножества **B** *разделяют* **A** и **C** в рассматриваемом графе. То есть если узлы подмножества **B** удаляются из графа, то между **A** и **C** не существует путей.

На рис. 6.3 показан пример марковской сети с пятью переменными $q_1, \ldots, q_5$. В этом примере подмножество q_1, q_4 (**A**) независимо от подмножества q_3 (**C**) с учетом подмножества q_2, q_5 (**B**).

Совместная вероятность марковской сети может быть выражена как произведение локальных функций в подмножествах переменных в рассматриваемой модели. Эти подмножества должны включать как минимум все клики (cliques) в сети. Для марковской сети на рис. 6.3 совместное распределение вероятностей можно записать в следующем виде:

$$P(q_1, q_2, q_3, q_4, q_5) = (1/k)P(q_1, q_4, q_5)P(q_1, q_2, q_5)P(q_2, q_3, q_5), \tag{6.2}$$

где k – постоянный коэффициент нормализации. По соображениям практического удобства другие подмножества переменных также могут рассматриваться при вычислении совместного распределения вероятностей. Если включить в вычисление еще и подмножества с размером

два, то формулу для вычисления совместного распределения из предыдущего примера можно записать так:

$$P(q_1, q_2, q_3, q_4, q_5) = (1/k)P(q_1, q_4, q_5)P(q_1, q_2, q_5)P(q_2, q_3, q_5)$$
$$P(q_1, q_2)P(q_1, q_4)P(q_1, q_5)P(q_2, q_3)P(q_2, q_5)P(q_3, q_5)P(q_4, q_5). \quad (6.3)$$

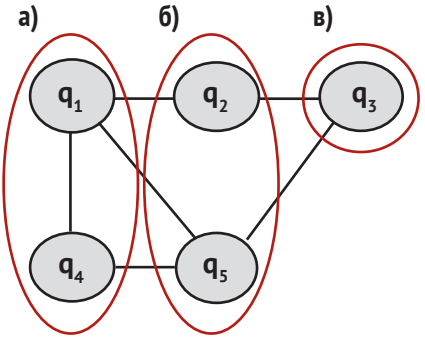


Рис. 6.3. Пример марковской сети, в которой узлы q_1, q_4 (a) независимы от узла q_3 (c) с учетом узлов q_2, q_5 (b)

Формально марковская сеть представляет собой множество случайных переменных $\mathbf{X} = X_1, X_2, \ldots, X_n$, проиндексированное по V так, что $G = (V, E)$ является неориентированным графом, который обладает марковским свойством: переменная X_i не зависит от всех прочих переменных с учетом своих соседей $Nei(X_i)$:

$$P(X_i \mid X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n) = P(X_i \mid Nei(X_i)). \quad (6.4)$$

Соседями переменной являются все переменные, которые напрямую связаны с ней в рассматриваемом графе. При определенных условиях (если распределение вероятностей является строго положительным) совместное распределение вероятностей марковского случайного поля может быть факторизовано по кликам графа:

$$P(\mathbf{X}) = (1/k) \prod_{C \in Cliques(G)} \phi_C(X_C), \quad (6.5)$$

где k – постоянный коэффициент нормализации, а ϕ_C – локальная функция от переменных в соответствующей клике C.

Марковское случайное поле может быть *регулярным* (regular) или *нерегулярным* (irregular). Если случайные переменные размещены в решетке, то марковское случайное поле считается регулярным, например случайные переменные могут представлять пикселы в изображении. Если случайные переменные не размещаются в решетке, то марковское случайное поле нерегулярно. В дальнейшем мы будем рассматривать регулярные марковские случайные поля.

6.2.1 Регулярные марковские случайные поля

Система соседства для регулярного марковского случайного поля **F** определяется следующим образом:

$$\mathbf{V} = \{Nei(F_i) \mid \forall i \in \mathbf{F}_i\}. \tag{6.6}$$

V обладает следующими свойствами:

1) позиция (местоположение) в поле не является соседом для самой себя;
2) соседские отношения симметричны, т. е. если $F_j \in Nei(F_i)$, то $F_i \in Nei(F_j)$.

Обычно марковское случайное поле организовано в форме регулярной решетки. Пример двумерной решетки показан на рис. 6.4. Для регулярной решетки соседские отношения порядка i определяются так:

$$Nei_i(F_i) = \{F_j \in \mathbf{F} \mid dist(F_i, F_j) \le r\}, \tag{6.7}$$

где $dist(x, y)$ – евклидово расстояние между x и y, а за единицу измерения принимается расстояние по вертикали и по горизонтали между позициями (узлами) решетки. Радиус r определяется для каждого порядка. Например, $r = 1$ для порядка один, когда каждый внутренний узел имеет 4 соседей, $r = \sqrt{2}$ для порядка два, когда каждый внутренний узел имеет 8 соседей, $r = 2$ для порядка три, когда каждый внутренний узел имеет 12 соседей, и т. д. На рис. 6.4 показан пример соседства первого порядка, а на рис. 6.5 – пример соседства второго порядка.

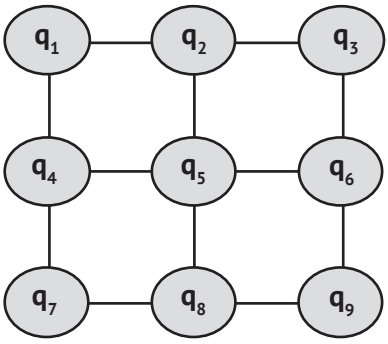


Рис. 6.4. Регулярное двумерное марковское случайное поле с соседством первого порядка

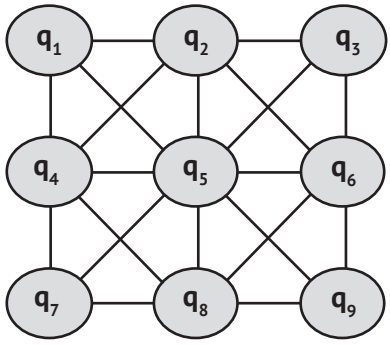


Рис. 6.5. Регулярное двумерное марковское случайное поле с соседством второго порядка

После установления структуры марковского случайного поля на основе порядка соседства обязательно должны быть определены его параметры. Параметры регулярного марковского случайного поля

определяются множеством локальных функций. Эти функции соответствуют совместным распределениям вероятностей подмножеств полностью связанных переменных в графе. Достаточно включить все клики в рассматриваемом графе, но могут быть включены также и другие полностью связанные подмножества. Например, в случае марковского случайного поля первого порядка существуют подмножества из 2 переменных, в марковском случайном поле второго порядка существуют подмножества из 2, 3 и 4 переменных. Вообще говоря, совместное распределение вероятностей для всего поля в целом может быть выражено как произведение локальных функций для различных подмножеств переменных:

$$P(\mathbf{F}) = (1/k) \prod_i f(X_i), \tag{6.8}$$

где $f(X_i)$ – локальные функции для подмножеств переменных X_i, k – постоянный коэффициент нормализации. Можно интерпретировать эти локальные функции как ограничения, с помощью которых будет отдаваться предпочтение конкретным конфигурациям. Например, в модели Изинга, если рассматриваются две соседние переменные X, Y, соответствующая локальная функция отдаст предпочтение (по более высоким вероятностям) конфигурациям, в которых $X = Y$, и будет отвергать (по более низким вероятностям) конфигурации, в которых $X \neq Y$. Эти локальные функции можно определить субъективно в зависимости от предметной области конкретного приложения или обучить локальные функции на имеющихся данных.

6.3 Случайные поля Гиббса

Совместная вероятность марковского случайного поля может быть выражена более удобным способом с учетом его равнозначности случайному полю Гиббса (Gibbs random field) в соответствии с теоремой Хаммерсли–Клиффорда (Hammersley-Clifford theorem) [4]. С учетом этой равнозначности можно записать формулу 6.8 следующим образом:

$$P(\mathbf{F}) = (1/z)\exp(-\mathbf{U}), \tag{6.9}$$

где **U** обозначает энергию (energy) по аналогии с физической энергией. Таким образом, максимизация $P(\mathbf{F})$ равнозначна минимизации **U**. Функция энергии также может быть записана в терминах локальных функций, но поскольку функция энергии – экспоненциальная функция, ее можно представить как сумму этих локальных функций (вместо произведения):

$$\mathbf{U}_\mathbf{F} = \sum_i U_i(X_i). \tag{6.10}$$

При рассмотрении регулярного марковского случайного поля порядка n функцию энергии можно выразить через функции подмножеств полностью связанных переменных различных размеров 1, 2, 3, …:

$$\mathbf{U}_{\mathbf{F}} = \sum_{i} U_1(F_i) + \sum_{i,j} U_2(F_i, F_j) + \sum_{i,j,k} U_3(F_i, F_j, f_k) + \ldots, \tag{6.11}$$

где U_i – локальные функции энергии, называемые потенциалами (potentials), для подмножеств с размером i. Следует отметить, что потенциалы обратны вероятностям, так что низкие потенциалы равнозначны высоким вероятностям.

С учетом равнозначности случайному полю Гиббса задача поиска конфигурации с максимальной вероятностью для марковского случайного поля преобразуется в задачу поиска конфигурации с минимальной энергией.

Подводя итог, следует отметить, что для создания марковского случайного поля необходимо определить:

- множество случайных переменных **F** и их возможные значения L;
- структуру зависимостей или для варианта регулярного марковского случайного поля – схему соседства;
- потенциалы для каждого подмножества полностью связанных узлов (как минимум для клик).

6.4 Логический вывод

Как уже было отмечено выше, более общее практическое применение марковских случайных полей состоит в поиске наиболее вероятной конфигурации, т. е. значения для каждой переменной, которое максимизирует совместную вероятность. С учетом равнозначности случайному полю Гиббса это то же самое, что минимизация функции энергии, выраженной как сумма локальных функций.

Множество всех возможных конфигураций марковского случайного поля обычно весьма велико, поскольку увеличивается экспоненциально при возрастании количества переменных в множестве **F**. Для дискретных МСП с m возможными метками количество возможных конфигураций равно m^N, где N – количество переменных в поле. Если рассматривается марковское случайное поле, представляющее бинарное изображение размером 100×100 пикселов (небольшое изображение), то количество конфигураций равно $2^{10\,000}$. Таким образом, практически невозможно вычислить энергию (потенциал) для каждой конфигурации, за исключением вариантов с очень маленькими полями.

Поиск наиболее вероятной конфигурации обычно формулируется как задача стохастического поиска. Сначала выполняется инициализация

со случайным присваиванием значений каждой переменной в марковском случайном поле, затем эта конфигурация улучшается с помощью локальных операций до тех пор, пока не будет получена конфигурация с минимальной энергией. В общем случае искомый минимум является локальным минимумом в функции энергии. Обеспечить глобальный минимум трудно.

Общая процедура стохастического поиска конфигурации с минимальной энергией показана в алгоритме 6.1. После инициализации всех переменных случайными значениями значение каждой переменной изменяется, и оценивается новая энергия. Если новая энергия меньше, чем предыдущая, то значение изменяется, в противном случае значение также может измениться с определенной вероятностью – это делается для того, чтобы избежать локальных минимумов. Процесс повторяется с определенным количеством итераций (или до достижения сходимости).

Алгоритм 6.1. Алгоритм стохастического поиска

Требования: марковское случайное поле **F**, функция энергии U_F, количество итераций N, количество переменных S, пороговое значение вероятности T, пороговое значение (предел) сходимости ε

```
for i = 1 to S do
  F(i) = l_k (Инициализация)
end for
for i = 1 to N do
  for j = 1 to S do
    t = l_k+1 (Альтернативное значение для переменной F(i))
    if U(t) < U(F(i)) then
      F(i) = t (Изменение значения F(i), если энергия больше)
    else
      if random(U(t) - U(F(i))) < T then
        F(i) = t (Изменение значения F(i) с определенной вероятностью,
                  если энергия больше)
      end if
    end if
  end for
end for
return F* (Возвращается окончательная конфигурация)
```

Существует несколько вариантов этого обобщенного алгоритма в зависимости от вариаций различных аспектов. В одном из вариантов определяется *оптимальная* конфигурация, для которой существуют две основные альтернативы: *оценка апостериорного максимума* (MAP – Maximum a posteriori probability) и *границы апостериорного максимума*

(MPM – Maximum posterior marginals). При варианте с оценкой апостериорного максимума (MAP) оптимальная конфигурация принимается как конфигурация в конце итерационного процесса. При варианте с границами апостериорного максимума (MPM) наиболее частое значение для каждой переменной во всех итерациях принимается как оптимальная конфигурация.

С учетом процесса оптимизации существуют три основные вариации:

1) ICM – iterated conditional modes – итерационные условные режимы: всегда выбирается конфигурация с минимальной энергией;
2) Metropolis: при фиксированной вероятности P выбирается конфигурация с более высокой энергией;
3) SA – simulated annealing – алгоритм имитации отжига: при изменяемой вероятности $P(T)$ выбирается конфигурация с более высокой энергией, здесь T – параметр, известный как *температура*. Вероятность выбора значения с более высокой энергией определяется на основе следующей формулы: $P(T) = e^{-\delta U/T}$, где δU – разность энергий. Алгоритм начинает работу с *большого* значения для T и постепенно уменьшает это значение на каждой итерации. При этом вероятность перехода к состояниям с более высокой энергией изначально велика, и эта вероятность постепенно уменьшается, стремясь к нулю в конце процесса.

6.5 Оценка параметров

Определение марковского случайного поля включает несколько характеристик:

- структура модели – для регулярного марковского случайного поля – система (схема) соседства;
- форма функций локального распределения вероятностей – для каждого полного множества в графе;
- параметры локальных функций.

В некоторых приложениях перечисленные выше характеристики могут быть определены субъективно, но это не всегда просто. Различные варианты выбора для структуры, распределения и параметров могут оказывать существенное воздействие на результаты применения модели в конкретных задачах. Таким образом, рекомендуется обучить модель на данных, которые могут иметь несколько уровней сложности. В самом простом, но нетривиальном случае известна структура и функциональная форма, поэтому необходимо только оценить параметры с

учетом чистой реализации (без шума и помех) марковского случайного поля f. Эта задача усложняется, если данные содержат шум (помехи), и усложняется в еще большей степени, если необходимо обучить функциональную форму распределения вероятностей и определить порядок системы соседства. Далее будет рассматриваться простейший случай – обучение параметров марковского случайного поля на имеющихся данных.

6.5.1 Оценка параметров с помощью данных с метками

Множество параметров θ марковского случайного поля F оценивается с помощью данных f, которые предположительно не содержат шума (помех). С учетом f механизм оценки максимального правдоподобия (maximum likelihood estimator) максимизирует вероятность этих данных при заданных параметрах $P(f \mid \theta)$, следовательно, оптимальные параметры определяются так:

$$\theta^* = ArgMax_{\theta} P(f \mid \theta). \tag{6.12}$$

Если априорная вероятность параметров $P(\theta)$ известна, то можно применить байесовский метод и максимизировать апостериорную плотность вероятности, полученную механизмом оценки марковского случайного поля:

$$\theta^* = ArgMax_{\theta} P(\theta \mid f), \tag{6.13}$$

где

$$P(\theta \mid f) \sim P(\theta) P(f \mid \theta). \tag{6.14}$$

Главное затруднение при оценке максимального правдоподобия для марковского случайного поля заключается в том, что требуется оценка нормализующей статистической суммы Z в распределении Гиббса, поскольку подразумевается суммирование по всем возможным конфигурациям. Напомним, что функция правдоподобия обусловлена следующим выражением:

$$P(f \mid \theta) = (1/Z)\exp(-U(f \mid \theta)), \tag{6.15}$$

где статистическая сумма равна

$$Z = \sum_{f \in F} \exp(-U(f \mid \theta)). \tag{6.16}$$

Таким образом, вычисление Z весьма затруднительно даже для марковского случайного поля среднего размера. Поэтому для эффективного решения этой задачи используются аппроксимации.

Одна из возможных аппроксимаций основана на условных вероятностях каждой переменной в поле f_i с учетом соседей N_i: $P(f_i \mid f_{N_i})$, и, предполагая их взаимную независимость, мы получаем то, что известно как псевдоправдоподобие (pseudo-likelihood) [1]. Тогда функцию энергии можно записать так:

$$U(f) = \sum_i U_i(f_i, f_{N_i}). \tag{6.17}$$

Если предположить, что рассматривается регулярное марковское случайное поле первого порядка и только отдельные узлы и пары узлов, то

$$U_i(f_i, N_i) = V_1(f_i) + \sum_j V_2(f_i, \text{fj}), \tag{6.18}$$

где V_1 и V_2 – потенциалы отдельных узлов и пар узлов соответственно, а f_j – соседи f_i.

Псевдоправдоподобие (PL – pseudo-likelihood) определяется как простое произведение условных правдоподобий:

$$PL(f) = \prod_i P(f_i \mid f_{N_i}) = \prod_i \frac{\exp(-U_i(f_i, f_{Ni}))}{\sum_{f_i} \exp(-U_i(f_i, f_{Ni}))}. \tag{6.19}$$

С учетом независимости f_i и f_{N_i} вычисленное по формуле 6.19 псевдоправдоподобие *PL* не является *истинным* правдоподобием, тем не менее доказано, что на большой ограниченной решетке *PL* сходится к истинному правдоподобию с вероятностью, равной единице [3].

Используя аппроксимацию псевдоправдоподобия и учитывая конкретную структуру и форму локальных функций, можно оценить параметры модели марковского случайного поля на основе имеющихся данных.

Если предположить дискретность марковского случайного поля и взять несколько реализаций (экземпляров), то параметры можно оценить с использованием методики гистограмм. Предположим, что существует N различных множеств экземпляров с размером k в наборе данных, а также что конкретная конфигурация $(f_i \mid f_{N_i})$ встречается H раз, тогда оценка вероятности этой конфигурации равна $P(f_i \mid f_{N_i}) = H/N$.

6.6 Условные случайные поля

Недостатком (ограничением) марковских случайных полей (и скрытых марковских моделей) является тот факт, что обычно предполагается, что наблюдения независимы с учетом каждой переменной состояния. Например, в скрытой марковской модели наблюдение O_t условно независимо от всех прочих наблюдений и рассматриваемых состояний S_t.

В обычных марковских случайных полях также предполагается, что каждое наблюдение зависит только от одной переменной и условно независимо от других переменных в рассматриваемом поле (см. раздел 6.7). Существуют приложения, в которых такие предположения о независимости неприемлемы, например присваивание меток словам в предложении на естественном языке, поскольку здесь возможны зависимости с широким диапазоном воздействия между наблюдениями (словами).

Скрытые марковские модели и обычные марковские случайные поля являются *порождающими* (generative) моделями, которые представляют совместное распределение вероятностей как произведение локальных функций на основе предположений о независимости. Если эти предположения об условной независимости исключены, то модель становится практически неуправляемой. Одним из альтернативных вариантов, для которого не требуются предположения об условной независимости, являются *условные случайные поля* (conditional random fields) [10, 11].

Условное случайное поле – это неориентированная графовая модель, глобально обусловленная **X** – случайной переменной, представляющей наблюдения [8]. Условные модели используются для присваивания меток последовательности наблюдений **X** с выбором той последовательности меток **Y**, которая максимизирует условную вероятность $P(\mathbf{Y} \mid \mathbf{X})$. Условная сущность таких моделей означает отсутствие трудозатрат на моделирование наблюдений и полное освобождение от необходимости формулировать предположения о независимости. В простейшей структуре условного случайного поля узлы, соответствующие элементам последовательности Y, формируют простую цепь первого порядка, как показано на рис. 6.6.

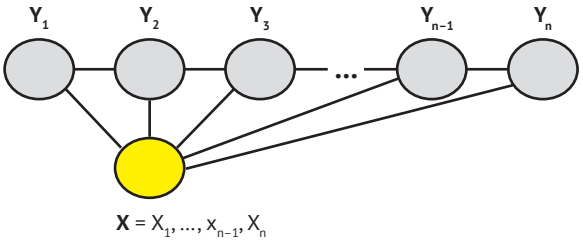


Рис. 6.6. Графовое представление структуры в форме цепи для условного случайного поля

Как и для марковских случайных полей, совместное распределение вероятностей факторизуется по множеству функций потенциалов, где каждая функция потенциала определяется через множество случайных переменных, соответствующих вершинам, образующим максимальную клику графа G, т. е. структуры условного случайного поля. Для условного случайного поля в форме цепи каждая функция потенциала определяется по парам соседних переменных меток Y_i и Y_{i+1}.

Функции потенциалов можно определить с помощью *функций признаков* (feature functions) [8], основанных на признаках с действительными значениями, которые выражают некоторые характеристики эмпирического распределения тренировочных обучающих данных. Например, признак может представлять присутствие (1) или отсутствие (0) слова в текстовой последовательности либо наличие конкретного элемента (грани, текстуры) в изображении. Для модели первого порядка (такой как цепь) функции потенциалов определяются через функции признаков полной последовательности наблюдений $\mathbf{X}$ и пары соседних (смежных) меток Y_{i-1}, Y_i: $U(Y_{i-1}, Y_i, \mathbf{X}, f)$.

При рассмотрении цепи первого порядка функцию энергии можно определить через функцию потенциала отдельной переменной и пару соседних переменных, как и для марковских случайных полей:

$$E = \sum_j \lambda_j t_j(Y_{i-1}, Y_i, \mathbf{X}, f) + \sum_k \mu_k(Y_i, \mathbf{X}, f), \tag{6.20}$$

где λ_j и μ_k – параметры, определяющие весовой коэффициент вклада пар переменных (внутреннее воздействие) и отдельных переменных (внешнее воздействие) соответственно. Эти параметры можно оценить по тренировочным данным. Главное отличие от марковских случайных полей – в данном случае потенциалы обусловлены полной последовательностью наблюдений. Определение оценки параметров и логический вывод выполняются так же, как и для марковских случайных полей.

6.7 Приложения

Марковские случайные поля применялись в некоторых задачах обработки изображений и компьютерного зрения. Например, марковские случайные поля используются для сглаживания изображений, восстановления изображений, сегментации, регистрации изображений, синтеза текстур, повышения разрешения, согласования изображений стереопары, аннотации изображений и извлечения информации из изображений. В следующих разделах рассматриваются два приложения: сглаживание изображений и расширенная аннотация изображений.

6.7.1 Сглаживание изображений

Цифровые изображения обычно искажены высокочастотным шумом. Для снижения уровня шума к изображению можно применить *процесс сглаживания* (image smoothing). Существует несколько различных вариантов решения этой задачи, один из них – использование марковских случайных полей.

Можно определить марковское случайное поле, связанное с цифровым изображением, в котором каждый пиксел соответствует случайной переменной. При использовании марковского случайного поля первого порядка каждая внутренняя переменная соединена с четырьмя своими соседями. Кроме того, каждая переменная соединена с переменной наблюдения, содержащей значение, соответствующее пикселу в изображении (см. рис. 6.7).

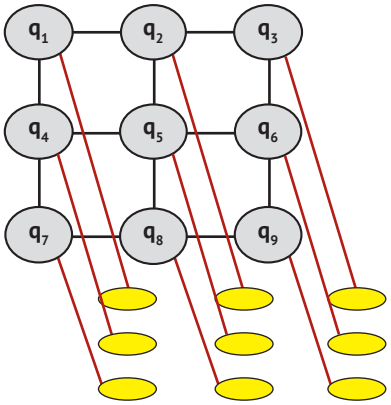


Рис. 6.7. Пример марковского случайного поля, связанного с изображением. В верхней части изображено двумерное МСП первого порядка для изображения 3×3. В нижней части представлены пикселы изображения, каждый из которых соединен с соответствующей переменной в поле

После формирования структуры марковского случайного поля необходимо определить локальные функции потенциалов. Свойством естественных изображений в общем случае является определенная непрерывность, т. е. соседние пикселы, как правило, имеют похожие значения. Таким образом, можно предположить ограничение, принуждающее соседние пикселы принимать схожие значения, и штрафовать (более высоким уровнем энергии) конфигурации, в которых соседи имеют отличающиеся значения. В то же время для каждой переменной в марковском случайном поле желательно иметь значение, близкое к значению в исходном изображении. При этом также штрафуются конфигурации, в которых переменные принимают значения, существенно отличающиеся от соответствующих наблюдений. Поэтому решение должно быть определенным компромиссом между этими двумя типами ограничений: схожестью с соседями и схожестью с наблюдениями.

В рассматриваемом здесь варианте функция энергии может быть выражена как сумма двух типов потенциалов: один связан с парами соседей $U_c(f_i, f_j)$, другой – для каждой отдельной переменной и соответствующего ей наблюдения $U_o(f_i, g_i)$. Таким образом, энергия вычисляется как сумма этих двух типов потенциалов:

$$\mathbf{U}_{\mathbf{F}} = \sum_{i,j} U_c(F_i, F_j) + \lambda \sum_i U_o(F_i, G_i), \tag{6.21}$$

где λ – параметр, управляющий степенью важности слагаемых: наблюдений ($\lambda > 1$) или соседей ($\lambda < 1$), G_i – переменная наблюдений, связанных с F_i.

В зависимости от требуемого поведения для каждого типа потенциала можно определить варианты штрафования отличий от соседей или от наблюдений. Разумным вариантом такой функции является квадрат разности, т. е. потенциал соседей равен

$$U_c(f_i, f_j) = (f_i - f_j)^2. \tag{6.22}$$

Потенциал наблюдений равен

$$U_o(f_i, g_i) = (f_i - g_i)^2. \tag{6.23}$$

При использовании этих потенциалов и с применением алгоритма стохастической оптимизации в результате получается сглаженное изображение как окончательная конфигурация **F**. На рис. 6.8 показано применение сглаживающего марковского случайного поля к цифровому изображению с изменением значения параметра λ.

Рис. 6.8. Пример сглаживания изображения с помощью марковского случайного поля. *Слева*: исходное изображение. *В центре*: изображение, обработанное с параметром $\lambda = 1$. *Справа*: изображение, обработанное с параметром $\lambda = 0.5$. Можно видеть, что при меньшем значении λ в результате получается более сглаженное изображение

6.7.2 Расширенная аннотация изображений

Автоматическая аннотация изображений – это задача автоматического присваивания аннотаций или меток изображениям либо фрагментам изображений на основе их локальных признаков. Аннотация изображений часто выполняется автоматическими системами. Это сложная задача из-за трудности извлечения правильных признаков, которые позволяют обобщить и выделить требуемый объект среди дру-

гих объектов, обладающих схожими визуальными свойствами. Ошибочная пометка фрагментов изображений в большинстве случаев является следствием недостаточно точных и правильных характеристик для классов, определяемых по признакам низкого уровня.

При назначении меток для сегментированного изображения можно включать дополнительную информацию для расширения (и улучшения) аннотации каждого фрагмента этого изображения. Метки для каждого фрагмента изображения обычно не являются независимыми, например в изображении животных в джунглях вполне предсказуемо обнаружение фрагмента неба над животным, деревьев и растений около животного и под ним. Таким образом, пространственные связи между различными фрагментами изображения могут помочь расширить и улучшить аннотацию [5].

Марковские случайные поля можно использовать для представления информации о пространственных связях между фрагментами изображения таким способом, что вероятность обнаружения конкретной пространственной связи между каждой парой меток могла бы использоваться для назначения наиболее вероятной метки для каждого фрагмента, т. е. для получения наиболее вероятной конфигурации меток для всего изображения в целом. Таким образом, применяя марковские случайные поля, можно объединять информацию, предоставляемую визуальными признаками каждого фрагмента (внешний потенциал), и информацию из пространственных связей с другими фрагментами того же изображения (внутренний потенциал). Объединив обе эти характеристики в функции потенциала и применяя процесс оптимизации, можно получить конфигурацию меток, которая наилучшим образом описывает все изображение в целом.

Основные этапы этой процедуры кратко описаны ниже (см. рис. 6.9):

1) изображение автоматически сегментируется (разделяется на фрагменты с использованием нормализованных разрезов);
2) полученным фрагментам присваивается список меток и соответствующие вероятности на основе их визуальных признаков с использованием классификатора;
3) одновременно определяются пространственные связи между теми же фрагментами;
4) применяется марковское случайное поле, объединяющее первоначальные метки и пространственные связи. В результате фрагментам назначаются новые метки с применением алгоритма имитации отжига;
5) смежные фрагменты с одинаковыми метками объединяются.

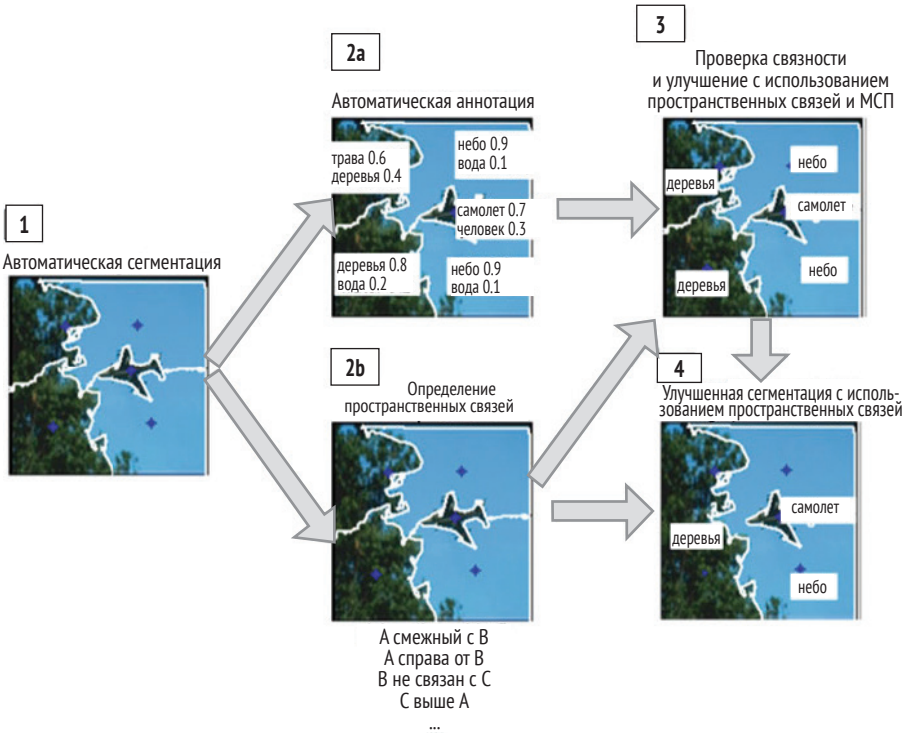


Рис. 6.9. Улучшение качества меток изображения посредством включения пространственных связей в марковское случайное поле: 1) автоматическое разделение изображения на фрагменты, 2а) первоначальное назначение меток для фрагментов, 2б) установление пространственных связей между фрагментами, 3) улучшение качества меток с помощью марковского случайного поля, 4) улучшенное разделение изображения на фрагменты (сегментация) [5]

Функция энергии, которая должна быть минимизирована, объединяет информацию, предоставленную классификаторами (вероятности меток), с пространственными связями (вероятности связей). На этом этапе пространственные связи разделяются на три группы: топологические связи, горизонтальные связи и вертикальные связи. Таким образом, функция энергии содержит четыре компоненты, по одной для каждого типа пространственных связей и еще одна для первоначальных меток. Итоговая формула для функции энергии:

$$U_p(f) = \alpha_1 V_T(f) + \alpha_2 V_H(f) + \alpha_3 V_V(f) + \lambda \sum_o V_o(f), \qquad (6.24)$$

где V_T – потенциал для топологических связей, V_H – потенциал для горизонтальных связей, V_V – потенциал для вертикальных связей, $\alpha_1, \alpha_2, \alpha_3$ – соответствующие постоянные коэффициенты для придания большего

или меньшего веса каждому типу связей. V_o – это потенциал классификации (метки), взвешенный с помощью постоянного коэффициента λ. Эти потенциалы можно оценить с помощью набора тренировочных изображений с метками. Потенциал для конкретного типа пространственных связей между двумя фрагментами классов A и B обратно пропорционален вероятности (частоте) обнаружения этой связи в тренировочном наборе изображений.

Применяя эту методику, можно значительно улучшить качество назначения меток в изображении по сравнению с первоначальным вариантом [6]. В некоторых случаях при использовании информации, предоставляемой новым набором меток, можно также улучшить качество сегментации изображения по сравнению с первоначальным вариантом, как показано на рис. 6.10.

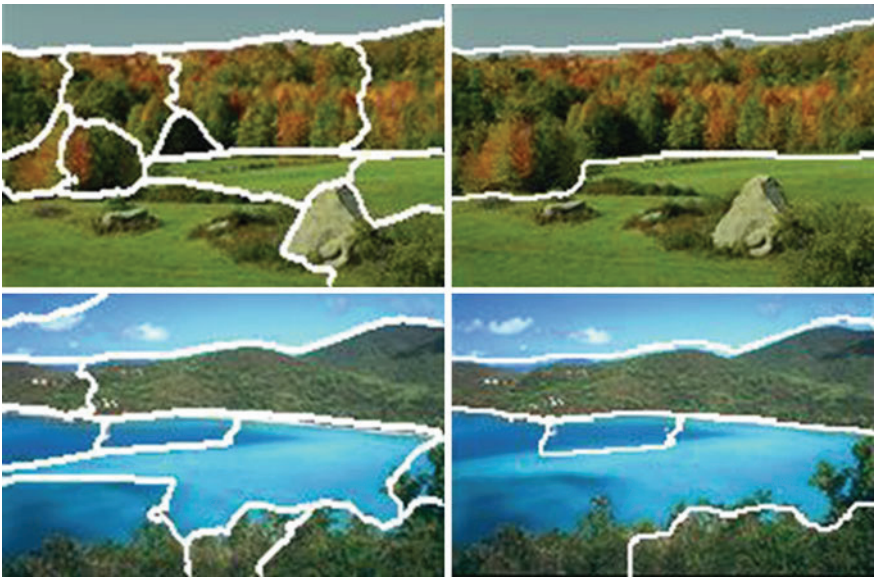


Рис. 6.10. Пример улучшения качества сегментации изображения посредством объединения смежных фрагментов с одинаковыми метками. *Слева*: первоначальная сегментация изображений. *Справа*: улучшенная сегментация изображений [5]

6.8 Материалы для дополнительного чтения

Введение в марковские случайные поля и их приложения приведено в [7]. Подробное описание применения марковских случайных полей для обработки изображений представлено в [9]. Общую вводную информацию об условных марковских случайных полях можно найти в [10, 11].

6.9 Задания и упражнения

1. Для марковской сети на рис. 6.3: а) определить клики в графе; б) определить совместную вероятность как произведение потенциалов клик; в) предполагая, что все переменные являются бинарными, определить требуемые параметры для этой модели.
2. Для марковского случайного поля первого порядка на рис. 6.4 определить минимальное марковское ограждение для каждой переменной. Марковское ограждение переменной q_i – это множество переменных, которые делают ее независимой от остальных переменных в графе.
3. Решить задачу 2 для марковского случайного поля второго порядка на рис. 6.5.
4. Для регулярного марковского случайного поля размером 4×4 узла с соседством первого порядка считать, что каждый узел может принимать одно из двух значений – 0 и 1. Также считать, что используются потенциалы сглаживания, как в приложении сглаживания изображений, с коэффициентом $\lambda = 4$, придающим больший вес наблюдениям. С учетом первоначальной конфигурации F и наблюдений G получить конфигурацию оценки апостериорного максимума (MAP), используя вариант итерационных условных режимов (ICM) алгоритма стохастической имитации.

```
    0 0 0 0
F:  0 0 0 0
    0 0 0 0
    0 0 0 0

    0 0 0 0
G:  0 1 1 0
    0 1 0 0
    0 0 1 0
```

5. Решить задачу 4 с использованием версии Metropolis алгоритма стохастической имитации с вероятностью $P = 0.5$. *Совет*: можно воспользоваться методикой подбрасывания монеты, для того чтобы решить, сохраняется ли конфигурация с более высокой энергией.
6. Решить задачи 4 и 5 с использованием метода границ апостериорного максимума (MPM) вместо оценки апостериорного максимума (MAP).

7. Граница изображения находится там, где происходит резкое изменение значений соседних пикселов (максимум первой производной, если рассматривать изображение как двумерную функцию). Определить потенциалы для марковского случайного поля первого порядка, которое выделяет границы изображения, считая, что каждый узел поля является бинарным, где 1 обозначает границу, а 0 – отсутствие границы. Наблюдением является уровень серого цвета в изображении, при котором значение каждого пиксела изменяется в пределах от 0 до 255.
8. Какова сложность по времени алгоритма стохастической имитации для его различных вариантов?
9. *** Реализовать алгоритм сглаживания изображений с использованием регулярного марковского случайного поля первого порядка. Изменять параметр λ и наблюдать влияние изменений на результат обработки изображения. Повторить реализацию для марковского случайного поля второго порядка.
10. *** Написать программу генерации изображения с повышением разрешения, используя марковские случайные поля. Например, сгенерировать изображение с удвоением размеров $2n \times 2m$ исходного изображения $n \times m$.

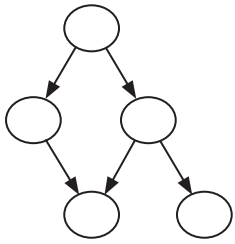
Ссылки на источники

1. Besag, J. Statistical analysis of non-lattice data. Statistician 24 (3), 179–195 (1975).
2. Binder, K. Ising Model. Hazewinkel, Michiel, Encyclopedia of Mathematics. Springer, New York (2001).
3. Geman, D., Geman, S., Graffigne, C. Locating Object and Texture Boundaries. Pattern recognition theory and applications. Springer, Heidelberg (1987).
4. Hammersley, J. M., Clifford, P. Markov fields on finite graphs and lattices. Unpublished Paper. http://www.statslab.cam.ac.uk/grg/books/hammfest/hamm-cliff.pdf (1971). Accessed 14 Dec 2014.
5. Hernández-Gracidas, C., Sucar, L. E. Markov randomfields and spatial information to improve automatic image annotation. Advances in Image and Video Technology. Lecture Notes in Computer Science, vol. 4872, p. 879–892. Springer (2007).
6. Hernández-Gracidas, C., Sucar, L. E., Montes, M. Improving image retrieval by using spatial relations. J. Multimed. Tools Appl. 62, 479–505 (2013).

7. Kindermann, R., Snell, J. L. Markov random fields and their applications. Am. Math. Soc. 34, 143–167 (1980).
8. Lafferty, J., McCallum, A., Pereira, F. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: International Conference on Machine Learning (2001).
9. Li, S. Z.: Markov Random Field Modeling in Image Analysis. Springer, London (2009).
10. Sutton, C., McCallum, A. An Introduction to Conditional RandomFields for Relational Learning. In: Geetor, L., Taskar, B. (eds.) Introduction to Statistical Relational Learning,MIT Press, Cambridge (2006).
11. Wallach, H. M.: Conditional random fields: an introduction. Technical Report MS-CIS-04-21, University of Pennsylvania (2004).

Глава 7

Байесовские сети: представление и логический вывод



7.1 Введение

В отличие от марковских сетей, являющихся неориентированными графовыми моделями, байесовские сети – это ориентированные графовые модели, которые представляют совместную вероятность множества случайных переменных. Некоторые методики, рассмотренные в предыдущих главах, такие как байесовские классификаторы и скрытые марковские модели, являются частными случаями (вариантами) байесовских сетей. Учитывая важность и объем исследований, выполненных по этой теме в последние годы, байесовским сетям посвящены две главы. В этой главе рассматриваются подробности представления и методики логического вывода. В следующей главе будет обсуждаться обучение байесовских сетей, в особенности обучение структуры и параметров.

Пример гипотетической медицинской байесовской сети показан на рис. 7.1. В этом графе узлы представляют случайные переменные, а дуги обозначают направления зависимостей между переменными. Структура графа кодирует множество отношений условной независимости между переменными. Например, следующие условные независимости можно логически вывести из рассматриваемого здесь примера:

- жар не зависит от боли в мышцах при гриппе (общая причинная связь);

- жар не зависит от нездорового питания при брюшном тифе (непрямая причинная связь);
- брюшной тиф не зависит от гриппа, если симптом «жар» НЕ известен (общая следственная связь). Известность симптома «жар» устанавливает зависимость между брюшным тифом и гриппом, например если теперь известно, что у пациента тиф и жар, то снижается вероятность гриппа.

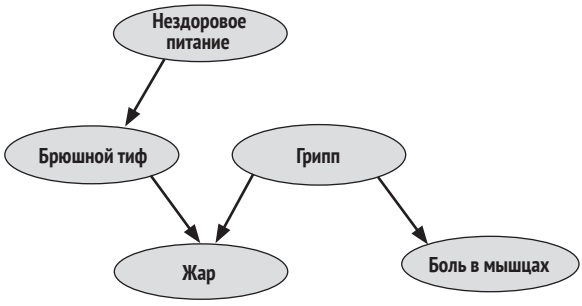


Рис. 7.1. Простой гипотетический пример медицинской байесовской сети

В дополнение к структуре байесовская сеть учитывает множество локальных параметров, которые являются условными вероятностями для каждой переменной с учетом их родителей в графе. Например, условная вероятность жара в зависимости от гриппа и тифа *P*(*жар* | *тиф*, *грипп*). Таким образом, совместная вероятность всех переменных в сети может быть представлена на основе этих локальных параметров. При этом обычно подразумевается важное сохранение в определенном количестве требуемых параметров.

Рассматривая байесовскую сеть (структуру и параметры), можно дать ответ на несколько вероятностных запросов. Например, в рассмотренном выше примере: какова вероятность жара при гриппе? Что более вероятно – тиф или грипп при жаре и нездоровом питании?

В следующем разделе будет формализовано представление байесовской сети, а затем будет представлено несколько алгоритмов для получения ответов (результатов) на различные типы вероятностных запросов.

7.2 Представление

Байесовская сеть представляет совместную вероятность множества n (дискретных) переменных $X_1, X_2, \ldots, X_n$ в форме направленного ациклического графа (НАГ) и множества таблиц условных вероятностей. Для каждого узла, соответствующего переменной, имеется связанная с ним таблица условных вероятностей, которая содержит вероятность каждого состояния переменной с учетом его родителей в графе. Структура

сети подразумевает наличие множества утверждений об условной независимости, придающее мощь этому представлению.

На рис. 7.2 изображен пример простой байесовской сети. Структура графа подразумевает наличие множества утверждений об условной независимости для этого множества переменных. Например, переменная R условно независима от C, G, F, D с учетом T, т. е.

$$P(R \mid C, T, G, F, D) = P(R \mid T). \tag{7.1}$$

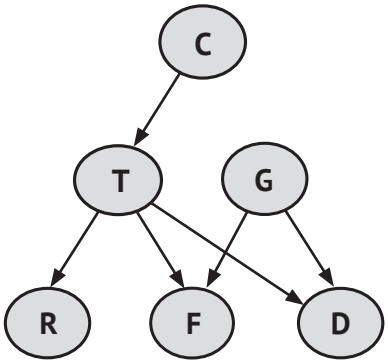


Рис. 7.2. Байесовская сеть

7.2.1 Структура

Утверждения об условной независимости, подразумеваемые структурой байесовской сети, должны соответствовать отношениям условной независимости для совместного распределения вероятностей, и наоборот. Эти отношения обычно представлены с использованием следующей нотации. Если переменная X условно независима от переменной Z с учетом Y:

- в распределении вероятностей: $P(X \mid Y, Z) = P(X \mid Y)$;
- в графе: $I < X \mid Y \mid Z >$.

Утверждения об условной независимости можно проверить (верифицировать) непосредственно из структуры байесовской сети, используя критерий, называемый *d-разделенностью* (d-separation). Перед определением этого критерия рассмотрим три простые структуры байесовских сетей для трех переменных и двух дуг:

- последовательная структура: $X \rightarrow Y \rightarrow Z$;
- расходящаяся структура: $X \leftarrow Y \rightarrow Z$;
- сходящаяся структура: $X \rightarrow Y \leftarrow Z$.

В первых двух случаях переменные X и Z условно независимы относительно Y, но в третьем случае это не так. Этот последний случай, назы-

ваемый *объяснение извне* (explaining away), вполне естественно соответствует наличию двух причин с общим следствием, т. е. знание следствия и одной из причин изменяет нашу уверенность в другой причине. Подобные причины могут быть связаны с разделяющимся узлом Y в графе. Таким образом, в зависимости от конкретного случая Y является последовательным, расходящимся или сходящимся узлом.

D-разделенность

Пусть в графе G множество переменных A является условно независимым от множества B с учетом множества C, если в графе G не существует пути между A и B, такого, что:

1) все узлы являются сходящимися или имеют потомков в множестве C;
2) все прочие узлы находятся вне множества C.

Например, для байесовской сети, изображенной на рис. 7.2, узел R независим от узла C относительно узла T, но T и G не являются независимыми относительно узла F.

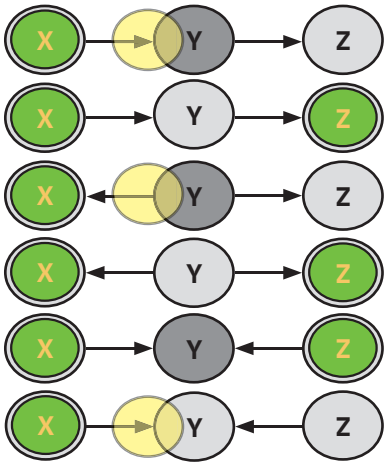


Рис. 7.3. Демонстрация различных случаев *d*-разделенности с использованием алгоритма Bayes ball. В тех случаях, когда шар блокируется узлом Y, критерий условной независимости не выполняется. Когда шар свободно проходит через узел Y, критерий условной независимости выполняется

Другой способ верификации *d*-разделенности – использование алгоритма Bayes ball. Будем считать, что имеется путь от узла X к узлу Z с узлом Y в середине (см. рис. 7.3). Узел Y затенен (серым цветом), если он известен (создан его экземпляр и задано значение), иначе узел Y не затенен (серым цветом). Мы бросаем шар из узла X в направлении

узла Z, и если шар достигает узла Z, то X и Z не являются независимыми относительно узла Y в соответствии со следующими правилами:

1) если узел Y последовательный или расходящийся и он не затенен (серым цветом), то шар свободно проходит через этот узел;
2) если узел Y последовательный или расходящийся и он затенен (серым цветом), то шар блокируется;
3) если узел Y сходящийся и он не затенен (серым цветом), то шар блокируется;
4) если узел Y сходящийся и он затенен (серым цветом), то шар свободно проходит через этот узел.

В соответствии с приведенным выше определением d-разделенности любой узел X условно независим от всех узлов в графе G, которые не являются потомками X с учетом его родителей в графе $Pa(X)$. Это утверждение известно как марковское свойство. Структуру байесовской сети можно определить с помощью родителей каждой переменной, таким образом, множество родителей переменной X известно как контур X. Для примера на рис. 7.2 структуру можно определить следующим образом:

1) Pa(C) = Ø
2) Pa(T) = C;
3) Pa(G) = Ø
4) Pa(R) = T;
5) Pa(F) = T, G;
6) Pa(D) = T, G.

Учитывая это условие и используя цепное правило, можно определить совместное распределение вероятностей для множества переменных в байесовской сети как произведение условной вероятности каждой переменной с учетом ее родителей:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa(X_i)). \tag{7.2}$$

Для примера на рис. 7.2:

$$P(C, T, G, R, F, D) = P(C)P(G)P(T \mid C)P(R \mid T)P(F \mid T, G)P(D \mid T, G).$$

Марковское ограждение (Markov blanket) узла X $MB(X)$ – это множество узлов, которые делают его независимым от всех прочих узлов в графе G, т. е. $P(X \mid G - X) = P(X \mid MB(X))$. Для байесовской сети марковское ограждение узла X состоит из:

- родителей узла X;
- потомков узла X;
- других родителей потомков узла X.

Например, в байесовской сети на рис. 7.2 марковским ограждением узла R является узел T, а марковским ограждением узла T являются узлы C, R, F, D, G.

Отображения

С учетом распределения вероятностей P для **X** и его графического представления G обязательно должно существовать соответствие между условной независимостью в распределении P и в графе G. Это соответствие называется *отображением* (mapping). Существуют три основных типа отображений:

- D-отображение (D-map): все отношения условной независимости в распределении P соблюдаются (с учетом d-разделенности) в графе G;
- I-отображение (I-map): все отношения условной независимости в графе G истинны в распределении P;
- P-отображение (P-map): или совершенное отображение, объединяющее D-отображение и I-отображение.

В общем случае не всегда возможно получить совершенное отображение отношений независимости между графом (G) и распределением (P), поэтому мы принимаем решение об использовании так называемого *минимального I-отображения* (minimal I-map): все отношения условной независимости, определяемые графом G, истинны в распределении P, и если удалить любую дугу в графе G, то это условие нарушается [14].

Аксиомы независимости

Учитывая некоторые отношения условной независимости между подмножествами случайных переменных, можно вывести другие отношения условной независимости в виде аксиом, т. е. без необходимости оценки вероятности или мер степени независимости. Существует несколько основных правил вывода новых отношений условной независимости из других отношений условной независимости. Эти правила известны как аксиомы независимости:

- симметричность: $I(X, Z, Y) \rightarrow I(Y, Z, X)$;
- декомпозиция: $I(X, Z, Y \cup W) \rightarrow I(X, Z, Y) \wedge I(X, Z, W)$;
- слабое объединение: $I(X, Z, Y \cup W) \rightarrow I(X, Z \cup W, Y)$;
- сокращение: $I(X, Z, Y) \wedge I(X, Z \cup Y, W) \rightarrow I(X, Z, Y \cup W)$;
- пересечение: $I(X, Z \cup W, Y) \wedge I(X, Z \cup Y, W) \rightarrow I(X, Z, Y \cup W)$.

Примеры практического применения аксиом независимости в графах показаны на рис. 7.4.

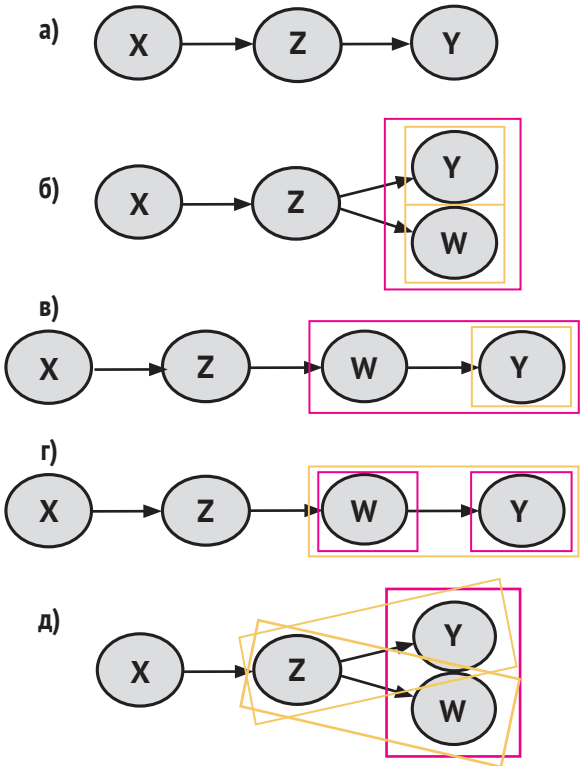


Рис. 7.4. Примеры практического применения аксиом независимости в графах: а) симметричность, б) декомпозиция, в) слабое объединение, г) сокращение, д) пересечение

7.2.2 Параметры

Для завершения определения байесовской сети необходимо определить ее параметры. Для байесовской сети параметрами являются условные вероятности каждого узла с учетом его родителей в графе. Если рассматривать дискретные переменные, то:

- корневые узлы: вектор частных распределений вероятностей;
- другие узлы: таблица условных вероятностей конкретной переменной с учетом ее родителей в соответствующем графе.

На рис. 7.5 показаны некоторые из таблиц условных вероятностей байесовской сети, изображенной на рис. 7.2. Для непрерывных переменных необходимо задать функцию, которая устанавливает связь функции плотности (распределения) каждой переменной с плотностью (распределения) родителей этой переменной (например, фильтры Кал-

мана (Kalman filters) рассматривают в этом качестве гауссовы распределенные переменные и линейные функции).

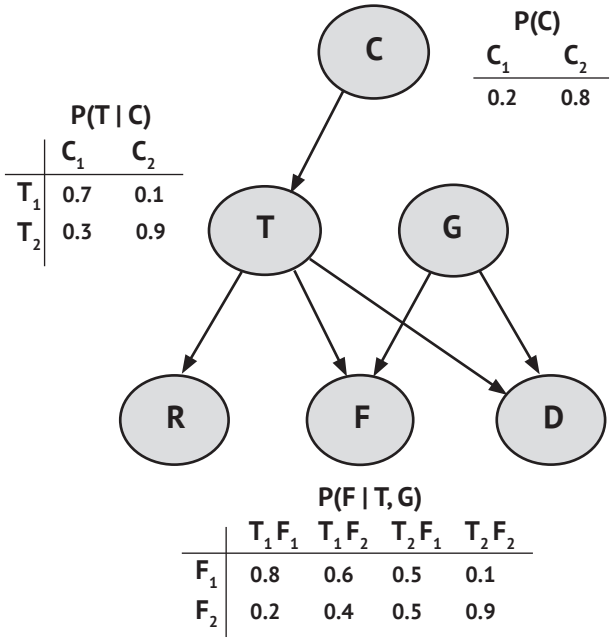


Рис. 7.5. Параметры байесовской сети, показанной на рис. 7.2. Здесь приведены таблицы условных вероятностей для некоторых переменных из этого примера: $P(C)$, $P(T \mid C)$ и $P(F \mid T, G)$. Предполагается, что в рассматриваемом тут примере все переменные являются бинарными

Для дискретных переменных количество параметров в таблице условных вероятностей возрастает экспоненциально при увеличении числа родителей рассматриваемого узла. Это может создавать проблемы при большом количестве родителей. Требования к объему памяти могут стать слишком высокими, а кроме того, трудно оценивать так много параметров. Для устранения этой проблемы были предложены два основных варианта решения: один на основе *канонических моделей* (canonical models), другой – использование представлений таблиц условных вероятностей в форме графов. В следующих разделах будут кратко рассмотрены обе схемы.

7.2.2.1 Канонические модели

Канонические модели представляют связи между множеством случайных переменных для конкретных взаимодействий с использованием нескольких параметров. Эту методику можно применять, если

вероятности случайной переменной в байесовской сети согласованы с конкретными *каноническими* (canonical) связями с учетом конфигураций родителей этой переменной. Существует несколько классов канонических моделей, наиболее часто используются *Noisy OR* и *Noisy AND* для бинарных переменных и их расширения для переменных с несколькими значениями – *Noisy Max* и *Noisy Min* соответственно.

Модель Noisy OR по своей сущности является расширением отношения OR (ИЛИ) в логике. Рассмотрим логический вентиль OR, в котором выводится значение *True* (истина), если любое из входных данных имеет значение *True*. Модель Noisy OR основана на концепции логического OR (ИЛИ), но отличается наличием определенной (небольшой) вероятности того, что значением переменной не является *True*, даже если один или несколько ее родителей имеют значение *True*. Похожим образом модель Noisy AND связана с логическим оператором AND (И). Эти модели применяются, только если все переменные являются бинарными, но существуют расширения для переменных с несколькими значениями, которые рассматривают множество упорядоченных значений для каждой переменной. Например, рассмотрим переменную, представляющую заболевание *D*. В бинарных канонических моделях эта переменная принимает два значения – *True* и *False*. В модели с несколькими значениями переменную можно определить как $D \in$ {*False* (ложь), *Mild* (легкая форма), *Intermediate* (средней тяжести), *Severe* (тяжелая форма)}, при этом перечисленные значения соблюдают предварительно заданный порядок. Модели Noisy Max и Noisy Min представляют собой обобщения моделей Noisy OR и Noisy AND соответственно для переменных с несколькими значениями.

В следующем подразделе будет подробно описана модель Noisy OR. Другие модели можно определить подобным образом.

Модель Noisy OR

Модель Noisy OR применяется, когда несколько переменных, или *причин* (causes), могут давать результат, или *следствие* (effect), если любая из переменных-причин имеет значение *True*, и чем больше причин являются истинными, тем более возрастает вероятность получения именно этого следствия. Например, следствием может быть определенный симптом *S*, а причинами – несколько возможных заболеваний D_1, $D_2, \ldots, D_m$, которые могут создавать этот симптом. При этом если ни одно из этих заболеваний не наблюдается (все имеют значение *False*), то симптом не появляется, а когда наблюдается любое из перечисленных заболеваний (*True*), симптом присутствует с высокой вероятностью, и эта вероятность возрастает при увеличении количества D_i = *True*. Графовое представление связей модели Noisy OR в байесовской сети показано на рис. 7.6.

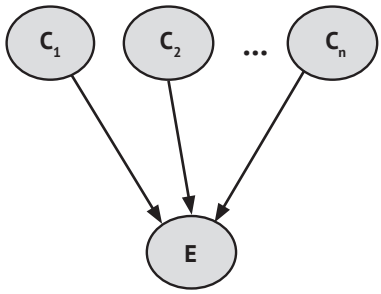


Рис. 7.6. Графовое представление структуры Noisy OR:
n переменных причин (C) являются родителями переменной следствия (E)

Формально следующие два условия обязательно должны выполняться, для того чтобы каноническая модель Noisy OR была применимой:

- обязательность: следствие является ложным, если все возможные причины являются ложными;
- независимость исключений: если следствие является проявлением нескольких причин, то механизмы, препятствующие проявлению следствия по одной причине, не зависят от механизмов, препятствующих проявлению следствия по прочим причинам.

Вероятность подавления следствия E (следствие не возникает) при наличии причины C_i определяется следующим образом:

$$q_i = P(E = False \mid C_i = True). \tag{7.3}$$

С учетом этого определения и описанных выше условий параметры в таблице условных вероятностей для модели Noisy OR можно вычислить с использованием следующих формул, где все m причин являются истинными (*True*):

$$P(E = False \mid C_1 = True, \ldots, C_m = True) = \prod_{i=1}^{m} q_i; \tag{7.4}$$

$$P(E = True \mid C_1 = True, \ldots, C_m = True) = 1 - \prod_{i=1}^{m} q_i. \tag{7.5}$$

В общем случае если k из m причин являются истинными (*True*), то $P(E = False \mid C_1 = True, \ldots, C_k = True) = \prod_{i=i}^{k} q_i$, поэтому если все причины являются ложными (*False*), то следствие является ложным (*False*) с вероятностью, равной единице. Таким образом, требуется только один параметр для каждой родительской переменной, чтобы сформировать таблицу условных вероятностей для вероятности препятствова-

ния возникновению следствия q_i. В этом случае количество параметров возрастает линейно, а не экспоненциально при увеличении количества родителей.

В качестве примера рассмотрим модель Noisy OR с тремя причинами C_1, C_2, C_3, где вероятность препятствования возникновению следствия одинакова для всех трех причин $q_1 = q_2 = q_3 = 0.1$. С учетом этих параметров можно получить таблицу условных вероятностей для переменной следствия, как показано в табл. 7.1.

Таблица 7.1. Таблица условных вероятностей для переменной Noisy OR с тремя родителями и параметрами $q_1 = q_2 = q_3 = 0.1$

C_1	0	0	0	0	1	1	1	1
C_2	0	0	1	1	0	0	1	1
C_3	0	1	0	1	0	1	0	1
$P(E = 0)$	1	0.1	0.1	0.01	0.1	0.01	0.01	0.001
$P(E = 1)$	0	0.9	0.9	0.99	0.9	0.99	0.99	0.999

7.2.2.2 Другие представления

Канонические модели применимы в определенных ситуациях, но они не предоставляют общее решение для получения компактных представлений таблиц условных вероятностей. Другой вариант представления основан на следующем наблюдении: в таблицах вероятности для многих предметных областей одинаковые значения вероятностей достаточно часто повторяются несколько раз в одной и той же таблице, например в таблице условных вероятностей обычно содержится много нулевых значений. Таким образом, нет необходимости многократного представления этих повторяющихся значений, вполне достаточно один раз представить каждое неповторяющееся значение.

Представлением, позволяющим воспользоваться преимуществом такого условия, является *дерево решений* (decision tree), поэтому его можно применять для компактного представления таблицы условных вероятностей. В дереве решений каждый внутренний узел соответствует переменной в таблице условных вероятностей, а ветви, исходящие из узла, соответствуют различным значениям, которые может принимать эта переменная. Узлы-листья в таком дереве представляют различные значения вероятности. Путь от корня до листа определяет значение вероятности для соответствующих переменных – по значениям в пути. Если переменная не учитывается в пути, это означает, что в таблице условных вероятностей содержатся одинаковые значения вероятностей для этой переменной.

Например, в табл. 7.2 отображена таблица условных вероятностей $P(X \mid A, B, C, D, E, F, G)$, предполагая, что все переменные являются бинарными (F, T). На рис. 7.7 показано дерево решений для таблицы условных вероятностей, приведенной в табл. 7.2. В этом примере экономия объема памяти незначительна, но для больших таблиц можно получить существенное снижение требований к объему памяти.

Таблица 7.2. Таблица условных вероятностей $P(X \mid A, B, C, D, E, F, G)$, в которой содержится несколько повторяющихся значений

A	B	C	D	E	F	G	X
T	T/F	T/F	T/F	T/F	T/F	T/F	0.9
F	T	T/F	T	T/F	T	T	0.9
F	T	T/F	T	T/F	T	F	0.0
F	T	T/F	T	T/F	F	T/F	0.0
F	T	T	F	T	T/F	T	0.9
F	T	T	F	T	T/F	F	0.0
F	T	T	F	F	T/F	T/F	0.0
F	T	F	F	T/F	T/F	T/F	0.0
F	F	T	T/F	T	T/F	T	0.9
F	F	T	T/F	T	T/F	F	0.0
F	F	T	T/F	F	T/F	T/F	0.0
F	F	F	T/F	T/F	T/F	T/F	0.0

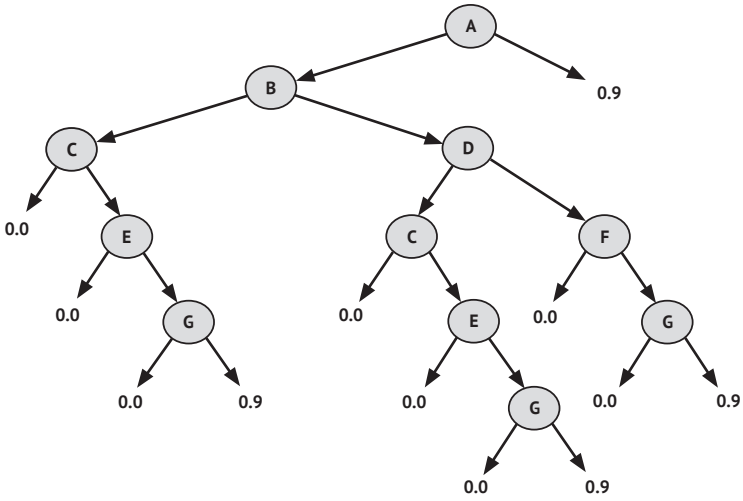


Рис. 7.7. Дерево решений для таблицы условных вероятностей, представленной в табл. 7.2. Для каждой переменной (узел в дереве) стрелка влево соответствует значению F, а стрелка вправо – значению T

Диаграмма решений (decision diagram) является расширением дерева решений, рассматривающим направленный ациклический граф, в котором нет ограничений, присущих структуре дерева. При этом исключается необходимость представления повторяющихся значений вероятности в узлах-листьях, а в некоторых случаях обеспечивается даже еще более компактное представление. Пример диаграммы решений для таблицы условных вероятностей, представленной в табл. 7.2, показан на рис. 7.8.

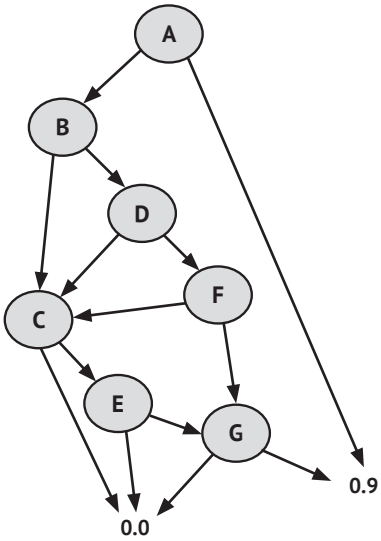


Рис. 7.8. Диаграмма решений, представляющая таблицу условных вероятностей из табл. 7.2. Здесь также стрелка влево соответствует значению F, а стрелка вправо – значению *T*

7.3 Логический вывод

Вероятностный логический вывод заключается в распространении (propagating) следствий (результатов) определенного свидетельства в байесовской сети для оценки его результата в неизвестных переменных. Таким образом, зная значения некоторого подмножества переменных в модели, можно получить апостериорные вероятности других переменных. Подмножество неизвестных переменных может быть пустым, тогда мы получаем априорные вероятности всех переменных.

Существуют два основных варианта задачи логического вывода в байесовских сетях. Первый вариант – получение апостериорной вероятности единственной переменной *H* с учетом подмножества известных (имеющих конкретное значение) переменных **E**, т. е. $P(H \mid \mathbf{E})$. Эта методика известна как вывод с помощью *конъюнктивного запроса* (conjunc-

tive query inference). Теоретически эту задачу можно решить, используя единственный вывод с помощью запроса несколько раз и применяя цепное правило, что усложняет задачу в целом. Например, $P(A, B \mid \mathbf{E})$ можно записать как $P(A \mid \mathbf{E})P(B \mid A, \mathbf{E})$, тогда потребуется два отдельных вывода с помощью запроса и операция умножения. В некоторых приложениях желательно знать, какие значения являются наиболее вероятными в множестве гипотез, т. е. $ArgMax_{\mathbf{H}}P(\mathbf{H} \mid \mathbf{E})$. Если множество гипотез **H** включает все ненаблюдаемые переменные, это называется *наиболее вероятным объяснением* (most probable explanation – MPE), или *задачей полной абдукции* (total abduction). Если нас интересует наиболее правдоподобное совместное состояние некоторых (но не всех) ненаблюдаемых переменных, это соответствует *оценке апостериорного максимума* (maximum a posteriori assignment – MAP), или *задаче частичной абдукции* (partial abduction).

В первую очередь мы сосредоточим внимание на задаче логического вывода с помощью единственного запроса, затем рассмотрим задачи MPE и MAP. Если необходимо решить задачу логического вывода с использованием прямого вычисления (полного перебора), т. е. по совместному распределению, то сложность вычислений возрастает экспоненциально при увеличении числа переменных, и задача становится практически неразрешимой даже при небольшом количестве переменных. Для решения этой задачи более эффективным способом было разработано много алгоритмов, которые можно упрощенно разделить на несколько следующих классов:

1) распространение вероятности (алгоритм Дж. Перла (Judea Pearl) [13]);
2) исключение переменных;
3) обусловливание;
4) дерево сочленений;
5) стохастическая имитация.

Алгоритм распространения вероятности применяется только к односвязным графам (деревьям и полидеревьям[15]), хотя существует расширение для обобщенных сетей – алгоритм распространения с петлями, который не гарантирует сходимости. Остальные четыре класса алгоритмов работают для любой структуры сети, при этом последняя является приблизительной методикой, но остальные три – точные методики.

В наихудшем случае задача логического вывода является NP-трудной (NP-hard) для байесовских сетей [1]. Но существуют эффективные (полиномиальные) алгоритмы для определенных типов структур (одно-

[15] Полидерево (polytree) – это односвязный направленный ациклический граф, в котором некоторые узлы имеют более одного родителя. В ориентированном (направленном) дереве каждый узел имеет не более одного родителя.

связных сетей). Для других структур эффективность решения зависит от связности конкретного графа. Во многих приложениях графы являются разреженными (sparse), и для этих случаев существуют весьма эффективные алгоритмы логического вывода.

Далее будет описан алгоритм распространения вероятности для односвязных сетей, затем наиболее часто используемые методики для многосвязных байесовских сетей.

7.3.1 Односвязные сети: алгоритм распространения доверия

В этом разделе описан алгоритм распространения доверия (вероятности) в дереве, предложенный Дж. Перлом (Judea Pearl, 1982) и предоставляющий основу для нескольких из наиболее продвинутых и обобщенных методик.

С учетом определенного свидетельства **E** (подмножества переменных с присвоенными конкретными значениями) апостериорная вероятность для значения *i* любой переменной *B* может быть получена с помощью применения правила Байеса:

$$P(Bi \mid \mathbf{E}) = P(Bi)P(\mathbf{E} \mid Bi) / P(\mathbf{E}). \tag{7.6}$$

Учитывая, что рассматриваемая здесь байесовская сеть имеет структуру дерева, любой узел делит эту сеть на два независимых поддерева. Следовательно, можно разделить свидетельство на следующие фрагменты (см. рис. 7.9):

- **E−**: свидетельство в дереве с корнем в узле *B*;
- **E+**: вся прочая часть свидетельства.

Тогда:

$$P(Bi \mid \mathbf{E}) = P(Bi)P(\mathbf{E}-, \mathbf{E}+ \mid Bi) / P(\mathbf{E}). \tag{7.7}$$

Учитывая, что **E+** и **E−** независимы, и еще раз применяя правило Байеса, получаем:

$$P(Bi \mid \mathbf{E}) = \alpha P(Bi \mid \mathbf{E}+)P(\mathbf{E}- \mid Bi), \tag{7.8}$$

где α – постоянный коэффициент нормализации. Если определить следующие компоненты:

$$\lambda(Bi) = P(\mathbf{E}- \mid Bi), \tag{7.9}$$

$$\pi(Bi) = P(Bi \mid \mathbf{E}+), \tag{7.10}$$

тогда формулу 7.8 можно записать так:

$$P(Bi \mid \mathbf{E}) = \alpha\pi(Bi)\lambda(Bi). \tag{7.11}$$

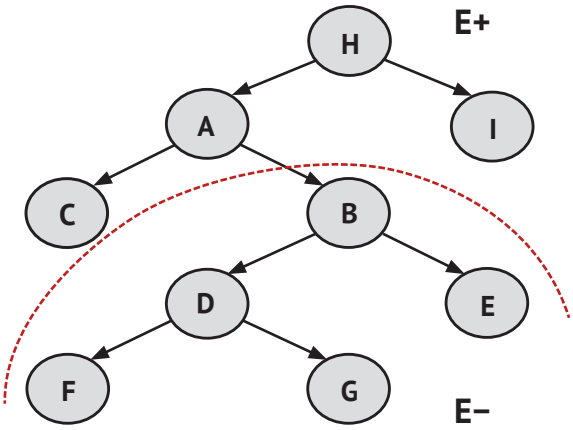


Рис. 7.9. В байесовской сети, имеющей структуру дерева, каждый узел (B) делит сеть на два условно независимых поддерева – **E+** и **E–**

Формула 7.11 представляет собой основу для алгоритма распределенного распространения с целью получения апостериорной вероятности всех узлов, для которых не заданы значения. Процесс вычисления апостериорной вероятности любого узла B разделяется на два этапа: (i) свидетельство, приходящее от потомков B в рассматриваемом дереве (λ), и (ii) свидетельство, приходящее от родителя узла $B(\pi)$. Можно интерпретировать каждый узел B в таком дереве как простой обработчик, который сохраняет свои векторы $\pi(B)$ и $\lambda(B)$ и свою таблицу условных вероятностей $P(B \mid A)$. Свидетельство распространяется с помощью механизма передачи сообщений, где каждый узел передает соответствующие сообщения своему родителю и своим потомкам в дереве. Выражение для сообщения, передаваемого из узла B своему родителю A:

$$\lambda_B(Ai) = \sum_j P(B_j \mid A_i)\lambda(B_j). \tag{7.12}$$

Выражение для сообщения, передаваемого из узла B своему потомку S_k:

$$\pi_k(Bi) = \alpha\pi(B_j)\prod_{l\neq k}\lambda_l(B_j), \tag{7.13}$$

где l обозначает ссылку на каждого отдельного потомка узла B.

Каждый узел может получать несколько λ-сообщений, которые объединяются методом почленного умножения λ-сообщений, полученных от каждого потомка. Следовательно, λ для узла A с m потомками вычисляется следующим образом:

$$\lambda(Ai) = \prod_{j=1}^{m}\lambda_{S_j}(Ai). \tag{7.14}$$

Алгоритм распространения начинает работу с присваивания свидетельства известным переменным, затем распространяет это свидетельство с помощью механизма передачи сообщений до тех пор, пока не будет достигнут корень дерева для λ-сообщений и узлы-листья для π-сообщений. На рис. 7.10 и 7.11 показаны соответствующие схемы распространения. В конце процесса распространения каждый узел содержит обновленные собственные векторы λ и π. Апостериорная вероятность любой переменной B определяется объединением этих векторов с использованием формулы 7.11 и нормализации.

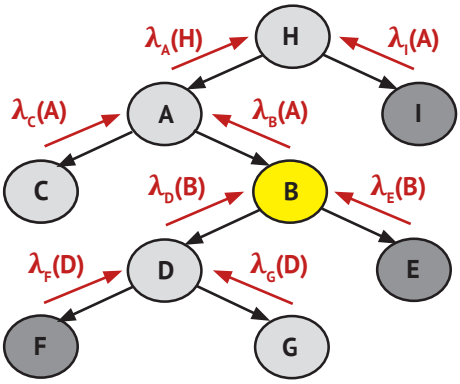


Рис. 7.10. Восходящее распространение. λ-сообщения распространяются от узлов-листьев к корню

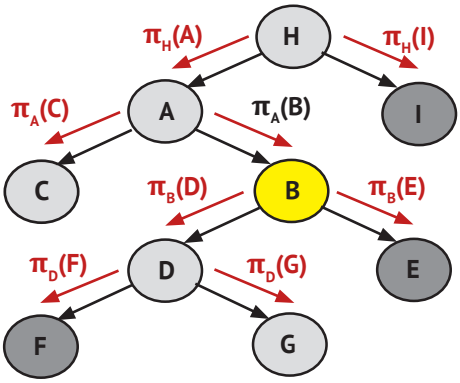


Рис. 7.11. Нисходящее распространение. π-сообщения распространяются от корня к узлам-листьям

Для корня и узлов-листьев необходимо определить некоторые начальные условия:

- узлы-листья: если узел неизвестен, то $\lambda = [1, 1, \ldots, 1]$ (равномерное распределение). Если узел известен, то $\lambda = [0, 0, \ldots, 1, \ldots, 0]$ (единица для конкретного присвоенного значения и нули для всех остальных значений);

- корневой узел: если неизвестен, то $\pi = P(A)$ (вектор априорного частного распределения вероятностей). Если известен, то $\pi = [0, 0, \ldots, 1, \ldots, 0]$ (единица для конкретного присвоенного значения и нули для всех остальных значений).

Теперь продемонстрируем выполнение алгоритма распространения доверия на простом примере. Рассмотрим байесовскую сеть, показанную на рис. 7.12, с 4 бинарными переменными (каждая переменная может принимать значения *false* и *true*) *C*, *E*, *F*, *D* с таблицами условных вероятностей, приведенными на рис. 7.12.

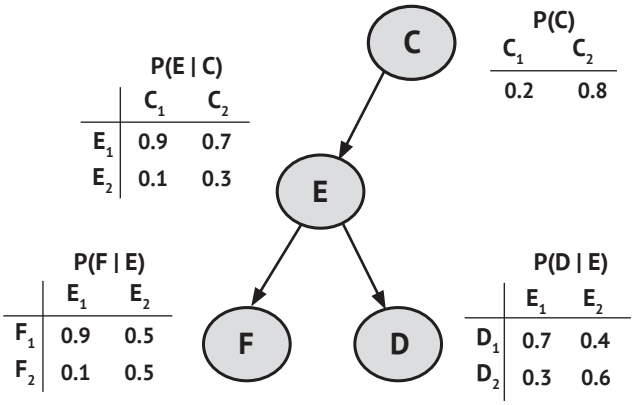


Рис. 7.12. Простая байесовская сеть, используемая в примере применения алгоритма распространения доверия

Пусть только свидетельство $F = false$. Тогда начальные условия для узлов-листьев: $\lambda_F = [1, 0]$ и $\lambda_D = [1, 1]$ (свидетельство отсутствует). Процесс распространения к родительскому узлу (E) реализуется простым умножением этих векторов λ на соответствующие таблицы условных вероятностей:

$$\lambda_F(E) = [1, 0] \begin{bmatrix} 0.9, 0.5 \\ 0.1, 0.5 \end{bmatrix} = [0.9, 0.5],$$

$$\lambda_D(E) = [1, 1] \begin{bmatrix} 0.7, 0.4 \\ 0.3, 0.6 \end{bmatrix} = [1, 1].$$

Далее $\lambda(E)$ определяется как объединение сообщений от двух потомков этого узла:

$$\lambda(E) = [0.9, 0.5] \times [1, 1] = [0.9, 0.5].$$

Теперь выполняется распространение к родителю C:

$$\lambda_E(C) = [0.9, 0.5] \begin{bmatrix} 0.9, 0.7 \\ 0.1, 0.3 \end{bmatrix} = [0.86, 0.78].$$

В этом примере $\lambda(C) = [0.86, 0.78]$, так как у узла C только один потомок. Таким образом, восходящее распространение завершено, и теперь можно перейти к нисходящему распространению.

Учитывая, что узел C не имеет присвоенного значения $\pi(C) = [0.8, 0.2]$, распространение выполняется к его потомку E, который также соответствует процедуре умножения вектора π на соответствующую таблицу условных вероятностей:

$$\pi(E) = [0.8, 0.2] \begin{bmatrix} 0.9, 0.7 \\ 0.1, 0.3 \end{bmatrix} = [0.86, 0.14].$$

Далее выполняется распространение к потомку D, но, учитывая, что узел E имеет еще одного потомка F, необходимо также принять во внимание λ-сообщение от этого другого потомка, следовательно:

$$\pi(D) = [0.86, 0.14] \times [0.9, 0.5] \begin{bmatrix} 0.7, 0.4 \\ 0.3, 0.6 \end{bmatrix} = [0.86, 0.14].$$

На этом завершается нисходящее распространение (нет необходимости выполнять распространение к узлу F, так как эта переменная известна). С учетом векторов λ и π для каждой неизвестной переменной нужно всего лишь почленно умножить эти векторы и нормализовать результат для получения апостериорных вероятностей:

$$P(C) = [0.86, 0.2] \times [0.86, 0.78] = \alpha[0.69, 0.16] = [0.815, 0.185],$$
$$P(E) = [0.86, 0.14] \times [0.9, 0.5] = \alpha[0.77, 0.07] = [0.917, 0.083],$$
$$P(D) = [0.57, 0.27] \times [1, 1] = \alpha[0.57, 0.27] = [0.67, 0.33].$$

На этом пример выполнения алгоритма распространения доверия завершается.

Распространение вероятностей представляет собой весьма эффективный алгоритм для байесовских сетей со структурой дерева. Сложность по времени для получения апостериорной вероятности для всех переменных в дереве пропорциональна диаметру (diameter) сети (числу дуг в пути от корня до наиболее удаленного листа).

Механизм передачи сообщений можно непосредственно распространить на полидеревья, поскольку они также являются односвязными сетями. В этом случае узел может иметь несколько родителей, поэтому λ-сообщения должны передаваться от узла всем его родителям. Сложность по времени имеет тот же порядок, что и для деревьев.

Алгоритм распространения применяется только к односвязным сетям. В следующем разделе представлены обобщенные алгоритмы, которые можно применить к любой структуре.

7.3.2 Многосвязные сети

Существует несколько классов алгоритмов для вероятностного логического вывода в многосвязных байесовских сетях. В этом разделе рассматриваются основные классы: (i) исключение переменных, (ii) обусловливание и (iii) дерево сочленений.

7.3.2.1 Исключение переменных

Метод исключения переменных основан на принципе вычисления вероятности посредством маргинализации совместного распределения. Но, в отличие от простейшего метода, здесь имеется преимущество в виде условий независимости байесовской сети и ассоциативных и дистрибутивных свойств операций сложения и умножения, которые повышают эффективность вычислений.

Рассмотрим байесовскую сеть, представляющую совместное распределение вероятностей переменной $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$. Необходимо вычислить апостериорную вероятность конкретной переменной или подмножества переменных X_H с учетом подмножества переменных свидетельств X_E. Остальные переменные обозначены как X_R, такие, что $\mathbf{X} = \{X_H \cup X_E \cup X_R\}$.

Апостериорная вероятность X_H с учетом свидетельства:

$$P(X_H \mid X_E) = P(X_H, X_E) / P(X_E). \tag{7.15}$$

Оба члена этой формулы можно вычислить с помощью маргинализации совместного распределения:

$$P(X_H, X_E) = \sum_{X_R} P(\mathbf{X}) \tag{7.16}$$

и

$$P(X_E) = \sum_{X_H} P(X_H, X_E). \tag{7.17}$$

Отдельным случаем, заслуживающим внимания, является вычисление частного распределения вероятностей переменных при отсутствии свидетельств. В этом случае $X_E = \emptyset$. Еще один интересный случай – вычисление вероятности свидетельства, которое выполняется по формуле 7.17.

Цель методики исключения переменных – эффективное выполнение всех перечисленных выше вычислений. Для этого можно сначала

представить совместное распределение как произведение локальных вероятностей в соответствии со структурой сети. Затем операции сложения можно выполнить только для тех подмножеств элементов, которые являются функциями переменных, подлежащих нормализации. Преимущества такого подхода заключаются в использовании свойств операций сложения и умножения, благодаря которым сокращается количество необходимых операций. Далее будет рассмотрен пример применения этого метода.

Рассмотрим байесовскую сеть, показанную на рис. 7.13, в которой необходимо вычислить вероятность $P(A \mid D)$. Для этого требуется определить $P(A, D)$ и $P(D)$. Для вычисления первого члена нужно обязательно исключить узлы B, C, E из совместного распределения, т. е.

$$P(A, D) = \sum_B \sum_C \sum_E P(A)P(B \mid A)P(C \mid A)P(D \mid B, C)P(E \mid C). \quad (7.18)$$

В соответствии с дистрибутивным свойством сложения можно перейти к следующему равнозначному выражению:

$$P(A, D) = P(A) \sum_B P(B \mid A) \sum_C P(C \mid A)P(D \mid B, C) \sum_E P(E \mid C). \quad (7.19)$$

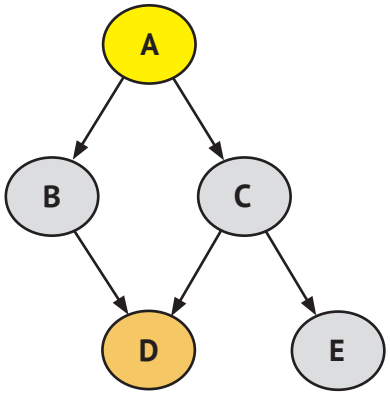


Рис. 7.13. Байесовская сеть, используемая в примере применения алгоритма исключения переменных

Если считать, что все эти переменные являются бинарными, то становится возможным сокращение с 32 до 9 операций. Разумеется, это сокращение будет более существенным для более крупных моделей или при наличии большего количества значений для каждой переменной.

В качестве примера рассмотрим байесовскую сеть на рис. 7.12 и предположим, что необходимо вычислить $P(E \mid F = f_1) = P(E, F = f_1) / P(F = f_1)$. С учетом структуры этой байесовской сети совместное распределение

вероятностей определяется как $P(C, E, F, D) = P(C)P(E \mid C)P(F \mid E)P(D \mid E)$. Сначала вычисляется $P(E, F)$ с изменением порядка выполнения операций:

$$P(E, F) = \sum_D P(F \mid E)P(D \mid E) \sum_C P(C)P(E \mid C).$$

Необходимо выполнить эти вычисления для каждого значения E с учетом $F = f_1$:

$$P(e_1, f_1) = \sum_D P(f_1 \mid e_1)P(D \mid e_1) \sum_C P(C)P(e_1 \mid C),$$

$$P(e_1, f_1) = \sum_D P(f_1 \mid e_1)P(D \mid e_1)[0.9 \times 0.8 + 0.7 \times 0.2],$$

$$P(e_1, f_1) = \sum_D P(f_1 \mid e_1)P(D \mid e_1)[0.86],$$

$$P(e_1, f_1) = [0.9 \times 0.7 + 0.9 \times 0.3][0.86].$$

$$P(e_1, f_1) = [0.9][0.86] = 0.774.$$

Аналогичным образом вычисляется $P(e_2, f_1)$, затем по этим полученным значениям можно вычислить $P(f_1) = \sum_E P(E, f_1)$. Наконец, вычисляется апостериорная вероятность переменной E с учетом f_1:

$$P(e_1 \mid f_1) = P(e_1, f_1) / P(f_1) \text{ и } P(e_2 \mid f_1) = P(e_2, f_1) / P(f_1).$$

Важнейшим аспектом алгоритма исключения переменных является выбор правильного порядка исключения каждой переменной, поскольку это оказывает важное воздействие на количество требуемых операций. Различные элементы, генерируемые во время вычислений, называются факторами (factors), которые являются функциями на подмножестве переменных, отображающими каждый экземпляр (значение) этих переменных в неотрицательное число (эти числа не обязательно должны быть вероятностями). В общем случае фактор можно представить как $f(X_1, X_2, \ldots, X_m)$. Например, в предыдущем примере один из факторов можно представить как $f(C, E) = P(C)P(E \mid C)$, т. е. как функцию от двух переменных.

Сложность вычислений по памяти и по времени для алгоритма исключения переменных определяется размером соответствующих факторов, т. е. количеством переменных w, для которых определен конкретный фактор. В основном сложность исключения (маргинализации) любого числа переменных является экспоненциальной зависимостью от количества переменных в факторе, т. е. $O(exp(w))$ [2]. Следовательно, порядок исключения переменных должен выбираться так, чтобы в наибольшем

факторе сохранялось минимальное количество переменных. Но поиск наилучшего порядка исключения переменных, вообще говоря, представляет собой NP-трудную задачу.

Существует несколько эвристик, которые помогают определить достаточно удачный («хороший») порядок исключения переменных. Эти эвристики можно описать на основе *графа взаимодействий* (interaction graph) – неориентированного графа, который формируется в процессе исключения переменных. Переменные каждого фактора образуют клику в графе взаимодействий. Первоначальный граф взаимодействий получается из исходной структуры байесовской сети посредством отмены направлений дуг и включения дополнительных дуг между каждой парой несвязанных переменных, которые имеют общего потомка. Затем при каждой операции исключения переменной X_j граф взаимодействий изменяется следующим образом: (i) добавляется дуга между каждой парой несвязанных между собой соседей узла X_j, (ii) переменная X_j удаляется из графа.

Графы взаимодействий, получаемые из байесовской сети на рис. 7.13 последовательным исключением переменных в порядке *E*, *C*, *B*, *D*, показаны на рис. 7.14.

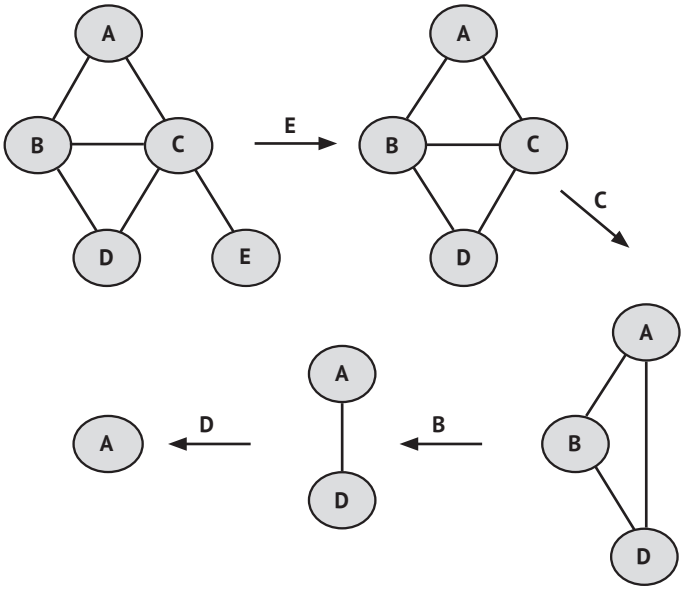


Рис. 7.14. Графы взаимодействий, получаемые из байесовской сети на рис. 7.13 последовательным исключением переменных в порядке *E*, *C*, *B*, *D*

Ниже кратко описаны две наиболее распространенные эвристики для определения порядка исключения переменных, которые можно получить из графа исключений:

- min-degree: исключается переменная, которая приводит к наименьшему возможному фактору; это равнозначно исключению переменной с наименьшим количеством соседей в текущем графе исключения;
- min-fill: исключается переменная, которая приводит к добавлению минимального количества ребер в граф взаимодействий.

Недостаток метода исключения переменных состоит в том, что он позволяет получить апостериорную вероятность только для одной переменной (или подмножества переменных). Для получения апостериорной вероятности каждой переменной без конкретно определенного значения в байесовской сети необходимо повторить вычисления для каждой такой переменной. Далее рассматриваются два алгоритма, которые одновременно вычисляют апостериорные вероятности для всех переменных.

7.3.2.2 Обусловливание

Метод обусловливания (conditioning) основан на том факте, что переменная с присвоенным значением блокирует распространение свидетельства в байесовской сети. Следовательно, можно выполнить разрез (cut) графа по переменной с определенным значением, что позволит преобразовать многосвязный граф в полидерево, для которого возможно применение алгоритма распространения вероятности.

В общем случае подмножеству переменных можно присвоить конкретные значения для преобразования многосвязной сети в односвязный граф. Если эти переменные в действительности неизвестны, то можно устанавливать для них каждое из всех возможных значений, затем выполнять алгоритм распространения вероятности для каждого значения. При выполнении каждого процесса распространения мы получаем вероятность для каждой неизвестной переменной. Далее конечные значения вероятностей вычисляются как взвешенные сочетания этих вероятностей.

В первую очередь будет разработан алгоритм обусловливания с предположением о том, что необходимо выделить только одну переменную, а затем алгоритм будет расширен для нескольких переменных. В формальном выражении следует определить вероятность для любой переменной B с учетом свидетельства E и при обусловленности переменной A. По правилу полной вероятности:

$$P(B \mid E) = \sum_i P(B \mid E, a_i) P(a_i \mid E), \tag{7.20}$$

где $P(B \mid E, a_i)$ – апостериорная вероятность B, которая определяется по распространению вероятности для каждого возможного значения A, $P(a_i \mid E)$ – весовой коэффициент.

Применяя правило Байеса, получим следующую формулу для оценки весовых коэффициентов:

$$P(a_i \mid E) = \alpha P(a_i)P(E \mid a_i). \tag{7.21}$$

Первый множитель $P(a_i)$ можно вычислить по распространению без свидетельства. Второй множитель $P(E \mid a_i)$ вычисляется по распространению при $A = a_i$ для получения вероятности переменных свидетельства, α – постоянный коэффициент нормализации.

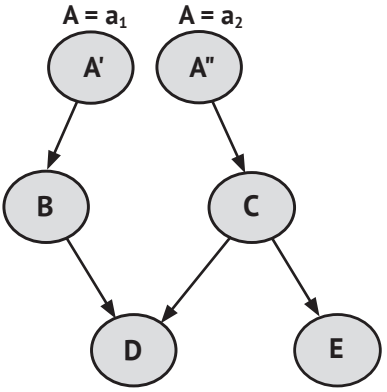


Рис. 7.15. Байесовская сеть, изображенная на рис. 7.13, преобразуется в односвязную сеть при условии присваивания конкретного значения переменной *A*

Например, рассмотрим байесовскую сеть на рис. 7.13. Эту многосвязную сеть можно преобразовать в полидерево, полагая, что *A* – переменная с присвоенным конкретным значением (см. рис. 7.15). Если свидетельством являются переменные *D*, *E*, то вероятности для других переменных *A*, *B*, *C* можно получить методом обусловливания, выполняя следующие шаги:

1) вычислить априорную вероятность для *A* (в рассматриваемом примере эта вероятность уже известна, так как *A* является корневым узлом);
2) вычислить вероятность узлов свидетельства *D*, *E* для каждого значения переменной *A* методом распространения вероятности в полидереве;
3) вычислить весовые коэффициенты $P(a_i \mid D, E)$ по результатам выполнения шагов 1 и 2 с использованием правила Байеса;
4) оценить вероятность *B* и *C* для каждого значения *A* с учетом свидетельства методом распространения вероятности в полидереве;
5) вычислить апостериорные вероятности для *B* и *C* по результатам выполнения шагов 3 и 4, применяя формулу 7.20.

В общем случае для преобразования многосвязной сети в полидерево необходимо присвоить конкретные значения m переменным. Следовательно, алгоритм распространения вероятности обязательно должен быть выполнен для всех сочетаний значений (векторное произведение) таких переменных. Если каждая переменная имеет k значений, то число комбинаций распространений равно k^m. Процедура в целом та же, что и для одной переменной, как описано в предыдущем разделе, но сложность возрастает.

7.3.2.3 Алгоритм дерева сочленений

Метод (алгоритм) дерева сочленений основан на преобразовании байесовской сети в *дерево сочленений* (junction tree), в котором каждый узел является группой или кластером переменных из исходной сети. Вероятностный логический вывод выполняется в этой новой структуре представления. Ниже будет кратко описан основной алгоритм, более подробное описание см. в разделе материалов для дополнительного чтения.

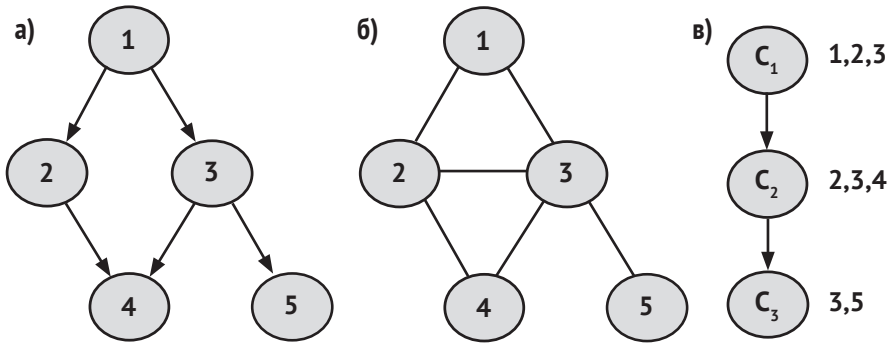


Рис. 7.16. Преобразование байесовской сети в дерево сочленений:
а) исходная байесовская сеть, б) триангулированный граф, в) дерево сочленений

Процедура преобразования описана ниже (см. рис. 7.16):

1) оценить направленность дуг;
2) упорядочить узлы в графе (на основе максимальной мощности);
3) морализовать граф (добавить дуги между парами узлов с общими потомками);
4) при необходимости добавить дополнительные дуги, чтобы сделать граф триангулированным;
5) определить клики в графе (подмножества узлов, которые являются полностью связными, но не являются подмножествами других полностью связных множеств);

6) сформировать дерево сочленений, в котором каждый узел является кликой, а его родителем является любой узел, который содержит все общие предшествующие переменные по порядку, определенному на шаге 2.

После завершения процедуры создания дерева сочленений логический вывод основывается на методе распространения вероятностей в дереве сочленений, применяемом так же, как и для байесовских сетей, имеющих структуру дерева. Сначала вычисляется совместная вероятность (потенциал) каждого макроузла, и с учетом некоторого свидетельства эта вероятность распространяется для получения апостериорной вероятности для каждого сочленения. Отдельное значение вероятности каждой переменной вычисляется по совместной вероятности соответствующего сочленения с применением маргинализации. Ниже эта процедура будет описана более подробно.

Алгоритм дерева сочленений можно разделить на два этапа: предварительная обработка и распространение. На этапе предварительной обработки потенциалы каждой клики вычисляются по следующей пошаговой схеме:

1) определение множества переменных для каждой клики C_i;
2) определение множества переменных, которые являются общими с предшествующей (родительской) кликой, S_i;
3) определение переменных, которые принадлежат множеству C_i, но не множеству S_i: $R_i = C_i - S_i$;
4) вычисление потенциалов каждой клики clq_i как произведения соответствующих таблиц условных вероятностей: $\psi(clq_i) = \prod_j P(X_j \mid Pa(X_j))$, где X_j – переменные в клике clq_i.

Например, рассмотрим байесовскую сеть на рис. 7.16 с кликами: $clq_1 = \{1, 2, 3\}$, $clq_2 = \{2, 3, 4\}$, $clq_3 = \{3, 5\}$. Тогда этап предварительной обработки будет таким:

C: $C_1 = \{1, 2, 3\}$, $C_2 = \{2, 3, 4\}$, $C_3 = \{3, 5\}$,
S: $S_1 = \varnothing$, $S_2 = \{2, 3\}$, $S_3 = \{3\}$,
R: $R_1 = \{1, 2, 3\}$, $R_2 = \{4\}$, $R_3 = \{5\}$.
Потенциалы: $\psi(clq_1) = P(1)P(2 \mid 1)P(3 \mid 2)$, $\psi(clq_2) = P(4 \mid 3, 2)$,
$\psi(clq_3) = P(5 \mid 3)$.

Этап распространения выполняется так же, как и процедура распространения доверия в деревьях, т. е. для λ-сообщений выполняется восходящее распространение, а для π-сообщений – нисходящее распространение.

Восходящее распространение

1. Вычисление λ-сообщения для передачи в родительскую клику: $\lambda(C_i) = \sum_R \psi(C_i)$.
2. Обновление потенциала каждой клики с использованием λ-сообщений от своих потомков: $\psi(C_j)' = \lambda(C_i)\psi(C_j)$.
3. Повторение шагов 1 и 2 до тех пор, пока не будет выполнен переход к корневой клике.
4. После перехода к корневой клике вычислить $P'(C_r) = \psi(C_r)'$.

Нисходящее распространение

1. Вычисление π-сообщения для передачи в каждый узел-потомок i: $\pi(C_i) = \sum_{C_j - S_i} P'(C_j)$.
2. Обновление потенциала каждой клики при получении π-сообщения от своего родителя: $P'(C_i) = \pi(C_i)\psi(C_i)'$.
3. Повторение шагов 1 и 2 до тех пор, пока не будет выполнен переход к узлам-листьям в дереве сочленений.

Если существует свидетельство, то потенциалы для каждой клики обновляются на основе этого свидетельства, и выполняется та же самая процедура распространения.

На завершающем этапе вычисляются частные апостериорные вероятности для каждой переменной по потенциалам соответствующих клик с использованием маргинализации: $P(X) = \sum_{C_i - X} P'(C_i)$.

Существуют два основных варианта алгоритма дерева сочленений, известных как архитектура Hugin [6] и архитектура Шеноя–Шафера (Shenoy-Shafer) [16]. Приведенное выше описание алгоритма основано на архитектуре Hugin. Основными различиями между этими архитектурами являются хранимая информация и способ вычисления сообщений. Эти различия определяют сложность вычислений каждой архитектуры. В общем, архитектура Шеноя–Шафера требует меньшего объема памяти, но больше времени.

7.3.2.4 Анализ сложности

В наихудшем случае вероятностный логический вывод для байесовских сетей является NP-трудной задачей [1]. Сложность по времени и по памяти определяется так называемой шириной дерева, или древесной шириной (tree-width), при этом учитывается, насколько близка структура сети к дереву. Таким образом, байесовская сеть со структурой дерева (максимум один родитель для каждого узла) имеет ширину дерева, равную единице. Полидерево, в котором существует не менее k родителей для каждого узла, имеет ширину дерева, равную k. В общем слу-

чае ширина дерева определяется по степени плотности (dense) топологии сети, и это оказывает воздействие на следующие характеристики: (i) размер наибольшего фактора в алгоритме исключения переменных; (ii) количество переменных, которым необходимо присвоить конкретные значения, в алгоритме обусловливания; (iii) размер наибольшей клики в алгоритме дерева сочленений.

На практике байесовские сети достаточно часто представляют собой разреженные графы, и в таких случаях методика дерева сочленений является весьма эффективной даже для моделей с сотнями переменных. В более сложных сетях предпочтительнее использовать алгоритмы аппроксимации, которые описываются в следующем разделе.

7.3.3 Приближенный логический вывод

7.3.3.1 Алгоритм распространения с петлями

Это просто частное приложение алгоритма распространения вероятности для многосвязных сетей. Несмотря на то что в этом случае условия для применения алгоритма не соблюдены, к тому же он предоставляет только приближенное решение задачи логического вывода, все же этот вариант алгоритма распространения вероятности весьма эффективен. Учитывая, что байесовская сеть не является односвязной, при распространении сообщений могут образовываться петли (loop) в сети. Следовательно, процедура распространения повторяется несколько раз. Ниже приведено пошаговое описание этой процедуры:

1) инициализация случайными значениями λ и π значений (сообщений) для всех узлов;
2) повторение следующих операций до обеспечения сходимости или достижения максимального числа итераций:
   a) выполнение процедуры распространения вероятности в соответствии с алгоритмом для односвязной сети;
   b) вычисление апостериорной вероятности для каждой переменной.

Этот алгоритм сходится, когда разность между апостериорными вероятностями для всех переменных на текущей и предыдущей итерациях меньше определенного порогового значения. Эмпирически было определено, что для некоторых конкретных структур данный алгоритм сходится к истинным значениям апостериорных вероятностей, но для других структур алгоритм не является сходящимся [11].

Важным приложением алгоритма распространения доверия с петлями является Turbo Codes – широко распространенная схема обнаружения и исправления ошибок, используемая в системах обмена данными.

7.3.3.2 Стохастическая имитация

Алгоритмы стохастической имитации, как можно понять из их названия, выполняют имитацию байесовских сетей несколько раз, при этом каждая имитация дает экземпляр значения (выборку значений) для всех переменных, которым изначально не было присвоено конкретное значение. Имитируемые значения выбираются случайно в соответствии с условной вероятностью каждой переменной. Этот процесс повторяется N раз, и апостериорная вероятность каждой переменной аппроксимируется с учетом частоты появления каждого значения в пространстве выборки. Это позволяет оценить апостериорную вероятность, которая зависит от количества выборок. Но при этом на стоимость вычислений не влияет сложность сети. В следующих подразделах представлены два алгоритма стохастической имитации для байесовских сетей: *алгоритм логической выборки* (logic sampling) и *оценка правдоподобия с весовыми коэффициентами* (likelihood weighting).

Алгоритм логической выборки

Алгоритм логической выборки (logic sampling) – это простой алгоритм стохастической имитации, который генерирует выборки в соответствии со следующей пошаговой процедурой:

1) генерация выборок значений для корневых узлов байесовской сети в соответствии с их априорными вероятностями, т. е. для каждой корневой переменной X генерируется случайное значение по распределению, соответствующему $P(X)$;
2) генерация выборок для следующего уровня, на котором расположены потомки узлов с ранее сгенерированными выборками, в соответствии с условными вероятностями этих потомков $P(Y \mid Pa(Y))$, где $Pa(Y)$ – родители узла Y;
3) повторение шага 2 до тех пор, пока не будет выполнена генерация выборок для узлов-листьев.

Описанная выше процедура выполняется N раз для генерации N выборок. Вероятность каждой переменной оценивается как отношение числа появлений (частоты) этого значения в N выборках к общему количеству выборок N, т. е. $P(X = x_i) \sim No(x_i) / N$, где $No(x_i)$ – число случаев $X = x_i$ во всех выборках.

Непосредственное применение описанной выше процедуры дает оценку частных (распределений) вероятностей всех переменных при отсутствии свидетельства. Если свидетельство существует (некоторым переменным присвоены конкретные значения), то все выборки, логически не согласованные с этим свидетельством, исключаются, и апостериорные вероятности оцениваются по оставшимся выборкам.

Например, рассмотрим байесовскую сеть на рис. 7.13 и 10 выборок, сгенерированных с помощью алгоритма логической выборки. Будем считать, что все переменные являются бинарными, тогда 10 сгенерированных выборок приведены в табл. 7.3.

Таблица 7.3. Выборки, сгенерированные с использованием алгоритма логической выборки для байесовской сети, показанной на рис. 7.13

Переменные	A	B	C	D	E
Выборка$_1$	T	F	F	F	T
Выборка$_2$	F	T	T	F	F
Выборка$_3$	T	F	F	T	F
Выборка$_4$	F	F	T	F	T
Выборка$_5$	T	F	T	T	F
Выборка$_6$	F	F	F	F	T
Выборка$_7$	F	T	T	T	F
Выборка$_8$	F	F	F	F	F
Выборка$_9$	F	F	F	T	F
Выборка$_{10}$	T	T	T	T	F

Все переменные бинарные с двумя возможными значениями: *True* = T или *False* = F.

Если свидетельство отсутствует, то с учетом выборок, приведенных в табл. 7.3, частные вероятности оцениваются следующим образом:

- $P(A = T) = 4/10 = 0.4$;
- $P(B = T) = 3/10 = 0.3$;
- $P(C = T) = 5/10 = 0.5$;
- $P(D = T) = 5/10 = 0.5$;
- $P(E = T) = 3/10 = 0.3$.

Остальные вероятности вычисляются просто как дополнения до единицы: $P(X = F) = 1 - P(X = T)$.

Если свидетельство существует для $D = T$, то исключаются все выборки, в которых $D = F$, и оцениваются частные вероятности по остальным четырем выборкам:

- $P(A = T \mid D = T) = 3/5 = 0.6$;
- $P(B = T \mid D = T) = 2/5 = 0.4$;
- $P(C = T \mid D = T) = 3/5 = 0.6$;
- $P(E = T \mid D = T) = 1/5 = 0.2$.

Недостатком алгоритма логической выборки при существующем свидетельстве является слишком большое количество исключаемых выборок. Это приводит к тому, что для качественной оценки требуется большее количество выборок. В следующем подразделе представлен алгоритм, который обращается с выборками не столь расточительно.

Оценка правдоподобия с весовыми коэффициентами

Алгоритм оценки правдоподобия с весовыми коэффициентами (likelihood weighting) генерирует выборки так же, как и алгоритм логической выборки, но при наличии свидетельства несогласованные логически выборки не исключаются. Вместо этого каждой выборке присваивается весовой коэффициент, соответствующий весовому коэффициенту свидетельства для данной конкретной выборки. Рассмотрим выборку s и переменные свидетельств $\mathbf{E} = \{E_1, \ldots, E_m\}$, тогда весовой коэффициент выборки s оценивается следующим образом:

$$W(\mathbf{E} \mid s) = P(E_1)P(E_2)\ldots P(E_m), \tag{7.22}$$

где $P(E_i)$ – вероятность переменной свидетельства E_i для данной конкретной выборки.

Апостериорная вероятность для каждой переменной X, принимающей значение x_i, оценивается посредством деления суммы весовых коэффициентов $W_i(X = x_i)$ для каждой выборки, где $X = x_i$, на общую сумму весовых коэффициентов для всех выборок:

$$P(X = x_i) \sim \sum_i W_i(X = x_i) \,/\, \sum_i W_i. \tag{7.23}$$

7.3.4 Наиболее вероятное объяснение

Наиболее вероятное объяснение (most probable explanation – MPE), или задача абдукции (abduction; обратный вывод), состоит в определении наиболее вероятных значений для подмножества переменных (подмножества объяснения) в байесовской сети с учетом некоторого свидетельства. Существует два варианта этой задачи: полная абдукция (total abduction) и частичная абдукция (partial abduction). В задаче полной абдукции подмножество объяснения представляет собой множество всех переменных, не имеющих конкретного присвоенного значения. В задаче частичной абдукции подмножеством объяснения является правильно подобранное подмножество переменных, не имеющих конкретного присвоенного значения. В общем случае наиболее вероятное объяснение – это не то же самое, что совокупность наиболее вероятных значений для каждой отдельной переменной в подмножестве объяснения.

Рассмотрим множество переменных $\mathbf{X} = \{X_E, X_R, X_H\}$, где X_E – подмножество переменных, не имеющих конкретного присвоенного значения, тогда можно формализовать задачи MPE следующим образом:

- задача полной абдукции: $ArgMax_{X_H, X_R} P(X_H, X_R \mid X_E)$;
- задача частичной абдукции: $ArgMax_{X_H} P(X_H \mid X_E)$.

Один из методов решения задачи MPE основан на измененной версии алгоритма исключения переменной. Для задачи полной абдукции операции суммирования заменяются операциями максимизации:

$$max_{X_H, X_R} P(X_H, X_R \mid X_E).$$

Для задачи частичной абдукции суммирование выполняется по тем переменным, которые не входят в подмножество объяснения, а операция максимизации выполняется над подмножеством объяснения:

$$max_{X_H} \sum_{X_R} P(X_H, X_R \mid X_E).$$

Задача MPE с точки зрения вычислений является более сложной, чем логический вывод с помощью единственного запроса.

7.3.5 Непрерывные переменные

До настоящего момента рассматривались байесовские сети с дискретными переменными, которые могут принимать несколько значений. При обработке непрерывных переменных одним из вариантов является дискретизация переменных, но такой подход может привести к потерям информации (при малом количестве интервалов) или к неоправданному росту требований к вычислительным ресурсам (при большом количестве интервалов). Другой вариант – работа непосредственно с непрерывными распределениями (вероятностей). Для некоторых семейств распределения вероятностей были разработаны методики вероятностного логического вывода, в частности для гауссовых переменных. Ниже будет описан простой алгоритм распространения вероятностей для байесовских сетей с линейными гауссовыми переменными [14].

Для этого простого алгоритма формулируются следующие предварительные предположения:

1) структурой сети является полидерево;
2) все источники неопределенности являются гауссовыми переменными, и они не коррелированны (не связаны друг с другом);
3) существует следующее линейное отношение между каждой переменной и ее родителями:

$$X = b_1U_1 + b_2U_2 + \ldots + b_nU_n + W_X,$$

где U_i – родители переменной X, b_i – постоянные коэффициенты, а W_X представляет гауссов шум с нулевой меаной (средним значением).

Процедура логического вывода аналогична алгоритму распространения доверия в дискретных байесовских сетях, но вместо распространения вероятностей здесь распространяются меаны (средние значения) и стандартные отклонения. Для гауссовых распределений частные распределения для всех переменных также являются гауссовыми. Таким образом, в общем случае апостериорную вероятность переменной можно записать в следующем виде:

$$P(X \mid E) = N(\mu_X, \sigma_X),$$

где μ_X и σ_X – соответственно меана (среднее значение) и стандартное отклонение для переменной X с учетом свидетельства E.

Далее рассматривается методика вычисления меаны (среднего значения) и стандартного отклонения с применением алгоритма распространения. Каждая переменная передает своему родителю переменную i:

$$\mu_i^- = (1 / b_i)\left[\mu_\lambda - \sum_{k \neq i} b_k \mu_k^+\right], \tag{7.24}$$

$$\sigma_i^- = (1 / b_i^2)\left[\sigma_\lambda - \sum_{k \neq i} b_k^2 \sigma_k^+\right]. \tag{7.25}$$

Каждая переменная передает своему узлу-потомку j:

$$\mu_j^+ = \frac{\sum_{k \neq j}\left[\mu_k^-/\sigma_k + \mu_\pi/\sigma_\pi\right]}{\sum_{k \neq j} 1/\sigma_k^- + \mu_\pi/\sigma_\pi}, \tag{7.26}$$

$$\sigma_j^+ = \left[\sum_{k \neq j} 1/\sigma_k^- + 1/\sigma_\pi\right]^{-1}. \tag{7.27}$$

Каждая переменная суммирует сообщения, получаемые от своих потомков и родителей, с помощью следующих формул:

$$\mu_\pi = \sum_i b_i \mu_i^+, \tag{7.28}$$

$$\sigma_\pi = \sum_i b_i^2 \sigma_i^+, \tag{7.29}$$

$$\mu_\lambda = \sigma_\lambda \sum_j \mu_j^- / \sigma_j^-, \tag{7.30}$$

$$\sigma_\lambda = \left[\sum_j 1/\sigma_j^- \right]^{-1}. \tag{7.31}$$

Наконец, каждая переменная получает собственное среднее значение и стандартное отклонение, объединяя информацию, полученную от своего родительского узла и узлов-потомков:

$$\mu_X = \frac{\sigma_\pi \mu_\lambda + \sigma_\lambda \mu_\pi}{\sigma_\pi + \sigma_\lambda}, \tag{7.32}$$

$$\sigma_X = \frac{\sigma_\pi \sigma_\lambda}{\sigma_\pi + \sigma_\lambda}. \tag{7.33}$$

Процедура распространения для других типов распределений более сложна, так как эти распределения не обладают теми же свойствами, что и гауссово распределение. В частности, произведение гауссовых распределений также является гауссовым распределением. Для других типов распределений иным вариантом является применение методик стохастической имитации.

7.4 Приложения

Байесовские сети применялись (и применяются) во многих предметных областях, в том числе в медицине, промышленности, образовании, финансовой сфере, биологии и т. д. В качестве примеров байесовских сетей в этой главе будут рассматриваться: (i) методика валидации (проверки корректности) информации и (ii) методика анализа надежности системы. В следующих главах будут продемонстрированы приложения байесовских сетей в других областях.

7.4.1 Валидация информации

Многие системы для принятия решений используют информацию. Если информация ошибочна, это может привести к неоптимальным решениям, а в некоторых случаях решения на основе ошибочных данных могут оказаться опасными. Рассмотрим пример стационарного больничного устройства интенсивной терапии, в котором сенсорные датчики наблюдают за состоянием оперируемого пациента, чтобы температура его тела поддерживалась в переделах нормы. Допустим, что датчики работают постоянно, но иногда они могут выдавать ошибочные показания. При этом возможны две ситуации:

- температурные датчики не показывают изменений температуры, даже если она увеличивается до опасных уровней;
- температурные датчики показывают опасный уровень, даже если температура нормальная.

В первой ситуации возможно нанесение серьезного вреда здоровью пациента. Во второй ситуации для пациента могут быть применены меры экстренного лечения, которые также могут ухудшить его состояние.

Во многих приложениях существуют разнообразные источники информации, например те же сенсорные датчики, которые не являются независимыми. Информация из одного источника дает некоторые сведения и о других источниках. Если есть возможность как-либо представить эти зависимости между различными источниками, то можно использовать их для выявления вероятных ошибок и избежания ошибочных решений. В этом разделе представлен алгоритм валидации (проверки корректности) информации на основе байесовских сетей [4]. Алгоритм начинает работу с создания модели зависимостей между источниками информации (переменными), представленными как байесовская сеть. Далее валидация выполняется в два этапа. На первом этапе выявляются потенциальные критические ошибки посредством сравнения действительного значения с прогнозируемым значением взаимосвязанных переменных с помощью процедуры распространения в байесовской сети. На втором этапе действительные критические ошибки изолируются с помощью создания дополнительной байесовской сети, основанной на свойстве марковского ограждения.

7.4.1.1 Выявление критических ошибок

Предполагается, что возможно создание вероятностной модели, связывающей все переменные в предметной области приложения. Например, рассмотрим сеть, показанную на рис. 7.17, которая представляет самые простые функции газовой турбины.

Предположим, что необходимо проверить измеряемые показания температуры в турбине. При считывании значений остальных сенсорных датчиков и применении процедуры распространения вероятностей появляется возможность вычисления апостериорного распределения вероятностей температуры с учетом всех свидетельств, т. е. $P(T \mid Mw, P, Fg, Pc, Pv, Ps)$. Предполагая, что все переменные являются дискретными или дискретизируются, если они непрерывные, с помощью процедуры распространения получаем распределения вероятностей для каждого значения температуры T. Если действительно наблюдаемое значение совпадает с допустимым верным (valid) значением, которое имеет высокую вероятность, то сенсорный датчик считается работающим пра-

вильно, иначе есть основания полагать, что датчик работает некорректно.

Эта процедура повторяется для всех сенсорных датчиков в модели. Но если валидация одного сенсорного датчика выполняется с использованием неисправного датчика, то можно ожидать и критически некорректно проведенную процедуру валидации. В приведенном выше примере что произойдет, если температура *T* проверяется с использованием неисправного датчика энергии *Mw*? Как узнать, который из датчиков неисправен? Таким образом, применяя описанную выше процедуру валидации, можно лишь выявить условие критической неисправности, но нет возможности точно определить, какой именно датчик действительно неисправен. Это называют видимой неисправностью (apparent fault). Поэтому необходим этап изоляции.

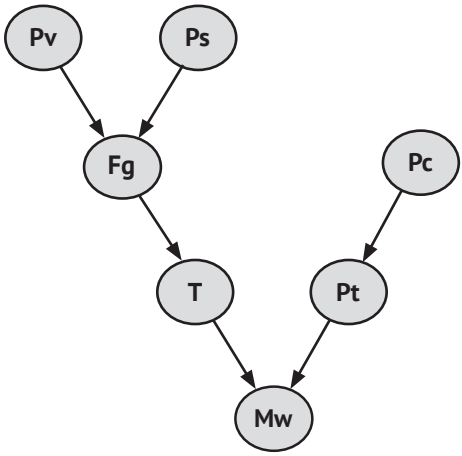


Рис. 7.17. Простая вероятностная модель газовой турбины. Мегаватты энергии, генерируемые газовой турбиной (узел *Mw*), зависят от температуры (узел *T*) и давления в турбине (узел *Pt*). Температура зависит от характеристик потока газа (узел *Fg*), а сам поток газа зависит от состояния клапана, регулирующего поток газа (узел *Pv*), и от давления, под которым подается газ (узел *Ps*). Давление в турбине зависит от давления на выходе компрессора (узел *Pc*)

7.4.1.2 Изоляция критической неисправности

Этап изоляции критической неисправности основан на свойстве марковского ограждения (Markov blanket). Например, в сети на рис. 7.17 $MB(T) = \{Mw, Fg, Pt\}$, а $MB(Pv) = \{Fg\}$. Множество узлов, составляющих марковское ограждение сенсорного датчика, может рассматриваться как защита этого датчика от изменений за пределами его марковского ограждения. Дополнительно мы определяем расширенное марковское ограждение (extended Markov blanket) узла *X* ($EMB(X)$) как множество датчиков, образованное самим этим датчиком и его марковским ограждением. Например, $EMB(Fg) = \{Fg, Pv, Ps, T\}$.

При использовании этого свойства, если неисправность существует в одном из датчиков, ее можно обнаружить в любом из датчиков, входящих в множество *EMB*. С другой стороны, если неисправность возникает за пределами *EMB* рассматриваемых датчиков, то она не влияет на оценку датчика, для которого определено это расширенное марковское ограждение. Тогда можно утверждать, что расширенное марковское ограждение датчика действует как его защита от других критических неисправностей, а также защищает другие датчики от неисправности этого датчика. Мы используем расширенное марковское ограждение EMB для создания модуля изоляции критической неисправности (fault isolation), который отделяет действительные критические неисправности (real faults) от видимых неисправностей (apparent faults). Полная теория разработана в [5].

После завершения цикла основных процедур валидации получено множество *S* видимо неисправных сенсорных датчиков. Таким образом, на основе сравнения множества *S* и *EMB* всех датчиков теория определяет следующие ситуации:

1) если $S = \emptyset$, то критических неисправностей нет;
2) если *S* = *EMB* для датчика *X* и нет другого *EMB*, являющегося подмножеством *S*, то обнаружена единственная действительная неисправность датчика *X*;
3) если *S* = *EMB* для датчика *X* и существует одно или несколько *EMB*, являющихся подмножеством *S*, то обнаружена действительная неисправность датчика *X* и, возможно, действительные неисправности датчиков, *EMB* которых являются подмножествами *S*. В этом случае возможны множественные неразличаемые действительные критические неисправности;
4) если *S* равно объединению нескольких *EMB* и это объединение является единственным в своем роде, то обнаружены множественные различаемые действительные критические неисправности во всех датчиках, *EMB* которых содержатся в множестве *S*;
5) если ни одно из перечисленных выше условий не выполнено, то обнаружены множественные критические неисправности, но их невозможно отличить друг от друга. Все переменные, *EMB* которых являются подмножествами *S*, могут иметь действительную критическую неисправность.

Например, при рассмотрении модели байесовской сети на рис. 7.17 могут возникнуть следующие ситуации (возможны и другие ситуации):

- $S = \{T, Pt, Mw\}$ – это соответствует ситуации 2 и подтверждает наличие одной действительной неисправности в узле *Mw*;

- $S = \{T, Pc, Pt, Mw\}$ – это соответствует ситуации 3, т. е. обнаружена действительная неисправность в узле *Pt* и, возможно, в узлах *Pv* и *Mw*;
- $S = \{Pv, Ps, Fg\}$ – это соответствует ситуации 4, т. е. обнаружены действительные неисправности в узлах *Pv* и *Ps*.

Изоляция действительной критической неисправности выполняется следующим образом. На основе свойства расширенного марковского ограждения *EMB*, описанного выше, действительная неисправность датчика *X* будет обнаружена, если выявлена видимая неисправность в его полном расширенном марковском ограждении. Кроме того, можно утверждать, что видимая неисправность будет обнаружена, если действительная неисправность существует в любом датчике расширенного марковского ограждения рассматриваемого здесь датчика. На основе этих фактов определяется сеть изоляции, образуемая двумя уровнями. Корневые узлы представляют действительные неисправности – по одному узлу для каждого датчика или переменной. Более низкий уровень формируется из отдельных узлов, каждый из которых представляет конкретную видимую неисправность для каждой переменной. Каждый узел является бинарной переменной с двумя возможными значениями: {*correct*, *faulty*}. Следует отметить, что дуги определяются *EMB* каждой переменной. На рис. 7.18 показана сеть изоляции для сети выявления неисправностей на рис. 7.17. Например, узел видимой неисправности, соответствующий переменной *Mw* (узел A_{mw}), соединен с узлами R_{mw}, R_T и R_P, которые представляют действительные неисправности узлов *EMB* переменной *Mw*. В то же время узел R_{mw} соединен со всеми узлами видимых неисправностей, причиной которых является эта действительная неисправность, т. е. с узлами A_{mw}, A_T и A_P. Изоляция критических неисправностей выполняется с помощью процедуры изоляции, описанной в алгоритме 7.1. Неисправные сенсорные датчики будут определяться по узлам действительных неисправностей с «высокой» вероятностью.

Алгоритм 7.1. Функция изоляции

Требования: Сенсорный датчик *n* и состояние сенсорного датчика *n*.

1. Присвоить значение (создать экземпляр) узлу видимого значения, соответствующему датчику *n*.
2. Выполнить процедуру распространения вероятностей и получить апостериорную вероятность для всех узлов действительных неисправностей (Real fault).
3. Обновить вектор сенсорных датчиков P_f(sensors).

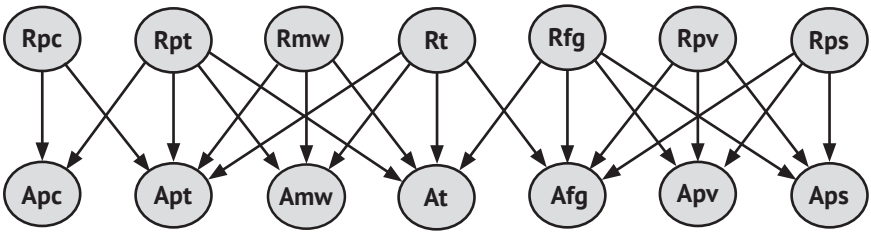


Рис. 7.18. Сеть изоляции для примера на рис. 7.17.
Узлы верхнего уровня представляют действительные неисправности, а узлы нижнего уровня представляют видимые неисправности

7.4.2 Анализ надежности

Для анализа надежности сложных систем общепринятой методикой является разделение системы на более мелкие элементы, блоки, подсистемы или компоненты. Основное предположение: каждая сущность имеет два состояния – *нормальное* (success) и *ошибочное* (failure). Такое разделение позволяет сгенерировать «блок-схему», похожую на описание работающей системы. Для каждого элемента определяется *коэффициент ошибки* (failure rate), и на этом основании вычисляется надежность всей системы в целом.

Обычно для анализа надежности используются *деревья ошибок* (fault trees), но эта методика имеет ограничения, такие как предположение о независимости событий, из-за чего трудно смоделировать зависимость между событиями, т. е. ошибками. Зависимые события могут быть определены при анализе надежности в следующих случаях: (i) общие причины – условие или событие, являющееся причиной нескольких элементарных ошибок; (ii) взаимоисключающие исходные события – возникновение одного простого события заранее исключает другое простое событие; (iii) избыточное резервирование – при возникновении ошибки в работающем компоненте в работу включается резервный компонент, и избыточная конфигурация продолжает функционировать; (iv) компоненты, поддерживающие рабочую нагрузку, – ошибка в одном из компонентов увеличивает рабочую нагрузку на другие компоненты поддержки этой функции. С помощью байесовских сетей можно в явной форме представить зависимости между ошибками и по такой методике моделировать сложные системы, которые трудно поддаются моделированию по более привычным методикам [17].

7.4.2.1 Моделирование надежности с помощью байесовских сетей

Анализ надежности начинается с представления структуры системы в форме блок-схемы надежности. В этом представлении существуют две основные структуры: последовательные и параллельные компоненты

(см. рис. 7.19). Последовательная структура предполагает, что две компоненты должны правильно работать для корректного функционирования системы, другими словами, если в одной компоненте возникает ошибка, то нарушается работа всей системы (это соответствует вентилю *AND* в дереве ошибок). В параллельных структурах достаточно, чтобы одна из компонент работала правильно для корректного функционирования системы (вентиль *OR* в дереве ошибок).

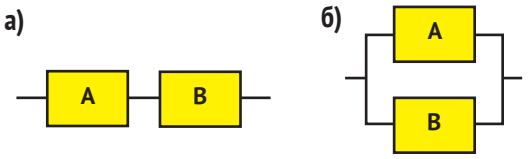


Рис. 7.19. Блок-схемы надежности для простых структур надежности с двумя компонентами: а) последовательная структура, б) параллельная структура

Описанные выше простые блок-схемы можно представить с помощью байесовской сети, показанной на рис. 7.20. Структура одинакова в обоих случаях, различаются только матрицы условных вероятностей. Матрицы условных вероятностей для обоих случаев показаны в табл. 7.4 и 7.5. Априорные вероятности основных компонент (A, B) представляют соответствующие коэффициенты ошибок в обоих вариантах. Таким образом, применяя вероятностный логический вывод в представлении байесовской сети, получаем коэффициент ошибок всей системы X.

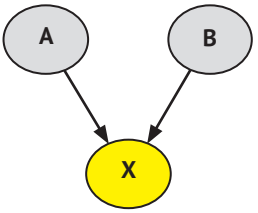


Рис. 7.20. Структура байесовской сети для двух простых блок-схем надежности, показанных на рис. 7.19

Таблица 7.4. Таблица условных вероятностей $P(X \mid A, B)$ для двух компонент с последовательной структурой

X	*A, B*	*A, ¬B*	*¬A, B*	*¬A, ¬B*
Success (Норма)	1	0	0	0
Failure (Ошибка)	0	1	1	1

A означает, что компонента *A* в рабочем состоянии, а ¬*A* – ошибка в компоненте.

Таблица 7.5. Таблица условных вероятностей $P(X \mid A, B)$ для двух компонент с параллельной структурой

X	*A, B*	*A, ¬B*	*¬A, B*	*¬A, ¬B*
Success (Норма)	1	1	1	0
Failure (Ошибка)	0	0	0	1

A означает, что компонента A в рабочем состоянии, а ¬*A* – ошибка в компоненте.

Представление в виде байесовской сети простых последовательных и параллельных вариантов можно напрямую обобщить для представления любой блок-схемы, которую можно преобразовать в множество последовательных и параллельных сочетаний компонентов, на практике представляющее большинство систем. Существуют некоторые структуры, которые невозможно разложить на элементарные последовательные и параллельные сочетания, например мост (bridge). Тем не менее также существует возможность моделирования таких вариантов с использованием байесовских сетей [17].

7.4.2.2 Моделирование зависимых критических ошибок

Главным преимуществом моделирования надежности с использованием байесовских сетей является возможность моделирования зависимых ошибок. Продемонстрируем это на примере системы с общими причинами ошибок.

Предположим, что система содержит две компоненты, на которые воздействуют три возможных источника ошибок. Источник S_1 воздействует на компоненту C_1, источник S_2 – на компоненту C_2, а источник S_3 – на обе компоненты (общая причина). Например, такой системой может быть электростанция с двумя подсистемами. Каждая подсистема содержит элементы, в которых могут возникать ошибки (сбои), но землетрясение может нарушить работу обеих подсистем. Модель байесовской сети для этого примера зависимых ошибок показана на рис. 7.21. В этой модели таблица условных вероятностей для всех трех некорневых узлов (C_1, C_2, X) равнозначна таблице условных вероятностей для последовательного сочетания компонент. X представляет коэффициент ошибок всей системы в целом. Этот коэффициент можно получить методом распространения вероятностей с учетом коэффициентов ошибок для трех источников ошибок.

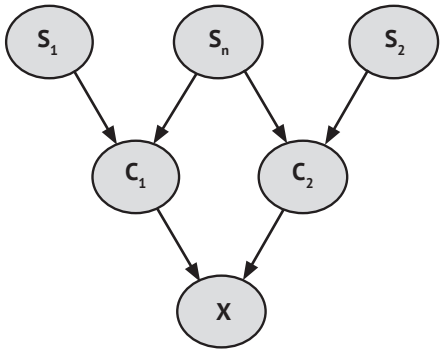


Рис. 7.21. Структура байесовской сети для системы с общими причинами ошибок

7.5 Материалы для дополнительного чтения

Введение в байесовские сети изложено в классической книге Джуды Перла (Judea Pearl) [14]. Другие широко известные книги по байесовским

сетям – [7, 12]. Недавнее описание БС с особым выделением моделирования и логического вывода приведено в [2], оно также включает анализ сложности для различных методик логического вывода. Книги, в которых больше внимания уделено практическим приложениям, – [8, 15]. Обзор канонических моделей представлен в [3]. Алгоритм дерева сочленений изначально был представлен в [9], а две основные архитектуры описаны в [6, 16]. Анализ алгоритма распространения с петлями можно найти в [11]. Методика логического вывода для непрерывных гауссовых переменных была описана в [14], а более обобщенный метод на основе усеченных экспоненциальных функций представлен в [10].

7.6 Задания и упражнения

1. Для байесовской сети на рис. 7.2 определить: а) контур каждой переменной; б) марковское ограждение каждой переменной; в) все отношения условной независимости, подразумеваемые структурой этой сети.
2. Вывести логически некоторые из отношений независимости в задании 1, используя аксиомы независимости.
3. Заполнить все таблицы условных вероятностей для байесовской сети на рис. 7.5, предполагая, что все переменные являются бинарными.
4. Исследовать модель Noisy AND и получить таблицы условных вероятностей для переменной с тремя причинами с вероятностями препятствования возникновению следствия, равными 0.05, 0.1 и 0.2 соответственно.
5. Рассмотреть пример распространения доверия в разделе 7.3.1, вычислить апостериорные вероятности для всех переменных с применением алгоритма распространения доверия, считая, что единственным свидетельством является $C = true$.
6. Повторить решение задания 5 с использованием процедуры исключения переменных.
7. Вычислить оценки апостериорных вероятностей в примере из раздела 7.3.1 при тех же условиях, что и в заданиях 5 и 6 ($C = true$), используя метод (алгоритм) логической выборки для различных количеств выборок (10, 20, ...), и сравнить полученные результаты, используя методику точного логического вывода.
8. Для байесовской сети на рис. 7.18: а) выполнить морализацию графа; б) выполнить триангуляцию графа; в) определить клики и

сформировать дерево сочленений; г) получить множества C, S и R для каждой клики в соответствии с алгоритмом дерева сочленений.

9. *** Разработать программу на основе процедуры (алгоритма) Bayes ball для демонстрации d-разделенности. Рассматривая структуру байесовской сети, пользователь выбирает два узла и подмножество разделенности. Программа должна найти все пути между этими двумя узлами, а затем определить, являются ли эти пути независимыми, с учетом подмножества разделенности, применяя процедуру Bayes ball. Показать в графической форме, проходит ли шар по пути полностью или блокируется.

10. *** Разработать обобщенную программу реализации алгоритма распространения доверия для полидеревьев с учетом дискретных переменных. Разработать версию с параллельными вычислениями этой программы с оптимальным распределением процессоров для эффективного распараллеливания алгоритма. Доработать обе версии этой программы для алгоритма распространения доверия с петлями.

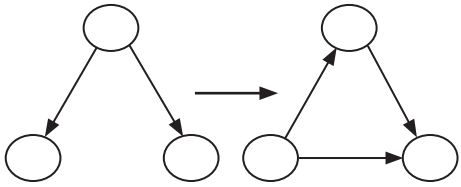
Ссылки на источники

1. Cooper, G. F. The computational complexity of probabilistic inference using Bayesian networks. Artif. Intell. 42, 393–405 (1990).
2. Darwiche, A. Modeling and Reasoning with Bayesian Networks. Cambridge University Press, New York (2009).
3. Díez, F. J., Druzdzel, M. J. Canonical probabilistic models for knowledge engineering. Technical Report CISIAD-06-01. Universidad Nacional de Educación a Distancia, Spain (2007).
4. Ibargüengoytia, P. H., Sucar, L. E., Vadera, S. A probabilistic model for sensor validation. In: Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence UAI-96, p. 332–339. Morgan Kaufmann Publishers Inc. (1996).
5. Ibargüengoytia, P. H., Vadera, S., Sucar, L. E. A probabilistic model for information validation. Br. Comput. J. 49(1), 113–126 (2006).
6. Jensen, F. V., Andersen, S. K. Approximations inBayesian belief universes for knowledge based systems. In: Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence UAI-90, p. 162–169. Elsevier, New York (1990).
7. Jensen, F. V. Bayesian Networks and Decision Graphs. Springer, New York (2001).
8. Korb, K. B., Nicholson, A. E. Bayesian Artificial Intelligence, 2nd edn. CRC Press, Boca Raton (2010).

9. Lauritzen, S., Spiegelhalter, D. J. Local computationswith probabilities on graphical structures and their application to expert systems. J. R. Stat. Soc. Ser. B. 50 (2), 157–224 (1988).
10. Moral, S., Rumi, R., Salmerón, A. Mixtures of truncated exponentials in hybrid Bayesian networks. Symb. Quant. Approaches Reason. Uncertain. 2143, 156–167 (2001).
11. Murphy, K. P., Weiss, Y., Jordan, M. Loopy belief propagation for approximate inference: an empirical study. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, p. 467–475. Morgan Kaufmann Publishers Inc. (1999).
12. Neapolitan, R. E. Probabilistic Reasoning in Expert Systems. Wiley, New York (1990).
13. Pearl, J. Fusion, propagation and structuring in belief networks. Artif. Intell. 29, 241–288 (1986).
14. Pearl, J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988).
15. Pourret, O., Naim, P., Marcot, B. (eds.) Bayesian Belief Networks: A Practical Guide to Applications. Wiley, New Jersey (2008).
16. Shenoy, P., Shafer, G. Axioms for probability and belief-function propagation. Uncertainty in Artificial Intelligence, vol. 4, p. 169–198. Elsevier, New York (1990).
17. Torres-Toledano, J. G., Sucar, L. E. Bayesian networks for reliability analysis of complex systems. In: Coelho, H. (ed.) IBERAMIA'98. Lecture Notes in Computer Science, vol. 1484, p. 195–206. Springer, Berlin (1998).

Глава 8

Байесовские сети: обучение



8.1 Введение

Обучение байесовской сети включает две составные части: обучение структуры и обучение параметров. Если структура известна, то обучение параметров заключается в определении оценок таблиц условной вероятности по имеющимся данным. Для обучения структуры существуют два основных типа методов: (i) глобальные методы на основе поиска и числовой оценки и (ii) локальные методы, использующие проверки условной независимости. В следующих разделах будут рассматриваться обе составные части обучения, начиная с обучения параметров.

8.2 Обучение параметров

Если в наличии имеется *достаточный* и полный объем данных для всех переменных и предполагается, что известна топология байесовской сети, то обучение параметров выполняется достаточно просто. Таблицу условных вероятностей для каждой переменной можно оценить по имеющимся данным на основе частоты появления каждого значения (или сочетания значений), получая при этом средство оценки *максимального правдоподобия* (maximum likelihood) параметров. Например, для оценки таблицы условных вероятностей для переменной B с учетом наличия у этой переменной двух родителей A, C:

$$P(B_i \mid A_j, C_k) \sim NB_iA_jC_k / NA_jC_k, \tag{8.1}$$

где $NB_iA_jC_k$ – число случаев в базе данных, в которых $B = B_i$, $A = A_j$ и $C = C_k$, а NA_jC_k – общее количество случаев, в которых $A = A_j$ и $C = C_k$.

8.2.1 Сглаживание

При оценке вероятностей по имеющимся данным иногда может возникать ситуация, в которой некоторое конкретное событие никогда не происходит в наборе данных. Из-за этого соответствующее значение вероятности становится равным нулю, что означает невозможный случай. Если вероятность этого события учитывается в процессе логического вывода, то результат вывода также будет равен нулю. Во многих случаях такая ситуация возникает из-за недостаточного объема данных, чтобы получить надежную и правильную оценку параметров, но не потому, что это событие действительно является невозможным.

Описанной выше ситуации можно избежать, если воспользоваться некоторым типом *сглаживания* (smoothing) для вероятностей, исключающим нулевые значения вероятностей. Существует несколько методик сглаживания, из которых наиболее часто применяемой и самой простой является *сглаживание Лапласа* (Laplacian smoothing).

Сглаживание Лапласа состоит в инициализации вероятностей равномерным распределением и последующим обновлением значений на основе имеющихся данных. Рассмотрим дискретную переменную X с k возможными значениями. Изначально каждое значение вероятности будет установлено как $P(x_i) = 1/k$. Затем с учетом набора данных из N элементов, в котором значение x_i встречается m раз, оценка вероятности этого значения вычисляется по следующей формуле:

$$P(x_i) = (1 + m) / (k + N). \tag{8.2}$$

8.2.2 Неопределенность параметров

Если объем данных недостаточен, что на практике встречается довольно часто, то мы имеем неопределенность параметров. Эту неопределенность можно смоделировать с использованием распределения вероятности второго порядка, и она может быть распространена в процессе логического вывода. Таким образом, мы получаем оценку неопределенности в полученных в результате вероятностях. Для бинарных переменных неопределенность параметров можно смоделировать, используя бета-распределение:

$$\beta(a, b) = \frac{(a + b + 1)!}{a!b!} x^a (1 - x)^b. \tag{8.3}$$

Для переменных с несколькими возможными значениями неопределенность параметров может быть представлена расширением бета-распределения, известным как распределение Дирихле.

Для варианта с бинарными переменными ожидаемое значение получаемого бета-распределения: $P(b_i) = (a + 1)/(a + b + 2)$, где a и b – параметры бета-распределения. Это представление также можно использовать при наличии экспертных оценок вероятностей. Параметры бета-распределения могут представлять меру (степени) достоверности (confidence) экспертных оценок, выражаемую изменением члена $a + b$ при одном и том же значении вероятности. Например:

- полная неосведомленность: $a = b = 0$;
- низкая достоверность: *малое* значение $a + b$ (10);
- средняя достоверность: *среднее* значение $a + b$ (100);
- высокая достоверность: *большое* значение $a + b$ (1000).

Этим представлением можно пользоваться для объединения экспертных оценок с данными. Например, для аппроксимации значения вероятности бинарной переменной b_i можно использовать следующую формулу:

$$P(b_i) = (k + a + 1) / (n + a + b + 2), \tag{8.4}$$

где a и $a + b$ представляют экспертную оценку, а k и n – вероятности, полученные по имеющимся данным (k – число появлений b_i в n выборках).

Например, предположим, что эксперт дает оценку 0.7 для определенного параметра, а экспериментальные данные предоставляют 40 положительных результатов из 100 выборок. Оценки этого параметра для различных степеней достоверности, назначаемых экспертной оценке, будут следующими:

- низкая достоверность: $(a + b = 10)$: $P(b_i) = \dfrac{(40 + 7 + 1)}{(100 + 10 + 2)} = 0.43$;
- средняя достоверность: $(a + b = 100)$: $P(b_i) = \dfrac{(40 + 70 + 1)}{(100 + 100 + 2)} = 0.55$;
- высокая достоверность: $(a + b = 1000)$: $P(b_i) = \dfrac{(40 + 700 + 1)}{(100 + 1000 + 2)} = 0.67$.

В первом случае здесь наблюдается преобладание влияния данных по сравнению с влиянием оценки, но в третьем случае вероятность ближе к экспертной оценке. Во втором случае обеспечивается определенный компромисс.

8.2.3 Недостаточный объем данных

Часто возникает другая ситуация – неполные данные. Здесь возможны два основных варианта:

- недостаточное количество значений (некоторые значения пропущены или потеряны) – в некоторых записях отсутствуют значения для одной или нескольких переменных;
- «исчезнувшие» узлы – переменные или множество переменных в модели, для которых вообще нет данных.

При работе с недостаточным объемом данных возможно несколько вариантов различных решений:

1) исключение записей с отсутствующими значениями;
2) принятие особенного «неизвестного» значения;
3) замена отсутствующего значения наиболее часто встречающимся значением (модой) для рассматриваемой переменной;
4) оценка отсутствующего значения на основе значений других переменных в соответствующей записи.

Первый и второй варианты решения приемлемы, если имеется достаточный объем данных, иначе можно потерять полезную информацию. В третьем варианте не рассматриваются другие переменные, в результате модель может быть искажена (может возникнуть предвзятость). Вообще говоря, самым лучшим вариантом является четвертый. В этом случае сначала обучаются параметры байесовской сети на основе полных записей, затем пополняется комплект данных и повторно оцениваются параметры с применением описанного ниже процесса для каждой записи с отсутствующими значениями:

1) присваивание значений всем известным переменным в текущей записи;
2) с помощью вероятностного логического вывода определение апостериорных вероятностей отсутствующих переменных;
3) присваивание каждой переменной значения с самой высокой апостериорной вероятностью;
4) добавление этой дополненной записи в базу данных и повторное вычисление оценок параметров.

В другом варианте описанного выше процесса вместо присваивания значения с наивысшей вероятностью присваивается «частный» вариант из каждого набора возможных значений для переменной пропорционально соответствующей апостериорной вероятности.

Для «исчезнувших» узлов методика оценки их параметров основана на EM-алгоритме.

8.2.3.1 «Исчезнувшие» узлы: EM-алгоритм

EM-алгоритм – это статистический метод, используемый для оценки параметров, если имеются ненаблюдаемые переменные. Процесс состоит из двух этапов, которые повторяются итеративно:

- этап E – отсутствующие значения данных оцениваются на основе текущих параметров;
- этап M – параметры обновляются на основе оцененных (на предыдущем этапе) данных.

Выполнение алгоритма начинается с инициализации отсутствующих параметров случайными значениями.

Для базы данных с одним или несколькими «исчезнувшими» узлами H_1, H_2, ..., H_k EM-алгоритм для оценки таблиц условных вероятностей этих узлов выполняется следующим образом:

1) определение таблиц условных вероятностей для всех полноценных переменных (значения этих переменных и всех их родителей находятся в базе данных) на основе механизма оценки максимального правдоподобия;
2) инициализация неизвестных параметров случайными значениями;
3) с учетом действительных параметров определение оценок значений «исчезнувших» узлов на основе известных переменных с помощью вероятностного логического вывода;
4) использование значений, оцененных на предыдущем шаге для «исчезнувших» узлов, для дополнения и/или обновления базы данных;
5) повторное выполнение оценки параметров для «исчезнувших» узлов с использованием обновленных данных;
6) повторение шагов 3–5 до сходимости (отсутствия значимых изменений параметров).

Этот EM-алгоритм оптимизирует неизвестные параметры и позволяет вычислить локальный максимум (окончательные оценки зависят от конкретной инициализации).

8.2.3.2 Пример

В этом разделе показано, как обрабатывать отсутствующие значения и «исчезнувшие» переменные, используя данные из примера *Golf*

(см. табл. 8.1). В этом наборе данных есть отсутствующие значения для переменной *Температура* (Temperature) (записи 1 и 9), кроме того, нет никакой информации о ветре (переменная *Ветер* (Wind)), который является «исчезнувшим» узлом. Сначала продемонстрируем, как добавляются отсутствующие значения для температуры, затем рассмотрим методику управления «исчезнувшим» узлом.

Таблица 8.1. Данные для примера *Golf* с отсутствующими значениями для переменной *Температура* и «исчезнувшей» переменной *Ветер*

Прогноз	Температура 1	Влажность	Ветер	Игра
Солнечно	xxx	Высокая	–	Нет
Солнечно	Высокая	Высокая	–	Нет
Облачно	Высокая	Высокая	–	Игра
Дождь	Средняя	Высокая	–	Игра
Дождь	Низкая	Нормальная	–	Игра
Дождь	Низкая	Нормальная	–	Нет
Облачно	Низкая	Нормальная	–	Игра
Солнечно	Средняя	Высокая	–	Нет
Солнечно	xxx	Нормальная	–	Игра
Дождь	Средняя	Нормальная	–	Игра
Солнечно	Средняя	Нормальная	–	Игра
Облачно	Средняя	Высокая	–	Игра
Облачно	Высокая	Нормальная	–	Игра
Дождь	Средняя	Высокая	–	Нет

Предположим, что обучается наивный байесовский классификатор (наивный байесовский классификатор – это частный тип байесовской сети) на основе доступных данных (12 полных записей без переменной *Ветер*), рассматривая переменную *Игра* (*Play*) как переменную класса, а все остальные переменные как атрибуты. Затем на основе этой модели можно оценить вероятность температуры в тех записях, где это значение отсутствует, с помощью вероятностного логического вывода с использованием значений других переменных в соответствующих записях как свидетельство. Таким образом:

Запись 1: $P(Температура \mid Солнечно, Высокая, Нет)$,

Запись 9: $P(Температура \mid Солнечно, Нормальная, Игра)$.

Затем можно выбрать в качестве значения температуры одно из значений с самой высокой апостериорной вероятностью и добавить отсутствующие значения, как показано в табл. 8.2.

Таблица 8.2. Данные для примера *Golf* после дополнения отсутствующих значений для переменной *Температура* и одной итерации EM-алгоритма для оценки значений переменной *Ветер*

Прогноз	Температура 1	Влажность	Ветер	Игра
Солнечно	Средняя	Высокая	Нет	Нет
Солнечно	Высокая	Высокая	Нет	Нет
Облачно	Высокая	Высокая	Нет	Игра
Дождь	Средняя	Высокая	Нет	Игра
Дождь	Низкая	Нормальная	Да	Игра
Дождь	Низкая	Нормальная	Да	Нет
Облачно	Низкая	Нормальная	Да	Игра
Солнечно	Средняя	Высокая	Нет	Нет
Солнечно	Средняя	Нормальная	Нет	Игра
Дождь	Средняя	Нормальная	Нет	Игра
Солнечно	Средняя	Нормальная	Да	Игра
Облачно	Средняя	Высокая	Да	Игра
Облачно	Высокая	Нормальная	Да	Игра
Дождь	Средняя	Высокая	Да	Нет

Для «исчезнувшего» узла *Ветер* (Wind) невозможно получить соответствующие таблицы условных вероятностей с помощью наивного байесовского классификатора $P(Ветер \mid Игра)$, так как полностью отсутствуют значения для переменной *Ветер*. Но можно применить EM-алгоритм с начальной инициализацией случайными значениями параметров для таблиц условных вероятностей. Например, это может быть равномерное распределение:

$$P(Ветер \mid Игра) = \begin{matrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{matrix}.$$

С учетом только что созданной таблицы условных вероятностей мы получили полную первоначальную модель для наивного байесовского классификатора и можем дать оценку вероятности ветреной погоды для каждой записи на основе значений других переменных в соответст-

вующей записи. Выбирая значение самой высокой вероятности в каждой записи, можно заполнить всю таблицу, как показано в табл. 8.2. На основе этой новой таблицы данных выполняется повторная оценка параметров, и вычисляется новая таблица условных вероятностей:

$$P(\textit{Ветер} \mid \textit{Игра}) = \begin{matrix} 0.60 & 0.44 \\ 0.40 & 0.56 \end{matrix}.$$

На этом завершается один цикл EM-алгоритма. Затем процесс повторяется до тех пор, пока все параметры в таблице условных вероятностей будут почти не отличаться от параметров, полученных на предыдущей итерации. В этот момент достигнута сходимость EM-алгоритма, и мы получили оценки отсутствующих параметров байесовской сети.

8.2.4 Дискретизация

Обычно байесовские сети работают с дискретными или номинальными значениями. Несмотря на то что существуют некоторые разработки для непрерывных переменных, они ограничены применением конкретных распределений, в частности использованием гауссовых переменных и линейных отношений. Другим вариантом включения непрерывных переменных в байесовские сети является дискретизация. Методы дискретизации могут быть (i) без учителя или (ii) с учителем.

8.2.4.1 Дискретизация без учителя

Методы дискретизации без учителя не рассматривают саму задачу, для которой предполагается использовать конкретную модель (например, задачу классификации), поэтому интервалы для каждой переменной определяются независимо. Существуют два основных типа методов дискретизации без учителя: с равными интервалами (по длине или ширине) и с равными наборами данных.

Методы с равными интервалами разделяют весь диапазон значений переменной $[X_{min}; X_{max}]$ на k равных элементов дискретизации, таких, что каждый элемент имеет размер $[X_{min}; X_{max}] / k$. Количество интервалов k обычно устанавливает пользователь.

Методы с равными наборами данных разделяют весь диапазон значений переменной на k интервалов, таких, что каждый интервал включает одинаковое число точек данных из тренировочного набора данных. Другими словами, если существует n точек данных, то каждый интервал будет содержать n/k точек данных. Это означает, что интервалы не обязательно имеют одинаковую ширину.

8.2.4.2 Дискретизация с учителем

Методы дискретизации с учителем рассматривают задачу, решаемую с помощью конкретной модели, поэтому дискретизация переменных вы-

полняется для оптимизации решения этой задачи, например для повышения точности классификации. Если рассматривать байесовскую сеть для классификации, т. е. байесовский классификатор, то метод дискретизации с учителем можно применить напрямую. Переменные непрерывных атрибутов дискретизируются в соответствии со значениями класса. Такой подход можно определить как задачу оптимизации.

Рассмотрим переменную атрибута X с диапазоном значений $[X_{min}; X_{max}]$ и переменную класса C с m значениями $c_1, c_2, \ldots, c_m$. При создании n тренировочных выборок, таких, что каждая из них содержит значение для C и X, задача состоит в определении оптимальной части диапазона значений X, такой, что точность классификатора становится максимальной. Это задача комбинаторики с высокой сложностью вычислений, которую можно решить с использованием описанного ниже процесса поиска:

1) генерация всех потенциально возможных делений на интервалы диапазона значений X, которые соответствуют значению в $[X_{min}; X_{max}]$, при изменении значения класса;
2) генерация первоначального множества из n интервалов на основе ранее определенных точек потенциально возможных делений;
3) проверка точности результатов байесовского классификатора (обычно на отдельном наборе данных, называемом набором валидации) в соответствии с текущим вариантом дискретизации;
4) улучшение дискретизации посредством дополнительного разделения какого-либо интервала или объединения двух интервалов;
5) повторение шагов 3 и 4 до тех пор, пока точность классификатора уже невозможно будет повысить или пока не будут выполнены некоторые другие условия завершения процесса дискретизации.

Можно использовать и другие методы поиска, в том числе самые простые, например алгоритм поиска восхождением к вершине, или более сложные методы, такие как *генетические алгоритмы*.

Описанный выше алгоритм не применяется для обобщенного варианта байесовской сети, которую можно использовать для прогнозирования различных переменных на основе разнообразных переменных свидетельств. Для таких случаев существует метод дискретизации с учителем [6], который дискретизирует непрерывные атрибуты во время обучения структуры байесовской сети. Этот метод основан на принципе *минимальной длины описания* (minimum description length – MDL), который описан в разделе 8.3.3. Для каждой непрерывной переменной количество интервалов определяется с учетом ее соседей в сети. Цель – минимизация (минимальной) длины описания (нахождение компромисса между точностью и сложностью модели) с использованием ал-

горитма поиска и метода тестирования по аналогии с процессом для байесовских классификаторов. Процедура повторяется итеративно для всех непрерывных переменных в сети.

8.3 Обучение структуры

Обучение структуры заключается в определении топологии байесовской сети по имеющимся данным. Это сложная задача, поскольку: (i) количество возможных структур огромно даже при небольшом числе переменных (существует гиперэкспоненциальная зависимость от числа переменных, например для 10 переменных количество возможных направленных ациклических графов имеет порядок 4×10^{18}), и (ii) требуется весьма большая база данных для вычисления качественных оценок статистических числовых характеристик, от которых зависят все методы.

Для конкретного варианта структуры в форме дерева существует метод, обеспечивающий создание наилучшего дерева. Для более общих вариантов было предложено несколько методов, которые можно разделить на два основных класса:

1) *глобальные методы*: [4, 5] выполняют эвристический поиск в пространстве структур сети, начиная с некоторой исходной структуры, и генерируют вариант структуры на каждом шаге. *Наилучшая* структура выбирается на основе числовой оценки, отображающей, насколько точно модель представляет имеющиеся данные. Наиболее часто используются числовые оценки байесовский информационный критерий (BIC) [4] и минимальная длина описания (MDL) [5];
2) *локальные методы*: основаны на вычислении оценок (не)зависимых связей между подмножествами переменных с учетом имеющихся данных для последующего определения структуры сети. Наиболее известный вариант этого метода – PC-алгоритм [10].

Оба класса методов позволяют получить приблизительно одинаковые результаты при достаточном объеме данных. Локальные методы обычно более чувствительны к ошибкам при малом количестве экземпляров (и выборок) данных, а глобальные методы, как правило, имеют бóльшую вычислительную сложность.

Далее будет рассматриваться алгоритм обучения дерева, разработанный Чоу (Chow) и Лю (Liu) [1], и его расширение для полидеревьев. Затем описываются методики обучения обобщенной структуры.

8.3.1 Обучение дерева

Чоу (Chow) и Лю (Liu) [1] разработали метод (алгоритм) для аппроксимации любого многомерного распределения вероятностей как про-

изведения распределений второго порядка, которое является основой для обучения байесовских сетей, имеющих форму дерева. Совместную вероятность n случайных переменных можно аппроксимировать следующим образом:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid X_{j(i)}), \tag{8.5}$$

где $X_{j(i)}$ – родитель X_i в дереве.

Задача заключается в определении наилучшего дерева, т. е. структуры дерева, которая лучше всего аппроксимирует действительное распределение. Мера точности аппроксимации основана на различии (разности) информации между действительным распределением (P) и аппроксимацией в форме дерева (P^*) и определяется следующей формулой:

$$DI(P, P^*) = \sum_{X} P(X) log(P(X)/P^*(X)). \tag{8.6}$$

Таким образом, теперь задача заключается в поиске дерева, которое минимизирует DI.

Взаимосвязанная информация между любой парой переменных определяется следующей формулой:

$$I(X_i, X_j) = \sum_{X_i, X_j} P(X_i, X_j) log(P(X_i, X_j)/P(X_i)P(X_j)). \tag{8.7}$$

С учетом структуры дерева для байесовской сети с переменными X_1, $X_2, \ldots, X_n$ определяется ее вес (weight) W как сумма взаимосвязанной информации дуг (пар переменных), из которых состоит это дерево:

$$W(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n-1} I(X_i, X_j), \tag{8.8}$$

где X_j – родитель узла X_i в дереве (дерево с n узлами содержит $n - 1$ дуг).

Можно показать [1], что минимизация DI равнозначна максимизации W. Следовательно, создание оптимального дерева равнозначно поиску *остовного дерева максимального веса* (maximum weight spanning tree) с использованием следующего алгоритма:

1) определение взаимосвязанной информации (I) между всеми парами переменных (для n переменных существует $n(n - 1)/2$ пар);
2) упорядочение значений взаимосвязанной информации в убывающем порядке;

3) выбор пары с максимальным значением I и соединение этих двух переменных дугой. Это позволит сформировать исходное дерево;
4) добавление пары со следующим максимальным значением I в сформированное дерево, если эта пара не создает цикл, иначе пропустить пару и перейти к следующей;
5) повторять шаг 4 до тех пор, пока все переменные не будут включены в дерево (n – 1 дуг).

Этот алгоритм позволяет получить скелет (как множество точек; skeleton) дерева, т. е. он не обеспечивает определение направления дуг в байесовской сети. Направления этих связей должны определяться с использованием внешней семантики (смысловых характеристик предметной области) или с использованием проверок зависимостей более высокого порядка (см. ниже).

Для демонстрации практического применения метода обучения дерева рассмотрим уже знакомый пример *golf* с 5 переменными: *игра* (play), *прогноз* (outlook), *влажность* (humidity), *температура* (temperature), *ветер* (wind). С учетом некоторых данных определяется взаимосвязанная информация, показанная в табл. 8.3.

Таблица 8.3. Взаимосвязанная информация в убывающем порядке для примера *golf*

Номер	Переменная 1	Переменная 2	Взаимосвязанная информация
1	Температура	Прогноз	0.2856
2	Игра	Прогноз	0.0743
3	Игра	Влажность	0.0456
4	Игра	Ветер	0.0074
5	Влажность	Прогноз	0.0060
6	Ветер	Температура	0.0052
7	Ветер	Прогноз	0.0017
8	Игра	Температура	0.0003
9	Влажность	Температура	0
10	Ветер	Влажность	0

В этом примере выбираются первые четыре пары (дуги) и создается дерево, показанное на рис. 8.1, в котором направления дуг были назначены произвольно.

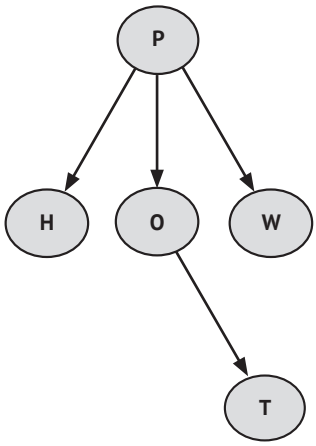


Рис. 8.1. Структура дерева, сформированная для примера *golf* (*P – игра* (play), *O – прогноз* (outlook), *H – влажность* (humidity), *T – температура* (temperature), *W – ветер* (wind)). Направления дуг назначены произвольно

8.3.2 Обучение полидерева

Ребане (Rebane) и Перл (Pearl) [9] разработали метод, который можно использовать для определения направления дуг в скелете дерева, а также в более общем случае – для обучения байесовской сети, имеющей форму полидерева (polytree). Алгоритм основан на проверках независимости троек переменных, и таким способом можно выделить сходящиеся структуры. Как только одна или несколько структур этого типа обнаружены в скелете, можно задать направление дополнительным дугам, применяя проверки независимости к соседним узлам. Но при этом не обязательно обеспечивается определение направления для всех дуг в дереве. Та же идея используется в PC-алгоритме для обучения обобщенных структур.

Алгоритм начинает работу со скелетом (неориентированной структурой), полученным с помощью алгоритма Чоу и Лю. Далее выполняется процесс обучения с назначением направления дуг и с использованием проверок независимости для троек переменных. При проверке трех переменных существуют три возможности:

1) последовательные дуги: $X \rightarrow Y \rightarrow Z$;
2) расходящиеся дуги: $X \leftarrow Y \rightarrow Z$;
3) сходящиеся дуги: $X \rightarrow Y \leftarrow Z$.

Первые два случая неразличимы при проверке статистической независимости, т. е. они равнозначны. В обоих вариантах X и Z независимы относительно Y. Третий случай отличается от первых двух, поскольку X и Z не являются независимыми относительно Y. Следовательно,

третий вариант можно использовать для определения направлений двух дуг, которые соединяют эти три переменные. Кроме того, можно применить этот элемент знаний для обучения (определения направления) других дуг, используя проверки независимости. С учетом данного принципа можно воспользоваться следующим алгоритмом для обучения полидеревьев:

1) создание скелета с использованием алгоритма Чоу и Лю;
2) итеративный проход по сети до тех пор, пока не будет сходящаяся тройка переменных. Переменная, в которой сходятся дуги, называется узлом с несколькими родителями (multi-parent node);
3) начиная с узла с несколькими родителями, определяются направления других дуг с использованием проверок независимости для троек переменных. Эта процедура продолжается до тех пор, пока не останется возможных вариантов (временная основа структуры);
4) шаги 2–3 повторяются до тех пор, пока не исчерпаются все возможности определения направлений дуг;
5) если какие-либо дуги остались ненаправленными, использовать внешние семантические (смысловые) правила для логического предположения об их направлении.

Для демонстрации работы этого алгоритма еще раз рассмотрим пример *golf* с уже созданным скелетом (неориентированной структурой). Предположим, что тройка переменных *H*, *P*, *W* относится к варианту сходящихся дуг. Тогда дуги будут направлены от *H* к *P* и от *W* к *P*. Далее зависимость между *H* и *W* измеряется с помощью отношения к *O* с учетом *P*. Если *H* и *W* независимы от *O* с учетом *P*, то должна существовать дуга, направленная от *P* к *O*. Наконец, проверяется отношение зависимости между *P* и *T* с учетом (относительно) *O*, и если *P* и *T* тоже признаны независимыми, то дуга направляется от *O* к *T*. На рис. 8.2 показана полученная в результате структура.

8.3.3 Методики поиска с оценкой

Применение описанных выше методов ограничивается только структурами деревьев и полидеревьев. В этом и следующем разделах будут рассматриваться методики обучения обобщенной структуры, начиная с самых общих методов.

Глобальные методы поиска наилучшей структуры основаны на глобальной метрике. То есть генерируются различные структуры и оцениваются по отношению к имеющимся данным с использованием некоторого метода числовой оценки. Существуют разнообразные варианты таких методов, но все они напрямую зависят от двух характеристик:

(i) меры соответствия между структурой и данными и (ii) метода поиска наилучшей структуры.

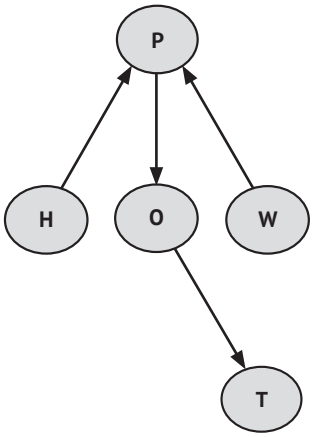


Рис. 8.2. Полидерево, полученное для примера *golf* с использованием алгоритма Ребане и Перла

8.3.3.1 Функции вычисления оценки

Существует несколько возможных мер соответствия, или функций вычисления оценки. Для функций вычисления оценки требуются два свойства [3]:

- возможность декомпозиции: функция вычисления оценки обладает возможностью декомпозиции, если значение, присваиваемое каждой структуре, можно выразить как сумму (в логарифмическом пространстве) локальных значений, которые зависят только от каждого конкретного узла и его родителей. Это важно для обоснования эффективности во время процесса поиска. Если учитывать это свойство, то при внесении локального изменения в структуру необходимо будет вычислить только соответствующую часть оценки;
- равнозначность (эквивалентность) оценки: функция вычисления оценки S обладает свойством равнозначности оценки, если она присваивает одно и то же значение всем направленным ациклическим графам, которые представлены одинаковым существенным графом. При таком подходе результат оценивания класса равнозначных структур будет одинаковым независимо от конкретного направленного ациклического графа, выбранного из этого класса. Структуры двух байесовских сетей соответствуют одному и тому же существенному графу, если они равнозначны с точки зрения отношений независимости, которые они представляют.

Ниже описываются некоторые наиболее часто используемые функции вычисления оценки, в том числе оценка максимального правдоподобия (maximum likelihood – ML), байесовский информационный критерий (Bayesian information criterion – BIC), байесовская оценка (Bayesian score – BD) и принцип минимальной длины описания (minimum description length – MDL).

Функция оценки максимального правдоподобия выбирает структуру, которая максимизирует вероятность данных D с учетом структуры G:

$$G^* = ArgMax_G\left[P(D \mid \Theta_G, G_i)\right], \tag{8.9}$$

где G_i – структура-кандидат, а Θ_G – соответствующий вектор параметров (вероятность каждой переменной с учетом ее родителей в оцениваемой конкретной структуре).

Непосредственное применение функции оценки максимального правдоподобия может привести к выбору весьма сложной сети, в которой обычно предполагается переподгонка (overfitting) данных (слабое обобщение), а кроме того, усложняет логический вывод. Поэтому требуется опция штрафования сложных моделей.

Наиболее часто используемой функцией вычисления оценки, которая включает компоненту штрафования, является байесовский информационный критерий, или просто BIC, определяемый следующей формулой:

$$BIC = logP(D \mid \Theta_G, G_i) - (d/2)logN, \tag{8.10}$$

где d – количество параметров в байесовской сети, а N – количество вариантов в имеющихся данных. Преимущество этой метрики – не требуется определение априорной вероятности, а кроме того, она связана с критерием минимальной длины описания (MDL), обеспечивая баланс между точностью и сложностью модели. Но, учитывая значительную штрафную компоненту за сложность модели, эта функция стремится выбирать слишком простые структуры.

Байесовские оценки

Другая метрика вычисляется на основе байесовской статистики и позволяет получить апостериорную вероятность структуры с учетом имеющихся данных по правилу Байеса:

$$P(G_i \mid D) = P(G_i)P(D \mid G_i) / P(D). \tag{8.11}$$

Учитывая, что $P(D)$ – постоянная величина, которая не зависит от рассматриваемой структуры, можно исключить этот элемент из метрики, чтобы получить байесовскую оценку, или просто BD:

$$BD = P(G_i)P(D \mid G_i). \tag{8.12}$$

$P(G_i)$ – это априорная вероятность модели. Она может быть определена экспертом или задана так, чтобы предпочтение отдавалось более простым структурам. Можно также установить для нее просто равномерное распределение.

Оценка BD является вариантом байесовской оценки, для которой приняты следующие предположения: (i) параметры независимы и имеют априорное распределение Дирихле; (ii) равнозначным структурам дается одинаковая оценка; (iii) экземпляры в выборках данных независимы и распределены одинаково (iid). При таких предположениях требуемые виртуальные счетчики (virtual count) для вычисления оценки могут быть определены следующим образом:

$$N_{ijk} = P(X_i = k, Pa(X_i) = j \mid G_i, \Theta_G) \times N'. \tag{8.13}$$

Это оцениваемый счетчик для конкретной конфигурации: $X_i = k$ при $Pa(X_i) = j$, N' – размер равнозначной выборки.

Предполагая, что гиперпараметры этих априорных распределений одинаковы, можно еще больше упростить вычисление байесовской оценки и получить так называемую метрику K2[16]. Для этой оценки существует возможность декомпозиции, и она вычисляется для каждой переменной X_i с учетом родителей этой переменной $Pa(X_i)$:

$$S_i = \prod_{j=1}^{q_i} [(r_i - 1)! / (N_{ij} + r_i - 1)!] \prod_{k=1}^{r_i} \alpha_{ijk}!, \tag{8.14}$$

где r_i – количество значений X_i, q_i – количество возможных конфигураций для родителей переменной X_i, α_{ijk} – число случаев в базе данных, когда $X_i = k$ и $Pa(X_i) = j$, N_{ij} – число случаев в базе данных, когда $Pa(X_i) = j$.

Эта метрика предоставляет практический вариант решения для вычисления оценки для байесовской сети. Другой широко применяемый вариант на основе принципа минимальной длины описания (MDL) описан в следующем подразделе.

Принцип минимальной длины описания (MDL)

Метрика (мера) MDL создает баланс между точностью и сложностью модели. Точность оценивается посредством измерения взаимосвязанной информации между атрибутами и классом. Сложность модели оценивается количеством параметров. Постоянная величина α в интервале [0, 1] используется для регулирования веса каждой характеристики, т. е. для баланса точности и сложности. Мера соответствия определяется следующей формулой:

$$MC = \alpha(W/W_{max}) + (1 - \alpha)(1 - L/L_{max}), \tag{8.15}$$

[16] K2 – это алгоритм для обучения байесовских сетей, описываемый ниже.

где W представляет точность модели, а L – сложность. W_{max} и L_{max} представляют максимальную точность и максимальную сложность соответственно. Для определения максимальных значений обычно устанавливается верхняя граница по количеству родителей, которое разрешено иметь каждому узлу. Значение $\alpha = 0.5$ обеспечивает одинаковую значимость точности и сложности модели, тогда как значение, близкое к нулю, придает бóльшую значимость сложности, а значение, близкое к единице, делает более важной точность.

Сложность определяется по числу параметров, требуемых для представления модели, и может быть вычислена по следующей формуле:

$$L = S_i\left[k_i \log_2 n + d(S_i - 1)F_i\right], \tag{8.16}$$

где n – количество узлов, k – количество родителей каждого узла, S_i – среднее число значений для каждой переменной, F_i – среднее число значений для каждой родительской переменной, d – количество битов на каждый параметр.

Оценку точности можно вычислить на основе «веса» каждого узла – такой подход аналогичен применению весовых коэффициентов в методике обучения деревьев. В данном случае вес каждого узла X_i оценивается на основе взаимосвязанной информации, получаемой от его родителей $Pa(X_i)$:

$$w(X_i, Pa(X_i)) = \sum_{x_i} P(X_i, Pa(X_i)) log\left[P(X_i, Pa(X_i)) / P(X_i)P(Pa(X_i))\right]. \tag{8.17}$$

Суммарный вес (точность) определяется суммой весов для каждого узла:

$$W = \sum_i w(X_i, Pa(X_i)). \tag{8.18}$$

8.3.3.2 Алгоритмы поиска

После определения меры (метрики) соответствия для выбираемых структур необходимо установить метод выбора «наилучшей» структуры из возможных вариантов. Поскольку количество возможных структур находится в экспоненциальной зависимости от количества переменных, практически невозможно вычислять оценку каждой структуры. Для ограничения числа оцениваемых структур выполняется эвристический поиск. При этом можно применять несколько различных методов поиска. Наиболее часто используется методика поиска *восхождением к вершине* (hill climbing): сначала берется простая структура дерева, которая улучшается до тех пор, пока не будет получена «наилучшая» структура. Простейший жадный алгоритм поиска наилучшей структуры приведен ниже:

1) генерация исходной структуры дерева;
2) вычисление меры (метрики) соответствия для исходной структуры;
3) добавление и/или изменение направления дуг в текущей структуре;
4) вычисление меры (метрики) соответствия для новой структуры;
5) если соответствие повышается (улучшается), то сохранить изменения, иначе – вернуться к предыдущей структуре;
6) повторять шаги 3–5 до тех пор, пока дальнейшее улучшение станет невозможным.

Приведенный выше алгоритм не обеспечивает нахождение оптимальной структуры, так как при его выполнении возможно достижение лишь локального максимума. На рис. 8.3 показана процедура поиска для примера *golf*, которая начинается со структуры дерева, улучшающейся до получения конечной структуры. Для получения наилучшей структуры могут применяться и другие методы поиска, такие как генетические алгоритмы, алгоритм имитации отжига, методы двунаправленного поиска и т. д.

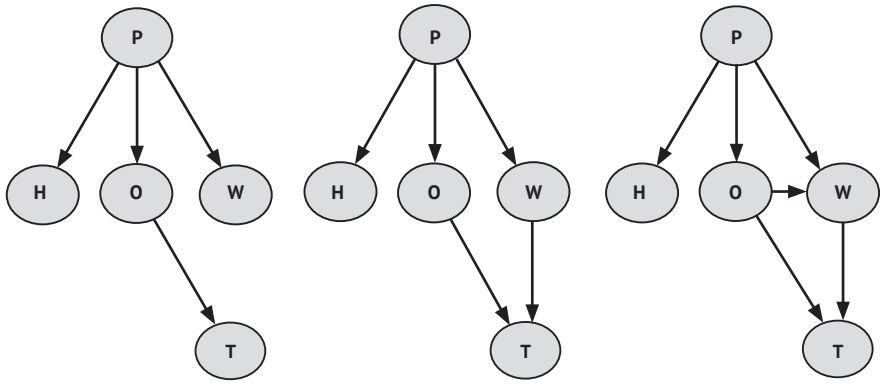


Рис. 8.3. Несколько шагов процедуры обучения структуры в примере *golf*, начинающейся со структуры простого дерева (слева) и завершающейся получением конечной структуры (справа)

Другой способ сокращения количества оцениваемых возможных структур – установление порядка переменных, метод, известный как причинное упорядочение (casual ordering). При таком подходе дуги в сети ограничиваются в соответствии с определенным порядком, т. е. может существовать дуга NO от вершины V_j к вершине V_i, если $j > i$ в соответствии с установленным порядком. Алгоритм K2 [4] использует преимущества этого подхода, предоставляя эффективный и широко применяемый метод обучения байесовских сетей.

8.3.3.3 Алгоритм K2

При использовании причинного упорядочения для всех переменных обучение до наилучшей структуры равнозначно независимому выбору наилучшего множества родителей для каждого узла. Изначально все переменные не имеют родителей. Затем алгоритм K2 постепенно добавляет родителей для каждого узла, пока эта операция увеличивает глобальную оценку. Когда добавление родителей к любому узлу не увеличивает оценку, поиск останавливается. Кроме того, причинное упорядочение обеспечивает отсутствие циклов в графе.

В алгоритме 8.1 показана общая схема процедуры K2. Входными данными для этого алгоритма является множество из n переменных с причинным упорядочением $X_1, X_2, \ldots, X_n$, база данных D, содержащая m вариантов, и, как правило, критерий ограничения максимального количества родителей каждой переменной u. Выходные данные представлены в виде множества родителей $Pa(X_i)$ для каждой переменной, которое определяет структуру сети. Начиная с первой переменной по заданному порядку, алгоритм проверяет всех возможных родителей этой переменной, которые пока еще не были добавлены в структуру, и включает родителя, обеспечивающего максимальное приращение общей оценки сети. Процедура повторяется для каждого узла сети до тех пор, пока не останется родителей, способных увеличить общую оценку.

Алгоритм 8.1. Алгоритм K2

Требования: Множество переменных X с причинным упорядочением, функция вычисления оценки S, максимальное количество родителей u
Обеспечиваемый результат: Множество родителей для каждой переменной $Pa(X_i)$

```
for i = 1 to n do
  oldScore = S(i, Pa(Xi))
  incrementScore = true
  Pa(Xi) =
  Pa(Xi) = Ø
  while incrementScore and |Pa(Xi)| < u do
    let Z – узел в множестве Predesessors(Xi) – Pa(Xi), который максимизирует S
    newScore = S(i, Pa(Xi) ∪ Z)
    if newScore > oldScore then
      oldScore = newScore
      Pa(Xi) = Pa(Xi) ∪ Z
    else
      incrementScore = false
    end if
  end while
```

```
end for
return Pa(X_1),Pa(X_2),...,Pa(X_n)
```

8.3.4 Методики проверки независимости

Другой класс методик обучения структуры применяет локальный, а не глобальный подход, используемый методиками числовой оценки и поиска. Основная идея локального подхода – применение проверок независимости ко множествам переменных для воспроизведения структуры байесовской сети. Примером этого типа методик является алгоритм Чоу и Лю для деревьев. В следующем разделе будет представлен метод обучения обобщенных структур – PC-алгоритм.

8.3.4.1 PC-алгоритм

PC-алгоритм [10] сначала формирует скелет (первоначальный неориентированный граф) байесовской сети, затем определяет ориентацию ребер этого графа.

Для определения скелета алгоритм начинает работу с полносвязного неориентированного графа и определяет условную независимость каждой пары переменных с учетом некоторого подмножества других переменных. Для этого алгоритм предполагает, что существует процедура, способная определить, являются ли две переменные X, Y независимыми с учетом подмножества переменных S, т. е. $I(X, Y \mid S)$. Другим вариантом этой процедуры является условная мера перекрестной энтропии. Если эта мера ниже некоторого порогового значения, установленного в соответствии с определенным уровнем достоверности, то ребро между оцениваемой парой значений удаляется. Такие проверки последовательно выполняются для всех пар переменных в графе.

На втором этапе устанавливается направление ребер в графе на основе проверок условной независимости между тройками переменных. Выполняется поиск в графе внутренних структур в форме $X - Y - Z$, таких, что не существует ребра $X - Y$. Если X, Y не являются независимыми относительно Z, то алгоритм устанавливает направление ребер, создающее V-образную структуру $X \rightarrow Z \leftarrow Y$. После того как все V-образные структуры определены, алгоритм пытается установить направление других ребер на основе проверок независимости, избегая при этом циклов. В алгоритме 8.2 показана общая схема основной процедуры[17].

Алгоритм 8.2. PC-алгоритм

```
Требования: Множество переменных X, процедура проверки независимости I
Обеспечиваемый результат: Направленный ациклический граф G
```

[17] ADJ(X) – множество узлов, смежных (соседних) с узлом X в графе.

```
1:  Инициализация полного неориентированного графа G'
2:  i=0
3:  repeat
4:    for X ∈ X do
5:      for Y ∈ ADJ(X) do
6:        for S ⊆ ADJ(X) – {Y}, | S | = i do
7:          if I(X, Y | S) then
8:            Удалить ребро X – Y из графа G'
9:          end if
10:       end for
11:     end for
12:   end for
13:   i = i + 1
14:   until | ADJ(X) | ⩽ i, ∀X
15:   Определить направления ребер в графе G'
16:   return G
```

Если множество независимостей является достоверным (заслуживающим доверия) для графа[18], а проверки независимости совершенны, то алгоритм создает граф, равнозначный исходному, т. е. структуре байесовской сети, которая сгенерировала эти данные.

Методики проверки независимости полагаются на наличие достаточного объема данных для получения качественных оценок по результатам проверок независимости. Алгоритмы поиска и оценки становятся более надежными с увеличением размера набора данных, но на производительность этих алгоритмов также влияет размер и качество доступных данных. При недостаточном объеме данных другим вариантом является объединение экспертных знаний и имеющихся данных.

8.4 Объединение экспертных знаний и имеющихся данных

При доступности экспертных знаний в предметной области можно объединить их с алгоритмами обучения, чтобы улучшить качество модели. При обучении параметров можно объединить имеющиеся данные и экспертные оценки на основе бета-распределения и распределения Дирихле, как описано в разделе 8.2.

При обучении структуры существуют две основные методики объединения экспертных знаний и имеющихся данных:

[18] Условие достоверности (faithfulness condition) можно интерпретировать как предположение о том, что отношения условной независимости существуют благодаря причинной структуре, а не из-за случайных свойств значений параметров [10].

- использование экспертных знаний как ограничений для сокращения пространства поиска для обучающих алгоритмов;
- изначальное принятие структуры, предложенной экспертом, и использование данных для проверки правильности и улучшения этой структуры.

Существует несколько способов использования экспертных знаний для поддержки выполнения алгоритмов обучения структуры:

1) определение порядка переменных (причинное упорядочивание), такого, что дуга от X_i к X_j может существовать, только если X_j следует после X_i в соответствии с установленным порядком;
2) определение ограничений в отношении направленных дуг, которые обязательно должны существовать между двумя переменными, т. е. $X_i \rightarrow X_j$;
3) определение ограничений по отношению к дуге между двумя переменными, которая может быть направлена по-другому;
4) определение ограничений в отношении пар переменных, которые не связаны прямым отношением, т. е. не должно существовать дуги между X_i и X_j;
5) различные сочетания описанных выше ограничений.

Несколько вариантов методик обоих типов, алгоритмов поиска и оценки и проверок независимости включают описанные выше ограничения.

Пример методики второго типа был представлен в главе 4 с применением алгоритма улучшения структуры. Эта методика начинает работу со структуры наивного байесовского классификатора, которая улучшается посредством исключения, объединения и/или вставки переменных. Этот принцип можно распространить и на обобщенные структуры байесовских сетей, в особенности на байесовские сети, имеющие структуру дерева.

8.5 Приложения

Существует много предметных областей, в которых применялись и применяются методы обучения байесовских сетей для лучшего понимания конкретной предметной области или для прогнозирования на основе частичных наблюдений. Примеры: медицина, финансовая сфера, промышленность, мониторинг окружающей среды и многие другие. В следующем разделе рассматривается пример моделирования загрязнения воздуха в Мехико-сити.

8.5.1 Модель загрязнения воздуха в Мехико-сити

Качество воздуха в Мехико-сити представляет собой серьезную проблему. Уровень загрязнения воздуха в Мехико один из самых высоких в мире с высоким средним ежедневным уровнем выброса нескольких основных загрязняющих веществ, таких как углеводороды, оксиды азота, монооксид углерода и т. п. Загрязнение в основном происходит из-за выхлопных газов транспорта и выбросов в атмосферу промышленных предприятий. При воздействии на основные загрязняющие вещества солнечного света происходят химические реакции с выделением разнообразных вторичных загрязняющих веществ, и уровень озона становится наиболее важной характеристикой. Кроме проблем со здоровьем, причинами которых может стать загрязнение воздуха, уровень озона считается показателем качества воздуха в городских районах.

Наблюдение за качеством воздуха выполняется на 25 пунктах в Мехико-сити, при этом пять из этих пунктов проводят наиболее полные исследования. В каждом из этих пяти пунктов измеряются девять переменных, в том числе направление и скорость ветра, температура (воздуха), относительная влажность, концентрация диоксида серы (сернистого газа), монооксида углерода, диоксида азота и озона. Измерения производятся каждую минуту 24 часа в сутки и усредняются по каждому часу.

Важно иметь возможность прогнозирования уровня загрязнения на несколько часов вперед или даже на следующий день по нескольким причинам:

1) обеспечение принятия срочных экстренных мер, если уровень загрязнения прогнозируется выше определенного порогового значения;
2) помощь промышленным предприятиям в разработке предварительных планов на случай чрезвычайных обстоятельств с целью минимизации расходов на принятие экстренных мер;
3) оценка (прогноз) уровня загрязнения в районе, где не выполняются измерения;
4) предварительные меры защиты в некоторых местах, например в школах, с целью снижения вероятности угроз здоровью из-за высокого уровня загрязнения.

В Мехико-сити уровень концентрации озона используется как глобальный показатель качества воздуха в различных частях города. Уровни концентрации озона определяются в единицах IMECA (Mexican air quality index). Важно спрогнозировать уровень концентрации озона на следующий день или, как минимум, на несколько следующих часов, используя другие переменные, измеряемые в различных пунктах.

Полезно знать зависимости между различными измеряемыми переменными и особенно их влияние на концентрацию озона. Это обеспечивает лучшее понимание задачи в целом и дает несколько потенциальных преимуществ:

- определение факторов, которые являются более важными для уровня концентрации озона в Мехико-сити;
- упрощение задачи вычисления оценки, если учитывать только важную информацию;
- выявление самых важных основных причин загрязнения воздуха в Мехико-сити – это может помочь при будущем планировании мер по снижению уровня загрязнения.

Начнем с применения алгоритма обучения для получения исходной структуры наблюдаемого явления [11]. Для этого рассматриваются 47 переменных: 9 переменных измерений для каждого из 5 главных пунктов плюс время суток и месяц, в которые были зафиксированы эти измерения. Было использовано около 400 случайных выборок и применен алгоритм Чоу и Лю для получения структуры дерева, которая наилучшим образом аппроксимирует распределение данных. Структура байесовской сети в форме дерева показана на рис. 8.4.

Затем принимается, что концентрация озона в одном пункте (Педрегаль) неизвестна, поэтому вычисляется ее оценка, прогнозируемая на один час вперед с использованием результатов других измерений. Таким образом, создается гипотетическая переменная *ozone-Pedregal* (O3_T), которая рассматривается как корень в вероятностном дереве, как показано на рис. 8.4. Из этой исходной структуры можно получить представление о значимости или влиянии других переменных для определения оценки переменной *ozone-Pedregal*. Самые близкие узлы к корню являются наиболее важными, а более удаленные узлы менее важны.

В этом примере можно видеть, что существуют три переменные (*ozone-Merced*, *ozone-Xalostoc* и скорость ветра в пункте Педрегаль), которые оказывают наибольшее воздействие на переменную *ozone-Pedregal*. Более того, если структура дерева является качественной аппроксимацией «действительной» структуры, то эти три узла обеспечивают независимость переменной *ozone-Pedregal* от остальных переменных (см. рис. 8.4). Таким образом, в качестве первой проверки этой структуры определяется оценка переменной *ozone-Pedregal* с использованием только этих трех переменных. Выполняются два эксперимента: (1) оценка переменной *ozone-Pedregal* с использованием 100 случайных выборок из тренировочных данных, (2) оценка переменной *ozone-Pedregal* с использованием 100 других выборок из отдельных данных, не используемых для тренировки. Мы наблюдаем, что даже с тремя параметрами оценки получаются достаточно качественными. Для тренировочных данных средняя ошибка

(абсолютная разность между действительной и оцениваемой (прогнозируемой) концентрацией озона) равна 11 IMECA, или 12%, а для данных, не используемых для тренировки, – 26 IMECA, или 22%.

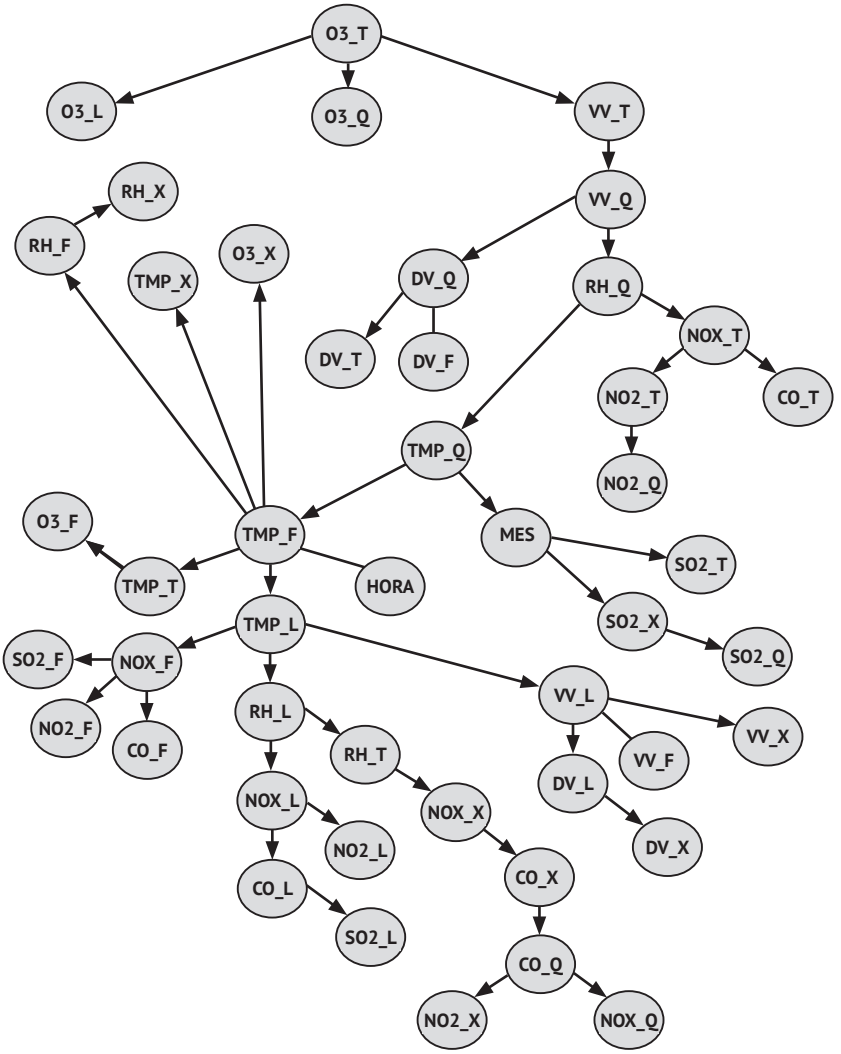


Рис. 8.4. Байесовская сеть в форме дерева, которое представляет наблюдения за концентрацией озона в пяти пунктах в Мехико-сити. Узлы представляют 47 переменных в соответствии со следующей системой условных обозначений. Для измеряемых переменных каждое имя образуется из двух частей «измерение–пункт» с использованием перечисленных ниже сокращений. Сокращения для измерений: O3 – озон, SO2 – диоксид серы, CO – монооксид углерода, NO2 – диоксид азота, NOX – оксиды азота, VV – скорость ветра, DV – направление ветра, TMP – температура, RH – относительная влажность. Сокращения названий пунктов наблюдения: T – Педрегаль, F – Тланепантла, Q – Мерсед, L – Халосток, X – Серро де ла Эстрелла. Другие две переменные соответствуют времени выполнения измерений: HORA – время, MES – месяц

В полученной структуре сделано интересное наблюдение: переменная *ozone-Pedregal* (расположение на юге города) зависит в основном от трех переменных: *ozone-Merced* и *ozone-Xalostoc* (расположение в центре и на севере города) и скорость ветра в Педрегале. Иначе говоря, загрязнение воздуха в южной части города зависит в основном от уровня загрязнения в центре и в северной части города (где более интенсивное транспортное движение и больше промышленных предприятий) и от скорости ветра, который переносит загрязняющие вещества с севера на юг. Это явление было уже известно и ранее, но оно автоматически обнаружено при обучении байесовской сети. Таким же образом могут быть обнаружены и другие, не столь известные связи, которые могут оказаться полезными для принятия решений по регулированию уровня загрязнения воздуха и принятию экстренных мер.

8.6 Материалы для дополнительного чтения

Основная книга по обучению байесовских сетей – [7]. Хекерман (Heckerman) [4] предлагает полное руководство по обучению байесовских сетей. Алгоритмы обучения дерева и полидерева описаны в [8]. Общее введение в обучение байесовских сетей с точки зрения статистики приведено в [10]. Анализ разнообразных функций вычисления оценки представлен в [3].

8.7 Задания и упражнения

1. В приведенной ниже табл. 8.4 содержатся исходные данные для примера *golf* с применением числовых значений для некоторых переменных. Выполнить дискретизацию этих переменных с использованием трех интервалов для каждой переменной: а) с равными интервалами (равной шириной) дискретизации, б) с равными наборами данных.

Таблица 8.4

Прогноз	Температура	Влажность	Ветер	Игра
Солнечно	19	Высокая	5	Нет
Солнечно	25	Высокая	3	Нет
Облачно	26	Высокая	3	Игра
Дождь	17	Высокая	6	Игра
Дождь	11	Нормальная	15	Игра
Дождь	7	Нормальная	17	Нет
Облачно	8	Нормальная	11	Игра

Прогноз	Температура	Влажность	Ветер	Игра
Солнечно	20	Высокая	7	Нет
Солнечно	19	Нормальная	1	Игра
Дождь	22	Нормальная	5	Игра
Солнечно	21	Нормальная	20	Игра
Облачно	22	Высокая	18	Игра
Облачно	28	Нормальная	16	Игра
Дождь	18	Высокая	Да	3

2. Используя дискретизированные данные из задания 1, сформировать таблицы условных вероятностей для структуры байесовской сети, показанной на рис. 8.2.
3. Продолжить выполнение EM-алгоритма для примера в разделе 8.2.3.2 до полной сходимости. Показать итоговую таблицу условных вероятностей *P*(*Ветер* | *Игра*) и итоговую таблицу данных.
4. На основе данных для примера *golf*, приведенных в табл. 8.2, обучить скелет дерева байесовской сети с использованием алгоритма Чоу и Лю.
5. Определить направления дуг в скелете из задания 4, применяя метод обучения полидерева.
6. На основе набора данных из задания 5 (табл. 8.2) обучить байесовскую сеть, применяя PC-алгоритм. Использовать меру (метрику) перекрестной энтропии (рассматривалась в главе 2) для проверки условной независимости.
7. Используя набор данных, приведенных ниже в табл. 8.5: а) обучить наивный байесовский классификатор, рассматривая *C* как класс; б) обучить байесовскую сеть со структурой дерева и закрепить направления дуг, рассматривая *C* как коренной узел; в) сравнить структуры обеих моделей.

Таблица 8.5

A1	A2	A3	C
0	0	0	0
0	1	1	1
0	1	0	1
0	0	1	1

A1	A2	A3	C
0	0	0	0
0	1	1	0
1	1	0	1
0	0	0	1
0	1	0	0
0	1	1	0

8. Для набора данных из задания 7 (табл. 8.5) использовать метод сглаживания Лапласа для получения таблиц условных вероятностей для обеих моделей: наивного байесовского классификатора и дерева байесовской сети. Сравнить полученные таблицы с таблицами условных вероятностей без сглаживания.
9. *** Разработать программу реализации алгоритма обучения полидерева. Применить эту программу к данным примера *golf* и сравнить полученный результат с результатами, полученными при выполнении задания 2.
10. *** Разработать программу реализации алгоритма обучения байесовской сети по имеющимся данным с использованием метода поиска и оценки на основе функции вычисления оценки MDL и другой вариант на основе проверок независимости (PC-алгоритм). Применить оба варианта к разнообразным наборам данных и сравнить результаты.

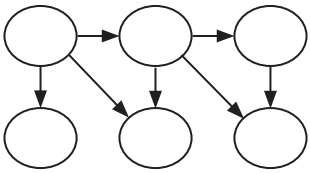
Ссылки на источники

1. Chow, C. K., Liu, C. N. Approximating discrete probability distributionswith dependence trees. IEEE Trans. Inf. Theory 14, 462–467 (1968).
2. Cooper, G. F., Herskovitz, E. A Bayesian method for the induction of probabilistic networks from data. Mach. Learn. 9 (4), 309–348 (1992).
3. De Campos, L. M. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. J. Mach. Learn. Res. 7, 2149–2187 (2006).
4. Heckerman, D. A Tutorial on Learning with Bayesian Networks. Innovations in Bayesian Networks, p. 33–82. Springer, Netherlands (2008).
5. Lam, W., Bacchus, F. Learning Bayesian belief networks: an approach based on the MDL principle. Comput. Intell. 10, 269–293 (1994).

6. Martínez, M., Sucar, L. E. Learning an optimal naive Bayes classifier. In: 18th International Conference on Pattern Recognition (ICPR), vol. 3, p. 1236–1239 (2006).
7. Neapolitan, R. E. Learning Bayesian Networks. Prentice Hall, New Jersey (2004).
8. Pearl, J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988).
9. Rebane, G., Pearl, J. The recovery of causal poly-trees from statistical data. In: Kanal, Laveen N., Levitt, Tod S., Lemmer, John F. (eds.) Uncertainty in Artificial Intelligence, p. 175–182 (1987).
10. Spirtes, P., Glymour, C., Scheines, R. Causation, Prediction, and Search. Springer, Berlin (1993).
11. Sucar, L. E., Ruiz-Suarez, J. C. Forecasting air pollution with causal probabilistic networks. In: Barnett, V., Turkman, K. F. (eds.) Statistics for the Environment 3: Statistical Aspects of Pollution, p. 185–197. Wiley, Chichester (2007).

Глава 9

Динамические и временны́е байесовские сети



9.1 Введение

Байесовские сети обычно представляют состояние конкретного явления в некоторый момент времени. Но во многих приложениях необходимо представить развитие рассматриваемого процесса во времени, т. е. показать, как различные переменные изменяются со временем. Такое представление также называют временно́й последовательностью (time series).

Существуют два типа моделей байесовских сетей для динамических процессов: основанные на состояниях (state) и основанные на событиях (event). Модели на основе состояний представляют состояние каждой переменной в дискретные интервалы времени, поэтому такие сети состоят из последовательности *интервалов (квантов) времени* (time slices), при этом каждый интервал времени определяет значение каждой переменной в момент времени t. Такие модели известны как динамические байесовские сети. Модели на основе событий представляют изменения состояния каждой переменной состояния. Затем каждой временно́й переменной ставится в соответствие время, в которое произошло изменение состояния. Этот тип моделей называют *сетями событий* (event networks) или *временны́ми сетями* (temporal networks).

В данной главе рассматриваются оба типа – динамические байесовские сети и временны́е сети, включая их представление, логический вывод и обучение.

9.2 Динамические байесовские сети

Динамические байесовские сети (dynamic Bayesian networks) представляют собой расширение байесовских сетей для моделирования динамических процессов. Динамическая байесовская сеть состоит из последовательности *интервалов (квантов) времени* (time slices), которые представляют состояние всех переменных в конкретный момент времени t, т. е. своего рода мгновенный снимок процесса, развивающегося во времени. Для каждого интервала времени определяется структура зависимости между переменными в этом интервале времени. Такая структура называется *базовой сетью* (base network). Обычно предполагается, что эта структура повторяется для всех интервалов времени (исключая первый интервал, для которого структура может отличаться). Кроме того, существуют ребра между переменными из различных интервалов с направлениями, заданными по времени, определяющие сеть переходов (transition network). Обычно для динамических байесовских сетей существует следующее ограничение: направленные связи могут существовать только между последовательными интервалами времени. Такая структура известна как марковская модель первого порядка. Но, вообще говоря, это ограничение не является обязательным. Пример динамической байесовской сети с 3 переменными и 4 интервалами времени показан на рис. 9.1.

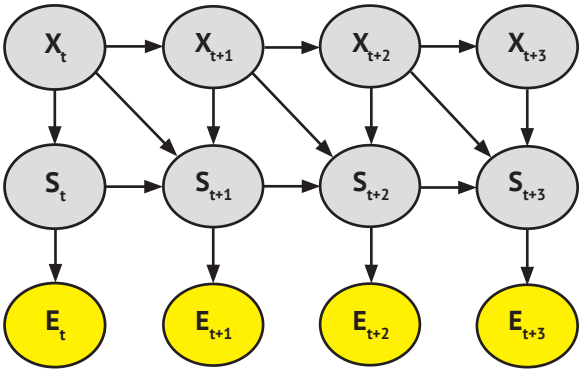


Рис. 9.1. Пример динамической байесовской сети с 3 переменными и 4 интервалами времени. В этом примере базовой структурой (сетью) является $X \rightarrow S \rightarrow E$, которая повторяется во всех 4 интервалах времени

Большинство динамических байесовских сетей, применяемых на практике, соблюдают следующие условия:

- марковская модель первого порядка – переменные состояния в момент времени t зависят только от переменных состояния в момент времени $t - 1$ (и от других переменных в момент времени t);

- стабильный процесс – структура и параметры модели не изменяются во времени.

Динамические байесовские сети можно рассматривать как обобщение марковских цепей и скрытых марковских моделей. Марковская цепь является простейшей динамической байесовской сетью, где в каждом интервале времени существует только одна переменная X_i, на которую напрямую воздействует лишь переменная из предшествующего интервала времени. В этом случае совместное распределение можно записать в виде следующей формулы:

$$P(X_1, X_2, \ldots, X_T) = P(X_1)P(X_2 \mid X_1)\ldots P(X_T \mid X_{T-1}). \tag{9.1}$$

Скрытая марковская модель содержит две переменные в каждом интервале времени: первую называют *переменной состояния* (state variable) S, вторую – *переменной наблюдения* (observation variable) Y. Обычно предполагается, что переменная S_t зависит только от переменной S_{t-1}, а переменная Y_t зависит лишь от S_t. Таким образом, совместную вероятность можно определить в виде следующего произведения:

$$P(\{S_{1:T}, Y_{1:T}\}) = P(S_1)P(Y_1 \mid S_1)\prod_{t=2}^{T} P(S_t \mid S_{t-1})P(Y_t \mid S_t). \tag{9.2}$$

Марковские цепи и скрытые марковские модели являются частными случаями динамических байесовских сетей, которые в общем случае могут содержать N переменных в каждом интервале времени, с любой базовой структурой и структурой переходов. Другим частным случаем динамических байесовских сетей являются фильтры Калмана (Kalman filters), которые также содержат одну переменную состояния и одну переменную наблюдения, но обе переменные являются непрерывными. Простые фильтры Калмана предполагают использование гауссовых распределений и линейных функций для моделей переходов и наблюдений.

9.2.1 Логический вывод

Существует несколько классов логического вывода, которые можно выполнять в динамических байесовских сетях. Ниже кратко определены основные типы логического вывода, где **X** – множество ненаблюдаемых (скрытых) переменных, **Y** – множество наблюдаемых переменных [9]:

- *фильтрация* (filtering) – прогноз следующего состояния на основе прошлых наблюдений: $P(X_{t+1} \mid Y_{1:t})$;
- *прогнозирование* (prediction) – прогноз будущих состояний на основе прошлых наблюдений: $P(X_{t+n} \mid Y_{1:t})$;

- *сглаживание* (smoothing) – оценка текущего состояния на основе прошлых и будущих наблюдений (полезно для обучения): $P(X_t \mid Y_{1:T})$;
- *декодирование* (decoding) – поиск наиболее правдоподобной (вероятной) последовательности скрытых переменных с учетом наблюдений: $ArgMax(X_{1:T})P(X_{1:T} \mid Y_{1:T})$.

Эффективные методы логического вывода разработаны для конкретных типов моделей, например для скрытых марковских моделей [10] (см. главу 5). Но для более сложных моделей логический вывод становится весьма трудновыполнимым с точки зрения вычислений. В таких случаях можно применить метод аппроксимации на основе сэмплирования, например метод Монте-Карло с использованием марковских цепей [6]. Широко распространенным методом аппроксимации является *многочастичный фильтр* (particle filter), который аппроксимирует распределение вероятностей состояния (состояние доверия) на множестве взвешенных *частиц* (particles) или элементов выборки [9].

9.2.2 Обучение

Как и в случае байесовских сетей, при обучении динамических байесовских сетей подразумевается два аспекта: (i) обучение структуры, или определение топологии графа, и (ii) обучение параметров, или определение таблиц условных вероятностей для каждой переменной. Дополнительно можно рассмотреть два варианта с точки зрения наблюдаемости переменных: а) полная наблюдаемость, когда существуют данные для всех переменных, и б) частичная наблюдаемость, когда некоторые переменные являются ненаблюдаемыми, или скрытыми, либо если отсутствуют данные для некоторых переменных. Существуют четыре основных варианта обучения динамических байесовских сетей, показанных в табл. 9.1.

Таблица 9.1. Обучение динамических байесовских сетей: 4 основных варианта

Структура	Наблюдаемость	Метод
Известна	Полная	Оценка максимального правдоподобия
Известна	Частичная	EM-алгоритм
Неизвестна	Полная	Поиск (глобальный) или проверки (локальные)
Неизвестна	Частичная	EM-алгоритм и глобальные или локальные методы

Для всех вариантов можно применять расширения методов обучения параметров и структуры для байесовских сетей, которые рассматривались в главе 8. Одно из таких расширений описывается ниже,

рассматривается вариант с неизвестной структурой и полной наблюдаемостью.

Предполагая, что динамическая байесовская сеть является стабильной (не изменяющейся во времени), можно считать, что модель определяется двумя структурами: (i) базовой структурой и (ii) структурой переходов. Таким образом, можно разделить процесс обучения динамической байесовской сети на два этапа: на первом обучается базовая структура, затем с учетом обученной базовой структуры обучается структура переходов, как показано на рис. 9.2.

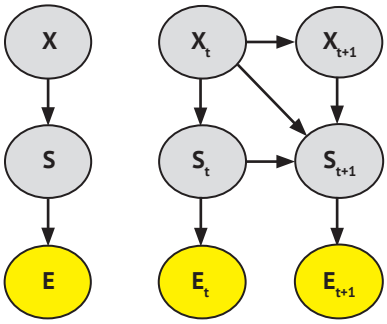


Рис. 9.2. Обучение динамической байесовской сети: сначала определяется базовая структура (слева), затем – структура переходов (справа)

Для обучения базовой структуры можно использовать все доступные данные для каждой переменной, не обращая внимания на временну́ю информацию. Такой подход равнозначен обучению обычной байесовской сети, поэтому можно применять любой метод, используемый для обучения байесовских сетей (см. главу 8).

Для обучения сети переходов рассматривается временна́я информация, в частности данные для всех переменных в двух смежных интервалах времени $\mathbf{X}_t$ и $\mathbf{X}_{t+1}$. Учитывая базовую структуру, можно затем обучить (установить) зависимости между переменными в момент времени t и в момент времени t+1 (предполагая, что рассматривается марковская модель первого порядка) и определить ограничения для направлений ребер: из прошлого в будущее.

Выше упрощенно, в общих чертах были описаны два этапа обучения динамической байесовской сети, но существует несколько вариантов реализации этой идеи, которые были разработаны ранее (см. раздел материалов для дополнительного чтения).

Помимо динамических байесовских сетей были разработаны также некоторые альтернативные представления байесовских сетей для описания процессов, протекающих во времени. Примером другого типа временно́й байесовской сети являются *сети событий* (event networks) [1, 5].

9.3 Временны́е сети событий

Временны́е сети событий (temporal event networks) – это другой тип динамических байесовских сетей для моделирования динамических процессов. Во временно́й сети событий узел представляет время возникновения некоторого события или изменения состояния конкретной переменной в противоположность узлу динамической байесовской сети, представляющему значение состояния переменной в определенный момент времени. Для некоторых задач, в которых происходит несколько изменений состояния в интересующий нас интервал времени, сети событий обеспечивают простое и более эффективное представление, но для других приложений, таких как мониторинг и фильтрация, в большей степени подходят динамические байесовские сети.

В настоящее время разработано и предложено несколько вариантов временны́х сетей событий, таких как сети времени (time nets) и *байесовские сети с временны́ми узлами* (temporal nodes Bayesian networks). В дальнейшем в этом разделе основное внимание будет уделено байесовским сетям с временны́ми узлами.

9.3.1 Байесовские сети с временны́ми узлами

Байесовские сети с временны́ми узлами [1, 5] состоят из множества временны́х узлов (temporal nodes). Временны́е узлы соединяются ребрами, при этом каждое ребро представляет причинно-следственную связь (отношение) между временны́ми узлами. В рассматриваемом интервале времени происходит не более одного изменения состояния каждой переменной (временно́го узла). Значение, принимаемое переменной, представляет интервал, в котором произошло событие. Время дискретизируется в виде конечного числа интервалов, при этом допускается различное количество интервалов и их различная продолжительность для каждого узла (множественная детализация). Каждый интервал, определяемый для узла-потомка, представляет возможные задержки между возникновением события (причина) одного из его родителей и соответствующим событием потомка (следствие). Некоторые временны́е узлы не имеют временны́х интервалов, они соответствуют моментальным узлам (instantaneous nodes). Корневые узлы являются моментальными по определению [1]. Формальное определение байесовских сетей с временны́ми узлами приведено ниже.

Байесовская сеть с временны́ми узлами определяется как пара $B = (G, \theta)$. G – направленный ациклический граф: $G = (\mathbf{V}, \mathbf{E})$. Граф G состоит из $\mathbf{V}$ – множества временны́х и моментальных узлов и $\mathbf{E}$ – множества ребер между узлами. Компонента θ соответствует множеству парамет-

ров, количественно определяющих сеть. Множество Θ содержит значения $\Theta_{vi} = P(v_i \mid Pa(v_i))$ для каждого $v_i \in \mathbf{V}$, где $Pa(v_i)$ представляет множество родителей узла v_i в графе G.

Временной узел v_i определяется множеством состояний **S**, а каждое состояние определяется упорядоченной парой $S = (\lambda, \tau)$, где λ – значение случайной переменной, а $\tau = [a, b]$ – соответствующий интервал с начальным значением a и конечным значением b. Эти значения соответствуют интервалу времени, в который произошло событие изменения состояния. Кроме того, каждый временной узел содержит дополнительное состояние по умолчанию (без изменений) $s = (\text{«no change»}, \emptyset)$, с которым не связан какой-либо интервал времени. Если узел не содержит интервалов, определенных для какого-либо из его состояний, то он именуется моментальным узлом. Приведенный ниже пример взят из [1].

Пример 9.1. Предположим, что во время $t = 0$ произошло дорожно-транспортное происшествие, т. е. столкновение. Этот тип ДТП можно классифицировать как *серьезное (тяжелое)* (severe), *средней тяжести* (moderate) или *легкое* (mild). Для упрощения модели будем считать, что существуют только два немедленных последствия для участника столкновения: травма головы (head injury) и внутреннее кровотечение (internal bleeding). Травма головы может повредить мозг, а травма грудной клетки может привести к внутреннему кровотечению. Это все моментальные события, которые могут стать причинами последующих изменений, например последствиями события «травма головы» могут стать расширенные зрачки и нестабильные основные показатели состояния организма. Предположим, что мы собрали информацию о ДТП, произошедших в некотором конкретном городе. Из этой информации ясно, что существует строгая причинная связь между серьезностью (тяжестью) ДТП и немедленным воздействием на состояние участника ДТП (пациента). Кроме того, врач-эксперт в предметной области терапии предоставил важную временну́ю информацию: при травме головы возникает опухоль мозга, и если не провести соответствующее обследование, то это приводит к расширению зрачков в течение 0–60 минут. Если начинается внутреннее кровотечение, то объем крови в кровеносной системе начнет уменьшаться, что приводит к дестабилизации основных показателей состояния организма. Время вероятной дестабилизации основных показателей состояния организма зависит от тяжести травмы, вызвавшей кровотечение: если кровопотеря велика, то время дестабилизации – от 0 до 15 минут, если кровопотеря незначительная, то время дестабилизации – от 15 до 45 минут.

> Травма головы также приводит к дестабилизации основных показателей состояния организма: проходит от 0 до 15 минут, прежде чем состояние становится нестабильным.

Представление в виде графа байесовской сети с временны́ми узлами для примера ДТП показано на рис. 9.3. В этой модели представлены три моментальных узла: Collision (столкновение), Head Injury (травма головы) и Internal Bleeding (внутреннее кровотечение). Эти события генерируют последующие изменения, которые не являются немедленными: Dilated Pupils (расширенные зрачки) и нестабильные Vital Signs (основные показатели состояния организма), которые зависят от тяжести столкновения, следовательно, имеют связанные с ним интервалы времени. Эта байесовская сеть с временны́ми узлами содержит только 5 узлов (для сравнения отметим, что равнозначная динамическая байесовская сеть потребовала бы создания 25 узлов).

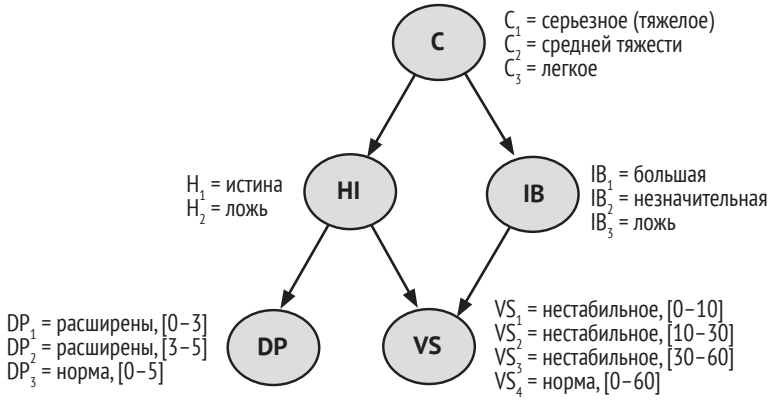


Рис. 9.3. Байесовская сеть с временны́ми узлами для примера ДТП:
C = столкновение, HI = травма головы, IB = внутреннее кровотечение,
DP = расширенные зрачки, VS = основные показатели состояния организма

В байесовских сетях с временны́ми узлами каждая переменная представляет событие или изменение состояния. Таким образом, требуется только один экземпляр (или несколько экземпляров) каждой переменной, если предположить, что происходит только одно изменение (или малое количество изменений) состояния переменной в рассматриваемом интервале времени. Не нужны никакие копии модели, не делается никаких предположений о марковской природе процесса. Байесовские сети с временны́ми узлами могут работать при множественной детализации, так как число и размер интервалов времени для каждого узла могут быть различными.

9.3.1.1 Логический вывод

Байесовские сети с временны́ми узлами позволяют обосновать вероятность возникновения определенных событий для диагностики (т. е. для поиска наиболее вероятной причины временно́го события) или для прогнозирования (т. е. для определения вероятных будущих событий, которые произойдут после наступления некоторого определенного события). Для этого можно применить стандартные методики распространения вероятностей в стандартных байесовских сетях (см. главу 7). Но с учетом того, что байесовские сети с временны́ми узлами представляют относительные времена между событиями, варианты прогнозирования и диагностики должны различаться для выполнения вероятностного логического вывода:

- *прогнозирование* – в том случае, когда как минимум один из корневых (моментальных) узлов байесовской сети с временны́ми узлами является частью свидетельства, временна́я связь для такой модели фиксируется, и процедура распространения вероятностей может выполняться напрямую, позволяя получить апостериорную вероятность последующих событий (вероятность для каждого интервала времени каждого немоментального временно́го узла). Рассмотрим пример на рис. 9.3: если известна степень тяжести столкновения (например, средняя), то с помощью процедуры распространения вероятности мы получаем апостериорные вероятности события *Dilated Pupils* (*расширенные зрачки*), происходящего в течение каждого интервала времени и не происходящего вообще (вариант *false*). Апостериорная вероятность других временны́х и моментальных узлов определяется подобным образом. Следует отметить, что интервалы времени для временны́х узлов будут определяться по отношению к возникновению события *Collision* (*столкновение*), которому соответствует время $t = 0$;
- *диагностика* – в том случае, когда моментальные узлы неизвестны, а свидетельство определяется только для временны́х узлов, необходимо рассматривать несколько сценариев (scenarios), так как неизвестно, какой интервал присвоить свидетельству при отсутствии конкретной ссылки на время. В этом случае необходимо учитывать все n возможных интервалов для рассматриваемого временно́го узла, выполняя логический вывод n раз, по одному для каждого интервала. Результаты каждого сценария должны обрабатываться до тех пор, пока не появится дополнительное свидетельство, такое как возникновение другого события, которое позволяет исключить некоторые сценарии. Рассматривая пример ДТП на рис. 9.3, предположим, что прибывает бригада

экстренной медицинской помощи и обнаруживает, что у пострадавшего расширены зрачки (*Pupils Dilated*). Так как время ДТП неизвестно, то свидетельство обязательно должно использоваться во всех трех интервалах времени для этой переменной, создавая три возможных сценария. Если в дальнейшем определяется точное время ДТП, то соответствующий сценарий сохраняется, а остальные исключаются.

9.3.1.2 Обучение

При обучении байесовской сети с временны́ми узлами рассматриваются три компонента этого процесса: (i) обучение интервалов времени для временны́х узлов, (ii) обучение структуры модели и (iii) обучение параметров модели. Поскольку эти три компонента взаимосвязаны, требуется итеративная процедура, которая выполняет обучение для получения начальной оценки для одного (или нескольких) из этих компонентов, а затем итеративно улучшает эти начальные оценки. Ниже будет представлен алгоритм обучения трех компонентов байесовской сети с временны́ми узлами [7].

Алгоритм полагает, что корневые узлы являются моментальными узлами, и позволяет получить конечное число неперекрывающихся интервалов для каждого временно́го узла. Алгоритм использует отрезки времени (задержки) между родительскими событиями и текущим узлом как входные данные для обучения создаваемых интервалов. При таком нисходящем методе алгоритм обеспечивает достижение локального максимума применительно к прогнозируемой оценке. Алгоритм изначально предполагает, что структура сети известна, а в дальнейшем обучающая компонента включается в структуру.

Алгоритм обучения (известный как LIPS) [7] для байесовских сетей с временны́ми узлами описан ниже.

1. В первую очередь выполняется первоначальная дискретизация временны́х переменных, например с использованием метода дискретизации по равным интервалам (равной ширины). Этот процесс позволяет получить начальную аппроксимацию интервалов для всех временны́х узлов.
2. Затем выполняется обучение структуры для стандартной байесовской сети. Часто используется алгоритм обучения K2 [3] (см. главу 8) для получения исходной структуры и соответствующих параметров.
3. Алгоритм обучения интервалов улучшает качество (точность) интервалов для каждого временно́го узла методом кластеризации. Для этого используется информация о конфигурации родитель-

ских узлов. Для получения множества интервалов применяется гауссова модель смеси (Gaussian mixture model)[19] как алгоритм кластеризации для временны́х данных. Теоретически каждый кластер соответствует некоторому интервалу времени. Интервалы определяются по среднему значению и стандартному отклонению кластеров. Алгоритм позволяет получить различные множества интервалов, которые объединяются и комбинируются, этот процесс генерирует различные множества интервалов, которые будут оцениваться с учетом точности прогнозирования. Алгоритм применяет две методики отсечения, чтобы исключать множества интервалов, которые могут оказаться бесполезными, а также для поддержания невысокой сложности для байесовской сети с временны́ми узлами. Наилучшее множество интервалов (которое, возможно, не будет получено на самом первом шаге выполнения) для каждого временно́го узла выбирается на основе точности прогнозирования. Если временной узел в качестве родителей имеет другие временные узлы, то конфигурации родительских узлов изначально неизвестны. Для решения этой задачи интервалы последовательно выбираются по нисходящей методике в соответствии со структурой байесовской сети с временны́ми узлами.

4. На завершающем этапе параметры (таблицы условных вероятностей) обновляются в соответствии с новым множеством интервалов для каждого временно́го узла.

Затем этот алгоритм выполняет итерационные циклы между обучением структуры и обучением интервалов.

Рассмотрим процесс определения интервалов для временно́го узла *Dilated Pupils* (DP) (расширенные зрачки) на рис. 9.3 (интервалы в рассматриваемом здесь примере отличаются от показанных на рис. 9.3). Можно видеть, что родительский узел (*Head Injury* (травма головы)) имеет две конфигурации: *true* (истина) и *false* (ложь). Следовательно, временны́е данные узла *Dilated Pupils* разделяются на две части, по одной для каждой конфигурации родительского узла. Затем для каждой части применяется первая аппроксимация шага обучения интервалов алгоритма, описанного выше. EM-алгоритм используется для получения гауссовых моделей смеси с параметрами 1, 2 и 3 как количеством кластеров. Это дает шесть различных множеств интервалов, как показано в табл. 9.2. Затем каждое множество интервалов оценивается с точки зрения эффективности прогнозирования для получения числовой оценки его качества, и выбирается множество интервалов с наилучшей числовой оценкой.

[19] Не следует путать со смешанной моделью (mixed model). – *Прим. перев.*

Таблица 9.2. Первоначальные множества интервалов, полученные для узла *Dilated Pupils*

Часть	Интервалы
Head Injury (травма головы) = true	[11–35]
	[11–27] [32–53]
	[8–21] [25–32] [45–59]
Head Injury (травма головы) = false	[3–48]
	[0–19] [39–62]
	[0–14] [28–40] [47–56]

Для каждой части создаются три множества интервалов.

Далее демонстрируется полный пример выполнения алгоритма обучения байесовской сети с временны́ми узлами с учетом имеющихся данных для примера ДТП, показанного на рис. 9.3. Предварительно предполагается, что у нас есть данные о других ДТП, которые близки к данным, показанным в верхней части табл. 9.3. Первые три столбца содержат качественные (категорийные) данные, а в двух последних столбцах приведены временны́е данные, которые представляют возникновение этих событий после ДТП (столкновения). Эти два столбца должны соответствовать временны́м узлам байесовской сети с временны́ми узлами. Выполнение начинается с применения дискретизации с равными интервалами (равной шириной) к числовым данным, при этом должны получиться результаты, близкие к данным, представленным в нижней части табл. 9.3.

Таблица 9.3. Данные, собранные для обучения байесовской сети с временны́ми узлами, показанной на рис. 9.3.

Collision (столкновение, ДТП)	Head Injury (травма головы)	Internal Bleeding (внутреннее кровотечение)	Dilated Pupils (расширенные зрачки)	Vital Signs (основные показатели состояния организма)
Severe (тяжелое)	True	Gross (сильное)	14	20
Moderate (средней тяжести)	True	Gross (сильное)	25	25
Mild (легкое)	False	False	–	–
…	…	…	…	…
Severe (тяжелое)	True	Gross (сильное)	[10–20]	[15–30]

Collision (столкновение, ДТП)	Head Injury (травма головы)	Internal Bleeding (внутреннее кровотечение)	Dilated Pupils (расширенные зрачки)	Vital Signs (основные показатели состояния организма)
Moderate (средней тяжести)	True	Gross (сильное)	[20–30]	[15–30]
Mild (легкое)	False	False	–	–
…	…	…	…	…

Исходные данные в верхней части таблицы показывают время возникновения временны́х событий. В нижней части таблицы представлены временны́е данные после первоначальной дискретизации. Для переменных *Dilated Pupils* (*расширенные зрачки*) и *Vital Signs* (*основные показатели состояния организма*) данные представляют время в минутах после столкновения.

Используя дискретизированные данные, можно применить алгоритм обучения структуры (например, K2) с частичным упорядочением временны́х событий: {Collision}, {Head Injury, Internal Bleeding} и {Dilated Pupils, Vital Signs}. Теперь получена исходная байесовская сеть с временны́ми узлами, но принятые в ней интервалы слишком упрощены, поэтому можно переходить на этап обучения интервалов для улучшения качества первоначальных интервалов времени. Этот процесс обучения позволяет получить байесовскую сеть с временны́ми узлами, весьма похожую на сеть, показанную на рис. 9.3.

9.4 Приложения

Практическое применение моделей динамических байесовских сетей будет показано в двух предметных областях. Сначала рассматривается использование динамических байесовских сетей для динамического распознавания жестов. Затем будет представлен пример применения временны́х сетей событий для прогнозирования вариантов мутаций ВИЧ.

9.4.1 Динамические байесовские сети: распознавание жестов

Динамические байесовские сети предоставляют вариант, отличный от скрытых марковских моделей, для распознавания жестов в динамическом режиме. Они используют преимущество большей гибкости с точки зрения структуры моделей. В этом разделе рассматривается конкретный тип ДБС, известный как классификатор на основе *динамической байесовской сети* (dynamic Bayesian network classifier) [2]. Как и в скрытых марковских моделях, в классификаторе на основе динамической байесовской сети имеется скрытая переменная состояния для

каждого момента времени S_t, а переменная наблюдения разделяется на m атрибутов $A_t^1, \ldots, A_t^m$, при этом предполагается, что эти атрибуты условно независимы относительно S_t. Таким образом, базовая структура для классификатора на основе динамической байесовской сети имеет структуру звезды со связями, направленными от S_t к каждому атрибуту A_t^i (см. рис. 9.4).

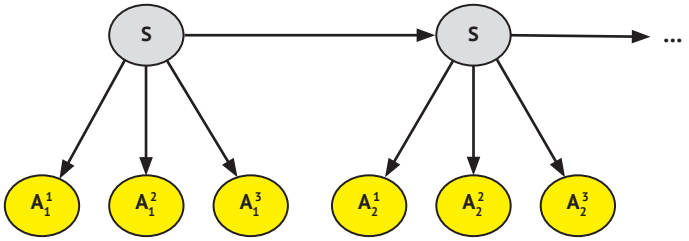


Рис. 9.4. Представление классификатора на основе динамической байесовской сети в форме графа с тремя атрибутами, воспроизведенными два раза

Совместная вероятность классификатора на основе динамической байесовской сети может быть выражена следующей формулой:

$$P\left(\{S_{1:T}, \mathbf{A}_{1:T}\}\right) = P(S_1)\left[\prod_{m=1}^{M} P(A_1^m \mid S_1)\right]\prod_{t=2}^{T} P(S_t \mid S_{t-1})\left[\prod_{m=1}^{M} P(A_t^m \mid S_t)\right], \qquad (9.3)$$

где $\mathbf{A} = A_1, \ldots, A_m$.

Отличием от совместной вероятности в скрытой марковской модели является то, что вместо $P(A_t \mid S_t)$ в формулу включено произведение по каждому атрибуту с учетом его состояния $\prod_{m=1}^{M} P(A_t^m \mid S_t)$.

Обучение параметров и классификация в классификаторах на основе динамической байесовской сети выполняется практически так же, как в скрытых марковских моделях, с использованием модифицированных версий алгоритмов Баума–Велша (Baum-Welch) и Forward соответственно.

9.4.1.1 Распознавание жестов с помощью классификатора на основе динамической байесовской сети

Классификатор на основе динамической байесовской сети применялся для распознавания различных жестов руки для подачи команд мобильному роботу. Рассматривался набор из 9 различных жестов. Ключевой кадр для каждого типа жеста показан на рис. 9.5. Изначально рука человека, выполняющего жест, обнаруживалась, и ее движение отслеживалось с использованием видеокамеры и специализированного программного обеспечения компьютерного зрения. Прямоугольник аппроксимирует положение руки в каждом изображении (кадре) видеопоследовательности, а из этих прямоугольников извлекается набор

признаков, которые являются наблюдениями для классификаторов на основе динамической байесовской сети.

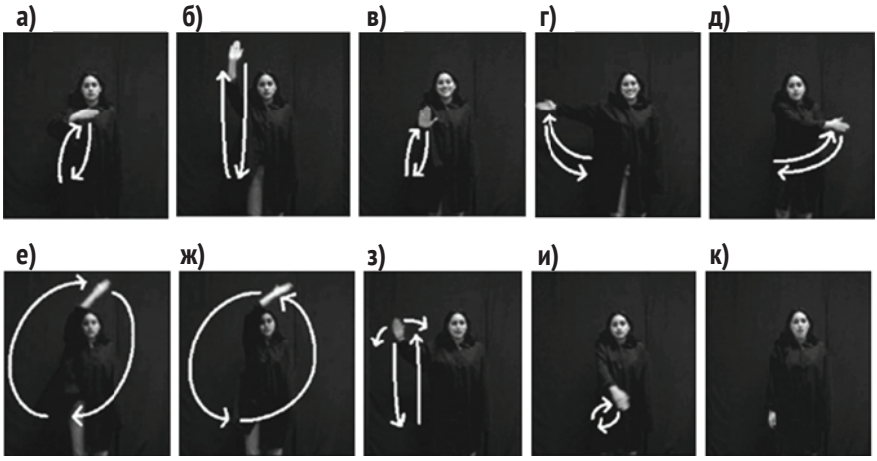


Рис. 9.5. Типы жестов, рассматриваемые в экспериментах: а) come (идти), б) attention (внимание), в) stop (остановиться), г) right (направо), д) left (налево), е) turn left (повернуться налево), ж) turn right (повернуться направо), з) waving-hand (волнообразные движения руки), и) pointing (указание (направления, места и т. п.)), к) начальное и конечное положения руки для каждого жеста

Признаки включают информацию о движении и положении (руки), в общей сложности семь атрибутов: (i) три признака для описания движения руки и (ii) четыре признака для описания положения руки. Признаки движения руки – Δarea или изменения в области движения руки, Δx и Δy – изменения положения руки в XY-плоскости изображения. Сочетание этих трех атрибутов позволяет оценить перемещение руки в декартовом пространстве *XYZ*. Каждый из признаков принимает только одно из трех возможных значений: +, –, 0, которые обозначают увеличение, уменьшение или отсутствие изменений, в зависимости от наблюдаемой области и положения руки на предыдущем изображении видеопоследовательности.

Признаки положения руки, обозначенные как *form* (форма, положение), *right* (справа), *above* (выше), *torso* (торс), описывают ориентацию руки и пространственные отношения между рукой и другими частями тела, такими как лицо и торс. Ориентация руки представлена признаком *form*. Этот признак дискретизируется по одному из трех возможных значений: +, если рука расположена вертикально, –, если рука расположена горизонтально, или 0, если рука наклонена влево или вправо в плоскости *XY*. Признак *right* означает, что рука находится справа от головы, *above* – рука выше головы, *torso* – рука перед торсом. Три последних атрибута принимают бинарные значения *true* и *false*, соответствующие

выполнению или невыполнению условия. Пример извлечения признака положения руки с учетом вышеописанных переменных приведен на рис. 9.6.

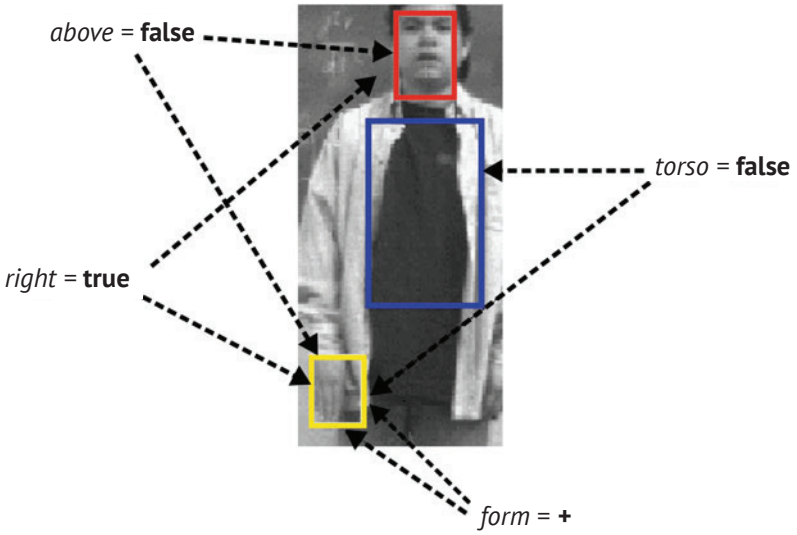


Рис. 9.6. На этом изображении показаны признаки положения руки. Видно, что рука в вертикальном положении, ниже головы, справа от фигуры пользователя и не находится перед его торсом. Следовательно, значения атрибутов: *above* = *false*, *right* = *true*, *torso* = *false*, *form* = +

Как и в скрытых марковских моделях, классификатор на основе динамической байесовской сети тренируется для каждого типа жеста. Для классификации вычисляется оценка вероятности каждой модели, и модель с самой высокой вероятностью выбирается как распознающая конкретный жест.

9.4.1.2 Эксперименты

Были проведены три эксперимента для сравнения производительностей классификации и обучения классификаторов на основе динамических байесовских сетей и скрытых марковских моделей. В первом эксперименте жесты одного и того же человека использовались для распознавания. Во втором эксперименте определялись оценки способностей обобщения классификаторов посредством тренировки и проверки жестов различных людей. В третьем эксперименте рассматривались варианты жестов, различающиеся по расстоянию (от исполнителя жестов) и углу зрения на выполняемый жест. Кроме того, сравнивались модели классификатора на основе динамических байесовских сетей и скрытых марковских моделей, использующие только признаки движения, и модели, применяющие информацию о движении и положении руки. Количество скрытых состояний в каждой модели изменялось от 3 до 18.

В первом эксперименте было записано (на видео) 50 выполнений каждого типа жестов одного человека, 20 экземпляров использовалось для тренировки, 30 для тестирования. Во втором эксперименте модели, обученные на жестах одного человека, оценивались на жестах, выполняемых другими 14 людьми. Каждый человек выполнил по 2 экземпляра каждого класса жеста. Для эксперимента с различными вариантами расстояния были случайно выбраны 15 экземпляров каждого жеста, выполненного на расстоянии 2 и 4 метров, т. е. был сформирован тестовый набор из 30 экземпляров на каждый жест. Для вариантов угла зрения также использовались 15 случайно выбранных экземпляров каждого жеста, наблюдаемого под углами +45 и −45 градусов.

Важным различием между классификаторами на основе динамических байесовских сетей и скрытыми марковскими моделями является количество параметров, требуемых для каждой модели. Количество параметров для определения распределений состояний наблюдений для скрытых марковских моделей с признаками положения и движения равно 648, а для данных только о движении – 27. В классификаторах на основе динамических байесовских сетей 21 параметр в первом случае и 12 во втором. Такое весьма существенное сокращение количества параметров для классификатора на основе динамических байесовских сетей оказывает важное воздействие на время тренировки, а также на точность классификации, когда количество тренировочных экземпляров невелико.

В экспериментах с одним человеком с использованием атрибутов движения и положения классификаторы на основе динамических байесовских сетей и скрытые марковские модели продемонстрировали весьма высокую производительность с коэффициентами успешного распознавания от 96 до 99% в зависимости от количества состояний (самые лучшие результаты были показаны для 12–15 состояний). Классификаторы на основе динамических байесовских сетей дают несколько лучшие результаты распознавания, но при значительном сокращении времени тренировки почти в десять раз быстрее скрытых марковских моделей.

В экспериментах с несколькими людьми вполне ожидаемо снизилась эффективность обоих вариантов – скрытых марковских моделей и классификаторов на основе динамических байесовских сетей – около 86% успешных распознаваний с атрибутами движения и положения. При использовании только атрибутов движения эффективность составила около 65%, т. е. почти на 20% ниже, чем при применении атрибутов положения. Как и в большинстве задач классификации, выбор наиболее подходящего набора атрибутов является чрезвычайно важным фактором.

В третьем эксперименте различия в расстоянии и углах зрения также воздействуют на результаты распознавания, при этом изменение

угла зрения оказывает большее воздействие. Изменение расстояния от видеокамеры до человека от 2 до 4 метров снижает эффективность распознавания приблизительно до 90 % (жесты одного человека). Если угол зрения изменяется в пределах от +45 до −45 градусов, то эффективность распознавания уменьшается приблизительно до 70 %. В обоих случаях результаты скрытых марковских моделей и классификаторов на основе динамических байесовских сетей практически одинаковы в плане эффективности.

Результаты этих экспериментов показывают конкурентоспособность в том, что касается показателей эффективности распознавания классификаторов на основе динамических байесовских сетей по сравнению со стандартными скрытыми марковскими моделями в различных задачах по распознаванию жестов. Факторизация атрибутов позволяет значительно сократить время тренировки для классификаторов на основе динамических байесовских сетей по сравнению с равнозначными скрытыми марковскими моделями, что позволяет выполнять обучение жестам в режиме онлайн. Кроме того, классификаторы на основе динамических байесовских сетей требуют меньшего количества тренировочных экземпляров для достижения эффективности, сравнимой с соответствующими скрытыми марковским моделями, в особенности при увеличении количества атрибутов.

9.4.2 Байесовская сеть с временны́ми узлами: прогнозирование вариантов мутаций ВИЧ

В этом разделе рассматривается приложение байесовских сетей с временны́ми узлами для исследования временны́х связей между устойчивыми к лекарственным средствам мутациями ВИЧ (вируса иммунодефицита человека) и антиретровирусными лекарственными средствами с выявлением возможных вариантов мутаций и установлением вероятностно-временно́й последовательности их появления [8].

Вирус иммунодефицита человека (ВИЧ) – один из самых быстро развивающихся организмов на нашей планете. Его удивительная способность к изменениям позволяет ВИЧ избежать воздействия многочисленных эволюционных факторов, влияющих на него естественным или искусственным образом. Это становится возможным благодаря развитию и отбору мутаций с высокой приспособляемостью. Это объект антиретровирусной терапии (АРТ) – мощное избирательное воздействие на ВИЧ. При неоптимальных условиях быстро выбираются мутации, которые позволяют вирусу воспроизводиться даже в присутствии высокоэффективных сочетаний антиретровирусных лекарственных средств. В частности, будет рассматриваться задача поиска связей мутация–мутация и лекарство–мутация у пациентов, лечащихся методом

антиретровирусной терапии. Мы сосредоточим внимание на ингибиторах протеазы (ИП), группе антиретровирусных лекарственных средств, широко применяемой в современной антиретровирусной терапии. Развитие устойчивых к лекарствам вирусов ставит под угрозу контроль над ВИЧ с дальнейшим ухудшением и ослаблением иммунной системы пациента. Таким образом, весьма важно глубокое понимание динамики появления и развития мутаций, устойчивых к лекарствам.

Хронологические данные, получаемые при наблюдении пациентов с ВИЧ, использовались для обучения байесовской сети с временны́ми узлами, затем эта модель оценивалась с точки зрения обнаруженных связей в соответствии с экспертными знаниями в предметной области и эффективности прогнозирования.

9.4.2.1 Данные

Клинические данные, полученные при наблюдении 2373 пациентов с ВИЧ подтипа B, извлечены из базы данных по ВИЧ Стэнфорда (HIVDB – HIV Stanford Database) [11]. Характеристики штаммов вируса в HIVDB были получены из профилей долговременного лечения, описывающих развитие мутаций в отдельных последовательностях. Для каждого пациента данные состоят из первоначально назначенного ему лечения (сочетания лекарственных средств) и из списка лабораторных исследований сопротивляемости, проведенных в различное время (по неделям). Каждое лабораторное исследование включает список наиболее часто встречающихся мутаций в среде вирусов внутри организма-«хозяина» в конкретный момент времени после начала лечения. Пример таких данных представлен в табл. 9.4. Количество доступных исследований изменяется от 1 до 10 в каждой истории болезни.

Таблица 9.4. Пример данных

Пациент	Первоначальное лечение	Список мутаций	Время (в неделях)
Пациент$_1$ (Pat_1)	LPV, FPV, RTV	L63P, L10I	15
		V77I	25
		I62V	50
Пациент$_2$ (Pat_2)	NFV, RTV, SQV	L10I	25
		V77I	45

Пациент Pat_1 с тремя временны́ми исследованиями, пациент Pat_2 с двумя временны́ми исследованиями.

Антиретровирусные средства обычно классифицируются по ферменту (энзиму), на который они ориентированы. Мы сосредоточились на вирусной протеазе, так как это наименьший из вирусных ферментов

по числу составляющих его аминокислот. В настоящее время доступны девять ингибиторов протеазы: ампренавир (Amprenavir – APV), атазанавир (Atazanavir – ATV), дарунавир (Darunavir – DRV), лопинавир (Lopinavir – LPV), индинавир (Indinavir – IDV), нельфинавир (Nelfinavir – NFV), ритонавир (Ritonavir – RTV), трипанавир (Tripanavir – TPV) и саквинавир (Saquinavir – SQV).

Для проверки способности этой модели прогнозировать важные с медицинской (клинической) точки зрения данные было выбрано подмножество пациентов из исходного набора данных, включая пациентов, лечащихся в режиме антиретровирусной терапии с применением ингибиторов протеазы LPV, IDV и SQV. Также были включены важные основные мутации, устойчивые к лекарствам, связанные с выбранными для лечения препаратами. Список выбранных мутаций: V32I, M46I, M46L, I47V, G48V, I54V, V82A, I84V, L90M. Поскольку используется только ограниченное подмножество лекарственных средств, количество пациентов в конечном наборе данных было сокращено до 300.

9.4.2.2 Модель и оценка

Байесовская сеть с временны́ми узлами обучалась по сокращенной базе данных HIVDB с применением алгоритма обучения, описанного в разделе 9.3. Были реализованы две модификации исходного алгоритма: первая для измерения мощности временно-вероятностных связей, вторая для изменения порядка переменных с учетом алгоритма обучения структуры (K2), поэтому результаты не являются предвзятыми из-за какого-то конкретного предварительно определенного порядка переменных.

Для определения оценки моделей и для измерения статистической значимости граничной интенсивности был использован метод «самораскрутки (самозагрузки) без параметров» (non-parametric bootstrapping) (получение нескольких моделей). Были определены два пороговых значения для рассмотрения связи как значимой. Связь считалась прочной, если она появлялась не менее чем в 90 % графов, или предполагаемой, если связь между значениями возникала в 70–90 % случаев. Поскольку методика выбора лекарственных средств и мутаций основана на мнениях экспертов, в рассматриваемом примере использовался более продуктивный способ определения порядка для алгоритма K2. В этом эксперименте рассматривались различные варианты упорядочения для алгоритма K2 и выбирался порядок с наивысшей точностью прогнозирования.

На рис. 9.7 изображена полученная байесовская сеть с временны́ми узлами. Узлы зеленого цвета представляют антиретровирусные лекарственные средства, узлы желтого цвета представляют мутации ВИЧ. Для каждой мутации (временно́го узла) показаны связанные с

ней интервалы времени с вероятностью появления мутации в конкретный интервал времени. Дуги, обозначающие прочные связи, помечены звездочкой (*).

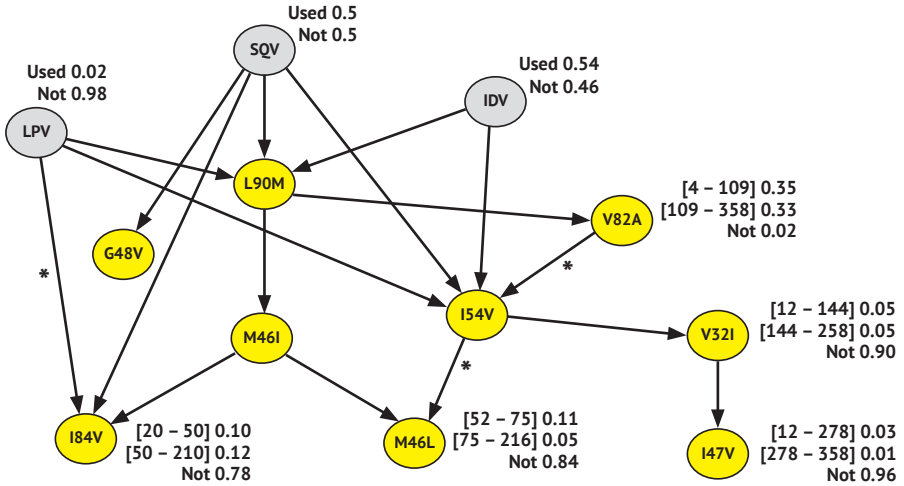


Рис. 9.7. Обученная модель байесовской сети с временны́ми узлами, отображающая временно-вероятностные связи между множеством антиретровирусных лекарственных средств (три верхних узла) и значимыми мутациями ВИЧ (все прочие узлы). Дуги, помеченные звездочкой (*), представляют прочные связи. Показаны только некоторые множества интервалов, связанные с соответствующими временны́ми узлами

Эта модель продемонстрировала способность прогнозировать клинически значимые связи между выбранными лекарственными средствами и мутациями ВИЧ. Разумеется, легко предсказуемыми в этой модели были прочные связи между SQV, G48V и I84V, несмотря на то что между этими двумя мутациями не наблюдается явных временны́х связей. Для всех трех лекарственных средств установлены явные прямые связи с мутацией L90M, отображая тот факт, что эта мутация является причиной общей сопротивляемости многим препаратам из этой группы ингибиторов протеазы. Показательно, что были спрогнозированы два возможных варианта мутации с сопротивляемостью препарату LPV [8]:

- I54V → V32I → I47V;
- L90M → M46IL → I84V.

Вне зависимости от важности временно́го порядка мутаций остается необходимость дальнейшей оценки модели. Кроме того, наблюдались совместные варианты мутаций между IDV и LPV, в том числе мутации L90M, M46IL, I54V, V82A и I84V.

9.5 Материалы для дополнительного чтения

Подробнейший обзор динамических байесовских сетей представлен в [9]. Фридман (Friedman) и др. [4, 6] описывают методики обучения динамических байесовских сетей. Байесовские сети с временны́ми узлами рассматриваются в [1], а их расширение с каноническими временны́ми моделями – в [5]. Алгоритм обучения байесовских сетей с временны́ми узлами описан в [7].

9.6 Задания и упражнения

1. Для динамической байесовской сети на рис. 9.1 определить, является ли она стабильной моделью. Какие параметры требуются для полной спецификации этой модели?
2. Предполагая, что все переменные являются бинарными (true, false), определить таблицы условных вероятностей для задания 1 (определить вероятности произвольным образом, так чтобы они соблюдали только аксиомы вероятности).
3. Для динамической байесовской сети на рис. 9.1 с таблицами условных вероятностей, определенными в задании 2: а) вычислить апостериорную вероятность для X_{t+1} при $X_t = S_t = true$ (фильтрация); б) вычислить апостериорную вероятность для X_{t+2} и X_{t+3} при $X_t = S_t = true$ (прогнозирование); в) вычислить апостериорную вероятность для X_{t+1} при $E_t = E_{t+1} = E_{t+2} = false$ (сглаживание).
4. Рассмотреть байесовскую сеть с временны́ми узлами на рис. 9.3. Определить таблицы условных вероятностей для всех переменных в соответствии со значениями и интервалами, показанными на рис. 9.3. Определить параметры, руководствуясь собственной интуицией (субъективные оценки).
5. Для структуры и параметров байесовской сети с временны́ми узлами из задания 4 вычислить апостериорную вероятность для всех переменных с учетом свидетельства $C = moderate$, используя вероятностный логический вывод (можно применить любую методику логического вывода для байесовских сетей).
6. Повторить решение задания 5, приняв в качестве свидетельства $DP = dilated$. Так как относительное время возникновения этого события неизвестно, рассмотреть различные возможные сценарии. Какой сценарий будет применяться, если в дальнейшем выяснится, что зрачки расширились через 4 единицы времени после столкновения?

7. Изменить алгоритмы логического вывода (Forward) и обучения (Баума–Велша) для скрытых марковских моделей (см. главу 5) так, чтобы их можно было применять к классификаторам на основе динамических байесовских сетей.
8. *** Найти наборы данных в различных динамических предметных областях для обучения динамических байесовских моделей (см. задания 9 и 10). Например, данные фондовой биржи, данные о погоде, долговременные медицинские исследования и т. п. Для каждого варианта приложения определить, какой тип модели – на основе состояний или на основе событий – является более подходящим.
9. *** Разработать программу, обучающую динамическую байесовскую сеть, с учетом двухэтапной процедуры: сначала обучается исходная структура и параметры (при $t = 0$), затем обучается структура переходов и параметры (для $t = k + 1$ при $t = k$).
10. *** Разработать программу, реализующую алгоритм *LIPS* для обучения байесовских сетей с временны́ми узлами.

Ссылки на источники

1. Arroyo-Figueroa, G., Sucar, L. E. A temporal Bayesian network for diagnosis and prediction. In: Proceedings of the 15thConference onUncertainty inArtificial Intelligence (UAI). Morgan-Kaufmann, San Mateo, p. 13–20 (1999).
2. Avilés-Arriaga, H. H., Sucar, L. E., Mendoza-Durán, C. E., Pineda-Cortés, L. A. Comparison of dynamic naive Bayesian classifiers and hidden Markov models for gesture recognition. J. Appl. Res. Technol. 9 (1), 81–102 (2011).
3. Cooper, G. F., Herskovitz, E. A Bayesian method for the induction of probabilistic networks from data. Mach. Learn. 9 (4), 309–348 (1992).
4. Friedman, N., Murphy, K., Russell, S. Learning the Structure of Dynamic Probabilistic Networks. In: Proceedings of the FourteenthConference onUncertainty inArtificial (UAI). Morgan-Kaufmann Publishers Inc., p. 139–147 (1998).
5. Galán, S. F., Arroyo-Figueroa, G., Díez, F. J., Sucar, L. E. Comparison of two types of event Bayesian networks: a case study. Appl. Artif. Intell. 21 (3), 185–209 (2007).
6. Ghahramani, Z. Learning Dynamic Bayesian Networks. Lecture Notes in Computer Science 1387, 168–197 (1998).

7. Hernández-Leal, P., González, J. A., Morales, E. F., Sucar, L. E. Learning temporal nodes Bayesian networks. Int. J. Approx. Reason. 54 (8), 956–977 (2013).

8. Hernández-Leal, P., Rios-Flores, A., Ávila-Rios, S., Reyes-Terán, G., González, J. A., Fiedler-Cameras, L., Orihuela-Espina, F., Morales, E. F., Sucar, L. E. Discovering HIV mutational pathways using temporal bayesian networks. Artif. Intell. Med. 57 (3), 185–195 (2013).

9. Murphy, K. Dynamic Bayesian networks: representation, inference and learning. dissertation, University of California, Berkeley (2002).

10. Rabiner, R., Juang, B. Fundamentals of Speech Recognition. Prentice Hall, New Jersey (1993).

11. Shafer, R. Rationale and uses of a public HIV drug-resistance database. J. Infect. Dis. 194 (1), 51–58 (2006).

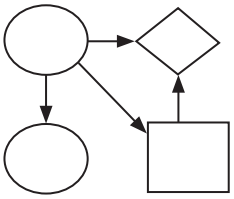
Часть III

Модели принятия решений

В этой части книги представлены вероятностные графовые модели, предполагающие принятие решений. Эти модели кроме случайных переменных содержат переменные принятия решения и переменные полезности. Цель таких моделей – помочь принять наилучшие решения в условиях неопределенности. В одной главе описаны модели, которые должны выбирать одно или несколько решений одновременно. В другой главе рассматриваются задачи последовательного принятия решений, в которых необходимо принимать множество решений в течение некоторого времени.

Глава 10

Графы принятия решений



10.1 Введение

Модели, которые рассматривались в части II, содержали только случайные переменные, поэтому их можно использовать для оценки апостериорной вероятности множества переменных с учетом некоторого свидетельства, например для классификации, диагностики или прогнозирования. Такие модели также могут предоставить наиболее вероятное сочетание значений некоторого подмножества переменных (наиболее вероятное объяснение) или глобальную (общую) вероятность всей модели в целом с учетом некоторых наблюдений. Но эти модели невозможно напрямую использовать для принятия решений.

В этой и в следующей главах будут представлены *модели принятия решений* (decision models), цель которых – помочь принять наилучшие решения в условиях неопределенности. Мы убедимся в том, что наилучшими решениями являются те, которые максимизируют ожидаемую полезность агента с учетом текущих имеющихся знаний (свидетельства) и заданных целей в рабочей среде теоретического принятия решений. Эти типы агентов называют *рациональными агентами* (rational agents).

После краткого введения в теорию принятия решений в этой главе будут описаны два типа методик моделирования для задач с одним или несколькими вариантами решений: деревья решений (decision trees) и диаграммы влияния (influence diagrams). Как и в случае применения вероятностных моделей, эти методики используют преимущества структуры зависимостей конкретной задачи для получения более компактного представления и более эффективной оценки.

10.2 Теория принятия решений

Теория принятия решений (decision theory) предоставляет нормативную структуру (рабочую среду) для лица, принимающего решение в условиях неопределенности. Теория принятия решений основана на концепции рациональности (rationality), т. е. агент должен пытаться максимизировать пользу или минимизировать издержки (затраты). При этом предполагается, что существует некоторый способ присваивания степени полезности (обычно это числовое значение, которое может соответствовать стоимости в денежном выражении или по любой другой шкале полезности) результату каждого варианта действий, чтобы наилучшим решением стал тот вариант, который имеет наивысшую полезность. В общем случае агент не уверен в результатах каждого из возможных решений, поэтому необходимо принимать во внимание оценки полезности каждого варианта решения. В теории принятия решений мы учитываем ожидаемую полезность (expected utility), которая вычисляет среднее значение всех возможных результатов решения, взвешенных по их вероятностям. Таким образом, если говорить кратко, рациональный агент обязательно должен выбрать решение, которое максимизирует ожидаемую полезность.

Теория принятия решений изначально была разработана в области экономики и исследования операций [11], но в последние годы она привлекла внимание исследователей в области искусственного интеллекта, заинтересованных в исследовании и создании интеллектуальных агентов (intelligent agents). Такие интеллектуальные агенты, например роботы, финансовые советники, интеллектуальные инструкторы и т. п., должны решать задачи, подобные тем, которые встречаются в экономике и исследовании операций, но с двумя основными отличиями. Во-первых, это размер (объем) задач, который в области искусственного интеллекта, как правило, намного крупнее, чем в обычных экономических приложениях. Второе основное отличие заключается в использовании знаний о предметной области задачи. Во многих приложениях искусственного интеллекта модель неизвестна заранее и может быть весьма трудной для определения.

10.2.1 Основы теории принятия решений

Принципы теории принятия решений изначально были определены в классической работе фон Неймана (von Neuman) и Моргенштерна (Morgenstern) «Theory of Games and Economic Behavior» (Теория игр и экономическое поведение) [11]. Авторы установили совокупность интуитивно определенных ограничений, которые должны управлять предпочтениями рационального агента (rational agent). Эти ограничения известны как аксиомы теории полезности (utility theory). Прежде

чем привести список этих аксиом, необходимо определить некоторые обозначения и термины. В процедуре принятия решения существуют четыре элемента:

- альтернативы (alternatives) – варианты выбора, доступные агенту; этими вариантами агент управляет (контролирует). Каждое решение имеет как минимум две альтернативы (например, выполнять или не выполнять некоторое действие);
- события (events) – генерируются окружающей средой и другими агентами. События не управляются (не контролируются) агентом. Каждое случайное событие приводит как минимум к двум результатам, и хотя заранее неизвестно, какой результат получится в действительности, можно присвоить некоторую вероятность каждому варианту;
- исходы (outcomes) – результаты сочетания решений, принятых агентом, и случайных событий. Каждый возможный исход имеет различную предпочтительность (полезность) для агента;
- предпочтения (preferences) – устанавливаются в соответствии с целями и задачами агента; агент присваивает их каждому возможному исходу. Предпочтения определяют для агента значение (ценность) каждого возможного результата решений.

В теории полезности различные сценарии называются лотереями (lotteries). В лотерее каждый возможный исход или состояние A имеет определенную вероятность p и соответствующее предпочтение для агента, выражаемое действительным числом U. Например, лотерея L с двумя возможными исходами: A с вероятностью p и B с вероятностью $1 - p$ может быть обозначена следующим образом:

$$L = [p, A; 1 - p, B].$$

Если агент предпочитает исход A в большей степени, чем исход B, то это записывается в виде $A \succ B$, а если агенту безразличны оба исхода, это обозначается как $A \sim B$. В общем случае лотерея может иметь любое количество исходов, при этом исход может быть элементарным состоянием или другой лотереей.

На основе этих концепций можно определить теорию полезности тем же методом, что и теорию вероятностей, т. е. устанавливая совокупность разумно обоснованных ограничений для предпочтений рационального агента – **аксиомы теории полезности**:

- *упорядоченность* (order) – сравнивая два состояния, агент предпочитает первое или второе, или же безразлично относится к обоим состояниям;

- *транзитивность* (transitivity) – если агент предпочитает исход A исходу B и исход B исходу C, то из этого непременно следует предпочтение исхода A относительно исхода C;
- *непрерывность* (continuity) – если $A \succ B \succ C$, то существует некоторая вероятность p, такая, что для агента не имеет значения различие между получением исхода B с вероятностью, равной единице, и лотереей $L = [p, A; 1 - p, C]$;
- *взаимозаменяемость* (substitutability) – если для агента не имеет значения различие между двумя лотереями A и B, то для агента не имеет значения и различие между двумя более сложными лотереями, подобными исходным, за исключением того, что B заменяет A в одной из этих лотерей;
- *монотонность* (monotonicity) – существуют две лотереи с одними и теми же исходами A и B. Если агент предпочитает исход A, то он обязательно должен выбрать лотерею, в которой исход A имеет более высокую вероятность;
- *способность к декомпозиции* (decomposability) – сложные составные лотереи можно разложить (выполнить декомпозицию) на простые лотереи, используя правила теории вероятностей.

Теперь из аксиом теории полезности можно вывести функцию (принцип) полезности.

Принцип полезности: если предпочтения агента соответствуют аксиомам теории полезности, то существует функция полезности U, использующая действительные значения, такая, что:

1) $U(A) \succ U(B)$ тогда и только тогда, когда агент предпочитает исход A исходу B;
2) $U(A) = U(B)$ тогда и только тогда, когда для агента не имеет значения различие между исходами A и B.

Принцип максимальной ожидаемой полезности: полезность лотереи равна сумме полезностей каждого исхода, умноженных на соответствующую вероятность:

$$U\left[P_1, S_1; P_2, S_2; P_3, S_3; \ldots\right] = \sum_j P_j U_j.$$

Теперь на основе этой концепции функции полезности можно определить ожидаемую полезность (expected utility – EU) конкретного решения D, принятого агентом, считая, что существует N возможных результатов этого решения, каждый из которых имеет вероятность P:

$$EU(D) = \sum_{j=1}^{N} P(результат_j(D))U(результат_j(D)).$$

Принцип максимальной ожидаемой полезности утверждает, что рациональный агент должен выбирать действие, которое максимизирует ожидаемую полезность для этого агента.

10.2.1.1 Полезность денег

Во многих случаях удобно измерять полезность в денежных единицах: чем больше денег получено на основе принятых решений, тем лучше. Таким образом, можно применять принцип максимальной ожидаемой полезности, измеряя полезность в денежном выражении. Но это не так просто, как кажется.

Предположим, что вы участвуете в игре, подобной обычным телевизионным шоу, и уже выиграли миллион долларов. Ведущий игры спрашивает, желаете ли вы сохранить свой выигрыш и завершить участие в игре или продолжить, перейти на следующий этап и попытаться выиграть $3 000 000. Вместо столь сложного вопроса ведущий просто подбросит монету, и если выпадет орел, то вы получите три миллиона долларов, но если выпадет решка, то потеряете все деньги, которые уже выиграли. Вам необходимо принять решение с двумя вариантами: (D1) сохранить выигранные деньги, (D2) перейти на следующий этап с получением возможности выиграть три миллиона (или потерять все). Каким будет ваше решение?

Посмотрим, что может подсказать нам принцип максимальной ожидаемой полезности, если измерять полезность в долларах (это называют ожидаемой денежной стоимостью – expected monetary value – EMV). Вычислим ожидаемую денежную стоимость (EMV) для обоих вариантов:

$$D1: EMV(D1) = 1 \times \$1\,000\,000 = \$1\,000\,000;$$
$$D2: EMV(D2) = 0.5 \times 0 + 0.5 \times \$3\,000\,000 = \$1\,500\,000.$$

Здесь можно видеть, что если мы хотим максимизировать ожидаемую полезность, выражаемую в долларах, то необходимо рисковать и продолжать игру. Но большинство из нас вероятнее всего выберут вариант с сохранением уже выигранного миллиона и не станут продолжать игру, рискуя потерять все. Как обосновано такое решение? Разве мы не должны быть рациональными?

Для большинства людей связь между полезностью и денежной стоимостью не является линейной, скорее, это логарифмическое отношение, которое обозначает неприятие риска (risk aversion) (см. рис. 10.1). Это отношение приблизительно линейно для небольших денежных

сумм (например, если в описанной выше игре у нас есть не миллион, а 10 долларов, то вероятнее всего мы продолжим игру «по линейному отношению»), но, имея крупную сумму (понятие «крупная сумма» каждый человек оценивает индивидуально), увеличение полезности, приносящее больше денег, перестает быть линейным.

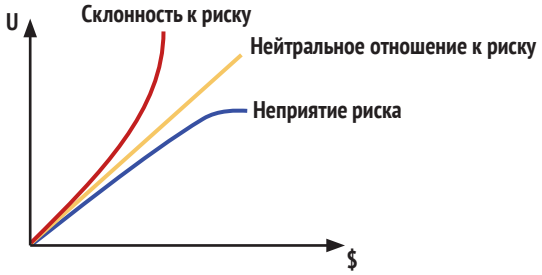


Рис. 10.1. На графиках показаны характерные связи между полезностью (U) и денежной стоимостью ($). *Верхний график*: склонность к риску, *средний график*: нейтральное отношение к риску, *нижний график*: неприятие риска

Отношение значений полезность–деньги различно для разных людей (и организаций) и зависит от их отношения к риску. Существует три основных типа отношения к риску: неприятие риска, нейтральное отношение к риску и склонность к риску, как показано на рис. 10.1.

Несмотря на то что применение принципа максимальной ожидаемой полезности для определения наилучшего решения кажется вполне очевидным, задачи принятия решений могут оказаться более сложными, включающими несколько различных решений, событий и возможных исходов. Решение таких задач не так просто, как может показаться на первый взгляд, – требуется систематический подход к моделированию и объяснению сложных задач принятия решений. Одним из самых первых инструментальных средств моделирования для задач принятия решений являются **деревья решений** (decision trees) [1].

10.3 Деревья решений

Дерево решений – это графическое представление задачи принятия решений, которое содержит три типа элементов или узлов, представляющих три основные компоненты задачи принятия решений: решения (decisions), неопределенные события (uncertain events) и результаты (results).

Узел решения (decision node) изображается как прямоугольник с несколькими ветвями (branches), каждая ветвь представляет один из возможных вариантов, имеющихся в этом пункте принятия решения. В конце каждой ветви может быть другой пункт принятия решения, событие или результат.

Узел события (event node) изображается в виде круга и также имеет несколько ветвей, которые представляют все возможные исходы этого неопределенного события. Эти исходы соответствуют всем возможным результатам события, т. е. исходы должны быть взаимоисключающими и исчерпывающими (полными). Каждой ветви присваивается значение вероятности, такое, что сумма вероятностей всех ветвей равна единице. В конце каждой ветви может быть другой узел события, узел решения или результат.

Результаты (results) помечаются полезностью, которую они имеют для агента, и обычно располагаются в конце каждой ветви дерева (т. е. являются листьями (leaves)).

Деревья решений обычно изображаются слева направо, т. е. корень дерева (узел решения) является левым крайним узлом, а листья дерева располагаются справа. Пример гипотетической задачи принятия решения (на основе примера из [6]) показан на рис. 10.2. Пример представляет принятие решения по инвестициям с тремя вариантами: (i) акции на фондовой бирже (Stocks), (ii) золото (Gold) и (iii) без инвестиций (No investment). Предполагая, что инвестиции производятся один раз в год, при инвестировании в акции в зависимости от положения на фондовой бирже (неопределенное событие) можно получить $1000 или потерять $300 с равной вероятностью. При инвестировании в золото необходимо принять еще одно решение: оформлять страховку или нет. При оформлении страховки гарантировано получение $200, в противном случае можно выиграть или проиграть в зависимости от того, поднимется ли цена на золото, останется неизменной или упадет – это представлено еще одним событием. Каждый возможный исход имеет определенное значение и присвоенную вероятность, как показано на рис. 10.2. Какое решение должен принять инвестор?

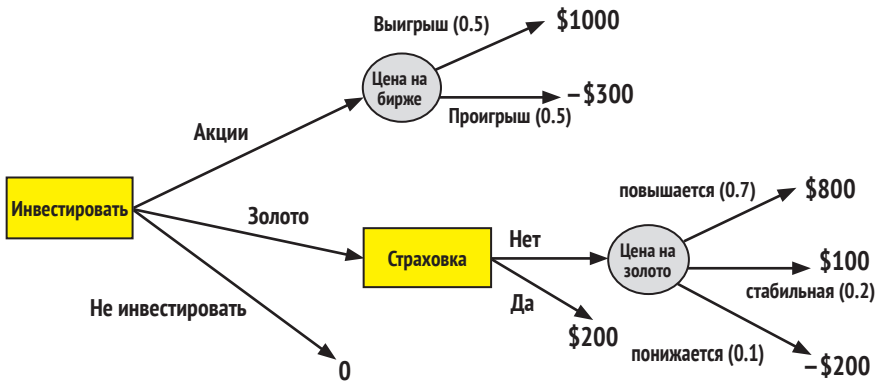


Рис. 10.2. Пример дерева решений (подробное описание см. в тексте)

Для определения наилучшего решения в каждом пункте принятия решения в соответствии с принципом максимальной ожидаемой по-

лезности необходимо вычислить оценку дерева решений. Вычисление оценки дерева решений состоит в определении значений узлов обоих типов – узлов решений и узлов событий. Эта процедура выполняется справа налево, начиная с любого узла, который имеет только результаты для всех своих ветвей:

- значением узла решений D является максимальное значение из всех значений исходящих ветвей этого узла:

$$V(D) = max_j U(результат_j(D));$$

- значение узла событий E – это ожидаемая полезность всех ветвей, исходящих из этого узла, получаемая как взвешенная сумма значений результатов, умноженных на соответствующие вероятности:

$$V(E) = \sum_j P(результат_j(E))U(результат_j(E)).$$

Следуя описанной выше процедуре, можно вычислить оценку дерева решений, показанного на рис. 10.2:

Событие 1 – Цена акций на фондовой бирже:

$$V(E_1) = 1000 \times 0.5 - 300 \times 0.5 = 350;$$

Событие 2 – Цена золота:

$$V(E_2) = 800 \times 0.7 + 100 \times 0.2 - 200 \times 0.1 = 560;$$

Решение 2 – Страховка:

$$V(D_2) = max(200, 560) = 560 - \text{Без страховки};$$

Решение 1 – Инвестиция:

$$V(D_1) = max(150, 560, 0) = 560 - \text{Инвестировать в золото}.$$

Таким образом, в этом примере наилучшим решением становится инвестиция в золото без страховки.

Деревья решений представляют собой инструментальные средства для моделирования и решения задач с принятием последовательных решений, так как решения должны быть представлены в последовательности, как в предыдущем примере. Но размер такого дерева (количество ветвей) возрастает экспоненциально по отношению к количеству узлов решений и узлов событий, поэтому такое представление на практике применимо только для небольших задач. Другим инструментальным средством моделирования является диаграмма влияния (influence diagram) [3, 8], которая обеспечивает компактное представление задачи принятия решений.

10.4 Диаграммы влияния

Диаграммы влияния (influence diagrams) представляют собой инструментальные средства для решения задач принятия решений. Диаграммы влияния были введены Ховардом (Howard) и Мэтисоном (Matheson) [3] как иной вариант деревьев решений для упрощения моделирования и анализа. С другой точки зрения можно рассматривать диаграммы влияния как расширение байесовских сетей, которое включает узлы решений и узлы полезности. В следующих разделах будет представлено краткое введение в диаграммы влияния, включая их представление и основные методы логического вывода.

10.4.1 Моделирование

Диаграмма влияния – это направленный ациклический граф G, который содержит узлы, представляющие случайные переменные, переменные решений и переменные полезности:

- случайные узлы (X) – представляют случайные переменные, как и в байесовских сетях, с соответствующими таблицами условных вероятностей. На диаграмме изображаются овалами;
- узлы решений (D) – представляют решения, которые должны быть приняты. Дуги, направленные к узлу решений, называют *информационными* (informational), т. е. это означает, что случайный узел или узел решений в начале такой дуги обязательно должен быть известен до начала процедуры принятия решения. На диаграмме обозначаются прямоугольниками;
- узлы полезности (U) – представляют стоимости или полезности, связанные с моделью. С каждым узлом связана функция, отображающая каждую операцию преобразования родителей этого узла в значение полезности. На диаграмме узлы полезности изображаются ромбами. Узлы полезности можно дополнительно подразделить на *обычные* (ordinary) узлы полезности, родителями которых являются случайные узлы и/или узлы решений, и *сверхценные* (super-value) узлы полезности, родителями которых являются обычные узлы полезности. Обычно сверхценные узлы полезности представляют собой (взвешенную) сумму обычных узлов полезности.

В диаграмме влияния существует три типа дуг:

- вероятностные (probabilistic) – обозначают вероятностные зависимости, направлены к случайным узлам;

- информационные (informational) – обозначают доступность информации, направлены к узлам решений. То есть $X \rightarrow D$ означает, что узел X известен до того, как будет принято решение D;
- функциональные (functional) – обозначают функциональные зависимости, направлены к узлам полезности.

Пример диаграммы влияния показан на рис. 10.3. Здесь демонстрируется упрощенная модель задачи определения места строительства нового аэропорта с учетом вероятности аварий, уровня шума и оценочной стоимости строительства как факторов, которые напрямую влияют на полезность.

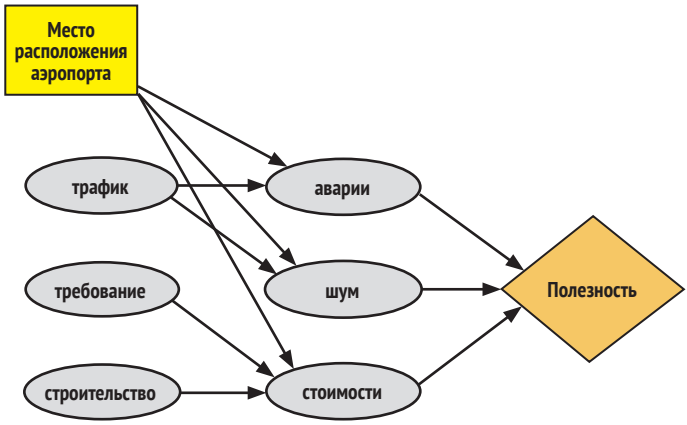


Рис. 10.3. Упрощенная модель задачи определения места расположения аэропорта, представленная в форме простой диаграммы влияния. Узел решения представляет различные варианты расположения нового аэропорта, узел полезности представляет полезность (или стоимость), которая зависит от нескольких факторов, в свою очередь зависящих от других случайных переменных

В любой диаграмме влияния обязательно должен существовать направленный (ориентированный) путь в представляющем диаграмму направленном графе. Этот путь должен включать все узлы решения, устанавливая порядок принятия решений. Этот порядок логически определяет часть случайных переменных в диаграмме влияния так, что если существует n переменных решений, то случайные переменные разделяются на $n + 1$ подмножеств. Каждое подмножество $\mathbf{R}_i$ содержит все случайные переменные, которые известны перед решением D_i, но неизвестны для предыдущих решений. Некоторые алгоритмы оценки диаграмм влияния используют преимущество этих свойств для повышения эффективности оценки.

Диаграммы влияния используются для оказания помощи лицу, принимающему решение, при поиске решений, максимизирующих ожидаемую полезность. Таким образом, целью анализа решений является

поиск оптимальной стратегии $\pi = \{d_1, d_2, \ldots, d_n\}$, при которой выбираются наилучшие решения для каждого узла решений для максимизации ожидаемой полезности $E_\pi(U)$. Если существует несколько узлов полезности, то в общем случае считается, что мы имеем дополнительную полезность, поэтому должна максимизироваться сумма этих отдельных полезностей:

$$E_\pi(U) = \sum_{u_i \in U} E_\pi(u_i). \tag{10.1}$$

10.4.2 Оценка

Оценка диаграммы влияния представляет собой поиск последовательности наилучших решений, или оптимальной стратегии. Сначала мы рассмотрим, как можно решить (оценить) простую диаграмму влияния с единственным решением. Затем будет приведен обзор обобщенных методик решения (оценки) диаграмм влияния.

Мы определяем простую диаграмму влияния как диаграмму с единственным узлом решения и единственным узлом полезности. Для этого варианта можно просто применить одну из методик логического вывода для байесовской сети, чтобы получить оптимальную стратегию, выполняя следующий алгоритм.

1. Для всех $d_i \in D$:
   a) установить $D = d_i$;
   b) присвоить значения всем известным случайным переменным;
   c) выполнить распространение вероятностей, как в байесовской сети;
   d) получить ожидаемое значение узла полезности U.
2. Выбрать решение d_k, которое максимизирует U.

Для более сложных задач принятия решений, в которых имеется несколько узлов решений, описанный выше алгоритм становится практически неприменимым. Вообще говоря, существуют три основных типа методик для решения диаграмм влияния:

- преобразование диаграммы влияния в дерево решений и применение стандартных методов решения (оценки) для деревьев решений;
- решение диаграммы влияния напрямую методом исключения переменных, применяя последовательность преобразований к графу, представляющему диаграмму;
- преобразование диаграммы влияния в байесовскую сеть и использование методик логического вывода для байесовских сетей.

В следующих подразделах будут рассматриваться второй и третий типы методик.

10.4.2.1 Метод исключения переменных

Алгоритм исключения переменных [8] основан на исключении узлов решений поочередно в определенном порядке. Узлы решений, которые уже были оценены, можно удалить из модели, и этот процесс продолжается до тех пор, пока не будут оценены все узлы решений. Для применения данной методики диаграмма влияния должна быть регулярной, т. е. соответствовать следующим условиям:

1) структурой диаграммы влияния является направленный ациклический граф;
2) узлы полезности не должны иметь потомков;
3) в представляющем структуру ориентированном графе существует направленный путь, включающий все узлы решений и определяющий порядок, в котором эти решения принимаются.

В общем случае для оценки узлов решений необходимо выполнить последовательность преобразований в диаграмме влияния. Эти преобразования обязательны для сохранения оптимальной последовательности решений, или оптимальной стратегии. Возможны следующие преобразования:

- исключение бесполезных (бессодержательных) узлов из числа случайных узлов или узлов решений, которые являются узлами-листьями в графе, – они не влияют на решения;
- исключение случайных узлов, которые являются родителями узлов полезности и не имеют других потомков, – полезность обновляется в соответствии со значением узла (если узлу не присвоено конкретное значение, то вычисляется ожидаемая полезность);
- исключение узлов решений, являющихся родителями узла полезности, когда родителями этих узлов решений также являются родители узла полезности, – вычисляется оценка узла решения, и принимается решение, которое максимизирует ожидаемую полезность; изменение производится в соответствии с функцией полезности;
- в том случае, когда ни одна из описанных выше операций неприменима, изменяется на противоположное направление дуги между двумя случайными переменными. Для инвертирования направления дуги между узлами i и j требуется отсутствие другого пути между этими узлами. Тогда дуга $i \rightarrow j$ меняет свое направление на противоположное и каждый узел «наследует» родителей другого узла.

Рассмотрим графическое представление алгоритма исключения переменных на примере. Пусть имеется исходная диаграмма влияния, показанная на рис. 10.4. Можно исключить бесполезный случайный узел в нижнем левом углу. После исключения этого узла его родитель также становится бесполезным узлом, который удаляется, и мы получаем диаграмму влияния, показанную на рис. 10.5.

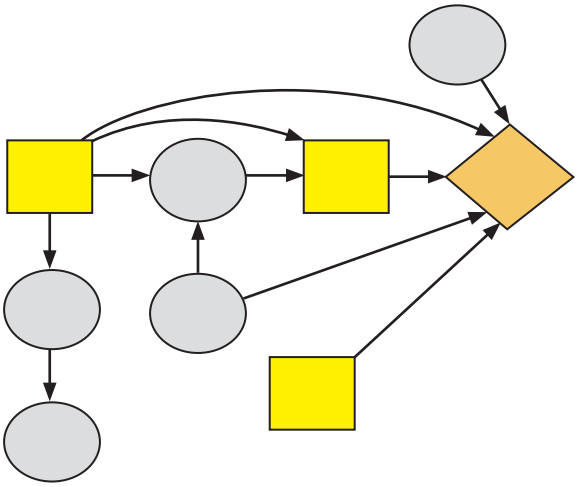


Рис. 10.4. Пример выполнения алгоритма исключения переменных: исходная диаграмма влияния

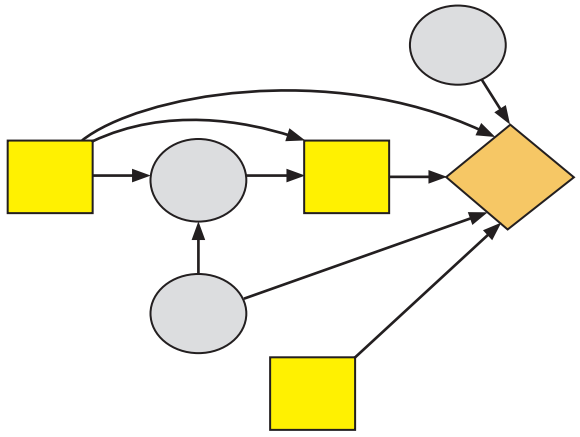


Рис. 10.5. Пример выполнения алгоритма исключения переменных: после исключения двух бесполезных узлов

Далее исключается случайный узел в верхней части диаграммы, родитель узла полезности, концентрирующего значение полезности, затем исключается первое решение, т. е. узел решения в нижней части диаграммы, и в результате получается диаграмма влияния, показанная на рис. 10.6.

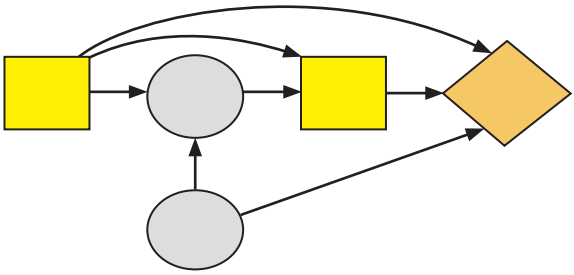


Рис. 10.6. Пример выполнения алгоритма исключения переменных: после исключения первого (узла) решения

Затем изменяется на противоположное направление дуги между двумя оставшимися случайными узлами, в результате чего появляется возможность исключения нижнего случайного узла, и получается модель, показанная на рис. 10.7. В этом графе можно оценить узел решения справа, после чего исключить оставшийся случайный узел и, наконец, оценить узел решения слева.

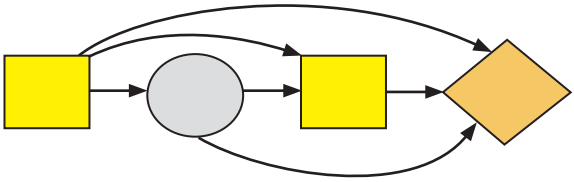


Рис. 10.7. Пример выполнения алгоритма исключения переменных: после инверсии направления дуги и исключения случайного узла

В следующем разделе будет описана другая методика исключения узлов для диаграмм влияния, которая использует преимущества эффективных алгоритмов логического вывода, разработанных для байесовских сетей.

10.4.2.2 Преобразование в байесовскую сеть

Методика приведения диаграммы влияния к байесовской сети впервые была предложена Купером (Cooper) [2]. Для преобразования диаграммы влияния в байесовскую сеть основным принципом является преобразование узлов решений и узлов полезности в случайные узлы с сохранением связанного с ними распределения вероятностей. Узел решений преобразуется в дискретную случайную переменную с учетом каждого варианта решения d_i как значения для этой переменной. При этом используется равномерное распределение как таблица условных вероятностей (узел решений не имеет родителей, поскольку все входящие дуги являются информационными). Узел полезности преобразуется в бинарную случайную переменную посредством нормализации

функции полезности так, чтобы ее значения находились в диапазоне от 0 до 1, т. е.

$$P(u_i = 1 \mid Pa(u_i)) = val(Pa(u_i)) / maximum(val(Pa(u_i))), \quad (10.2)$$

где $Pa(u_i)$ – родители узла полезности в диаграмме влияния, *val* – значение, присвоенное каждому сочетанию значений родительских узлов.

После описанного выше преобразования с учетом единственного узла полезности задача поиска оптимальной стратегии сводится к поиску значений узлов решений, которые максимизируют вероятность узла полезности: $P(u = 1 \mid D, R)$, где D – множество узлов решений, R – множество других случайных переменных в диаграмме влияния. Вероятность можно вычислить, используя стандартные методики логического вывода для байесовских сетей, но для этого потребуется возрастающее экспоненциально число шагов логического вывода, по одному для каждой перестановки элементов множества D.

Учитывая, что в регулярной диаграмме влияния узлы решений упорядочены, можно более эффективно вычислить оценку, если оценивать решения в существующем (или в обратном) порядке [9]. Таким образом, вместо максимизации $P(u = 1 \mid D, R)$ максимизируется $P(D_j \mid u = 1, R)$. Можно рекурсивно оптимизировать каждый узел решений D_j, начиная с самого последнего решения, затем переходя к предыдущему решению и т. д. до достижения самого первого решения. Это гораздо более эффективная процедура оценки. Для описанного здесь алгоритма были предложены дополнительные улучшения, основанные на диаграммах влияния с возможностью их декомпозиции (decomposable IDs) (см. раздел материалов для дополнительного чтения).

В обычных методиках решения диаграмм влияния принимаются два важных предположения:

- полное упорядочение – все решения соблюдают полное упорядочение в соответствии с направленным путем в графе;
- запоминание (non-forgetting) – все предшествующие наблюдения запоминаются для принятия последующих решений.

Эти предположения ограничивают применимость диаграмм влияния лишь некоторыми предметными областями, в частности временны́ми задачами, в которых подразумевается принятие нескольких решений в различные интервалы времени. В некоторых предметных областях, таких как принятие медицинских решений, полное упорядочение решений является неосуществимым предположением, так как существуют ситуации, в которых лицу, принимающему решение, неизвестно заранее, какое решение должно быть принято в первую очередь для максимизации ожидаемой полезности. Для системы, развивающейся в

течение длительного интервала времени, количество наблюдений возрастает линейно по времени, поэтому требование запоминания подразумевает, что размер стратегий возрастает экспоненциально.

10.4.3 Расширения

10.4.3.1 Диаграммы влияния с ограниченной памятью

Для устранения описанных в предыдущем разделе ограничений диаграмм влияния Лауритцен (Lauritzen) и Нильссон (Nilsson) [5] предложили диаграммы влияния с ограниченной памятью (limited memory influence diagrams – LIMID) как расширение обычных диаграмм влияния. Термин «ограниченная память» отображает свойство, согласно которому нет необходимости запоминать переменную, известную при принятии решения, для принятия последующего решения. Исключение некоторых переменных снижает сложность модели, и модель становится разрешимой с помощью компьютера, но за счет применения не самой оптимальной стратегии.

10.4.3.2 Динамические сети принятия решений

Другое расширение применяется для задач последовательного принятия решений, в которых подразумевается принятие нескольких решений в течение некоторого интервала времени. Как и в байесовских сетях, можно рассматривать задачи принятия решений, в которых последовательность решений должна быть принята в различные интервалы времени. Этот тип задач известен как задача последовательного принятия решений (sequential decision problem). Задача последовательного принятия решений может моделироваться как динамическая сеть принятия решений (dynamic decision network – DDN), также известная как динамическая диаграмма влияния (dynamic influence diagram), которую можно рассматривать как расширение динамической байесовской сети с дополнительными узлами решений и узлами полезности для каждого интервала времени (см. рис. 10.8).

Теоретически можно вычислить оценку динамической сети принятия решений так же, как и оценку диаграммы влияния, учитывая, что решения должны быть упорядочены по времени. Таким образом, каждый узел решений D_t имеет входящие информационные дуги, направленные от всех предшествующих узлов решений D_{t-1}, D_{t-2} и т. д. Но поскольку количество интервалов времени («эпох») увеличивается, сложность возрастает, что может привести к неосуществимости с точки зрения вычислений. Кроме того, в некоторых приложениях заранее неизвестно количество интервалов времени принятия решений («эпох»), более того, теоретически возможно даже бесконечное количество решений.

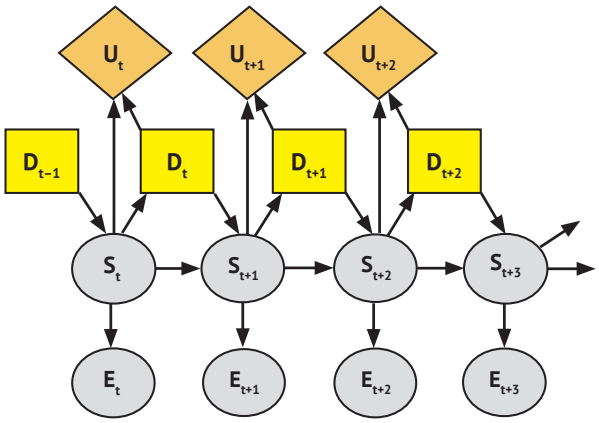


Рис. 10.8. Пример динамической сети принятия решений с четырьмя этапами («эпохами») принятия решений

Динамические сети принятия решений тесно связаны с марковскими процессами принятия решений, которые будут рассматриваться в следующей главе.

10.5 Приложения

Практическое применение графов принятия решений демонстрируется на примере системы помощи при мытье рук пожилым людям и людям с физическими недостатками.

10.5.1 Медработник, принимающий теоретические решения

Цель медработника (осуществляющего уход за пациентами) – помощь пациенту в завершении выполнения какой-либо задачи с использованием правильного выбора одного из предлагаемых вариантов. Здесь рассматривается конкретная задача: мытье рук (cleaning one's hands). Рассматриваемая система действует как медработник, который направляет пожилого человека или человека с физическими недостатками для правильного выполнения этой задачи [7].

Важные объекты в среде умывальной комнаты (ванной): мыло, водопроводный кран, полотенце. Система определяет поведение пользователя при взаимодействии с этими объектами. Затем система выбирает действие (используются звуковые подсказки) для направления пользователя к завершению задачи. Возможно также отсутствие подсказок (так называемое нулевое действие – null action), если пользователь правильно выполняет последовательность требуемых шагов (действий).

10.5.1.1 Модель

Динамическая сеть принятия решений используется для моделирования поведения пользователя и принятия оптимальных решений в каждом интервале времени на основе поведения пользователя (наблюдений) и целей системы (полезностей). Оптимальными действиями могут считаться предварительно оцененные моделью многие шаги, выполняемые до тех пор, пока не будет завершена задача мытья рук, а баланс между оптимальностью и эффективностью достигается посредством предварительного анализа *k* шагов (lookahead). При этом оптимальные решения для *k* узлов решений получаются вычислением динамической сети принятия решений с помощью одной из методик, описанных в предыдущих разделах этой главы. В рассматриваемом здесь примере динамическая сеть принятия решений была преобразована в динамическую байесовскую сеть и решена с использованием логического вывода для байесовских сетей.

Динамическая сеть принятия решений для этого примера показана на рис. 10.9. В этой модели переменные состояния (*S*) представляют действия пользователя в каждом интервале времени, которые не являются непосредственно наблюдаемыми. Узлы наблюдений (*O*) представляют информацию от системы наблюдения, используемой для распознавания действий, выполняемых пользователем. Узлы действий (*A*) соответствуют различным действиям (подсказкам), которые могут быть выбраны управляющей системой в любой момент времени. Поощрения (*R*) представляют немедленные поощрения, зависящие от текущего состояния и выбранного действия. Ниже эти элементы модели описаны более подробно:

- **состояния** – пространство состояний характеризуется действиями (жестами рук), выполняемыми человеком. В рассматриваемом примере переменная состояний может принимать шесть возможных значений: s_1 = *открывание водопроводного крана*, s_2 = *закрывание водопроводного крана*, s_3 = *использование мыла*, s_4 = *вытирание рук*, s_5 = *взятие полотенца*, s_6 = *мытье рук*;
- **наблюдения** – соответствуют информации, получаемой системой визуального распознавания жестов [7], которая пытается распознавать действия, выполняемые человеком при мытье рук. Значения наблюдений полностью совпадают со значениями состояний;
- **действия** – звуковые подсказки, помогающие человеку завершить выполнение задачи. Существует восемь действий, соответствующих возможным подсказкам, предлагаемым системой: a_1 = *открыть водопроводный кран*, a_2 = *закрыть водопроводный кран*,

a_3 = взять мыло в руки, *a_4 = намылить и вымыть руки*, *a_5 = взять полотенце*, *a_6 = вытереть руки*, *a_7 = нулевое действие*, *a_8 = запрос помощи*;

- **поощрения** – связаны с предпочтительными вариантами выбора различных действий, выбираемых системой. В контексте медработника-сиделки необходимо учитывать подсказки, понятность этих подсказок и ответную реакцию пользователя. Использовались три различных значения поощрения: +3 обозначает предпочтительное действие, −3 обозначает штраф, −6 используется для выбора обращения к помощи (call for help). Здесь важно, что запрос помощи должен быть самым последним вариантом действия.

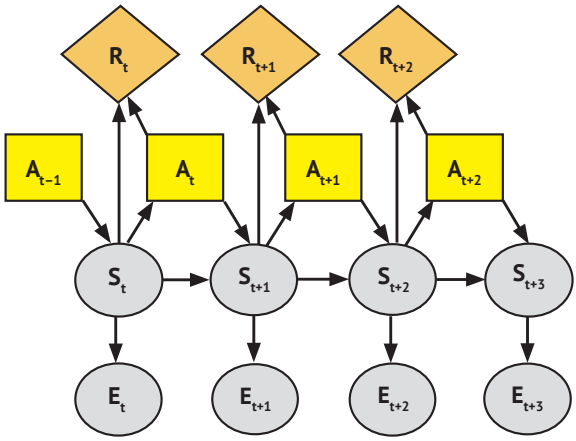


Рис. 10.9. Динамическая сеть принятия решений с четырьмя этапами, моделирующая образ действий медработника (сиделки). S_t представляет действия пользователя, O_t соответствует наблюдаемым действиям, A_t – выбранные действия, R_t – немедленное поощрение

Кроме того, для этой модели требуются две таблицы условных вероятностей: функция переходов $P(S_{t+1} \mid S_t)$ и функция наблюдений $P(O_t \mid S_t)$:

- **функция переходов** – определяет вероятность следующего состояния (следующего жеста) с учетом текущего состояния и действия. В рассматриваемом здесь примере это предсказуемо с некоторой степенью неопределенности. Функции переходов определялись субъективно;
- **функция наблюдений** – в рассматриваемом здесь примере функция наблюдений состоит из вероятности наблюдаемого жеста с учетом состояния, т. е. действительно выполняемого жеста. Эту вероятность можно получить по достоверности (матрицы несоответствий (неточностей)) системы распознавания жестов.

Предполагается, что модель не изменяется во времени, т. е. полученные таблицы условных вероятностей не изменяются со временем.

10.5.1.2 Вычисление оценки

Модель оценивалась по следующим характеристикам: (i) чувствительность относительно количества состояний или предварительного анализа; (ii) эффективность по времени, требуемому для решения модели при выборе следующего действия; (iii) эффективность при сравнении действий, выбранных системой, с действиями медработника-человека.

Динамическая сеть принятия решений была решена выполнением от 2 до 11 этапов, а действия, выбираемые в различных сценариях, сравнивались с моделями других размеров. Ожидаемая полезность увеличивалась при расширении предварительного анализа, обычно стабилизируясь после 6–7 этапов. Но выбранные действия не изменялись после 4 этапов предварительного анализа, поэтому было выбрано именно это значение.

Модель с 4 интервалами времени (4 узлами решений) оценивалась по ее времени отклика, и на ее решение потребовалось около 3 секунд на обычном персональном компьютере. Таким образом, это количество интервалов времени обеспечивает разумный баланс между скоростью (производительностью) и эффективностью.

Для оценки качества действий, выбранных системой, ее решения сравнивались с решениями (действиями) человека, выполняющего ту же задачу. Предварительная оценка была определена по действиям обычных людей, имитирующих проблемы при мытье рук. Десять взрослых людей, участвующих в эксперименте, были разделены на две группы, по 5 человек в каждой (тестовая и контрольная группы). Первая (тестовая) группа для завершения задачи мытья рук руководствовалась звуковыми подсказками, выдаваемыми системой. Контрольная группа получала словесные инструкции от человека-ассистента. Каждый участник эксперимента в опросном листе оценивал следующие характеристики: (i) понятность подсказки; (ii) подробность подсказки; (iii) эффективность системы. Подробность подсказки соответствует уровню ее конкретности. Понятность оценивает, насколько легко пользователю понять сообщение. Эффективность оценивает уровень руководства системы для успешного выполнения задачи.

Полученные результаты приведены в табл. 10.1. По этим результатам можно видеть небольшое преимущество человека при выборе наилучшей подсказки, но различия между системой и человеком-ассистентом незначительны. В двух характеристиках наблюдается небольшое различие (0.6 и менее), в одной (подробность подсказки) различие более значимо. Эту последнюю характеристику необходимо учесть при звуковой

записи сообщений, связанных с каждой подсказкой. Этот аспект легко улучшить, он почти не влияет на решения, принимаемые системой.

Таблица 10.1. Результаты, полученные при эксперименте с системой-сиделкой и человеком, имитирующим проблемы с памятью, которым руководит эта система для завершения задачи мытья рук

	Человек-сиделка	Система
Понятность	4.4	3.9
Подробность	4.6	3.6
Эффективность	4.2	3.6

Сравниваются решения человека-ассистента и системы теоретического принятия решений на основе динамической сети принятия решений. Шкала оценок от 1 (наихудшая оценка) до 5 (наилучшая оценка).

10.6 Материалы для дополнительного чтения

Первоначальным источником информации о диаграммах влияния является книга Ховарда (Howard) и Мэтисона (Matheson) [3]. В книге Дженсена (Jensen) рассматриваются сети принятия решений [4]. Деревья решений описаны в [1]. Алгоритм исключения переменных для оценки диаграмм влияния представлен в [8]. Другая методика оценки на основе преобразования в байесовскую сеть описана в [2]. Диаграммы влияния с ограниченной памятью представлены в [5], а их расширение до динамических моделей – в [10].

10.7 Задания и упражнения

1. Определить функцию полезности в денежном выражении с учетом неприятия риска, такую, что оптимальным решением для примера из раздела 10.2.1.1 является сохранение денег ($D1$) при вычислении ожидаемой полезности для обоих возможных решений.
2. Рассмотреть дерево решений на рис. 10.2. Будущее состояние фондовой биржи изменилось, и теперь потенциальная прибыль увеличилась до $3000. Кроме того, цена страховки золота увеличилась до $600. После применения этих изменений в дереве решений повторно вычислить его оценку. Изменились ли решения по сравнению с исходным примером?
3. Определить требуемые таблицы условных вероятностей для диаграммы влияния на рис. 10.3. Рассмотреть два возможных места расположения аэропорта Loc_A и Loc_B и учесть, что все слу-

чайные переменные являются бинарными: *traffic* = {*low*, *high*}, *demand* = {*no*, *yes*}, *construction* = {*simple*, *complex*}, *accidents* = {*very-low*, *low*}, *noise* = {*low*, *medium*}, *costs* = {*medium*, *high*}. Определить параметры, руководствуясь собственной интуицией и аксиомами теории вероятностей.

4. Для диаграммы влияния на рис. 10.3 определить функцию полезности с учетом аварий, уровня шума и затрат, используя значения из задания 3: а) определить функцию полезности как математическую функцию *U* = *f*(*аварии*, *уровень шума*, *затраты*); б) определить функцию полезности как таблицу.

5. На основе параметров и функций полезности из заданий 3 и 4 определить оценку диаграммы влияния, изображенной на рис. 10.3, вычисляя функцию полезности для каждого возможного места расположения аэропорта при отсутствии свидетельства. Какое из мест расположения аэропорта является наилучшим в соответствии с описанной здесь моделью?

6. Повторно выполнить задание 5 для различных сценариев, например: *traffic* = *low*, *demand* = *no*, *construction* = *simple* и *traffic* = *high*, *demand* = *yes*, *construction* = *complex*. Изменится ли оптимальное место расположения аэропорта при этих сценариях?

7. Рассмотреть задачу о принятии решения: брать ли с собой зонт на основании прогноза погоды. Существуют два варианта решения этой задачи: ознакомиться с прогнозом погоды (нет, да) и взять зонт (нет, да) и одна случайная переменная *weather* (*погода*) = {*sunny* (*солнечно*), *light-rain* (*слабый дождь*), *heavy-rain* (*сильный дождь*)}. Смоделировать эту задачу как дерево решений, установив стоимости/полезности для различных сценариев, например: взять зонт и солнечно, взять зонт и слабый дождь, не брать зонт и сильный дождь и т. д.

8. Определить диаграмму влияния для задачи о зонте из задания 7.

9. *** Разработать программу, которая выполняет преобразования диаграммы влияния в байесовскую сеть. Затем использовать вероятностный логический вывод для оценки этой байесовской сети. Применить программу для решения модели размещения аэропорта, используя параметры и полезности, определенные в задании 8.

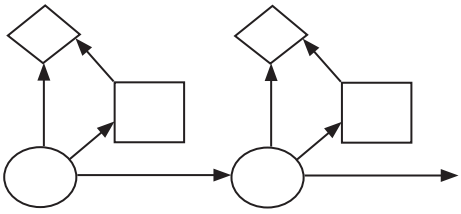
10. *** Исследовать процедуру, используемую для преобразования диаграммы влияния в дерево решений. Применить эту процедуру для решения модели размещения аэропорта.

Ссылки на источники

1. Cole, S., Rowley, J. Revisiting decision trees. Manag. Decis. 33 (8), 46–50 (1995).
2. Cooper, G. A method for using belief networks as influence diagrams. In: Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI), p. 55–63 (1988).
3. Howard, R., Matheson, J. Influence diagrams. In: Howard, R., Matheson, J. (eds.) Readings on the Principles and Applications of Decision Analysis. Strategic Decisions Group, Menlo Park (1984).
4. Jensen, F. V. Bayesian Networks and Decision Graphs. Springer, New York (2001).
5. Lauritzen, S., Nilsson, D. Representing and solving decision problems with limited information. Manag. Sci. 47, 1235–1251 (2001).
6. Ley-Borrás, R. Análisis de Incertidumbre y Riesgo para la Toma de Decisiones. Comunidad Morelos, Orizaba, Mexico (2001) (In Spanish).
7. Montero, A., Sucar, L. E. Decision-theoretic assistants based on contextual gesture recognition. Ann. Inf. Syst. (to be published).
8. Shachter, R. D. Evaluating influence diagrams. Oper. Res. 34 (6), 871–882 (1986).
9. Shachter, R. D., Peot, M. Decision making using probabilistic inference methods. In: Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence (UAI), p. 276–283 (1992).
10. van Gerven, M. A. J., Díez, F. J., Taal, B. G., Lucas, P. J. F. Selecting treatment strategies with dynamic limited-memory influence diagrams. Artif. Intell. Med. 40 (3), 171–186 (2007).
11. von Neumann, J., Morgenstern, O. Theory of Games and Economic Behavior. Princeton University Press, Princeton (1944).

Глава 11

Марковские процессы принятия решений



11.1 Введение

В этой главе рассматриваются задачи последовательного принятия решений, т. е. задачи, в которых подразумевается последовательность принятия решений в течение некоторого времени. Предполагается, что агент принятия решений рационален, поэтому целью является максимизация ожидаемой полезности в течение длительного времени. Учитывая существование неопределенности в результатах решений агента, этот тип задач можно моделировать как марковские процессы принятия решений (Markov decision processes – MDP). Решая модель марковского процесса принятия решений, мы получаем то, что называют *стратегией* (policy), которая указывает агенту, какое действие выбрать на каждом временно́м шаге на основе текущего состояния. Оптимальной стратегией является та, которая выбирает действия так, чтобы максимизировать ожидаемое значение.

Сначала будет приведено формализованное определение модели марковского процесса принятия решений, затем будут представлены два стандартных метода ее решения: итерация значения и итерация стратегии. Несмотря на то что сложность решения марковского процесса принятия решений является квадратической по отношению к числу состояний-действий, эта модель может стать практически невыполнимой в отношении требований к памяти и времени, когда число состояний-действий слишком велико. Факторизованные марковские

процессы принятия решений обеспечивают представление на основе графовых моделей для решения весьма больших марковских процессов принятия решений.

В конце главы будут представлены частично наблюдаемые марковские процессы принятия решений (partially observable Markov decision processes – POMDP), в которых существует неопределенность не только в результатах действий, но и в состояниях.

11.2 Моделирование

Марковский процесс принятия решений (Markov decision process – MDP) [13] моделирует задачу последовательного принятия решений, в которой система развивается во времени и управляется агентом. Динамика системы регулируется вероятностной функцией перехода Φ, которая отображает состояния **S** и действия **A** в новые состояния **S'**. В каждый момент времени агент принимает поощрение R, зависящее от текущего состояния s и предпринятого действия a. Решая представление этой задачи в форме марковского процесса принятия решений, мы получаем рекомендуемую стратегию, которая максимизирует ожидаемое поощрение во времени, а также учитывает неопределенность результатов любого действия.

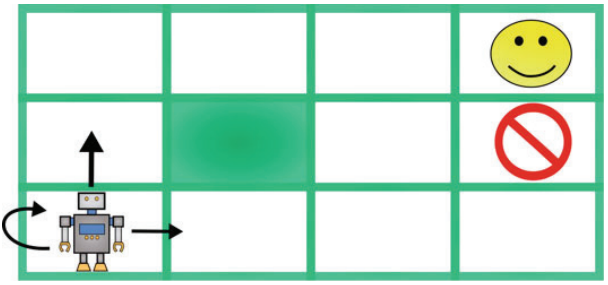


Рис. 11.1. Робот в мире, ограниченном строгой сетью координат. Каждая ячейка представляет возможные состояния робота, улыбающееся лицо – цель, запрещающий знак – опасность. Робот изображен в ячейке со стрелками, демонстрируя вероятность перехода в следующее состояние с учетом действия *up* (вверх)

Например, рассмотрим агента (имитацию робота), который живет в мире, ограниченном строгой сетью координат, при этом состояние робота определяется ячейкой, в которой он находится (см. рис. 11.1). Робот должен двигаться к цели (к ячейке с улыбающимся лицом), обходить препятствия (закрашенные ячейки) и избегать опасностей (ячейка с запрещающим знаком). Возможные действия робота – перемещение в соседние ячейки (вверх, вниз, влево, вправо). Предполагается, что робот получает немедленное определенное поощрение, когда проходит через каждую ячейку, например +100, когда достигает цели, –100, если прохо-

дит через ячейку с запрещающим знаком (которая может представлять опасное место), –1 для всех прочих ячеек (это заставляет робота искать самый короткий путь к цели).

Будем считать, что существует неопределенность в результатах каждого действия, выполняемого роботом. Например, если выбрано действие *up* (вверх), то робот переходит в верхнюю ячейку с вероятностью 0.8 и с вероятностью 0.2 в другие ячейки (влево или вправо). Это обозначено с помощью толщины стрелок на рис. 11.1. Имея в своем распоряжении определенные состояния, действия, поощрения и функцию перехода, можно смоделировать эту задачу как марковский процесс принятия решений.

Задача робота – прийти в целевую ячейку как можно быстрее и избежать опасностей. Эта цель достигается с помощью решения марковского процесса принятия решений, который представляет эту задачу, и с помощью максимизации ожидаемого поощрения[20]. Найденное решение обеспечивает агента стратегией, т. е. определяет наилучшее действие, выполняемое в каждом состоянии, как показано стрелками на графической схеме этого небольшого примера на рис. 11.2.

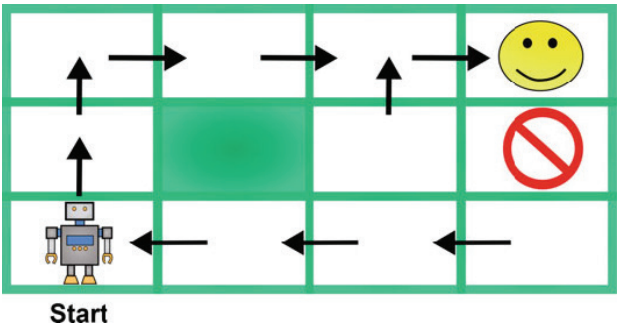


Рис. 11.2. Итоговая стратегия, определенная как результат решения модели марковского процесса принятия решений. Стрелки показывают оптимальное действие в каждой ячейке

С формальной точки зрения марковский процесс принятия решений представляет собой кортеж $M = <S, A, \Phi, R>$, где S – конечное множество состояний $\{s_1, \ldots, s_n\}$, A – конечное множество действий $\{a_1, \ldots, a_m\}$, Φ: $A \times S \times S \rightarrow [0, 1]$ – функция перехода между состояниями, определяемая как распределение вероятностей. Вероятность достижимого состояния s' посредством выполнения действия a из состояния s записывается как $\Phi(a, s, s')$. R: $S \times A \rightarrow \Re$ – функция поощрения. $R(s, a)$ – это поощрение, получаемое агентом, если он выполняет действие a в состоянии s.

В зависимости от того, насколько велик интервал, рассматриваемый в будущем (horizon), существуют два основных типа марковских процессов принятия решений: (i) с конечным интервалом (finite horizon) и

[20] При этом предполагается, что определяемая здесь функция поощрения правильно моделирует требуемую цель.

(ii) с бесконечным интервалом (infinite horizon). В задачах с конечным интервалом считается, что существует фиксированное, предварительно определенное количество временны́х шагов, для которых необходимо максимизировать ожидаемое поощрение (или минимизировать затраты (издержки)). Например, рассмотрим пример с инвестором, который покупает и продает акции каждый день (временной шаг) и желает максимизировать свою прибыль в течение года (интервал в будущем – horizon). В задачах с бесконечным интервалом нет фиксированного, предварительно определенного количества временны́х шагов; это количество может быть различным и теоретически бесконечным. Это подходящий вариант для обобщенной задачи планирования действий робота, так как изначально неизвестно количество перемещений (действий), которое потребуется роботу для достижения цели или нескольких целей.

В оставшейся части главы мы сосредоточимся на задачах с бесконечным интервалом, так как они в основном встречаются на практике. Кроме того, преимуществом таких задач является то, что при определенных условиях оптимальная стратегия стабильна, т. е. зависит только от состояния, а не от временны́х шагов.

Стратегия (policy) π для марковского процесса принятия решения представляет собой функцию $\pi : S \rightarrow A$, определяющую для каждого состояния s_i действие, которое необходимо выполнить, a_i. С учетом конкретной стратегии ожидаемое суммарное поощрение для конкретного состояния s известно как значение (value) для этого состояния, соответствующее применяемой стратегии $V^{\pi}(s)$. Это значение можно вычислить, используя следующую рекурсивную формулу:

$$V^{\pi}(s) = R(s, a) + \sum_{s \in \mathbf{S}} \Phi(a, s, s') V^{\pi}(s'), \tag{11.1}$$

где $R(s, a)$ представляет немедленное поощрение с учетом действия a, а сумма $\sum_{s \in \mathbf{S}} \Phi(a, s, s') V^{\pi}(s')$ – ожидаемое значение следующего состояния в соответствии с выбранной стратегией.

Для варианта задачи с бесконечным интервалом параметр, известный как *коэффициент дисконта* (discount factor) $0 \leqslant \gamma < 1$, включается для того, чтобы сумма сходилась. Этот параметр можно интерпретировать как коэффициент, придающий большее значение (ценность) поощрению, получаемому в настоящее время, чем поощрению в будущем.[21]

После включения коэффициента дисконта функция значения выглядит следующим образом:

$$V^{\pi}(s) = max_a \{R(s, a) + \gamma \sum_{s \in \mathbf{S}} \Phi(a, s, s') V^{\pi}(s'), \tag{11.2}$$

[21] Это значение ценности очевидно при финансовых инвестициях, связанных с инфляцией и процентной ставкой. Для других приложений обычно не совсем понятен способ определения коэффициента дисконта, и в общем случае используется значение, близкое к единице, например 0.9.

Требуется найти стратегию, которая максимизирует ожидаемое поощрение, т. е. стратегию, дающую наибольшее значение для всех состояний. Для варианта задачи с бесконечным интервалом и с любым учитываемым коэффициентом дисконта γ существует стратегия π^*, которая оптимальна независимо от начального состояния и соответствует хорошо известной формуле *Беллмана* (Bellman) [2]:

$$V^{\pi}(s) = max_a\left\{R(s, a) + \gamma \sum_{s \in \mathbf{S}} \Phi(a, s, s')V^{\pi}(s')\right\}. \quad (11.3)$$

Следовательно, стратегия, которая максимизирует формулу 11.3, и является оптимальной стратегией π^*:

$$\pi^*(s) = argmax_a\left\{R(s, a) + \gamma \sum_{s \in \mathbf{S}} \Phi(a, s, s')V^{\pi}(s')\right\}. \quad (11.4)$$

Формула *Беллмана* является рекурсивным выражением, которое невозможно вычислить напрямую. Но существует несколько методов ее эффективного вычисления. Эти методы будут рассматриваться в следующем разделе.

11.3 Вычисление оценки

Существуют три основных метода решения (вычисления оценки) марковского процесса принятия решений и поиска оптимальной стратегии: а) итерация значения; б) итерация стратегии; в) линейное программирование [13]. Два первых метода решают задачу итеративно, уточняя значение функции или стратегии соответственно. Третий метод преобразовывает исходную задачу в задачу линейного программирования, которую можно решить с помощью стандартных методов оптимизации, например симплекс-методом. Мы подробно рассмотрим только два первых метода, подробности третьего метода см. в разделе материалов для дополнительного чтения.

11.3.1 Итерация значения

Процедура итерационного уточнения значения начинается с присваивания начального значения каждому состоянию. Обычно это значение немедленного поощрения для соответствующего состояния. Таким образом, на итерации 0 значение $V_0(s) = R(a, s)$. Далее эти оценки значений уточняются на каждой итерации посредством максимизации с использованием формулы Беллмана. Процесс завершается, когда значения всех состояний сходятся, т. е. когда разность значений на предыдущей и текущей итерациях меньше предварительно установленного порогового значения. Действия, выбранные на последней итерации, соответствуют оптимальной стратегии. Выполнение метода показано в алгоритме 11.1.

Алгоритм 11.1. Алгоритм итерации значения

1: $\forall_s V_0(s) = R(s, a)$ {Инициализация}
2: $t = 1$
3: **repeat**
4: $\forall_s V_t(s) = \max_a\{R(s, a) + \gamma\sum_{s\in S}\Phi(a, s, s')V_{t-1}(s')\}$ {Итеративное уточнение}
5: **until** $\forall_s \mid V_t(s) - V_{t-1}(s) \mid < \varepsilon$
6: $\pi^*(s) = \text{argmax}_a\{R(s, a) + \gamma\sum_{s\in S}\Phi(a, s, s')V_t(s')\}$ {Получение оптимальной стратегии}

Сложность по времени этого алгоритма является квадратической относительно количества состояний-действий.

Обычно стратегия сходится раньше, чем сходится значение; это означает, что стратегия не изменяется, даже если значение пока еще не сходится. Это дает повод для подробного рассмотрения второго метода – итерации стратегии.

11.3.2 Итерация стратегии

Процедура итерационного уточнения стратегии начинается с выбора случайной начальной стратегии (если в распоряжении имеются некоторые знания из предметной области, то их можно использовать для «посева» начальной стратегии). Затем стратегия итеративно уточняется (улучшается) при выборе действий для каждого состояния, увеличивающих большинство ожидаемых значений. Алгоритм завершается, когда стратегия сходится, т. е. когда стратегия не изменяется по сравнению с предыдущей итерацией. Выполнение метода показано в алгоритме 11.2.

Алгоритм 11.2. Алгоритм итерации стратегии

1: $\pi_0 : \forall_s a_0(s) = a_k$ {Инициализация стратегии}
2: $t = 1$
3: **repeat**
4: {Итеративное уточнение (улучшение) стратегии}
5: $\forall_s V_t^{\pi_{t-1}}(s) = \{R(s, a) + \gamma\sum_{s\in S}\Phi(a, s, s')V_{t-1}(s')\}$ {Вычисление значений для текущей стратегии}
6: $\forall_s \pi_t(s) = \text{argmax}_a\{R(s, a) + \gamma\sum_{s\in S}\Phi(a, s, s')V_t(s')\}$ {Итеративное уточнение (улучшение)}
7: **until** $\pi_t = \pi_{t-1}$

Как правило, процесс итерационного уточнения стратегии сходится за меньшее число итераций по сравнению с итерациями значения, но накладные расходы на вычисления в каждой итерации выше, так как значения должны обновляться.

Решение небольших марковских процессов принятия решений с использованием алгоритма 11.2 весьма эффективно, но при значительном уве-

личении пространства состояний-действий возникают затруднения. Например, рассмотрим задачу с 10000 состояний и 10 действиями, что часто встречается в приложениях, подобных задаче о движении робота. В этом случае пространство, требуемое для хранения таблицы переходов, будет иметь размер $10000 \times 10000 \times 10 = 10^9$, а обновление функции значения потребует порядка 10^8 операций на каждой итерации. Поэтому решение весьма крупного марковского процесса принятия решений может стать чрезвычайно затруднительным даже при современном уровне компьютерных технологий. Другим вариантом является декомпозиция пространства состояний и использование преимуществ независимых отношений (связей) для снижения требований к памяти и вычислительным ресурсам с применением представления на основе графовой модели марковских процессов принятия решений, известной как факторизованные марковские процессы принятия решений (factored Markov decision process).

11.4 Факторизованные марковские процессы принятия решений

В факторизованных марковских процессах принятия решений множество состояний описывается через множество случайных переменных $\mathbf{X} = \{X_1, \ldots, X_n\}$, где каждая переменная X_i принимает значения из некоторой конечной области (домена) $Dom(X_i)$. Состояние s определяет значение $x_i \in Dom(X_i)$ для каждой переменной X_i. Модель переходов и функция поощрения могут увеличиваться по экспоненциальному закону и стать весьма большими, если они явно представлены как матрицы, но рабочие среды для динамических байесовских сетей (см. главу 7) и деревья решений [14] предоставляют инструментальные средства для описания модели переходов и функции поощрения в компактной форме.

Пусть X_i обозначает переменную в текущий момент времени, а X_i' – переменную на следующем шаге. Функция переходов для каждого действия a представлена как двухэтапная динамическая байесовская сеть, т. е. двухуровневый направленный ациклический граф G_T с узлами $\{X_1, \ldots, X_n, X_i', \ldots, X_n'\}$ (см. рис. 11.3, слева). Каждый узел X_i' связан с *условным распределением вероятностей* $P_\Phi(X_i' \mid Parents(X_i'))$, которое обычно представлено матрицей (*таблицей условных вероятностей*) или в более компактной форме дерева решений. Тогда вероятность перехода $\Phi(a, s_i, s_i')$ определяется как $\prod_i P_\Phi(x_i' \mid \mathbf{u_i})$, где $\mathbf{u_i}$ представляет значения переменных в множестве $Parents(X_i')$.

Поощрение, связанное с некоторым состоянием, часто зависит только от значений конкретных признаков этого состояния. Отношение между поощрениями и переменными состояния может быть представлено с помощью узлов значений в диаграммах влияния, как показано на рис. 11.3 (в центре). Таблица условных поощрений (conditional reward table – CRT)

для такого типа узла представляет собой таблицу, связывающую поощрение с каждым сочетанием значений для родителей этого узла в графе. Эта таблица является экспоненциальной по числу связанных с ней переменных. Несмотря на то что в наихудшем случае таблица условных поощрений будет экспоненциально увеличивать пространство для хранения функции поощрений, во многих случаях функция поощрений имеет структуру, позволяющую представить ее компактно с использованием деревьев решений или графов, как показано на рис. 11.3 (справа).

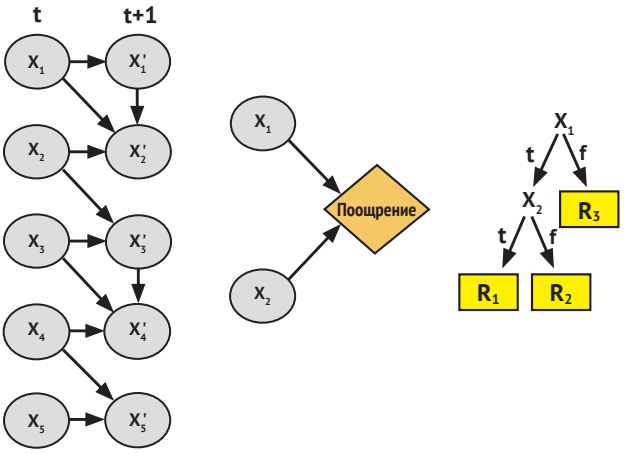
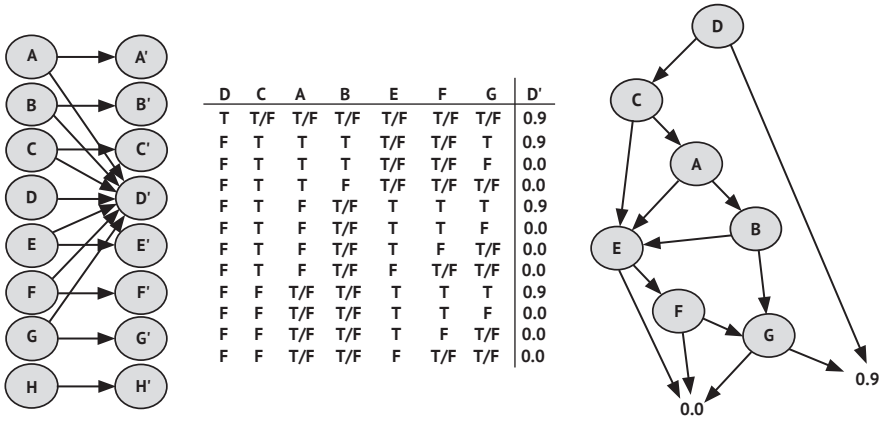


Рис. 11.3. Слева: динамическая байесовская сеть с пятью переменными состояния, представляющими функцию перехода для одного действия. В центре: диаграмма влияния, определяющая функцию поощрения. Справа: структурированное условное поощрение (conditional reward – CR), представленное как бинарное дерево решений

Во многих случаях таблицы условных вероятностей в динамической байесовской сети соответствуют определенной структуре, в частности некоторые значения вероятностей достаточно часто повторяются много раз (например, нулевая вероятность). Воспользовавшись преимуществом таких свойств, можно обеспечить еще более компактное представление таблицы условных вероятностей в форме дерева или графа, так как повторяющиеся значения вероятностей отображаются только один раз в листьях этих графов (деревьев). Чрезвычайно эффективным представлением является *алгебраическая диаграмма решений* (algebraic decision diagram – ADD). Пример таблицы условных вероятностей, представленной в форме алгебраической диаграммы решений, показан на рис. 11.4.

Представление функций переходов в марковских процессах принятия решений как двухэтапных динамических байесовских сетей и функций поощрения как дерева решений с дальнейшим повышением их компактности на основе деревьев или алгебраических диаграмм решений во многих случаях позволяет весьма существенно сэкономить

память для хранения очень крупных марковских процессов принятия решений. Пример такой экономии будет продемонстрирован в разделе приложений в текущей главе.



D	C	A	B	E	F	G	D'
T	T/F	T/F	T/F	T/F	T/F	T/F	0.9
F	T	T	T	T/F	T/F	T	0.9
F	T	T	T	T/F	T/F	F	0.0
F	T	T	F	T/F	T/F	T/F	0.0
F	T	F	T/F	T	T	T	0.9
F	T	F	T/F	T	T	F	0.0
F	T	F	T/F	T	F	T/F	0.0
F	T	F	T/F	F	T/F	T/F	0.0
F	F	T/F	T/F	T	T	T	0.9
F	F	T/F	T/F	T	T	F	0.0
F	F	T/F	T/F	T	F	T/F	0.0
F	F	T/F	T/F	F	T/F	T/F	0.0

Рис. 11.4. Пример таблицы условных вероятностей, представленной в форме алгебраической диаграммы решений. Слева: двухэтапная динамическая байесовская сеть, представляющая функцию переходов. В центре: таблица условных вероятностей для одной из переменных. Справа: алгебраическая диаграмма решений, представляющая таблицу условных вероятностей, – для каждого узла стрелка вправо соответствует значению *T*, стрелка влево соответствует значению *F*

Кроме того, на основе такого компактного представления были разработаны чрезвычайно эффективные версии алгоритмов итераций значения и стратегии, которые также сокращают время вычислений, требуемое для решения сложных моделей марковских процессов принятия решений. Примером является алгоритм SPUDD [8].

Дальнейшего снижения вычислительной сложности можно достичь, используя другие методики, такие как абстракция и декомпозиция, описанные в следующих подразделах.

11.4.1 Абстракция

Главный принцип абстракции (abstraction) – сокращение пространства состояний посредством создания абстрактной модели, в которой состояния с похожими признаками объединяются в группы [9].

Равнозначные состояния (equivalent states) – это состояния, которые имеют одинаковые функции переходов и поощрений. Такие состояния можно объединить в группы без изменения исходной модели, так что оптимальная стратегия останется той же самой для сокращенной модели. Но сокращение пространства состояний только при помощи объединения равнозначных состояний в общем случае не является существенным. Дальнейшего сокращения можно достичь, группируя схожие

состояния. В результате получаются приблизительные модели и достигается баланс между точностью модели (и получаемой в итоге стратегии) и ее сложностью.

Для создания сокращенных приблизительных моделей применялись разнообразные стратегии. Одна из таких стратегий заключается в разделении пространства состояний на множество блоков, таких, что каждый блок является стабильным. Таким образом, сохраняются те же вероятности переходов, что и в исходной модели [5]. В другом варианте предлагается разделение пространства состояний на *качественные* (qualitative) состояния, которые имеют схожие функции поощрения [15].

11.4.2 Декомпозиция

Процедура декомпозиции состоит в разделении общей (глобальной) задачи на более мелкие подзадачи, которые решаются независимо, после чего их решения объединяются [4, 10]. Существует два основных типа декомпозиции: (i) последовательная, или иерархическая, и (ii) параллельная.

Иерархические марковские процессы принятия решений поддерживают последовательную декомпозицию, при которой достигаются различные подцели в последовательном порядке, что позволяет достичь главной итоговой цели. Таким образом, на конкретном этапе выполнения в любой рассматриваемый момент времени активной является только одна (под)задача. Иерархические марковские процессы принятия решений ускоряют решение сложных задач, определяя различные подзадачи, соответствующие промежуточным целям. Достигается каждая подцель, затем все эти подпроцессы объединяются для решения общей глобальной задачи. Примерами иерархических методов являются HAM [11] и MAXQ [6].

В параллельных марковских процессах принятия решений подзадачи выполняются в параллельном режиме для решения глобальной задачи. В общем случае эти методики предполагают, что задачу можно разделить на несколько относительно независимых подзадач, которые можно решить независимо друг от друга, а затем объединить полученные решения для выполнения глобальной задачи. Если подзадачи не являются полностью независимыми, то требуется определение некоторых дополнительных условий. Например, слабо связанные *марковские процессы принятия решений* (Loosely Coupled MDP) [10] рассматривают несколько независимых подпроцессов, которые являются связанными из-за ограничений использования общего ресурса. Для решения таких задач применяется итеративная процедура на основе эвристического распределения ресурсов для каждой подзадачи. При другом подходе, описанном в [4], изначально независимо решается каждая подзадача, а при объединении полученных решений учитываются потенциально

возможные конфликты между частными (частично определенными) стратегиями. Эти конфликты разрешаются для получения глобальной, приблизительно оптимальной стратегии. Пример применения этой методики в робототехнике описан ниже в разделе 11.6.

11.5 Частично наблюдаемые марковские процессы принятия решений

В некоторых предметных областях состояние невозможно наблюдать в полной мере, поэтому доступна только частичная информация о состоянии системы. Этот тип задач известен как частично наблюдаемые марковские процессы принятия решений (partially observable Markov decision processes – POMDP). В данном случае имеются некоторые наблюдения, по которым состоянию можно дать только вероятностную оценку. Например, рассмотрим задачу из предыдущего раздела о движении робота в плоском мире, разделенном на ячейки. Возможно, робот не всегда может точно определить, в какой ячейке он находится, но может оценить вероятность нахождения в каждой ячейке по наблюдениям за окружающей средой. Точно такая же ситуация и для настоящего мобильного робота в реальных окружающих средах, где робот не может точно знать свое местоположение в конкретной окружающей среде, но может дать ему вероятностную оценку, используя свои сенсорные датчики и определенные ориентиры в этой среде.

Формально частично наблюдаемый марковский процесс принятия решений определяется как кортеж $M = <S, A, \Phi, R, O, \Omega, \Pi>$. Первые четыре элемента те же самые, что и в определении обычного марковского процесса принятия решений. O – конечное множество наблюдений $\{o_1, \ldots, o_t\}$. $\Omega : S \times O \rightarrow [0, 1]$ – функция наблюдений, определяемая как распределение вероятностей, которое дает вероятность наблюдения o с учетом того, что процесс находится в состоянии s, $P(o \mid s)$. Π – распределение начального состояния, которое определяет вероятность нахождения в состоянии s в момент времени $t = 0$.

В частично наблюдаемом марковском процессе принятия решений текущее состояние не является известным с определенностью, известно только распределение вероятностей этого состояния, которое называют *доверительным состоянием* (belief state). Поэтому для решения частично наблюдаемого марковского процесса принятия решений требуется поиск формы отображения пространства доверия в пространство действий, такой, что выбирается оптимальное действие. Это равнозначно непрерывному пространству состояний марковского процесса принятия решений. Таким образом, решение частично наблюдаемого марковского процесса принятия решений намного более трудное, чем решение обычного марковского процесса принятия решений, так

как пространство доверия (доверительное пространство) теоретически бесконечно.

Точное решение частично наблюдаемого марковского процесса принятия решений практически невыполнимо с точки зрения вычислений, за исключением очень маленьких задач. Поэтому было разработано несколько методик приближенного решения. Некоторые из основных типов методик поиска приближенных решений для частично наблюдаемого марковского процесса принятия решений перечислены ниже:

- дерево стратегий и методы динамических сетей принятия решений – предполагается задача с бесконечным интервалом, а частично наблюдаемый марковский процесс принятия решений может быть представлен как *дерево стратегий* (policy tree) (по аналогии с деревом решений) или как динамическая сеть принятия решений. Затем можно применить алгоритмы для решения этих типов моделей;
- методы выборки – функция значений вычисляется для множества точек в пространстве доверия (доверительном пространстве); используется интерполяция для определения оптимального действия с целью получения других доверительных состояний, которые не входят в множество точек выборки;
- методы аппроксимации функции значений – учитывая, что функция значений является выпуклой и кусочно-линейной, ее можно представить как множество векторов (называемых α-векторами). Таким образом, функцию значений можно аппроксимировать множеством векторов, которые доминируют над другими векторами, и таким способом найти приблизительное решение.

В разделе материалов для дополнительного чтения приведены ссылки на дополнительные источники информации о частично наблюдаемых марковских процессах принятия решений.

11.6 Приложения

Практическое применение марковских процессов принятия решений демонстрируется в двух различных предметных областях. Сначала рассматривается система помощи операторам электростанции в сложных ситуациях. Во втором примере описано регулирование и согласование набора модулей при решении сложных задач для обслуживающих роботов.

11.6.1 Управление электростанцией

Система парогенерации электростанции со смешанным циклом подает разогретый до очень высокой температуры пар в паровую тур-

бину. Система состоит из парогенератора отбора тепла (утилизатора), насоса рециркуляции, управляющих клапанов и соединительных труб. *Парогенератор отбора тепла (утилизатор)* (heat recovery steam generator – HRSG) – агрегат, способный извлекать остаточную энергию из отработанных газов газовой турбины для генерации пара под высоким давлением (*Pd*) в специальном резервуаре (*паровом коллекторе* – steam drum). *Насос рециркуляции* (recirculation pump) – это устройство, которое извлекает отработанную воду из парового коллектора для последующей непрерывной подачи в HRSG (*Ffw*). Результатом этого процесса является поток пара под высоким давлением (*Fms*), который постоянно вращает *паровую турбину* (steam turbine) для производства электроэнергии (*g*) в *силовом (электрическом) генераторе* (power generator). Главные элементы управления – *клапан подачи воды* (feedwater valve – *fwv*) и *клапан свежего пара* (main steam valve – *msv*).

При нормальной работе система управления подачей воды, состоящая из трех элементов, подает команды на клапан, контролирующий подачу воды (*fwv*), для регулирования уровня (*dl*) и давления (*pd*) в паровом коллекторе. Но это типовое устройство управления не учитывает возможность критических сбоев в цикле управления (клапаны, оборудование или любые другие устройства). Более того, не принимается во внимание, помогут ли в будущем результаты принимаемых решений увеличить срок службы парового коллектора, улучшить безопасность и производительность. Задача заключается в получении функции, которая выполняет преобразование состояний системы управления электростанцией в рекомендации для оператора. Эта функция должна учитывать все перечисленные выше аспекты.

Описанная здесь задача была смоделирована как марковский процесс принятия решений, который послужил основой для разработки инструментального средства для подготовки и помощи операторам электростанций.

11.6.1.1 Система помощи операторам электростанции

AsistO [16] – это интеллектуальная система помощи, которая предоставляет рекомендации для обучения, подготовки и помощи в онлайн-режиме для предметной области управления электростанцией. Эта система помощи объединена с системой, имитирующей работу электростанции и способной в определенной степени воспроизводить рабочий процесс электростанции со смешанным циклом.

Система AsistO основана на модели теории принятия решений, которая представляет основные элементы системы парогенерации электростанции со смешанным циклом. Основные переменные в системе парогенерации представляют ее состояние в факторизованной форме. Некоторые из этих переменных являются непрерывными, поэтому

они дискретизируются в конечное число интервалов. Действия, соответствующие операциям управления основными клапанами в этой подсистеме электростанции, в особенности действия, управляющие уровнем парового коллектора (это особо важный элемент электростанции): клапан подачи воды (*fwv*) и клапан подачи свежего пара (*msv*). Функция поощрения определена в форме кривой рекомендуемых операций для соотношения между давлением в паровом коллекторе и потоком пара (см. рис. 11.5). Главная цель – сохранение баланса между эффективностью и безопасностью электростанции. Поэтому управляющие действия должны стремиться сохранить состояние электростанции в пределах этой кривой рекомендуемых операций. При любых отклонениях необходимо вернуть состояние в безопасную точку – это схематически показано стрелками на рис. 11.5.

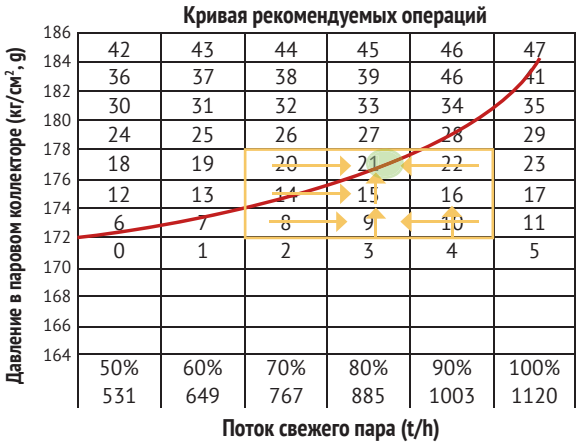


Рис. 11.5. Кривая рекомендуемых операций для парового коллектора электростанции со смешанным циклом, определяющая требуемое соотношение между давлением в паровом коллекторе и потоком пара. Стрелки схематически показывают основной принцип работы системы рекомендаций – возвращение в нормальный режим работы с безопасным соотношением (обозначено зеленым кружком), когда состояние электростанции отклоняется от кривой рекомендуемых операций

Последним определяемым элементом для завершения создания модели марковского процесса принятия решений является функция переходов. В рассматриваемом здесь приложении ее можно обучить, используя имитатор работы электростанции и выборки из пространств состояний и действий. После завершения создания марковского процесса принятия решений его можно решить для получения оптимальной стратегии, используя которую, система может давать рекомендации оператору при различных условиях функционирования электростанции.

11.6.1.2 Результаты экспериментов

Рассматривался относительно простой пример с пятью переменными состояния: *Fms*, *Ffw*, *Pd*, *g*, *d* – и четырьмя действиями: открыть/закрыть клапаны подачи воды (*fwv*) и свежего пара (*msv*) при определенных количественных показателях. Функция поощрения была определена на основе кривой рекомендуемых операций. Для обучения функции переходов были сделаны выборки динамических характеристик с использованием имитационной системы.

Сравнивались требования к памяти для простого плоского представления марковского процесса принятия решений и его факторизованного представления. Для плоского представления марковского процесса принятия решений потребовалось 589 824 параметра (значения вероятностей), в то время как для факторизованного марковского процесса принятия решений – только 758. Оптимальное решение для факторизованного марковского процесса принятия решений было получено менее чем за две минуты на обычном персональном компьютере.

Также сравнивалась система рекомендованных управляющих действий на основе марковского процесса принятия решений и обычная система автоматического управления с использованием системы, имитирующей работу электростанции. Действия, выполненные обеими системами, одинаковы, но система управления на основе марковского процесса принятия решений затратила меньше времени на возврат в точку безопасного режима работы при возникновении отклонений.

11.6.2 Согласование задач робота

Обслуживающие роботы – это в основном мобильные роботы, разработанные для помощи людям при выполнении различных действий, например для уборки дома, для помощи пожилым людям в интернатах для престарелых, для соблюдения графика приема лекарств пациентами в больницах. Для выполнения таких задач обслуживающий робот должен обладать несколькими обязательными способностями, такими как определение своего местонахождения (локализация) и навигация (выбор маршрута движения), обход препятствий, обнаружение и распознавание людей, обнаружение объектов и взаимодействие с ними и т. д. Для упрощения разработки обслуживающих роботов и обеспечения их многократного использования эти разнообразные способности можно реализовать как независимые программные модули, которые в дальнейшем могут объединяться в различных сочетаниях для решения конкретных задач, таких как доставка сообщений или предметов между сотрудниками в офисе. В таких случаях необходимо согласование различных модулей выполнения элементарных задач, желательно наиболее оптимальным способом.

Марковские процессы принятия решений предоставляют вполне подходящую рабочую среду и инструментарий для согласования задач обслуживающих роботов [7]. Пространство состояний можно определить с учетом количества переменных, определяющих ситуацию самого высокого уровня в элементарных задачах. Действия соответствуют командам, т. е. обращениям (вызовам) к различным программным модулям, например к данным навигационной системы для перемещения робота в определенную позицию в окружающей среде. Функцию поощрения можно определить на основе целей выполняемой задачи. Например, робот доставки сообщений получает некоторое небольшое поощрение, когда принимает сообщение от отправителя, и другое, более существенное поощрение после доставки сообщения получателю. Если задача моделируется как марковский процесс принятия решений, то этот марковский процесс принятия решений можно решить для получения стратегии выполнения этой конкретной задачи. В общем случае это лучше, чем обычный план действий, поскольку марковский процесс принятия решений предоставляет план для любого начального состояния (как особый тип генерального плана), и этот план более надежен с учетом неопределенности результатов различных действий.

В этой рабочей среде на основе программных модулей общего назначения и в системе их согласования на основе марковского процесса принятия решений обслуживающий робот способен относительно легко (теоретически) решать разнообразные задачи. Необходимо лишь изменять функцию поощрений марковского процесса принятия решений в соответствии с новыми целями конкретной задачи и снова решать измененный марковский процесс принятия решений для получения стратегии для другой задачи.

Кроме того, желательно, чтобы робот выполнял несколько действий одновременно, например выбор маршрута и перемещение в заданное место, обход препятствий и поиск людей – все сразу. Но если задача согласования элементарных задач робота представлена как один марковский процесс принятия решений, то необходимо учитывать все возможные сочетания одновременно выполняемых действий. Это вызывает взрывной рост пространства действий-состояний и, как следствие, значительное увеличение сложности решения марковского процесса принятия решений. К тому же становится гораздо труднее определить или обучить модель.

Другим вариантом является моделирование каждой подзадачи (элементарной задачи) как независимого марковского процесса принятия решений. Далее осуществляется решение каждого марковского процесса принятия решений для получения соответствующей оптимальной стратегии, затем эти стратегии выполняются в параллельном режиме. Такая методика известна как параллельные *марковские процессы принятия решений* (concurrent MDP) [4].

11.6.2.1 Параллельные марковские процессы принятия решений

На основе функциональной декомпозиции сложная задача разделяется на несколько подзадач (элементарных задач). Каждая подзадача представлена как марковский процесс принятия решений и решается независимо от других, а стратегии выполняются параллельно с предположением об отсутствии конфликтов между ними. Все подзадачи имеют общую цель и могут совместно использовать часть пространства состояний, представленную в факторизованной форме. Тем не менее между подзадачами все же могут возникать конфликты.

Существуют два основных типа конфликтов: (i) *конфликты ресурсов* (resource conflicts) и (ii) *конфликты поведения* (behavior conflicts). Конфликты ресурсов возникают, когда для двух действий требуется один и тот же физический ресурс (например, при управлении колесами робота), который невозможно использовать в параллельном режиме. Этот тип конфликта разрешается в режиме офлайн при помощи двухэтапного процесса [3]. На первом этапе мы получаем оптимальную стратегию для каждой подзадачи (марковского процесса принятия решений). Начальная глобальная стратегия получается при объединении локальных стратегий, так что если существует конфликт между действиями, выбранными каждым марковским процессом принятия решений для конкретного состояния, то рассматривается действие с максимальным значением, а состояние помечается как конфликтное. Это начальное решение улучшается на втором этапе с использованием метода итерации стратегии. Принимая предыдущую стратегию как начальную стратегию и учитывая только состояния, помеченные как конфликтные, можно существенно снизить сложность по времени и получить почти оптимальную глобальную стратегию.

Конфликты поведения возникают при наличии возможности выполнения двух (и более) действий в одно и то же время, но это нежелательно в данном конкретном приложении. Например, для мобильного робота нежелательно одновременное движение и передача предмета человеку (эта ситуация затруднительна и для человека). Конфликты поведения разрешаются в режиме онлайн на основе набора ограничений, определяемых пользователем. Если ограничений нет, то все действия выполняются параллельно, иначе модуль соблюдения ограничений выбирает набор действий с наивысшей ожидаемой полезностью.

11.6.2.2 Эксперименты

Эксперименты выполнялись с участием обслуживающего робота Markovito, который выполнял задачи доставки, при этом учитывались возможные конфликты. Markovito – это обслуживающий робот на основе робототехнической платформы ActivMedia PeopleBot, оснащенный

лазером, сонаром, инфракрасными сенсорными датчиками, видеокамерой, механизмом захвата и двумя компьютерами (см. рис. 11.6) [1].

Рис. 11.6. Обслуживающий робот Markovito

В рассматриваемой здесь задаче для Markovito целью являлось выполнение операций приема и доставки сообщения, предмета или сообщения и предмета по запросу пользователя. Пользователь отдает приказ на отправку сообщения/предмета, а робот запрашивает имя отправителя и получателя. Робот либо записывает сообщение, либо использует механизм захвата для удержания предмета и перемещается к месту расположения получателя для доставки. Эта задача подразделяется на пять подзадач, каждая из которых представлена как марковский процесс принятия решений:

1) *navigation – движение* – робот двигается с соблюдением мер безопасности в различных сценариях;
2) *vision – зрение* – для поиска и распознавания людей и объектов;
3) *interaction – взаимодействие* – для того, чтобы слушать пользователя и говорить с ним;
4) *manipulation – выполнение операции* – для приема и доставки предмета с соблюдением мер безопасности;

5) *expression – выражение чувств* – для выражения эмоций с помощью анимируемого лица.

Каждая подзадача представлена как факторизованный марковский процесс принятия решений:

- navigation – движение – состояния: 256 – декомпозиция в 6 переменных состояния (*located – место определено*, *has-place – на месте*, *has-path – на пути*, *direction – направление*, *see-user – вижу пользователя*, *listen-user – слушаю пользователя*). Действия: 4 (*go-to – идти к*, *stop – остановиться*, *turn – повернуться*, *move – двигаться*);
- vision – зрение – состояния: 24 – декомпозиция в 3 переменных состояния (*user-position – позиция пользователя*, *user-known – пользователь известен*, *user-in-db – пользователь в базе данных*). Действия: 3 (*look-for-people – поиск людей*, *recognize-user – распознавание пользователя*, *register-user – регистрация пользователя*);
- interaction – взаимодействие – состояния: 9216 – декомпозиция в 10 переменных состояния (*user-position – позиция пользователя*, *user-known – пользователь известен*, *listen – слушаю*, *offer-service – предложение услуги*, *has-petition – прием просьбы*, *has-message – получено сообщение*, *has-object – получен предмет*, *has-user-id – получен идентификатор пользователя*, *has-receiver-id – получен идентификатор получателя*, *greeting – приветствие*). Действия: 8 (*hear – слушать*, *offer-service – предложить услугу*, *get-message – получить сообщение*, *deliver-message – доставить сообщение*, *ask-receiver-name – запросить имя получателя*, *ask-user-name – запросить имя пользователя*, *hello – привет*, *bye – пока*);
- manipulation – выполнение операции – состояния: 32 – декомпозиция в 3 переменных состояния (*has-petition – прием просьбы*, *see-user – вижу пользователя*, *has-object – получен предмет*). Действия: 3 (*get-object – получить предмет*, *deliver-object – доставить предмет*, *move-gripper – переместить механизм захвата*);
- expression – выражение чувств – состояния: 192 – декомпозиция в 5 переменных состояния (*located – место определено*, *has-petition – прием просьбы*, *see-user – вижу пользователя*, *see-receiver – вижу получателя*, *greeting – приветствие*). Действия: 4 (*normal – обычный*, *happy – доволен*, *sad – печален*, *angry – рассержен*).

Следует отметить, что несколько переменных состояния являются общими для двух или нескольких марковских процессов принятия решений. Если представить эту задачу как единый простой марковский процесс принятия решений, то в нем будет существовать 1 179 648 состояний (с учетом всех недублирующихся переменных со-

стояний) и 1536 сочетаний действий. В целом получается около двух миллиардов комбинаций состояния-действия. Решение такой задачи в форме единственного марковского процесса принятия решений будет крайне трудным даже при современном уровне разработок для марковских процессов принятия решений.

Модель марковского процесса принятия решений для каждой подзадачи была разработана с использованием метода структурированного представления [8]. Функции переходов и поощрений были определены пользователем на основе имеющихся знаний о задаче и интуиции. Марковский процесс принятия решений для каждой подзадачи относительно прост, поэтому формулирование спецификаций вручную не вызвало особых затруднений. В этой задаче могут возникать конфликты между подзадачами, поэтому были определены методы их разрешения на основе набора ограничений, приведенных в табл. 11.1.

Таблица 11.1. Набор ограничений для робота, доставляющего сообщения (и предметы)

Действия	Ограничение	Действия
get_message (получить сообщение)	**not_during** (не во время)	поворот OR (ИЛИ) движение вперед
ask_user_name (запросить имя пользователя)	**not_before** (не перед)	recognize_user (распознавание пользователя)
recognize_user (распознавание пользователя)	**not_start** (не начинать)	avoid_obstacle (обход препятствия)
get_object (получить предмет)		directed_toward (движение в направлении)
OR (ИЛИ)	**not_during** (не во время)	OR (ИЛИ) поворота
deliver_object (доставить объект)		OR (ИЛИ) движения

Для сравнения задача доставки была решена при двух условиях: (i) без ограничений и (ii) с ограничениями.

В варианте без ограничений все действия могли выполняться параллельно, но робот продемонстрировал некоторые нежелательные образцы поведения. Например, в эксперименте робот долго не мог идентифицировать человека, желающего отправить сообщение, и только после идентификации получил возможность выполнить задачу. Это произошло потому, что марковский процесс принятия решений для *зрения* (vision) не получал качественного образа для анализа и распознавания пользователя из-за того, что марковский процесс принятия решений для *движения* (navigation) пытался в это время обойти пользователя как препятствие. Кроме того, по той же причине пользователь вынужден

был идти за роботом, чтобы обеспечить правильный ввод речевого сообщения через микрофон.

Вариант с применением ограничений позволил получить более гибкое и эффективное решение. Например, по сравнению с экспериментом без ограничений марковский процесс принятия решений для зрения при установленных ограничениях смог обнаружить и идентифицировать пользователя гораздо раньше. При активизации марковского процесса принятия решений для *взаимодействия* (interaction) действия марковского процесса принятия решений для *движения* (navigation) не выполнялись, что обеспечило эффективное взаимодействие и распознавание.

В среднем версия с ограничениями выполнила около 50% временны́х шагов, требуемых для версии без ограничений для завершения выполнения задачи. В итоге метод разрешения конфликтов с помощью введения ограничений не только позволил роботу демонстрировать ожидаемое поведение, но также обеспечил более устойчивую производительность, устранил конфликты и позволил существенно сократить время, требуемое для завершения задачи.

11.7 Материалы для дополнительного чтения

Книга Путермана (Puterman) [13] представляет собой превосходный справочник по марковским процессам принятия решений, включая их формальное определение и разнообразные методики решения. Недавний обзор моделей теории принятия решений и их приложений приведен в [18]. В справочнике [12] можно найти обзоры марковских процессов принятия решений и частично наблюдаемых марковских процессов принятия решений. Представление марковских процессов принятия решений с использованием алгебраических диаграмм решений и SPUDD описано в [8]. Обзор разнообразных методов решения частично наблюдаемых марковских процессов принятия решений представлен в [17].

11.8 Задания и упражнения

1. Для примера мира, состоящего из прямоугольных ячеек на рис. 11.1, считать, что каждая ячейка является состоянием и существуют четыре возможных действия: *up* (вверх), *down* (вниз), *left* (влево), *right* (вправо). Завершить описание (спецификацию) марковского процесса принятия решений, включая функции переходов и поощрений.

2. Решить марковский процесс принятия решений из задания 1 методом итераций значения. Инициализировать значения для каждого

состояния для немедленного поощрения и показать, как изменяются значения на каждой итерации.

3. Решить марковский процесс принятия решений из задания 1 методом итераций стратегии. Инициализировать стратегию значением *up* (вверх) для всех состояний. Показать, как изменяется стратегия на каждой итерации.

4. Определить факторизованное представление для примера мира, состоящего из прямоугольных ячеек (задание 1), считая, что состояние представлено в форме двух переменных *row* (строка) и *column* (столбец). Определить функцию перехода как двухэтапную динамическую байесовскую сеть, а функцию поощрения – как дерево решений.

5. Рассмотреть сценарий с примером мира, состоящего из прямоугольных ячеек (задание 1), в котором сетка делит мир на несколько комнат и коридор, соединяющий эти комнаты. Разработать иерархическое решение для движения робота в этом сценарии, определив один марковский процесс принятия решений для перехода из любой ячейки в комнате к двери, которая ведет в коридор, второй марковский процесс принятия решений, который позволяет пройти по коридору от одной двери к другой, и третий марковский процесс принятия решений для прохода от двери, ведущей в комнату, к определенной цели в этой комнате. Полностью определить каждый марковский процесс принятия решений, а затем управляющий механизм верхнего уровня, который координирует стратегии более низкого уровня.

6. Разработать решение задачи перемещения по сетке на основе двух параллельных марковских процессов принятия решений: один для определения пути движения к цели (без учета препятствий), второй для обхода препятствий. Определить модели для каждой из этих двух подзадач и способ объединения двух полученных стратегий для нахождения пути к цели с обходом препятствий. Позволяет ли это решение всегда приходить к цели или возможно возникновение «ловушки» в локальных максимумах?

7. Доказать, что решение по формуле Беллмана с использованием метода итераций значения всегда сходится.

8. *** Разработать обобщенную программу, которая реализует алгоритм итераций значения.

9. *** Разработать программу для решения задачи о движении робота в мире, состоящем из прямоугольных ячеек, включая графический

интерфейс для определения размера сетки ячеек, препятствий и местонахождения цели. Определить высокое положительное поощрение для целевой ячейки, высокие отрицательные поощрения (штрафы) для препятствий и небольшие положительные поощрения для всех прочих ячеек. Представить задачу как марковский процесс принятия решений и получить оптимальную стратегию, используя метод итераций значения.

10. *** Когда модель для марковского процесса принятия решений неизвестна, альтернативой является обучение оптимальной стратегии методом проб и ошибок с помощью обучения с подкреплением (reinforcement learning). Изучить основные алгоритмы обучения с подкреплением, такие как Q-обучение, и реализовать программу для обучения стратегии для решения задачи о движении робота в мире, состоящем из прямоугольных ячеек (рис. 11.1) (использовать те же поощрения, что и в исходном примере). Совпадает ли полученная стратегия с решением модели марковского процесса принятия решений?

Ссылки на источники

1. Avilés-Arriaga, H. H., Sucar, L. E., Morales, E. F., Vargas, B. A., Corona, E. Markovito: a flexible and general service robot. In: Liu, D., Wang, L., Tan, K. C. (eds.) Computational Intelligence in Autonomous Robotic Systems, p. 401–423. Springer, Berlin (2009).
2. Bellman, R. Dynamic Programming. Princeton University Press, Princeton (1957).
3. Corona, E., Morales, E. F., Sucar, L. E. Solving policy conflicts in concurrent Markov decision processes. In: ICAPSWorkshop on Planning and Scheduling Under Uncertainty, Association for the Advancement of Artificial Intelligence (2010).
4. Corona, E., Sucar, L. E. Task coordination for service robots based on multiple Markov decision processes. In: Sucar, L. E., Hoey, J., Morales, E. (eds.) Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions. IGI Global, Hershey (2011).
5. Dean, T., Givan, R. Model minimization in Markov decision processes. In: Proceedings of the 14th National Conference on Artificial Intelligence (AAAI), p. 106–111 (1997).
6. Dietterich, T. Hierarchical reinforcement learning with the MAXQ value function decomposition. J. Artif. Intell. Res. 13, 227–303 (2000).

7. Elinas, P., Sucar, L., Reyes, A., Hoey, J. A decision theoretic approach for task coordination in social robots. In: Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN), p. 679–684 (2004).
8. Hoey, J., St-Aubin, R., Hu, A., Boutilier, C. SPUDD: stochastic planning using decision diagrams. In: Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI), p. 279–288 (1999).
9. Li, L., Walsh, T. J., Littman, M. L. Towards a unified theory of state abstraction for MDPs. In: Proceedings of the Nineth International Symposium on Artificial Intelligence and Mathematics, p. 21–30 (2006).
10. Meuleau, N., Hauskrecht, M., Kim, K. E., Peshkin, L., Kaelbling, L. P., Dean, T., Boutilier, C. Solving very large weakly coupled Markov decision processes. In: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI), p. 165–172 (1998).
11. Parr, R., Russell, S. J. Reinforcement learning with hierarchies of machines. In: Proceeding of the Advances in Neural Information Processing Systems (NIPS) (1997).
12. Poupart, P. An introduction to fully and partially observable Markov decision processes. In: Sucar, L.E., Hoey, J., Morales, E. (eds.) Decision TheoryModels for Applications in Artificial Intelligence: Concepts and Solutions. IGI Global, Hershey (2011).
13. Puterman, M. L. Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (1994).
14. Quinlan, J. R. Induction of decision trees. Mach. Learn. 1 (1), 81–106 (1986).
15. Reyes, A., Sucar, L. E., Morales, E. F., Ibargüngoytia, P. Hybrid Markov Decision Processes. Lecture Notes in Computer Science, vol. 4293. Springer, Berlin (2006).
16. Reyes, A., Sucar, L. E., Morales, E. F. AsistO: a qualitative MDP-based recommender system for power plant operation. Computacion y Sistemas 13 (1), 5–20 (2009).
17. Ross, S., Pineau, J., Paquet, S., Chaib-draa, B. Online planning algorithms for POMDPs. J. Artif. Intell. Res. 32, 663–704 (2008).
18. Sucar, L. E., Hoey, J., Morales, E. Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions. IGI Global, Hershey (2011).

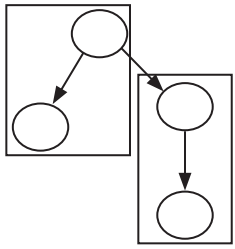
Часть IV

Реляционные и причинно-следственные модели

В четвертой, и последней, части книги описываются два интересных расширения вероятностных графовых моделей: реляционные вероятностные модели и причинно-следственные графовые модели. Реляционные вероятностные модели наращивают возможности представления стандартных вероятностных графовых моделей, объединяя функциональную мощь логики первого порядка со средствами обоснования в условиях неопределенности вероятностных моделей. Причинно-следственные графовые модели выходят за рамки представления вероятностных зависимостей и позволяют установить связи между причинами и следствиями (результатами) на основе все той же рабочей среды и инструментария графовых моделей.

Глава 12

Реляционные вероятностные графовые модели



12.1 Введение

Стандартные вероятностные графовые модели, которые рассматривались до настоящего момента, должны представлять в явной форме каждый объект в предметной области, поэтому по существу они равнозначны по своим средствам логического выражения логике высказываний (пропозициональной логике). Но существуют задачи, в которых количество объектов (переменных) может существенно возрастать, так что потребуется более выразительное (компактное) представление. Например, предположим, что необходимо смоделировать знания студента по некоторой теме (эта задача известна как моделирование обучающегося или в более общем смысле моделирование пользователя) и что нужно включить в эту модель всех студентов некоторого колледжа, где каждый студент обучается по нескольким темам (курсам). Если моделировать эту задачу с использованием обычной вероятностной графовой модели, например как байесовскую сеть, то, вероятнее всего, модель станет огромной и трудной для понимания и хранения. Вместо этого был бы более эффективным подход, при котором можно было бы получить обобщенную модель, которая представляет отношения зависимостей для каждого студента S и любого курса C. Затем эту модель можно было бы параметризовать для конкретных вариантов. Это можно сделать, используя логику предикатов, но стандартные логические представления

не учитывают неопределенность. Поэтому требуется формализация, которая объединяет логику предикатов и вероятностные модели.

Реляционные вероятностные графовые модели (relational probabilistic graphical models – RPGM) объединяют выразительную мощь логики предикатов с возможностями обоснования в условиях неопределенности вероятностных графовых моделей. Некоторые из этих моделей расширяют вероятностные графовые модели, такие как байесовские сети или марковские сети, представляя объекты, их атрибуты и связи (отношения) с другими объектами. При иных подходах расширяются возможности представления на основе логики, чтобы включить в модель фактор неопределенности, посредством описания распределения вероятностей в дополнение к логическим формулам.

Были предложены различные методы формализации для объединения логики и вероятностных графовых моделей. Ниже предлагается система классификации реляционных вероятностных графовых моделей (частично на основе [5]):

1. Расширения логических моделей
   a) Неориентированные графовые модели
      i. Марковские логические сети
   b) Ориентированные графовые модели
      i. Байесовские логические программы
      ii. Байесовские логические сети
2. Расширения вероятностных моделей
   a) Неориентированные графовые модели
      i. Реляционные марковские сети
      ii. Реляционные сети зависимостей
   b) Ориентированные графовые модели
      i. Реляционные байесовские сети
      ii. Вероятностные реляционные модели
3. Расширения языков программирования
   a) Программы стохастической логики
   b) Программирование вероятностной индуктивной логики
   c) Байесовская логика (BLOG)
   d) Язык вероятностного моделирования (IBAL)

В этой главе будут рассматриваться две реляционные вероятностные графовые модели. Во-первых, *вероятностные реляционные модели* (probabilistic relational models) [3], которые расширяют байесовские сети для включения объектов и связей (отношений), как в реляционных базах данных. Во-вторых, *марковские логические сети* (Markov logic

networks) [11], которые добавляют весовые коэффициенты в логические формулы и могут рассматриваться как расширение марковских сетей. Сначала будет представлен краткий обзор логики первого порядка, затем описываются две вышеназванные реляционные вероятностные методики. В конце главы демонстрируется практическое применение этих методик в двух предметных областях: моделирование обучающегося (студента) и визуальное распознавание объектов.

12.2 Логика

Логика – это весьма тщательно продуманный язык представлений с точно определенным синтаксисом и семантическими (смысловыми) компонентами. В этом разделе приведено лишь краткое введение в логику, а для получения более подробной информации необходимо обратиться к разделу материалов для дополнительного чтения. Раздел начинается с определений логики высказываний (пропозициональной логики), затем мы перейдем к логике первого порядка.

12.2.1 Логика высказываний

Логика высказываний (пропозициональная логика – propositional logic) позволяет давать обоснование выражениям или высказываниям (propositions), которые являются истинными (*True* – T) или ложными (*False* – F). Например, «*Джо – студент технологического факультета*» (*Joe is an engineering student*). Высказывания обозначаются заглавными (прописными) буквами, например *P*, *Q*, …, известными как атомарные высказывания, или просто атомы (atoms). Высказывания объединяются с использованием логических связок или логических операторов, в результате получаются так называемые составные высказывания (compound propositions). Ниже перечислены логические операторы:

- отрицание: ¬;
- конъюнкция: ∧
- дизъюнкция: ∨
- импликация: →;
- двойная импликация: ↔.

Например, если *P* = «Джо – студент технологического факультета» и *Q* = «Джо молод» (Joe is young), то $P \wedge Q$ означает «Джо – студент технологического факультета AND (И) Джо молод».

Атомы и логические связки можно комбинировать в соответствии с набором синтаксических правил. Допустимые корректные сочетания называются *правильно построенными формулами* (well-formed formulas – WFF). Правильно построенная формула в логике высказываний – это выражение, полученное в соответствии со следующими правилами:

1) любой атом является правильно построенной формулой;
2) если P – правильно построенная формула, то $\neg P$ – правильно построенная формула;
3) если P и Q – правильно построенные формулы, то $P \wedge Q$, $P \vee Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ – правильно построенные формулы;
4) никакие другие формулы не являются правильно построенными формулами.

Например, $P \rightarrow (Q \wedge R)$ – правильно построенная формула, но $\rightarrow P$ и $\vee$ Q не являются правильно построенными формулами.

Смысл (семантика) логической формулы может быть выражен с помощью функции, которая возвращает значение *True* (истина) или *False* (ложь) для этой формулы при каждой возможной интерпретации (истинные значения атомов в этой формуле). В логике высказываний *интерпретация* (interpretation) может быть представлена в форме *таблицы истинности* (truth table). В табл. 12.1 показаны таблицы истинности для основных логических операторов. Интерпретация, которая присваивает значение «истина» формуле *F*, является моделью (model) формулы *F*.

Таблица 12.1. Таблицы истинности для логических операторов (связок)

P	Q	¬P	P ∧ Q	P ∨ Q	P → Q	P ↔ Q
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Формула F является логическим следствием (logical consequence) множества формул $\mathbf{G} = \{G_1, G_2, \ldots, G_n\}$, если при каждой интерпретации, для которой **G** истинно, формула F истинна. Это обозначается как $G \models F$.

Логика высказываний не может выражать общие свойства, как, например, «*все студенты технологического факультета молоды*» (all engineering students are young). Для этого необходима логика первого порядка.

12.2.2 Логика предикатов первого порядка

Рассмотрим два высказывания: «Все студенты технического университета специализируются по курсу технологии» (All students in a technical university are engineering majors) и «Конкретный человек *t* из конкретного университета $\mathcal{F}$ специализируется по курсу информатики» (A particular person *t* of a particular university $\mathcal{F}$ is a computer science

major). Первое высказывание объявляет свойство, которое применяется ко всем, кто учится в техническом университете, тогда как второе высказывание применимо только к конкретному человеку из конкретного университета. Логика первого порядка позволяет учитывать такие различия.

Выражения или формулы в логике предикатов первого порядка формируются с использованием четырех типов символов: постоянных, или *констант* (constants), *переменных* (variables), *функций* (functions) и *предикатов* (predicates). Постоянные символы представляют конкретные объекты, значимые в рассматриваемой предметной области (например, люди, курсы, университеты и т. д.). Переменные символы охватывают группы объектов в предметной области. Переменные, которые могут быть аргументами предиката или функции, обозначаются строчными буквами (буквами нижнего регистра), т. е. x, y, z.

Предикаты – это выражения, которые могут быть истинными (*True*) или ложными (*False*). Предикаты содержат несколько аргументов. Предикаты могут представлять отношения (связи) между объектами в предметной области (например, *Above*) или атрибуты объектов (например, *IsRed* (красный)). Если число аргументов предиката равно нулю, то он является атомом, как в логике высказываний. Предикаты обозначаются заглавными (прописными) буквами (буквами верхнего регистра), например $P(x)$.

Функции (функциональные символы) (например, *neighborOf* (сосед)) представляют отображения (mappings) кортежей объектов в объекты. Функции представлены буквами нижнего регистра (строчными) и также могут иметь несколько аргументов. Например, $f(x)$ представляет функцию с одним аргументом x. Функция без аргументов является постоянной (constant).

Логика предикатов включает логические связки (операторы) из логики высказываний, а также другие операторы, известные как *кванторы* (quantifiers):

- квантор всеобщности: $\forall x$ (для всех x);
- квантор существования: $\exists x$ (существует x).

Терм (term) – это константа, переменная или функция от термов. Атомарная формула (atomic formula) или атом (atom) – это предикат, который имеет в качестве аргументов N термов.

Теперь можно переписать пример, приведенный в начале этого подраздела, в терминах логики первого порядка. Если формула $M(t_x, \mathcal{F})$ представляет специализацию студента t_x в техническом университете $\mathcal{F}$, то высказывание «Все студенты технического университета специализируются по курсу технологии» (All students in a technical university are engineering majors) можно переписать следующим образом:

$$\forall t_x \, \forall \mathcal{F} : M(t_x, \mathcal{F}) = engineering. \quad (12.1)$$

Как и в логике высказываний, в логике предикатов первого порядка существует набор синтаксических правил, которые определяют, какие выражения являются правильно построенными формулами:

1. Любой атом является правильно построенной формулой.
2. Если P – правильно построенная формула, то $\neg P$ – правильно построенная формула.
3. Если P и Q – правильно построенные формулы, то $P \wedge Q$, $P \vee Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ – правильно построенные формулы.
4. Если предикат P является правильно построенной формулой и x – свободная переменная в предикате P, то $\forall xP$ и $\exists xP$ – правильно построенные формулы.
5. Никакие другие формулы не являются правильно построенными формулами.

Грубо говоря, база знаний (knowledge base) – это множество утверждений или формул в формате логики первого порядка [4].

$\mathcal{L}$-интерпретация определяет, какими символами представлены конкретные объекты, функции и связи (отношения) в предметной области. Переменные и константы могут быть типизированы, в этом случае диапазон значений охватывает только объекты соответствующего типа, а константы могут представлять лишь объекты соответствующего типа. Например, переменная x может включать в диапазон значений университеты (например, общественные университеты, частные университеты и т. д.), а константа может представлять некоторый университет или специально подобранное множество университетов (например, u_2, ($u_6 \cup u_2$) и т. д.). Атомарная формула (или просто атом) – это символ-предикат, примененный к кортежу термов: $Near(u_4, u_1)$.

В стандартной логике предикатов все предикаты либо истинны (*True*), либо ложны (*False*), поэтому их невозможно напрямую использовать в условиях вероятностной неопределенности. Например, если неизвестно, обучается ли некоторый человек в конкретном университете, то можно лишь сказать, что $Univ(p) = u_1$ OR $Univ(p) = u_2$, но невозможно точно сказать, какой вариант более вероятен: обучение в университете u_1 или u_2.

Чтобы получить в свое распоряжение выразительную мощь логики предикатов и в то же время получить возможность представления и обоснования (в вероятностных терминах), необходимо объединить логику предикатов с вероятностными моделями в единой рабочей среде. В следующем разделе будут описаны две такие рабочие среды.

12.3 Вероятностные реляционные модели

Вероятностные реляционные модели (probabilistic relational models) [6] представляют собой расширение байесовских сетей, которое обеспечивает более выразительное объектно-ориентированное представление, упрощая получение знаний. Кроме того, они позволяют без затруднений распространить применение модели на другие предметные области. При очень большой модели в любой момент времени рассматривается только ее часть, поэтому сложность логического вывода уменьшается.

Основными сущностями в вероятностных реляционных моделях являются объекты или элементы предметной области. Объекты в предметной области разделяются на множество непересекающихся классов $X_1, \ldots, X_n$. Каждый класс связан с множеством атрибутов $A(X_i)$. Каждый атрибут $A_{ij} \in A(X_i)$ (т. е. атрибут j класса i) принимает значения из некоторой фиксированной области значений $V(A_{ij})$. $X : A$ обозначает атрибут A объекта из класса X [6]. Определяется также множество связей (отношений) R_j между классами. Бинарная связь $R(X_i, X_j)$ между классами X_i и X_j может быть обозначена как *слот* (slot) класса X_i. Классы и связи (отношения) определяют *схему* (schema) модели. Затем вероятностная реляционная модель определяет распределение вероятностей во множестве экземпляров схемы, в частности распределение для атрибутов объектов конкретной модели.

Модель зависимостей определяется на уровне классов, что позволяет использовать ее для любого объекта класса. Вероятностные реляционные модели явно применяют реляционную структуру модели, поэтому атрибут объекта будет зависеть от некоторых атрибутов связанных с ним объектов. Вероятностная реляционная модель определяет распределение вероятностей, используя те же теоретические базовые принципы, что и в байесовских сетях. Каждая из случайных переменных в вероятностной реляционной модели, т. е. атрибуты $x.a$ отдельных объектов x, подвержена прямому влиянию других атрибутов, которые являются их родителями. Таким образом, вероятностная реляционная модель для каждого атрибута определяет множество родителей, которые напрямую воздействуют на этот атрибут, и локальную вероятностную модель, которая устанавливает вероятностные параметры.

Существуют два основных различия между вероятностными реляционными моделями и байесовскими сетями [6]: (i) в вероятностной реляционной модели модель зависимостей определяется на уровне классов, что позволяет использовать ее для любого объекта класса; (ii) вероятностная реляционная модель явно использует реляционную структуру модели, позволяя установить зависимость атрибута объекта от атрибутов связанных с ним объектов.

Пример вероятностной реляционной модели для предметной области «Образование» на основе материала [6] показан на рис. 12.1. В этом примере 4 класса с 2 атрибутами у каждого класса:

- профессор (professor): преподавательская_квалификация (teaching-ability), востребованность (popularity);
- студент (student): интеллект (intelligence), успеваемость (ranking);
- курс (course): рейтинг_посещаемости (rating), сложность (difficulty);
- список_студентов (registration): удовлетворенность_качеством_обучения (satisfaction), общая_оценка (grade).

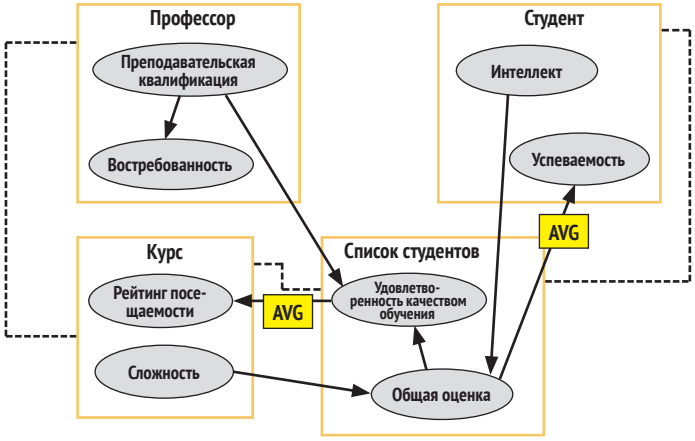


Рис. 12.1. Пример структуры вероятностной реляционной модели для предметной области «Образование». Штриховые ребра представляют связи между классами, стрелки соответствуют вероятностным зависимостям. Метка AVG в связи означает, что условные вероятности зависят от этой переменной

Это представление позволяет определить два типа атрибутов каждого класса: (i) информационные переменные и (ii) случайные переменные. К случайным относятся переменные, которые связаны в некоторую разновидность байесовской сети, называемую *скелетом* (skeleton). Из этого скелета можно сгенерировать разнообразные байесовские сети с учетом других переменных в используемой модели. Например, в модели студента, описанной в разделе 12.5, определяется общий скелет для экспериментов, из которого генерируются конкретные экземпляры для каждого эксперимента. Это придает модели бóльшую гибкость и обобщенность и упрощает накопление знаний. Кроме того, логический вывод становится более эффективным, поскольку в каждом конкретном варианте используется только часть общей модели.

Распределение вероятностей для скелетов определяется так же, как в байесовских сетях. Таким образом, вероятностная реляционная модель определяет для каждого атрибута *x.a* множество родителей, которые

оказывают прямое воздействие на этот атрибут, и локальную вероятностную модель, которая, в свою очередь, определяет условную вероятность того же атрибута с учетом его родителей. Чтобы локальные модели гарантированно определяли логически согласованное глобальное распределение вероятностей, структура зависимостей более низкого уровня должна быть ациклической, как в байесовской сети. Затем определяется совместное распределение вероятностей, равное произведению локальных распределений, как и в байесовских сетях.

12.3.1 Логический вывод

Механизм логического вывода в вероятностных реляционных моделях тот же, что и в байесовских сетях, когда создается экземпляр модели для конкретных объектов в предметной области. Но вероятностные реляционные модели могут использовать преимущества двух свойств, чтобы повысить эффективность логического вывода. Эти свойства становятся явными в представлении вероятностной реляционной модели в противоположность байесовским сетям, где эти свойства являются всего лишь неявными.

Первое свойство – локальность влияния, т. е. большинство атрибутов в основном зависят от атрибутов того же класса при наличии нескольких зависимостей между классами. Методы вероятностного логического вывода могут воспользоваться преимуществом этого свойства локальности, применяя принцип «разделяй и властвуй».

Другим свойством, которое может повысить эффективность логического вывода, является возможность повторного использования. В вероятностной реляционной модели обычно имеется несколько объектов одного класса с аналогичной структурой и параметрами. Поэтому когда логический вывод выполняется для одного объекта, его можно повторно использовать для других родственных объектов.

12.3.2 Обучение

С учетом того, что вероятностные реляционные модели используют те же базовые принципы, что и байесовские сети, методики обучения, разработанные для байесовских сетей, можно применять и для вероятностных реляционных моделей. Алгоритм ожидаемой максимизации был доработан для обучения параметров вероятностной реляционной модели, а методики обучения структуры были разработаны для обучения структуры зависимостей по реляционной базе данных [3].

12.4 Марковские логические сети

В противоположность вероятностным реляционным моделям марковские логические сети (Markov logic networks) начинают работу с логи-

ческого представления, добавляя весовые коэффициенты (weights) в формулы для учета неопределенности.

В логике $\mathcal{L}$-интерпретация, которая нарушает корректность формул, принятых в базе знаний, имеет нулевую вероятность. Это означает, что появление $\mathcal{L}$-интерпретации невозможно, поскольку все «возможные миры» (possible worlds) обязательно должны быть согласованы с базой знаний. В марковских логических сетях это предположение ослаблено. Если интерпретация нарушает целостность базы знаний, то она (интерпретация) имеет меньшую вероятность, чем другие интерпретации, которые не допускают нарушений. Меньшая вероятность означает, что интерпретация имеет ненулевую вероятность. В марковской логической сети весовой коэффициент добавляется в каждую формулу, чтобы отобразить, насколько строгим является ограничение: при более высоких весовых коэффициентах происходят более значимые изменения вероятности вне зависимости от того, соответствует ли данная интерпретация этой формуле или не соответствует.

Прежде чем дать формальное определение марковской логической сети, необходимо краткое напоминание о марковских сетях (см. главу 6). Марковская сеть – это модель совместного распределения для множества переменных $X = (X_1, X_2, \ldots, X_n) \in \mathfrak{X}$. Марковская сеть сформирована из неориентированного графа G и множества функций потенциалов ϕ_k. Связанный с ней граф содержит узлы для каждой переменной, а модель содержит функции потенциалов для каждой клики в соответствующем графе. Совместное распределение вероятностей, представленное марковской сетью, вычисляется по формуле

$$P(X = x) = \frac{1}{Z}\prod_k \phi_k(x_{|k|}), \tag{12.2}$$

где $x_{|k|}$ – состояние k-й клики, а Z – статистическая сумма.

Обычно марковские сети могут быть также представлены с использованием логлинейных (логарифмически линейных) моделей, в которых функция потенциалов каждой клики заменяется экспоненциально взвешенной суммой:

$$P(X = x) = \frac{1}{Z}\exp\sum_j w_j f_j(x), \tag{12.3}$$

где w_j – весовой коэффициент (действительное числовое значение), а f_j – принятая для наших целей бинарная формула $f_j(x) \in \{0, 1\}$.

Теперь можно дать формальное определение марковской логической сети [1].

Марковская логическая сеть L – это множество пар (F_i, w_i), где F_i – формула логики первого порядка, а w_i – действительное число. Вместе с конечным множеством констант $C = \{c_1, c_2, \ldots, c_{|C|}\}$ она определяет марковскую сеть $M_{L,C}$ с учетом формул 12.2 и 12.3:

1) марковская сеть $M_{L,C}$ содержит по одному бинарному узлу для каждого возможного обоснования каждой формулы, имеющейся в марковской логической сети L. Значение этого узла равно 1, если атом обоснования истинный (*true*), или 0, если атом обоснования ложный (*false*);
2) марковская сеть $M_{L,C}$ содержит по одному признаку для каждого возможного обоснования каждой формулы F_i в марковской логической сети L. Значение этого признака равно 1, если формула обоснования истинна (*true*), или 0, если формула обоснования ложна (*false*). Весовой коэффициент этого признака w_i связан с формулой F_i в L.

Марковские логические сети представляют собой обобщение марковских сетей, поэтому их можно рассматривать как шаблоны для компоновки марковских сетей. Если взять некоторую марковскую логическую сеть и множество различных констант, то можно создавать разнообразные марковские сети, которые известны как марковские сети обоснования (ground Markov networks). Совместное распределение вероятностей для марковской сети обоснования определяется таким же способом, как и для марковской сети с применением формулы 12.3.

Графовая структура марковской логической сети основана на ее определении. Ребро между двумя узлами марковской логической сети существует, если соответствующие атомы обоснования появляются вместе как минимум в одной формуле обоснования в базе знаний L. Например [1], рассмотрим следующую марковскую логическую сеть, состоящую из двух логических формул:

$$\forall x\ Курение(x) \rightarrow Рак(x);$$
$$\forall x\ \forall y\ Друзья(x) \rightarrow (Курение(x) \leftrightarrow Курение(y)).$$

Если переменным x и y присвоены постоянные значения A и B, то получим структуру, показанную на рис. 12.2.

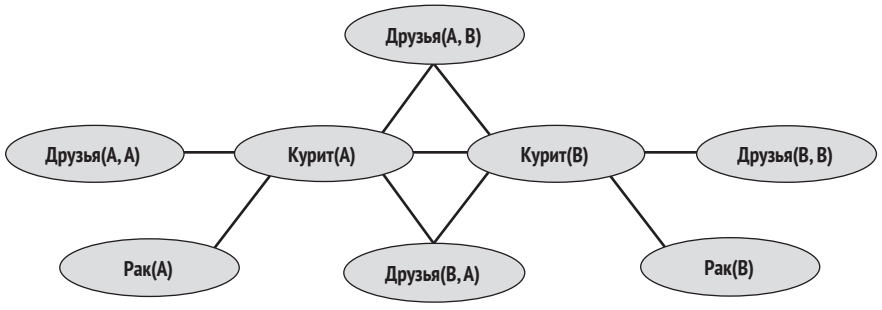


Рис. 12.2. Структура марковской логической сети обоснования, полученной из двух логических формул примера (см. в тексте). (Схема основана на [1])

12.4.1 Логический вывод

Логический вывод в марковской логической сети состоит в оценке вероятности логической формулы F_1 с учетом того, что другая формула (или несколько формул) F_2 является истинной. Таким образом, необходимо вычислить $P(F_1 \mid F_2, L, C)$, где L – марковская логическая сеть, состоящая из множества взвешенных логических формул, а C – множество постоянных значений. Для вычисления этой вероятности можно оценить пропорциональное отношение возможных миров, в которых F_1 и F_2 истинны, ко всем возможным мирам, в которых содержится F_2. При таком вычислении вероятность каждого возможного мира принимается в соответствии с весовыми коэффициентами формул и структурой соответствующей марковской сети обоснования.

Непосредственное выполнение описанного выше вычисления связано с чрезвычайно высокой стоимостью вычислений, поэтому такой подход становится практически недоступным, за исключением чрезвычайно малых моделей. Для применения более эффективного способа вычислений можно использовать некоторые другие методики вероятностного логического вывода. Одним из таких вариантов является использование стохастической имитации: с помощью выборок экземпляров возможных миров можно получить оценку требуемой вероятности, например применяя метод Монте-Карло с использованием марковских цепей.

Еще один вариант – формулирование некоторых разумных предположений о структуре логических формул для упрощения процесса логического вывода. В [1] авторы описывают разработку алгоритма эффективного логического вывода для случая, когда F_1 и F_2 являются конъюнкциями литералов обоснования.

12.4.2 Обучение

При обучении марковской логической сети рассматриваются два аспекта, как и при обучении байесовских сетей. Первый аспект – обучение логических формул (структуры), второй аспект – обучение весовых коэффициентов для каждой формулы (параметров).

Для обучения логических формул можно применять методики из области индуктивного логического программирования (ИЛП; inductive logic programming). В области ИЛП существуют разнообразные методики, позволяющие выводить логические связи (отношения) из данных, обычно учитывая при этом некоторые базовые знания. Для получения более подробной информации см. раздел материалов для дополнительного чтения.

Весовые коэффициенты в логических формулах можно обучать по реляционной базе данных. В основном весовые коэффициенты в формулах пропорциональны соответствующему им количеству истинных обоснований в данных с учетом их ожидания в соответствии с моделью. Подсчет количества истинных обоснований формул также может привести к чрезвычайно дорогостоящим вычислениям в крупных предметных областях. При другом подходе вычисляется оценка на основе выборок обоснований формул и верификация их истинности по имеющимся данным.

Пример практического применения марковской логической сети в приложении для визуального распознавания объектов представлен в разделе 12.5.

12.5 Приложения

Практическое применение двух классов реляционных вероятностных графовых моделей, представленных в предыдущих разделах, будет продемонстрировано в двух различных предметных областях. Сначала мы рассмотрим, как можно создать в некотором роде обобщенную модель студента для виртуальных лабораторных занятий на основе вероятностной реляционной модели. Затем будет описано использование марковской логической сети для представления визуальных грамматик для распознавания объектов.

12.5.1 Моделирование студента

Особенно сложной областью моделирования студента являются виртуальные лабораторные занятия. Виртуальная лаборатория предоставляет имитационную модель некоторого оборудования, с которым может взаимодействовать студент, обучаясь на практике. Преподаватель выступает в роли виртуального ассистента в такой лаборатории, предоставляя помощь и давая полезные советы пользователю, а также устанавливает уровень сложности экспериментов в соответствии с уровнем подготовки студента. В общем случае нежелательно беспокоить студента вопросами и тестами для уточнения и обновления его модели. Когнитивное состояние должно быть получено исключительно на основе взаимодействий студента с виртуальной лабораторией и результатов экспериментов. Для этого требуется модель студента. Эта модель по взаимодействиям студента с лабораторией логически выводит когнитивное состояние. На основе такой модели интеллектуальный преподаватель (также виртуальный) может давать советы и подсказки, персонально ориентированные на конкретного студента [13].

12.6 Вероятностная реляционная модель студента

Вероятностные реляционные модели обеспечивают компактное и естественное представление для моделирования студента. Вероятностные реляционные модели позволяют представить каждого присутствующего студента в одной и той же модели. Каждый класс представляет множество параметров нескольких студентов, как в базах данных, но модель также включает вероятностные зависимости между классами для каждого студента.

Для применения вероятностных реляционных моделей к моделированию студента необходимо определить основные объекты в этой предметной области. Обобщенная модель студента была спроектирована со специализацией для виртуальных лабораторий, начиная со структуры самого верхнего уровня на уровне классов и заканчивая специализированными байесовскими сетями для различных экспериментов на нижнем уровне. Как показано на рис. 12.3, были определены основные классы, связанные со студентами и экспериментами. В рассматриваемом здесь примере определено 8 классов с несколькими атрибутами для каждого класса, описанными ниже:

- студент: student-id (идентификатор студента), student-name (имя студента), major (специализация), quarter (семестр), category (категория);
- раздел знаний: student-id (идентификатор студента), knowledge-theme-id (идентификатор раздела знаний), knowledge-theme-known (известный раздел знаний);
- подраздел знаний: student-id (идентификатор студента), knowledge-sub-theme-id (идентификатор подраздела знаний), knowledge-sub-theme-known (известный подраздел знаний);
- элементы знаний: student-id (идентификатор студента), knowledge-item-id (идентификатор элемента знаний), knowledge-item-known (известный элемент знаний);
- теоретические основы: previous-course (предыдущий курс), grade (общая оценка);
- поведение студента: student-id (идентификатор студента), experiment-id (идентификатор эксперимента), behavior-var1, behavior-var2, …;
- результаты экспериментов: student-id (идентификатор студента), experiment-id (идентификатор эксперимента), experiment-repetition (повторение эксперимента), result-var1, result-var2, …;

- эксперименты: experiment-id (идентификатор эксперимента), experiment-description (описание эксперимента), exp-var1, exp-var2, ….

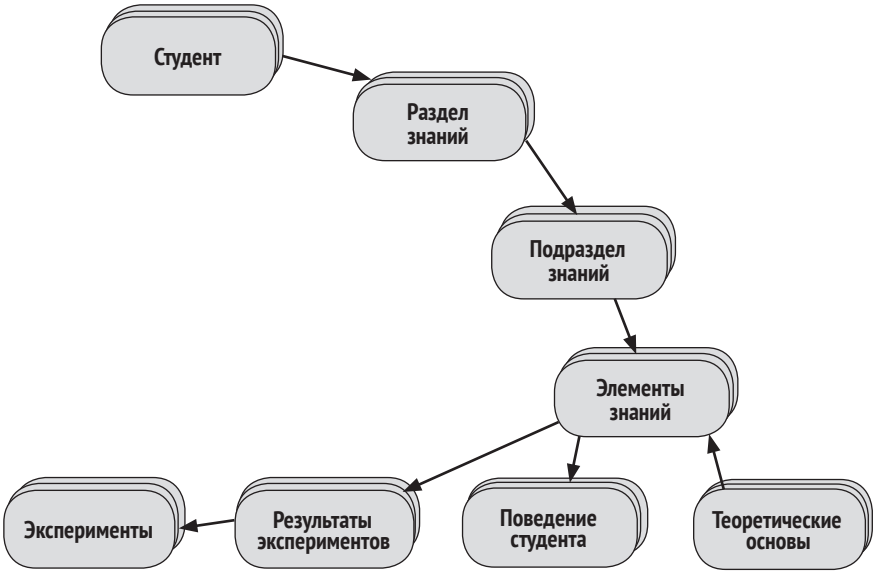


Рис. 12.3. Представление высокого уровня структуры вероятностной реляционной модели для модели студента, отображающей основные классы и их отношения

Модель зависимостей определяется на уровне классов, что позволяет использовать ее для любого объекта класса. Некоторые атрибуты в этой модели представляют вероятностные значения. Это означает, что атрибут представляет случайное значение, которое связано с другими атрибутами в том же классе или в других классах.

По вероятностной реляционной модели студента можно определить обобщенную байесовскую сеть – скелет, который можно инициализировать конкретными значениями для различных сценариев, в данном случае – для различных экспериментов. В этой модели легко организовать классы по уровням для улучшения понимания модели. Из класса модели мы получаем иерархический скелет, как показано на рис. 12.4. Класс эксперимента подразделяется в соответствии с интересующим нас объектом, при этом создаются два подкласса: выполнение эксперимента (experiment performance) и поведение эксперимента (experiment behavior), – которые образуют самый нижний уровень в иерархии. Промежуточный уровень представляет различные элементы знаний (концепции), связанные с каждым экспериментом. Эти элементы связаны с самым верхним уровнем, который объединяет элементы в подразделы (sub-themes) и разделы (themes) и, наконец, в общую категорию студентов. В рассматриваемом примере были определены три категории

студентов: *novice* (новичок), *intermediate* (средний уровень) и *advanced* (опытный). Для каждой категории определена одинаковая структура байесовской сети, полученная из скелета, но для каждой категории используются различные таблицы условных вероятностей.

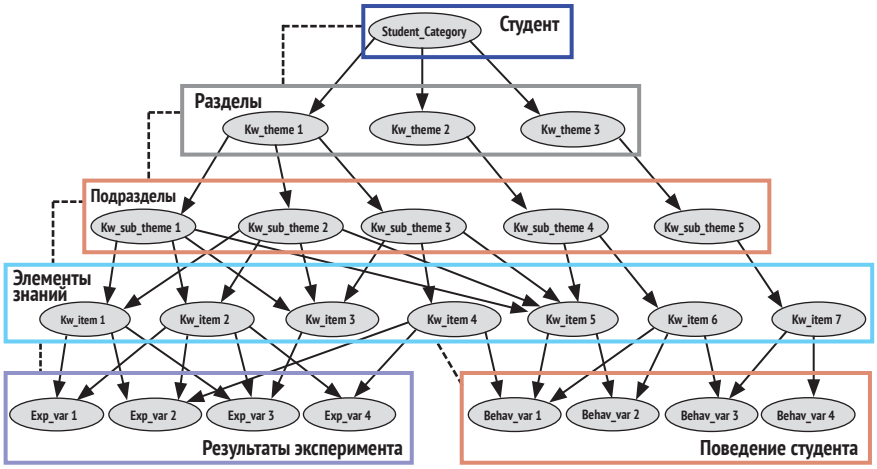


Рис. 12.4. Общий скелет для экспериментов, выведенный из вероятностной реляционной модели студента для виртуальных лабораторий. Сеть имеет иерархическую структуру, начиная с узла, представляющего категорию студента, на вершине, затем следуют три уровня переменных, представляющих знания студента в данной предметной области на различных уровнях абстракции: разделы, подразделы и элементы. На нижнем уровне существуют два множества переменных, соответствующих результатам взаимодействия студента с экспериментом в виртуальной лаборатории, разделенным на результаты экспериментов и поведение студента [13]

С помощью полученного скелета можно определить различные экземпляры, соответствующие значениям конкретных переменных в модели. Например, из общего скелета для экспериментов, показанного на рис. 12.4, можно определить конкретные экземпляры для каждого эксперимента (например, в области робототехники это могут быть эксперименты, связанные с проектированием робота, управлением роботом, планированием движения робота и т. п.) и уровень подготовки студента (новичок, средний уровень, опытный).

После генерации конкретной байесовской сети ее можно использовать для обновления модели студента с помощью стандартных методик распространения вероятностей. В рассматриваемом здесь примере одна из таких методик используется для распространения свидетельства от оценки эксперимента до элементов знаний, а затем до уровней подразделов и разделов знаний. После каждого эксперимента уровень знаний на каждом отдельном уровне детализации – элементов, подразделов и разделов – используется преподавателем для принятия реше-

ния о необходимости оказания помощи студенту и о степени подробности подсказок. Например, если в целом эксперимент был проведен успешно, но в каком-то аспекте был замечен недостаток, то студенту предлагается небольшой урок по соответствующей концепции (элементу знаний). Если эксперимент завершился неудачно, то рекомендуется расширенный урок по полному разделу или подразделу знаний. На основе категории студента преподаватель определяет уровень сложности следующих экспериментов, предлагаемых студенту.

12.6.1 Визуальные грамматики

Визуальная грамматика (visual grammar) описывает объект в иерархической форме. Это может быть диаграмма, геометрический чертеж или изображение. Например, описание блок-схемы выполняется методом декомпозиции: сложные конструкции разделяются на простые элементы (от полного изображения блок-схемы к стрелкам или простым прямоугольникам).

Для визуального распознавания объектов необходима грамматика, позволяющая моделировать процесс декомпозиции визуального объекта на составные части и определяющая связь каждой части с другими составными частями [12]. Интересным типом реляционной грамматики являются *грамматики символ–связь* (symbol-relation grammars; SR-grammars) [2], так как они способны обеспечить этот тип описания, а кроме того, обладают возможностью добавления переписанных правил для определения взаимосвязей (отношений) между терминальными и нетерминальными символами после завершения декомпозиции для всех нетерминальных символов.

12.6.1.1 Представление объекта с помощью SR-грамматик

Классы объектов представляются на основе грамматик символ–связь (SR-грамматик). В представление включаются три основные части: (i) основные элементы грамматики, или лексикон (lexicon), (ii) пространственные связи, (iii) правила преобразований. Ниже будут кратко описаны все три части.

Главная идея заключается в использовании простых общих признаков как основных элементов, поскольку их можно применять к различным классам объектов. Эти области (признаки) определяют визуальный словарь (visual dictionary). Визуальными признаками считаются *области одинакового цвета* (uniform color regions) (цвет классифицирован по 32 уровням) и границы по различным ориентациям (направлениям) (получаемые с помощью фильтров Габбора (Gabbor filters)). Эти признаки образуют множество тренировочных экземпляров, которые объединяются в кластеры. Центроиды полученных кластеров составляют *визуальный лексикон* (visual lexicon).

Пространственные связи включают топологические взаимосвязи и взаимосвязи порядка. Используются следующие взаимосвязи: *Inside_of*(*A*, *B*) – область *A* находится внутри области *B*, *Contains*(*A*, *B*) – область *A* полностью покрывает область *B*, *Left*(*A*, *B*) – область *A* касается области *B* и *A* расположена слева от *B*, *Above*(*A*, *B*) – область *A* касается области *B* и *A* расположена над *B*, *Invading*(*A*, *B*) – область *A* частично покрывает область *B*, больше, чем *Above* и *Left*, но меньше, чем *Contains*.

Следующий шаг – генерация правил, которые формируют грамматику. Используя тренировочные изображения для класса интересующих нас объектов, определяются наиболее часто встречающиеся взаимосвязи между кластерами. Такие взаимосвязи становятся правилами-кандидатами для создания грамматики. Это итеративный процесс, в котором правила комбинируются и преобразовываются в новые нетерминальные элементы создаваемой грамматики. Процесс повторяется до тех пор, пока не будет достигнуто пороговое значение (определяемое как минимальное количество элементов). Начальный символ грамматики представляет класс распознаваемых объектов.

В процессе визуального распознавания объектов подразумевается наличие неопределенности: помехи (шум) в изображении, окклюзии, неточная обработка на нижнем уровне и т. д. SR-грамматики не учитывают неопределенность, поэтому для ввода фактора неопределенности их можно расширить с помощью реляционных вероятностных графовых моделей, в частности с помощью марковских логических сетей.

12.6.1.2 Преобразование SR-грамматики в марковскую логическую сеть

SR-грамматика для некоторого класса объектов преобразовывается непосредственно в формулы на языке марковской логической сети. При этом получается структурная схема марковской логической сети. Параметры – весовые коэффициенты, связанные с каждой формулой, – определяются с помощью набора тренировочных изображений.

Рассмотрим простой пример SR-грамматики для распознавания лиц на основе признаков высокого уровня: глаз, рта, носа, головы. Продукциями этой простой SR-грамматики для лиц являются:

$1 : FACE^0 \rightarrow < \{eyes^2, mouth^2\}, \{above(eyes^2, mouth^2)\} >$

$2 : FACE^0 \rightarrow < \{nose^2, mouth^2\}, \{above(nose^2, mouth^2)\} >$

$3 : FACE^0 \rightarrow < \{eyes^2, head^2\}, \{inside\_of(eyes^2, head^2)\} >$

$4 : FACE^0 \rightarrow < \{nose^2, head^2\}, \{inside\_of(nose^2, head^2)\} >$

$5 : FACE^0 \rightarrow < \{mouth^2, head^2\}, \{inside\_of(mouth^2, head^2)\} >$

Преобразование в марковскую логическую сеть выполняется достаточно просто. Сначала необходимо объявить формулы:

```
aboveEM(eyes, mouth)
aboveNM(nose, mouth)
insideOfEH(eyes, head)
insideOfNH(nose, head)
insideOfMH(mouth, head)
isFaceENMH(eyes, nose, mouth, head)
```

Далее следует объявить предметную область:

```
eyes={E1, E2, E3, E4}
nose={N1, N2, N3, N4}
mouth={M1, M2, M3, M4}
head={H1, H2, H3, H4}
```

Для завершения процесса следует записать взвешенные формулы логики первого порядка. Используется валидационный набор данных об изображениях и вероятности, преобразованные в весовые коэффициенты:

```
1.58 isFaceENMH(e, n, m, h) => aboveEM(e, m)
1.67 isFaceENMH(e, n, m, h) => aboveNM(n, m)
1.16 isFaceENMH(e, n, m, h) => insideOfEH(e, h)
1.25 isFaceENMH(e, n, m, h) => insideOfNH(n, h)
1.34 isFaceENMH(e, n, m, h) => insideOfMH(m, h)
```

Для распознавания лица в изображении самые важные детали изображения преобразовываются в базу знаний логики первого порядка. Для этого выявляются терминальные элементы (в рассматриваемом здесь примере это глаза, рот, нос и голова) в изображении, а также пространственные связи между этими элементами. Затем база знаний конкретного изображения объединяется с обобщенной моделью, представленной как марковская логическая сеть. По этой объединенной конструкции генерируется марковская сеть обоснований. Распознавание объекта (лица) выполняется с использованием стандартного вероятностного логического вывода (см. главу 6) в марковской сети.

12.7 Материалы для дополнительного чтения

Существует несколько книг для начинающих изучать логику, например [8, 9]. С точки зрения искусственного интеллекта [4] предлагает качественное введение в логику предикатов и представления на основе логики. Большинство современных реляционных вероятностных моделей описано в [5], также включены главы по каждой методике. Вероятностные реляционные модели представлены в [6]. Методики обучения вероятностных реляционных моделей по имеющимся данным описаны в [3]. Обзор марковских логических сетей включен в [1].

[7, 10] – книги для начинающих изучать индуктивное логическое программирование.

12.8 Задания и упражнения

1. Если p и r ложны, а q и s истинны, определить значения (истинно или ложно) следующих выражений:
   - $p \vee q$;
   - $\neg p \wedge \neg(q \wedge r)$;
   - $p \rightarrow q$;
   - $(p \rightarrow q) \wedge (q \rightarrow r)$;
   - $(s \rightarrow (p \wedge \neg r)) \wedge ((p \rightarrow (r \vee q)) \wedge s)$.
2. Какие из следующих выражений являются истинными:
   - $\forall x\ ((x^2 - 1) > 0)$;
   - $\forall x\ (x^2 > 0)$;
   - $\exists x\ (1/(x^2 + 1) > 1)$;
   - $\neg\exists x\ ((x^2 - 1) \leqslant 0)$.
3. Определить, являются ли следующие выражения правильно построенными формулами:
   - $\forall x\ \neg(p(x) \rightarrow q(x))$;
   - $\exists x\ \forall x\ (p(x) \leftrightarrow q(x))$;
   - $\exists x \vee q(x) \wedge q(y)$;
   - $\forall x\ \exists y\ p(x) \vee (p(x) \leftrightarrow q(y))$.
4. Записать следующие высказывания в терминах логики предикатов первого порядка: (i) все люди имеют отца и мать; (ii) у некоторых людей есть братья, или сестры, или братья и сестры; (iii) если у человека есть брат, то его отец также является отцом его брата; (iv) не существует людей, у которых два родных отца или две родные матери.
5. Для вероятностной реляционной модели, показанной на рис. 12.1, предположить, что все переменные являются бинарными, т. е. Teaching-Ability = {Good, Average} (Преподавательская_квалификация = {Хорошая, Средняя}), Popularity = {High, Low} (Востребованность = {Высокая, Низкая}) и т. д. В соответствии со структурой этой модели определить требуемые таблицы условных вероятностей.
6. На основе вероятностной реляционной модели из задания 5 предположить, что существуют два профессора, три студента, три курса и пять регистрационных записей (один студент записался на два курса, другой на три курса). Расширить вероятностную реляцион-

ную модель для перечисленных выше объектов и сгенерировать соответствующую байесовскую сеть.

7. Рассмотреть пример марковской логической сети из раздела 12.4 с двумя логическими формулами и дополнительной третьей формулой: $\forall x\ UnhealthyDiet(x) \rightarrow Cancer(x)$. Принимая, что переменным x и y присваиваются значения *Tim*, *Sue* и *Maria*, получить графовую структуру зависимостей этой марковской логической сети.
8. Определить весовые коэффициенты для двух логических формул из примера марковской логической сети из раздела 12.4, предполагая, что из базы данных уже извлечены следующие статистические данные: (i) 19 человек курят и болеют раком; (ii) 11 человек курят и не болеют раком; (iii) 10 человек не курят и не болеют раком; (iv) 5 человек не курят и болеют раком; (v) 15 пар людей были друзьями и оба курили; (vi) 5 пар людей были друзьями, один курил, второй не курил.
9. *** Изучить другие методики формализации, которые объединяют логику и вероятность. Проанализировать их преимущества и недостатки с точки зрения выразительной мощи и эффективности вычислений.
10. *** Разработать программу реализации логического вывода для вероятностных реляционных моделей. Учитывая, что вероятностная реляционная модель описывается на уровне классов (можно воспользоваться объектно-ориентированной базой данных) и на основе множества объектов, преобразовать ее в байесовскую сеть. Затем выполнить логический вывод в этой байесовской сети (использовать ранее разработанные алгоритмы).

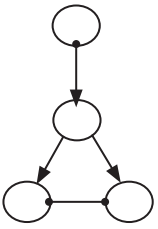
Ссылки на источники

1. Domingos, P., Richardson, M. Markov Logic: A Unifying Framework for Statistical Relational Learning. In: Getoor, L., Taskar, B. (eds.) Introduction to Statistical Relational Learning, p. 339–371. MIT Press, Cambridge (2007).
2. Ferrucci, F., Pacini, G., Satta, G., Sessa, M. I., Tortora, G., Tucci, M., Vitiello, G. Symbol-Relation Grammars: A Formalism for Graphical Languages. Information and Computation 131 (1), 1–46 (1996).
3. Friedman, N., Getoor, L., Koller, D., Pfeffe, A. Learning Probabilistic Relational Models. In: Proceeding of the International Joint Conference on Artificial Intelligence (IJCAI), p. 1300–1309 (1999).

4. Genesereth, M. R., Nilsson, N. J. Logical Foundations of Artificial Intelligence. Morgan Kaufmann (1988).
5. Getoor, L., Taskar, B. Introduction to Statistical Relational Learning. MIT Press, Cambridge (2007).
6. Koller D. Probabilistic Relational Models. In: Proceedings of the 9th International Workshop on Inductive Logic Programming. Lecture Notes in Artificial Intelligence, vol. 1634, Springer, 3–13 (1999).
7. Lavrac, N., Dzeroski, S. Inductive Logic Programming: Techniques and Applications. Ellis Horwood, New York (1994).
8. Lemmon, E. J. Beginning Logic. Hackett Publishing Company (1978).
9. Newton-Smith, W. H. Logic: An Introductory Course. Routledge, Milton Park (1985).
10. Nienhuys-Cheng, S., de Wolf, R. Foundations of Inductive Logic Programming. Springer-Verlag, Berlin (1991).
11. Richardson, M., Domingos, P. Markov logic networks. Machine Learning. 62 (1–2), 107–136 (2006).
12. Ruiz, E., Sucar, L. E. An object recognition model based on visual grammars and Bayesian networks. In: Proceedings of the Pacific Rim Symposium on Image and Video Technology, LNCS 8333, p. 349–359, Springer-Verlag (2014).
13. Sucar, L. E., Noguez, J. Student Modeling. In: O. Pourret, P. Naim, B. Marcot (eds.) Bayesian Belief Networks: A Practical Guide to Applications, p. 173–186. Wiley and Sons (2008).

Глава 13

Графовые причинно-следственные модели



13.1 Введение

Причинная связь (causality) должна обрабатывать отношения (связи) типа причина–следствие (cause-effect), т. е. при наличии двух (и более) взаимосвязанных явлений определять, какое из них является причиной (cause), а какое – следствием (effect). Но может существовать и третье объяснение, а именно: существует другое явление, которое представляет собой общую причину (common cause) первых двух изучаемых явлений.

Вероятностные модели не обязательно должны представлять причинно-следственные связи. Например, рассмотрим две ориентированные байесовские сети: BN1: $A \rightarrow B \rightarrow C$ и BN2: $A \leftarrow B \leftarrow C$. С вероятностной точки зрения обе сети представляют одно и то же множество отношений зависимости и независимости: прямые зависимости между A и B, между B и C, а также $I(A, B, C)$, т. е. A и C независимы относительно B. Но если мы определяем, что направленная связь $A \rightarrow B$ означает, что A является причиной B, то эти модели представляют абсолютно различные причинно-следственные связи.

Учитывая, что мы можем моделировать и решать сложные явления (задачи) с помощью вероятностных графовых моделей и что причинно-следственные связи имеют значение для многих сложных и противоречивых концепций, можно задать вопрос: зачем нужны причинно-следственные модели? Причинно-следственные модели и в

особенности графовые причинно-следственные модели обладают несколькими преимуществами.

Во-первых, если байесовская сеть создается на основе причинно-следственных связей, то такие модели обычно проще формировать и понимать (например, при общении с экспертами в какой-либо предметной области), кроме того, такие модели, как правило, сами по себе более просты. Например, в медицине часто наблюдаются болезни с несколькими симптомами, которые обычно условно независимы от конкретной болезни. Если представить эту ситуацию как байесовскую сеть по правилам причинно-следственных связей, то получится структура, показанная на рис. 13.1а. На рис. 13.1б показана другая структура, в которой требуются связи между переменными симптомов, так как они условно независимы относительно болезни, но не являются абсолютно независимыми. Модель, созданная на основе информации о причинно-следственных связях, оказывается более простой.

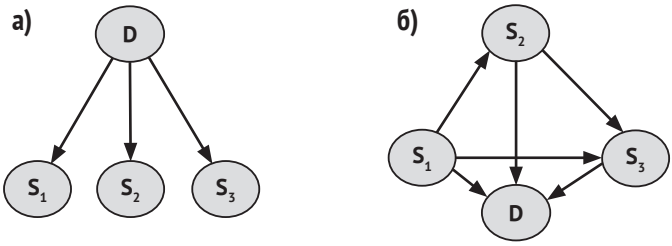


Рис. 13.1. Байесовская сеть, представляющая болезнь (*D*) и три ее симптома (S_1, S_2, S_3), которые условно независимы относительно *D*: а) структура на основе причинно-следственных связей, б) другая структура

Во-вторых, с помощью причинно-следственных моделей можно выполнять другие типы обоснований, которые невозможно осуществить, по крайней мере непосредственным образом, с помощью вероятностных графовых моделей, таких как байесовские сети. Существуют и иные варианты логического вывода: (i) интервенции (interventions), в которых необходимо найти следствия присваивания (изменения) определенной переменной конкретного значения внешним агентом (следует отметить, что это отличается от обычного наблюдения за переменной), и (ii) контрфактуальный анализ (counterfactuals), когда необходимо обосновать то, что могло бы произойти, если известная информация отличалась от того, что действительно произошло. Мы будем рассматривать оба этих варианта более подробно в следующих разделах главы.

Разработано несколько методик причинно-следственного моделирования, например функциональные уравнения, диаграммы пути, модели структурных уравнений и др. В этой главе основное внимание уделено

графовым моделям, в частности причинно-следственным байесовским сетям.

13.2 Причинно-следственные байесовские сети

Причинно-следственная байесовская сеть (casual Bayesian network) – это направленный ациклический граф G, в котором каждый узел представляет переменную, а дуги в этом графе представляют причинно-следственные связи. Таким образом, связь $A \rightarrow B$ – это некоторый физический механизм, такой, что на значение B непосредственно влияет значение A. Данную связь можно интерпретировать как интервенции (interventions) – присваивание значения некоторой переменной или нескольким переменным внешним агентом. Например, предположим, что A обозначает дождевальную установку (OFF/ON), а B представляет состояние увлажненности травы (FALSE/TRUE). Если трава (B) изначально не увлажнена (not WET), а дождевальная установка включена (ON) в результате интервенции (внешним агентом), то B изменяет значение на TRUE.

Как и в обычных байесовских сетях, если существует дуга от A до B (A является непосредственной причиной B), то A является родителем B, а B – потомком A. Если рассматривать любую переменную X в причинно-следственной байесовской сети, то $Pa(X)$ – это множество всех родителей переменной X. Кроме того, аналогично байесовским сетям, когда прямые или непосредственные причины – родители – некоторой переменной известны, то наиболее удаленные причины (или далекие предки) не имеют значения. Например, если известно, что трава увлажнена (WET), это делает ее скользкой (SLIPPERY), и не важно, как именно трава стала влажной (кто-то включил дождевальную установку или прошел дождь).

Причинно-следственные сети представляют более строгие предположения, чем обычные байесовские сети, так как все связи, определяемые структурой сети, должны соответствовать причинно-следственным связям. Таким образом, все родительские узлы $Pa(X)$ конкретной переменной X соответствуют прямым причинам X. Это означает, что если для любой из переменных-родителей X или для любого сочетания переменных-родителей установлено некоторое значение посредством интервенции, то это окажет воздействие (следствие) на переменную X. В причинно-следственной байесовской сети переменная, являющаяся корневым узлом (переменная без родителей), называется экзогенной (exogenous), а все остальные переменные называются эндогенными (endogenous).

Простой пример причинно-следственной байесовской сети показан на рис. 13.2. Здесь представлены следующие причинно-следственные связи: (i) *Дождевальная установка* (Sprinkler) является причиной *Увлажнения* (Wet), (ii) *Дождь* (Rain) является причиной *Увлажнения* (Wet), (iii) *Увлажнение* (Wet) является причиной *Скользкого состояния травы* (Slippery). В этом примере *Дождевальная установка* (Sprinkler) и *Дождь* (Rain) – экзогенные переменные, а *Увлажнение* (Wet) и *Скользкое состояние травы* (Slippery) – эндогенные переменные.

Рис. 13.2. Простой пример причинно-следственной байесовской сети, которая представляет следующие связи: *Дождевальная установка* (Sprinkler) – причина *Увлажнения* (Wet), *Дождь* (Rain) – причина *Увлажнения* (Wet), а *Увлажнение* (Wet) – причина *скользкого состояния травы* (Slippery) (пример взят из книги Дж. Перла (J. Pearl) [3])

Если $P(\mathbf{X})$ – совместное распределение вероятностей множества переменных $\mathbf{X}$, то $P_y(\mathbf{X})$ определяется как распределение, являющееся результатом установки значений для подмножества переменных $\mathbf{Y}$ посредством интервенции. Это можно представить как $do(\mathbf{Y} = \mathbf{y})$, где $\mathbf{y}$ – множество постоянных значений. Например, в причинно-следственной байесовской сети на рис. 13.2 при установлении значения для дождевальной установки значения ON (включена): $do(Sprinkler = ON)$ – результатом будет распределение $P_{Sprinkler=ON}(Sprinkler, Rain, Wet, Slippery)$.

Формальное определение причинно-следственной байесовской сети может быть приведено в следующей форме [3]:

Причинно-следственная байесовская сеть G – это направленный ациклический граф на множестве переменных $\mathbf{X}$, совместимый со всеми распределениями, полученными из пересечений по $\mathbf{Y} \subseteq \mathbf{X}$, при соблюдении следующих условий:

1. Распределение вероятностей $P_y(\mathbf{X})$, полученное в результате интервенции, является совместимым по Маркову с графом G, т. е. равнозначным произведению условных вероятностей для каждой переменной $X \in G$ с учетом родителей этой переменной: $P_y(\mathbf{X}) = \prod_{X_i}(X_i \mid P_a(X_i))$.

2. Вероятность каждой переменной, являющейся частью интервенции, равна единице для устанавливаемого значения: $P_y(X_i) = 1$, если $X_i = x_i$ согласовано (не противоречит) с $Y = y$, $\forall X_i \in \mathbf{Y}$.
3. Вероятности всех остальных переменных, не являющихся частью интервенции, равны вероятности каждой отдельной переменной с учетом ее родителей, и она согласована (не противоречит) с интервенцией: $P_y(X_i \mid P_a(X_i)) = P(X_i \mid P_a(X_i))$, $\forall X_i \notin \mathbf{Y}$.

С учетом приведенного выше определения и в особенности того факта, что вероятность переменных, значения которых установлены в результате интервенции, равна единице (условие 2), совместное распределение вероятностей можно вычислить как усеченную факторизацию (truncated factorization):

$$P_y(\mathbf{X}) = \prod_{X_i \notin \mathbf{Y}} (X_i \mid P_a(X_i)), \tag{13.1}$$

при условии что все переменные X_i согласованы с интервенцией $\mathbf{Y}$.

Другое следствие приведенного выше определения: после того как для всех родителей переменной X_i установлены значения в результате интервенции, установка значения для любой другой переменной $\mathbf{W}$ не влияет на вероятность переменной X_i:

$$P_{Pa(X_i), \mathbf{W}}(X_i) = P_{Pa(X_i)}(X_i), \tag{13.2}$$

при условии что $\mathbf{W} \cap (X_i, Pa(X_i)) = \emptyset$.

Рассмотрим еще раз пример, показанный на рис. 13.2. Если трава увлажнена любым способом $do(Wet = TRUE)$, то на вероятность переменной *Slippery* (Скользкая) не влияют переменные *Rain* (Дождь) или *Sprinkler* (Дождевальная установка).

13.3 Обоснование причин

Обоснование причин должно выполняться с помощью ответов на причинные запросы из причинно-следственной модели, в частности в нашем случае, из графовых причинно-следственных моделей. Существует несколько типов причинных запросов, которые можно рассмотреть, и мы начнем с анализа причинных прогнозов, затем перейдем к контрфактуальному анализу.

13.3.1 Прогноз

Рассмотрим причинно-следственную байесовскую сеть, содержащую множество переменных: $X_G = \{X_C, X_E, X_O\}$, где X_C – подмножество причин (causes), X_E – подмножество следствий (effects), X_O – остальные перемен-

ные. Необходимо выполнить причинный запрос в этой модели: каким будет следствие X_E при установке значения $X_C = x_C$? То есть нужно получить распределение вероятностей переменной X_E как результат интервенции $X_C = x_C$:

$$P_C(X_E \mid do(X_C = x_C)). \tag{13.3}$$

Для получения причинного прогноза в причинно-следственной байесовской сети G выполняется следующая процедура:

1. Исключаются все входящие стрелки в графе для всех узлов подмножества X_C, чтобы получить измененную причинно-следственную байесовскую сеть G_r.
2. Устанавливаются значения для всех переменных подмножества X_C, $X_C = x_C$.
3. Вычисляется итоговое распределение в измененной модели G_r (с применением методики распространения вероятностей, как в обычных байесовских сетях).

Например, рассмотрим гипотетическую причинно-следственную байесовскую сеть, показанную на рис. 13.3а, которая представляет определенные причинно-следственные знания о сердечном приступе (stroke). Если необходимо оценить следствие Unhealthy diet (нездорового питания) в переменной Stroke (сердечный приступ), то в соответствии с описанной выше процедурой исключается связь, направленная от узла Health consciousness (забота о своем здоровье) к узлу Unhealthy diet (нездоровое питание), и получается модель, показанная на рис. 13.3б. Затем необходимо установить для переменной Unhealthy diet (нездоровое питание) значение *TRUE* (истина) и методом распространения вероятностей получить распределение вероятностей для переменной Stroke (сердечный приступ). Если распределение для переменной Stroke (сердечный приступ) изменяется в зависимости от значения переменной Unhealthy diet (нездоровое питание), то по этой модели можно сделать вывод о влиянии переменной Unhealthy diet (нездоровое питание) на переменную Stroke (сердечный приступ) (т. е. установить причинно-следственную связь).

Возникает вопрос: когда распределение, полученное в результате интервенции, равно распределению, полученному в результате наблюдения? В математических терминах это записывается так: $P(X_E \mid X_C) = P_C(X_E \mid do(X_C = x_C))$. Обе части выражения равны, если X_C включает всех родителей X_E и ни одного из своих потомков, учитывая то, что любая переменная в байесовской сети независима от своих «не-потомков» с учетом своих родителей. В других случаях равенство не всегда (не обязательно) соблюдается, эти распределения зависят от других условий.

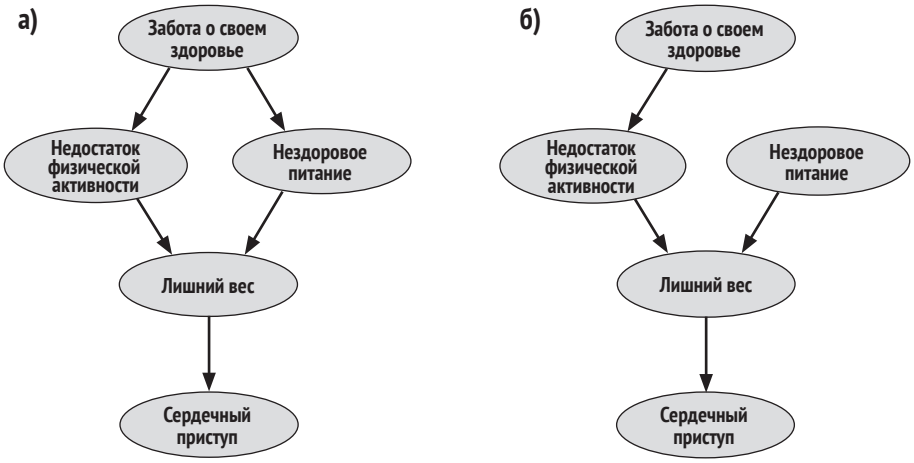


Рис. 13.3. Пример причинно-следственного прогноза: а) простая гипотетическая причинно-следственная байесовская сеть, представляющая некоторые причинно-следственные связи для Stroke (сердечный приступ), б) итоговая графовая модель, полученная из (а) посредством интервенции *do*(*unhealthy-diet* = *TRUE*)

13.3.2 Контрфактуальный анализ

Контрфактуальный анализ – это метод обоснования того, что мы весьма часто выполняем в нашей повседневной жизни. Например, рассмотрим причинно-следственную модель на рис. 13.3а. Обычный вопрос контрфактуального анализа может выглядеть так: если у некоторого человека наблюдается сердечный приступ, то мог бы случиться этот сердечный приступ (Stroke = TRUE), если бы он проявлял большую физическую активность (Lack of exercise = FALSE)?

Контрфактуальный анализ включает три основных шага:

1) абдукция (диагностирование, или обратный вывод): изменение модели посредством ввода в нее нового свидетельства (в рассматриваемом здесь примере: изменение значения Stroke на unknown (неизвестно));
2) действие: выполнение минимальной интервенции в модели в соответствии с введенным гипотетическим свидетельством (в рассматриваемом здесь примере: присваивание переменной Lack of exercise значения FALSE и удаление связи от переменной Health consciousness к переменной Lack of exercise);
3) прогноз: выполнение вероятностного логического вывода в измененной модели и получение вероятности интересующей нас переменной (или нескольких переменных) (в рассматриваемом здесь примере: выполнение распространения вероятностей для оценки $P_{Lack\text{-}exercise}(Stroke \mid do(Lack\text{-}exercise = FALSE))$).

13.4 Обучение причинно-следственных моделей

Обучение причинно-следственных моделей по имеющимся данным без прямых интервенций создает множество затруднений. Как уже было отмечено ранее, если обнаруживается некоторая зависимость между двумя переменными *X* и *Y*, то невозможно без дополнительной информации определить, является ли *X* причиной *Y* или же *Y* является причиной *X*. Кроме того, может существовать некоторый другой фактор, который создает зависимость между этими двумя переменными.

Например, будем считать, что на основании имеющихся данных мы обнаружили, что люди, употребляющие вино, в меньшей степени подвержены сердечным приступам. Затем мы могли бы прийти к выводу о том, что употребление вина приводит к снижению вероятности возникновения сердечного приступа (рис. 13.4а). Но может существовать и другая переменная, которая создает эту видимую причинно-следственную связь, известную как скрытая частая причина (latent common cause). Возможно, что оба явления – употребление вина и сердечный приступ – необходимо оценивать с точки зрения уровня доходов конкретного человека, поскольку люди с высоким уровнем доходов склонны употреблять больше вина, и в то же время вероятность возникновения сердечного приступа у таких людей меньше, так как им доступно более качественное медицинское обслуживание (см. рис. 13.4б). Таким образом, сложность обучения причинно-следственных связей заключается в том, как включить в модель все действительно важные факторы.

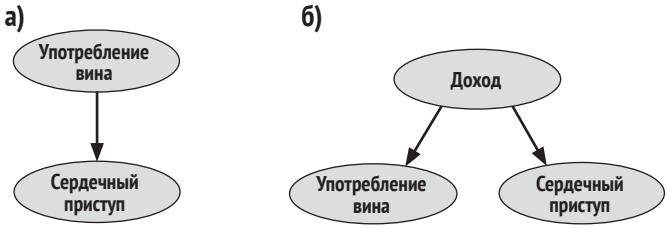


Рис. 13.4. Пример сложности, возникающей при обучении причинно-следственных моделей: а) первоначальная сеть, показывающая видимую причинно-следственную связь между *Употребление вина* (Drinking wine) и *Сердечный приступ* (Heart Attack), б) другая сеть с (скрытой) частой причиной *Доход* (Income), которая описывает зависимость между *Употребление вина* (Drinking wine) и *Сердечный приступ* (Heart Attack)

Разработано несколько алгоритмов обучения причинно-следственных сетей. В общем случае при обучении структуры причинно-следственных сетей используются следующие предположения:

- марковское детерминирующее (причинное) условие (Causal Markov Condition): переменная независима от своих «не-потомков» с учетом ее прямых причин (родителей в графе);
- достоверность (Faithfulness): не существует дополнительных независимостей между переменными в модели, которые не подразумеваются марковским детерминирующим (причинным) условием;
- причинная достаточность (Causal Sufficiency): в модели не существует общих смешанных связей (отношений) наблюдаемых переменных.

Один из алгоритмов обучения причинно-следственных сетей – Bayesian Constraint-Based Casual Discovery (BCCD) [2]. Эта методика является расширением PC-алгоритма обучения структуры байесовской сети (см. главу 8), состоящего из двух этапов:

1) сначала рассматривается полностью связанный граф и оценивается надежность каждой причинно-следственной связи $X - Y$ посредством измерения условной независимости между X и Y. Если пара переменных является условно независимой с надежностью, превышающей определенное пороговое значение, то удаляется ребро между этими переменными;
2) оставшиеся причинно-следственные связи (неориентированные ребра в графе) упорядочиваются в соответствии с их надежностью. Затем ребра в графе ориентируются (определяется их направление), начиная с наиболее надежной связи, на основе проверок условной независимости для троек переменных.

Для оценки надежности причинно-следственной связи $R = X \rightarrow Y$ этот алгоритм использует байесовскую оценку:

$$P(R \mid D) = \frac{P(D \mid M_R)P(M_R)}{P(D \mid M)p(M)}, \tag{13.4}$$

где D – данные, M – все возможные структуры, M_R – все структуры, содержащие связь R. Таким образом, $P(M)$ обозначает априорную вероятность структуры M, а $P(D \mid M)$ – вероятность данных с учетом этой структуры. Вычисления по формуле 13.4 связаны с большими накладными расходами, поэтому используются приближенные вычисления по предельному правдоподобию данных с учетом структуры. Кроме того, обычно ограничивается максимальное количество переменных в сети.

В зависимости от порогового значения надежности полученная в результате сеть может содержать неориентированные ребра —, которые

означают, что для определения направления этой связи недостаточно информации, двунаправленные ребра ↔, обозначающие существование общей смешанной связи. Этот тип структуры называют *максимальным родовым графом* (подвид смешанного графа) (Maximal ancestral graph – MAG). Равнозначным классом является частичный родовой граф (Partial ancestral graph – PAG). В частичном родовом графе существует три типа ребер: направленные → и ненаправленные —, когда они являются согласованными для всех графов в равнозначном классе, а также несогласованные ребра, обозначаемые небольшим кружком ○.

В следующем разделе будет показан частичный родовой граф (PAG), полученный в результате обучения причинно-следственной модели для реально используемого приложения.

13.5 Приложения

Существует много практических приложений, в которых причинно-следственные модели оказываются полезными. Ниже перечислены некоторые области их применения:

- прогнозирование следствий (результатов) определенных интервенций (вмешательств);
- обучение причинно-следственных графовых моделей по имеющимся данным;
- диагностика физических (материальных) систем;
- генерация объяснений (обоснований);
- понимание причинно-следственных выражений.

В следующем подразделе рассматривается практическое приложение для обучения причинно-следственных графовых моделей по имеющимся данным.

13.5.1 Обучение причинно-следственной модели для синдрома дефицита внимания и гиперактивности

В [4] авторы описывают расширение алгоритма BCCD для объединенного множества дискретных и непрерывных переменных и применение этого расширенного алгоритма к набору данных, содержащему фенотипическую информацию о детях с синдромом дефицита внимания и гиперактивности (СДВГ) (Attention Deficit Hyperactivity Disorder – ADHD). Авторы использовали сокращенный набор данных, содержащий данные по 223 объектам наблюдения, с созданием следующих девяти переменных для каждого наблюдаемого ребенка:

1. Gender – Пол (мужской/женский);
2. Attention deficit score – Числовая оценка дефицита внимания (непрерывная переменная);
3. Hyperactivity/Impulsivity score – Числовая оценка гиперактивности/импульсивности (непрерывная переменная);
4. Verbal IQ – Вербальная шкала IQ (непрерывная переменная);
5. Performance IQ – Невербальная (поведенческая) шкала IQ (непрерывная переменная);
6. Full IQ – Общий IQ (непрерывная переменная);
7. Aggressive behavior – Агрессивное поведение (да/нет);
8. Medication status – Стадия лечения (простое, первичное/непростое, расширенное);
9. Handedness – Ведущая рука (правая/левая).

Учитывая небольшой размер набора данных, авторы включили в модель некоторую дополнительную информацию, в частности в сети нет переменной, которая может стать причиной *Gender*. При использовании значения порога надежности, равного 50 %, была получена сеть, изображенная на рис. 13.5, представленная как частичный родовой граф (PAG).

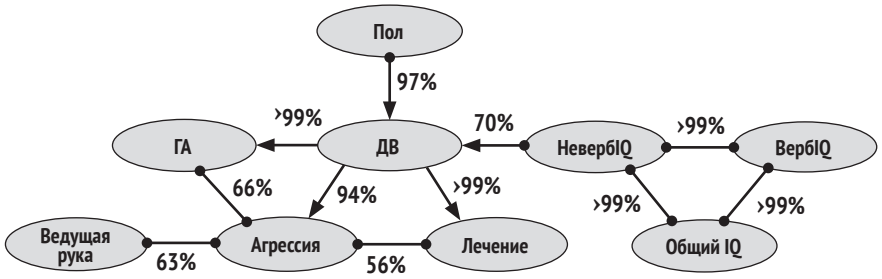


Рис. 13.5. Причинно-следственная модель, полученная из набора данных ADHD (синдром дефицита внимания и гиперактивности). Граф представляет полученный в результате PAG, в котором ребра помечены как → или – для неизменяемых связей и как ○ для изменяемых связей. Для каждого ребра показана его надежность (в процентах). (*ДВ* (AD) – оценка дефицита внимания, *ГА* (HI) – гиперактивность, *НевербIQ* (PerfIQ) – невербальная (поведенческая) шкала IQ, *ВербIQ* (VerbIQ) – вербальная шкала IQ, *Лечение* (Med) – стадия лечения.) Схема основана на [4]

Из этой итоговой сети было выведено несколько интересных причинно-следственных связей, некоторые из которых были уже известны по результатам более ранних медицинских исследований [4]:

- существует строгая зависимость (следствие) от пола на уровне дефицита внимания;

- уровень дефицита внимания влияет на гиперактивность/импульсивность и на агрессивность поведения;
- ведущая рука (левая) связана с агрессивностью поведения;
- связь между невербальным (поведенческим) IQ, вербальным IQ и общим IQ объясняется скрытой частой причиной;
- только невербальный (поведенческий) IQ имеет прямую причинно-следственную связь с дефицитом внимания.

13.6 Материалы для дополнительного чтения

Графовые причинно-следственные модели были впервые применены в генетике [6]. Две подробнейшие книги по графовым причинно-следственным моделям – [3, 5]. Алгоритм BCCD для обучения причинно-следственных графов представлен в [2]. Обзор других методик обучения причинно-следственных сетей можно найти в [1].

13.7 Задания и упражнения

1. Объяснить различия между ориентированной графовой моделью, например байесовской сетью, и причинно-следственной моделью.
2. Рассмотреть причинно-следственную байесовскую сеть на рис. 13.2. Получить несколько другой вариант моделей байесовской сети для этих четырех переменных, которые не обязательно являются причинно-следственными. Сравнить полученные сети по простоте и понятности с исходной моделью.
3. Для причинно-следственной байесовской сети, показанной на рис. 13.3а: а) предполагая, что все переменные являются бинарными, определить таблицы условных вероятностей на основе собственной интуиции; б) получить априорную вероятность для всех переменных методом вероятностного логического вывода; в) с учетом интервенции *do*(*Unhealthy-diet* = *true*) вычислить апостериорную вероятность переменных *Overweight* (лишний вес) и *Stroke* (сердечный приступ).
4. В соответствии с результатами выполнения задания 3 воздействует ли нездоровое питание (*Unhealthy-diet*) на переменную *Stroke* (сердечный приступ)? Почему?
5. Вычислить апостериорную вероятность переменных *Overweight* (лишний вес) и *Stroke* (сердечный приступ) с учетом наблюдения *Unhealthy-diet* = *true* для сети, показанной на рис. 13.3а (используя параметры из задания 3). Соответствуют ли полученные вероятности результатам, полученным при соответствующей интервенции? Почему?

6. Для причинно-следственной байесовской сети, показанной на рис. 13.3а, рассмотреть вопрос контрфактуального анализа: будет ли уменьшаться вероятность сердечного приступа (*Stroke*), если человек обеспечивает здоровое питание (*Unhealthy-diet = false*)? Выполнить требуемые операции в этой модели (используя параметры из задания 3) для ответа на этот вопрос.
7. Используя данные примера *Golf* (глава 8), применить стандартный PC-алгоритм для обучения структуры байесовской сети. Затем применить алгоритм BCCD, используя оценки надежности для каждой причинно-следственной связи, чтобы упорядочить эти причинно-следственные связи, и установить пороговое значение, чтобы определить, имеет ли конкретная связь некоторое направление или остается ненаправленной. Сравнить структуры, полученные с применением PC-алгоритма и алгоритма BCCD.
8. Для частичного родового графа (PAG), показанного на рис. 13.5, показать все максимальные родовые графы (MAG), входящие в этот PAG.
9. *** Разработать программу, реализующую алгоритм BCCD для дискретных переменных.
10. *** Получить данные из какой-либо реальной предметной области и использовать программу, разработанную в задании 9, для получения причинно-следственной модели. Изменить пороговое значение надежности и сравнить полученные модели.

Ссылки на источники

1. Aitken, J. S. Learning Bayesian networks: approaches and issues. Knowl. Eng. Rev. 26 (2), 99–157 (2011).
2. Claassen, T., Heskes, T. A Bayesian approach to constraint based causal inference. In: Proceedings of Uncertainty in Artificial Intelligence (UAI), AUAI Press, p. 207–216 (2012).
3. Pearl, J. Causality: Models. Reasoning and Inference. Cambridge University Press, New York (2009).
4. Sokolova, E., Groot, P., Classen, T., Heskes, T. Causal discovery from databases with discrete and continuous variables. In: Probabilistic graphical models (PGM). Springer, p. 442–457 (2014).
5. Spirtes, P., Glymour, C., Scheines, R. Causation, Prediction, and Search. MIT Press, Cambridge (2000).
6. Wright, S. Correlation and causation. J. Agric. Res. 20, 557–585 (1921).

Словарь терминов

Bayesian Classifier – байесовский классификатор – классификатор, который присваивает вероятности различным меткам объектов на основе байесовского правила.

Bayesian network – байесовская сеть – ориентированный циклический граф, представляющий совместное распределение вероятностей множества случайных переменных, такого, что каждая переменная условно независима от своих «не-потомков» с учетом своих родителей в этом графе.

Casual Bayesian Network – причинно-следственная байесовская сеть – направленный ациклический граф, в котором узлы представляют случайные переменные, а дуги – причинно-следственные связи.

Casual reasoning – причинно-следственное обоснование – процедура ответа на причинно-следственные запросы в форме причинно-следственной модели.

Classifier – классификатор – метод или алгоритм присваивания меток объектам.

Clique – клика – полностью связанное подмножество узлов в графе, которое является максимальным.

Conditional independence – условная независимость – две переменные являются условно независимыми относительно третьей, если они становятся независимыми, когда эта третья переменная известна.

Conditional Probability – условная вероятность – вероятность некоторого события, при условии что произошло некоторое другое событие.

Conditional Random Field – условное случайное поле – случайное поле, в котором все переменные глобально обусловлены наблюдениями.

Decision Tree – дерево (принятия) решений – дерево, представляющее задачу принятия решений и содержащее три типа узлов: решения, неопределенные события и результаты.

Directed Acyclic Graph – направленный ациклический граф – ориентированный граф, которые не содержит ориентированных контуров (ориентированный контур – контур, в котором все ребра в последовательности следуют в направлениях, указанных стрелками).

D-separation – *D*-разделенность – графовый критерий для определения, являются ли два подмножества переменных условно независимыми относительно третьего подмножества переменных в байесовской сети.

Dynamic Bayesian Network – динамическая байесовская сеть – расширение байесовских сетей для моделирования динамических процессов. ДБС состоит из последовательности интервалов времени, каждый из которых представляет состояние всех переменных в определенный момент времени.

Expectation-Maximization – EM-алгоритм – статистический метод, используемый для оценки параметров при наличии ненаблюдаемых переменных.

Graph – граф – графическое представление бинарных связей (отношений) между множеством объектов.

Hidden Markov Model – скрытая марковская модель – марковская цепь, в которой состояния не являются непосредственно наблюдаемыми.

Independent variables – независимые переменные – две случайные переменные являются независимыми, если известное значение одной из них не влияет на распределение вероятностей другой переменной.

Influence diagram – диаграмма влияния – графовая модель для решения задач принятия решений. Это расширение байесовских сетей, которое включает узлы решений и узлы полезности.

Junction tree – дерево сочленений – дерево, в котором каждый узел соответствует подмножеству переменных вероятностной графовой модели.

Limited Memory Influence Diagram – диаграмма влияния с ограниченной памятью – диаграмма влияния, в которой переменные, известные во время принятия решения, не обязательно запоминаются для принятия дальнейших решений.

Markov Blanket – марковское ограждение – множество переменных, которые делают переменную независимой от всех прочих переменных в вероятностной графовой модели.

Markov Chain – марковская цепь – машина (механизм) состояний, в которой переходы между состояниями являются недетерминированными и соответствуют марковскому свойству.

Markov Decision Process – марковский процесс принятия решений – графовая модель для последовательного принятия решений, состоящая из конечного множества состояний и действий, в которой состояния соответствуют марковскому свойству.

Markov Network – марковская сеть – случайное поле, представленное как неориентированный граф, соответствующий свойству локальности, – каждая переменная в этом поле независима от всех прочих переменных относительно своих соседей в графе.

Markov Property – марковское свойство – вероятность следующего (будущего) состояния не зависит от предыдущих состояний (в прошлом) относительно текущего (настоящего) состояния.

Markov Random Field – марковское случайное поле – марковская сеть.

Multidimensional classifier – многомерный классификатор – классификатор, который может присваивать более одной метки каждому объекту.

Naïve Bayes Classifier – наивный байесовский классификатор – байесовский классификатор, который предполагает, что все атрибуты являются независимыми относительно переменной класса.

Partially Observable Markov Decision Process – частично наблюдаемый марковский процесс принятия решений – марковский процесс принятия решений, в котором состояния не являются непосредственно наблюдаемыми.

Policy – стратегия – функция, отображающая состояния в действия.

Probabilistic Graphical Model – вероятностная графовая модель – компактное представление совместного распределения вероятностей множества случайных переменных, состоящее из графа и множества локальных распределений вероятностей.

Probabilistic Inference – вероятностный логический вывод – процедура вычисления апостериорной вероятности неизвестных переменных в вероятностной графовой модели с учетом некоторого свидетельства (подмножества известных переменных или переменных с присвоенными значениями).

Probability – вероятность – функция, которая присваивает действительное число каждому событию (подмножеству пространства состояний) и соответствует определенным аксиомам, известным как аксиомы теории вероятностей.

Random Field – случайное поле – набор случайных переменных, проиндексированных по их позициям в этом наборе.

Random Variable – случайная переменная – форма отображения пространства состояний в действительные числа.

Rational Agent – рациональный агент – агент, который выбирает решения для максимизации ожидаемой полезности, соответствующей его предпочтениям.

Relational Probabilistic Graphical Models – реляционные вероятностные графовые модели – расширение вероятностных графовых моделей, которое обладает большей выразительной мощью благодаря включению некоторого типа реляционного представления (представления связей (отношений)).

Sample space – пространство выборок – множество возможных исходов (результатов) эксперимента.

Temporal Event Network – временна́я сеть событий – байесовская сеть для моделирования динамических процессов, в которой каждый узел представляет время возникновения события или изменения состояния некоторой переменной.

Tree – дерево – связный граф, который не содержит простых контуров.

Предметный указатель

В

Г

П

Р

С

Т

Ф

Ш

Э

Книги издательства «ДМК ПРЕСС» можно купить оптом и в розницу
в книготорговой компании «Галактика»
(представляет интересы издательств
«ДМК ПРЕСС», «СОЛОН ПРЕСС», «КТК Галактика»).
Адрес: г. Москва, пр. Андропова, 38;
Тел.: **+7(499) 782-38-89**. Электронная почта: **books@alians-kniga.ru**.
При оформлении заказа следует указать
адрес (полностью), по которому должны быть высланы книги;
фамилию, имя и отчество получателя.
Желательно также указать свой телефон и электронный адрес
Эти книги вы можете заказать и в интернет-магазине: **www.a-planeta.ru**.

Луис Энрике Сукар

Вероятностные графовые модели

Принципы и приложения

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Перевод с английского *Снастин А. В.*
Корректор *Синяева Г. И.*
Верстка *Паранская Н. В.*
Дизайн обложки *Мовчан А. Г.*

Формат 70×100 1/16. Печать цифровая.
Усл. печ. л. 27.46. Тираж 200 экз.

Отпечатано в ПАО «Т8 Издательские Технологии»
109316, Москва, Волгоградский проспект, д. 42, корп. 5

Веб-сайт издательства: **www.dmkpress.com**