ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

В 1814 году Пьер-Симон Лаплас писал: «по большей части важнейшие жизненные вопросы являются на самом деле лишь задачами теории вероятностей». Спустя сто лет после этих слов на такие вопросы можно было ответить только одним способом (сохраняя верность мнению Лапласа): проанализировать каждую задачу на бумаге, выразить результат в виде формулы и вычислить значение формулы, вручную подставив в нее числа. Наступление эры компьютеров мало что изменило. Просто стало возможно вычислять более сложные формулы, а анализ с помощью «пера и бумаги» теперь может занимать сотни страниц.

Для анализа вероятностной задачи необходимо построить *вероятностную модель*, в которой описывается пространство возможных исходов и каждому из них каким-то образом сопоставляется числовая вероятность. Раньше вероятностные модели формулировались на смеси естественного языка и полуформальной математической нотации. На основе модели с помощью некоторых математических манипуляций выводилась формула или алгоритм для вычисления ответов. Обе стадии были трудоемкими, чреватыми ошибками и зависели от конкретной задачи, поэтому применение теории вероятностей на практике сталкивалось с серьезными ограничениями. Вопреки Лапласу, важнейшие жизненные вопросы оставались неразрешенными.

Первым крупным продвижением стала разработка *формальных языков*, в частности байесовских и марковских сетей, для выражения вероятностных моделей. У формального языка имеется точный синтаксис, определяющий, какие выражения допустимы, и точная семантика, определяющая, что означает каждое допустимое выражение (т. е. какая именно вероятностная модель представлена данным выражением). Поэтому появилась возможность описывать вероятностные модели в машиночитаемом виде и разработать единый алгоритм вычисления следствий *любой* выразимой вероятностной модели.

Но в этой бочке меда есть одна ложка дегтя: отсутствие *выразимых* вероятностных моделей. Байесовские и марковские сети как формальные языки обладают очень ограниченными выразительными возможностями. В каком-то смысле их можно назвать вероятностными аналогами булевых схем. Чтобы понять, в чем состоит ограничение, рассмотрим написание программы расчета платежной ведомости для крупной компании. На языке высокого уровня, например Java, она может состоять из десятков тысяч строк кода. А теперь представьте себе реализацию той же функциональности посредством соединения логических вентилей. Эта задача,

по всей видимости, абсолютно неразрешима. Схема оказалась бы невообразимо огромной, сложной и непонятной, поскольку логическим схемам недостает выразительной силы, соответствующей структуре задачи.

В 1997 году Ави Пфеффер, автор этой книги, тогда еще студент, в соавторстве со своим научным руководителем Дафной Коллер и коллегой Дэвидом Макаллестером, опубликовал пионерскую работу по языкам вероятностного программирования (ЯВП) (probabilistic programming language – PPL), в которой высказана важнейшая идея, связывающая теорию вероятностей с выразительными возможностями языков программирования высокого уровня. Идея заключалась в том, что программу можно рассматривать как вероятностную модель, если ввести некоторые стохастические элементы и определить смысл программы как вероятность каждого возможного пути выполнения. Эта идея вводит полезную связь между двумя важнейшими областями математики, и мы еще только приступаем к исследованию открывающихся на этом пути возможностей.

Книга представляет собой неформальное введение в круг этих идей на примере языка Figaro для иллюстрации основных концепций и их применения. Автор избегает ненужной математики и акцентирует внимание на реальных тщательно подобранных примерах, которые подробно объясняет. Книга доступна любому человеку со стандартной подготовкой в области программирования. Заодно трудолюбивый читатель с меньшими, чем обычно, усилиями освоит принципы и технику байесовского вывода и статистического обучения. Но, пожалуй, еще важнее тот факт, что читатель приобретет навыки моделирования – одно из важнейших умений любого ученого или инженера. Figaro и другие ЯВП позволяют выражать такие модели непосредственно, быстро и точно.

Эта книга – важный шаг на пути вывода вероятностного программирования из научно-исследовательских лабораторий, где оно зародилось, в реальный мир. Без сомнения, такое столкновение с реальностью выявит ограничения имеющихся систем ВП, и у лабораторий появятся новые задачи. С другой стороны, читатели этой книги обязательно откроют неожиданные способы применения Figaro и подобных ему языков к широкому кругу новых задач, о которых авторы даже не подозревали.

Стюарт Рассел
профессор информатики
Калифорнийский университет в Беркли

ВСТУПЛЕНИЕ

Вероятностное программирование – новая захватывающая область исследований, которая привлекает все больший интерес. Постепенно она прокладывает себе дорогу из академических кругов в мир программистов. По сути дела, вероятностное программирование – это новый способ создания вероятностных моделей, позволяющих предсказывать или выводить новые факты, которых нет в результатах наблюдений. Вероятностные рассуждения давно считаются одним из основных подходов к машинному обучению, где модель описывает то, что известно из опыта. Но раньше такие системы были ограничены простыми фиксированными структурами типа байесовских сетей. Вероятностное программирование освобождает системы вероятностных рассуждений от этих оков, предоставляя всю мощь языков программирования для представления моделей. Можно считать, что это аналог перехода от логических схем к языкам высокого уровня.

Я начал заниматься вероятностным программированием с подростковых лет, когда писал футбольный симулятор на BASIC, хотя в то время еще не осознавал этого. В программе встречались предложения вида «GOTO 1730 + RANDOM * 5», призванные выразить случайную последовательность событий. После тщательной настройки эмулятор оказался настолько реалистичным, что я мог развлекаться с ним часами. Разумеется, с тех пор вероятностное программирование далеко ушло от предложений GOTO со случайным адресом.

В 1997 я в соавторстве с Дафной Коллер и Дэвидом Макаллестером написал одну из первых работ по вероятностному программированию. В ней описывался Lisp-подобный язык, но главным новшеством был алгоритм выведения вероятных аспектов программы на основе наблюдаемых результатов ее работы. Это и вывело вероятностное программирование из разряда типичных языков вероятностного моделирования, предоставив средства, позволявшие программе не только продвигаться вперед по одному из возможных путей выполнения, но и строить обратные рассуждения и делать выводы о том, почему получен наблюдаемый результат.

В начале 2000-х годов я разработал IBAL (произносится «айболл») – первую систему вероятностного программирования общего назначения, основанную на функциональном программировании. IBAL обладала высокой выразительной способностью и содержала инновационные алгоритмы вывода, но с годами меня все сильнее раздражали ее ограничения и главное из них – трудность взаимодействия с данными и интеграции с приложениями. Поэтому в 2009 году я начал разрабатывать новую систему вероятностного программирования, которую назвал

Figaro. При проектировании Figaro на первый план выдвигалась практичность, но не жертвуя мощью вероятностного программирования. Это навело меня на мысль реализовать Figaro в виде библиотеки на языке Scala, что упрощает интеграцию вероятностных моделей с приложениями для виртуальной машины Java. В то же время Figaro предлагала, пожалуй, самый широкий спектр средств представления и алгоритмов выводов из всех известных мне систем вероятностного программирования. В настоящее время Figaro – проект с открытым исходным кодом на GitHub, его текущая версия имеет номер 3.3.

Овладеть вероятностным программированием не так-то просто, потому что необходимы разнообразные навыки и, прежде всего, умение строить вероятностные модели и писать программы. Для многих программистов написание программ – естественный процесс, но вероятностные модели окутаны тайной. Эта книга призвана сорвать покров таинственности, показать, как можно эффективно программировать в процессе создания моделей и помочь в освоении систем вероятностного программирования. Не предполагается, что читатель имеет подготовку в области машинного обучения или вероятностных рассуждений. Знакомство с функциональным программированием и языком Scala будет полезно, но для чтения книги не нужно быть опытным специалистом по Scala. Зато вполне может статься, что после прочтения вы станете программировать на Scala увереннее.

Прочитав книгу, вы сможете проектировать вероятностные модели для различных приложений, извлекающих осмысленную информацию из данных, и степень доктора по машинному обучению для этого не потребуется. Если вы специализируетесь в какой-то предметной области, то книга поможет выразить модели, которые имеются у вас в голове или на бумаге, и сделать их операционными, т. е. допускающими вычисление и анализ вариантов. Если вы – специалист по анализу данных, то, прочитав книгу, сможете разрабатывать более развитые, детальные и потенциально более точные модели, чем позволяют другие средства. Программисту или архитектору, стремящемуся включить в свою систему умение рассуждать в условиях неопределенности, книга поможет не только построить вероятностную модель, но и интегрировать ее с приложением. По какой бы причине вы ни выбрали эту книгу, я надеюсь, что она доставит вам удовольствие и окажется полезной.

БЛАГОДАРНОСТИ

Эта книга потребовала многих лет работы: от первых идей, касающихся вероятностного программирования, через создание систем IBAL и Figaro до замысла, написания и шлифовки книги в сотрудничестве с издательством Manning. Не счесть людей, усилия которых помогали созданию книги.

Своим существованием эта книга в значительной степени обязана трудам моих коллег из компании Charles River Analytics: Джо Гормана (Joe Gorman), Скотта Харрисона (Scott Harrison), Майкла Ховарда (Michael Howard), Ли Келлога (Lee Kellogg), Элисон О'Коннор (Alison O'Connor), Майка Репоза (Mike Reposa), Брайна Раттенберга (Brian Ruttenberg) и Гленна Таката (Glenn Takata). Также спасибо Скотту Нилу Рейли (Scott Neal Reilly), который поддерживал Figaro с самого начала.

Большую часть того, что я знаю об искусственном интеллекте и машинном обучении, я почерпнул от Дафны Коллер, моего научного руководителя и коллеги. Стюарт Рассел предоставил мне первую возможность изучать искусственный интеллект и воодушевлял на протяжении всей моей карьеры. А недавно мы начали работать в одном проекте, и он написал предисловие к этой книге. Майкл Стоунбрейкер предоставил мне шанс заняться исследованиями в своем проекте СУБД Postgres, и я много узнал о построении систем, работая в его группе. Алон Халеви (Alon Halevy) пригласил меня провести лето вместе с ним с в AT&T Labs, где я имел первые беседы о вероятностном программировании с Дэвидом Макаллестером; в результате родилась на свет статья о вероятностном диалекте Lisp, написанная в соавторстве с Дафной. С Лайзой Гетур (Lise Getoor), делившей со мной кабинет и общую работу, я мог обсуждать идеи, которые вынашивал.

Я глубоко признателен Алексу Ихлеру (Alex Ihler), который любезно проверил книгу на предмет отсутствия технических ошибок. Алекс также оказался великолепным слушателем, на котором последние пару лет я опробовал все идеи, относящиеся к логическому выводу.

Многие люди делились своими замечаниями на различных этапах разработки, в том числе Равишанкар Раджагопалан (Ravishankar Rajagopalan) и Шабеш Балан (Shabeesh Balan), Крис Хенеган (Chris Heneghan), Клеменс Баадер (Clemens Baader), Кроистофер Вебер (Cristofer Weber), Эрл Бингэм (Earl Bingham), Джузеппе де Марко (Giuseppe de Marco), Джауме Валлс (Jaume Valls), Хавьер Гуэрра Гиральдес (Javier Guerra Giraldez), Костас Пассадис (Kostas Passadis), Лука Кампобассо (Luca Campobasso), Лукас Наллиндо (Lucas Gallindo), Марк Элстон

(Mark Elston), Марк Миллер (Mark Miller), Нитин Годе (Nitin Gode), Одиссеас Пентаколос (Odisseyas Pentakolos), Петр Рабинович (Peter Rabinovitch), Филлип Брэдфорд (Phillip Bradford), Стивен Уэйкли (Stephen Wakely), Тапош Дутта Рой (Taposh Dutta Roy), Унникришнан Кумар (Unnikrishnan Kumar).

Я благодарен многим замечательным сотрудникам издательства Manning, которые способствовали выходу книги из печати. Отдельное спасибо редактору Дэну Махарри (Dan Maharry), который сделал книгу куда лучше, чем мог бы я сам, и Фрэнку Полманну (Frank Pohlmann), который посоветовал мне написать книгу и помогал на подготовительных этапах.

Спасибо Исследовательской лаборатории ВВС и Управлению перспективных научно-исследовательских проектов Министерства обороны США (DARPA) за финансирование части описанной в этой книге работы в рамках программы PPAML (вероятностное программирование для развития методов машинного обучения). Отдельная благодарность нескольким руководителям программ в DARPA: Бобу Кохоуту (Bob Kohout), Тони Фальконе (Tony Falcone), Кэтлин Фишер (Kathleen Fisher) и Сурешу Джаганнатану (Suresh Jagannathan), которые поверили в вероятностное программирование и прилагали все силы, чтобы оно стало реальностью. Этот материал основан на работе, которая финансировалась ВВС США по контракту № FA8750-14-C-0011. Мнения, открытия и заключения или рекомендации автора, встречающиеся в этом материале, не обязательно отражают точку зрения ВВС США.

И наконец, эта книга не состоялась бы без любви и поддержки со стороны моей семьи. Спасибо моей супруге Дебби Гелбер и моим детям Дине, Номи и Рути за то, что они такие чудесные. И вечная благодарность моей маме Клэр Пфеффер, воспитавшей меня с любовью. Эту книгу я посвящаю твоей памяти.

ОБ ЭТОЙ КНИГЕ

Многие решения, будь то в бизнесе, науке, военном деле или повседневной жизни, принимаются в условиях неопределенности. Если разные факторы толкают в разных направлениях, то как узнать, на что обращать больше внимания? Вероятностные модели дают средство выразить всю информацию, относящуюся к конкретной ситуации. Вероятностные рассуждения позволяют использовать эти модели, чтобы найти вероятности величин, наиболее существенных для принятия решения. С помощью вероятностных рассуждений можно *предсказать* наиболее вероятное развитие событий: хорошо ли будет продаваться ваш продукт по назначенной цене, будет ли пациент отвечать на предложенное лечение, сможет ли кандидат победить на выборах, если займет определенную позицию? Вероятностные рассуждения можно использовать и для *вывода* вероятных причин уже случившихся событий: если продажи провалились, то не потому ли, что была назначена слишком высокая цена?

Вероятностное рассуждение также является одним из основных подходов к машинному обучению. Мы представляем исходные допущения о предметной области в виде вероятностной модели, например общей модели поведения пользователей в ответ на появление продуктов на рынке. Затем, получив обучающие данные, скажем о реакции пользователей на конкретные продукты, мы модифицируем первоначальные представления и получаем новую модель. Эту модель можно уже использовать для предсказания будущих событий, например успеха перспективного продукта, или для объяснения причин наблюдавшихся событий, скажем провала нового продукта.

Раньше для выражения вероятностных рассуждения применялись специализированные языки представления вероятностных моделей. В последние годы мы поняли, что можно использовать обычные языки программирования, и это положило начало вероятностному программированию. У такого подхода три основных достоинства. Во-первых, при построении моделей в нашем распоряжении все средства языка программирования: развитые структуры данных, поток управления и т. д. Во-вторых, вероятностную модель легко интегрировать с другими приложениями. И, в-третьих, для рассуждений о моделях можно использовать универсальные алгоритмы логического вывода.

Задача этой книги – вооружить вас знаниями, необходимыми для использования вероятностного программирования в повседневной деятельности. В частности, объясняется:

- как строить вероятностные модели и выражать их в виде вероятностных программ;
- как устроены вероятностные рассуждения и как они реализованы в различных алгоритмах вывода;
- как с помощью системы вероятностного программирования Figaro создавать практичные вероятностные программы.

Система Figaro реализована в виде библиотеки на языке Scala. Как и в Scala, в ней сочетаются функциональный и объектно-ориентированный стили программирования. Некоторое знакомство с функциональным программированием было бы полезно. Но в этой книге не используются сложные концепции функционального программирования, так что для понимания хватит и скромных познаний. Знание Scala также не помешало бы. Хотя встречающиеся конструкции Scala часто объясняются, эта книга не является введением в Scala. Но экзотические особенности Scala почти не используются, так что поверхностного знакомства будет достаточно.

Структура книги

Часть 1 представляет собой введение в вероятностное программирование и систему Figaro. В начале главы 1 объясняется, что такое вероятностное программирование и почему оно полезно, а затем приводится краткое введение в Figaro. Глава 2 – учебное пособие по работе с Figaro, она позволит вам быстро приступить к написанию вероятностных программ. В главе 3 вы найдете полное приложение вероятностного программирования – фильтр спама – включающее компонент, который рассуждает о том, является данное почтовое сообщение хорошим или спамным, и компонент, который обучает вероятностную модель на данных. Цель главы 3 – дать общую картину взаимосвязей разных частей, перед тем как приступать к детальному изучению приемов моделирования.

Часть 2 целиком посвящена построению вероятностных программ. В главе 4 представлены основные сведения о вероятностных моделях и вероятностных программах, которые необходимы для понимания того, что в действительности стоит за созданием таких программ. В главе 5 описаны две системы моделирования, лежащие в основе вероятностного программирования: байесовские и марковские сети. Главы 6–8 содержат набор полезных приемов программирования, применяемых при создании более сложных программ. В главе 6 речь идет о коллекциях в Scala и Figaro, позволяющих скомпоновать много переменных одного типа. Глава 7 посвящена объектно-ориентированному программированию, которое в функциональных программах не менее полезно, чем в обычных. В главе 8 рассказано о моделировании динамических систем. Динамической называется система, состояние которой изменяется со временем, и в этой главе подробно рассматривается это весьма распространенное и важное применение вероятностных рассуждений.

Из части 3 вы узнаете о вероятностных алгоритмах вывода. Понимать, что такое вывод, необходимо для эффективного применения вероятностного программирования; это позволит выбрать подходящий для решения задачи алгоритм, правиль-

но настроить его и выразить модель так, чтобы она эффективно поддерживала рассуждения. Я старался соблюсти баланса между изложением теории, стоящей за алгоритмами, и практическими примерами их использования. В основополагающей главе 9 сформулированы три правила, в которых заключены основные идеи вероятностного вывода. В главах 10 и 11 описаны два основных семейства алгоритмов вывода. В главе 10 рассматриваются факторные алгоритмы, в том числе введение в факторы и принципы их работы, а также алгоритмы исключения переменных и распространения доверия. В главе 11 обсуждаются выборочные алгоритмы с акцентом на выборку по значимости и методы Монте-Карло по схеме марковской цепи. Если в главах 10 и 11 упор сделан на базовом запросе – вычислении вероятностей величин, представляющих интерес, то в главе 12 показано, как факторные и выборочные алгоритмы можно использовать для вычисления других запросов, например, совместной вероятности нескольких переменных, наиболее вероятных значений переменных и вероятности эмпирических данных. Наконец, в главе 13 обсуждаются две более сложных, но важных задачи вывода: мониторинг динамической системы, изменяющейся во времени, и нахождение числовых параметров вероятностной модели на основе обучающих данных.

Каждая глава сопровождается упражнениями. Это могут быть как простые вычисления, так и задачи на размышление, не предполагающие однозначного ответа.

В книге есть еще два приложения. Приложение A содержит инструкцию по установке Figaro, а приложение B – краткий обзор других систем вероятностного программирования.

О коде и упражнениях

Код набран моноширинным шрифтом, чтобы не путать его с обычным текстом. Многие листинги сопровождаются аннотациями, в которых объяснены важные концепции. В некоторых случаях за листингом следует несколько комментариев, оформленных как отдельные пункты.

Многие примеры кода имеются в сети, найти их можно на сайте книги по адресу www.manning.com/books/practical-probabilistic-programming. Там же есть ответы на избранные упражнения.

Об авторе

Ави Пфеффер – пионер вероятностного программирования, работающий в этой области с момента ее зарождения. Ави – ведущий проектировщик и разработчик Figaro. В компании Charles River Analytics Ави занимается применением Figaro к различным задачам, в том числе к анализу вредоносного ПО, мониторингу исправности транспортных средств, моделированию климата и оценке инженерных систем.

В свободное время Ави поет, сочиняет музыку и продюсирует музыкальные произведения. Ави живет в Кембридже, штат Массачусетс, с женой и тремя детьми.

Автор в сети

Приобретение книги «Практическое вероятностное программирование» открывает бесплатный доступ к закрытому форуму, организованному издательством Manning Publications, где вы можете оставить свои комментарии к книге, задать технические вопросы и получить помощь от автора и других пользователей. Получить доступ к форуму и подписаться на список рассылки можно на странице www.manning.com/books/practical-probabilistic-programming. Там же написано, как зайти на форум после регистрации, на какую помощь можно рассчитывать, и изложены правила поведения в форуме.

Издательство Manning обязуется предоставлять читателям площадку для общения с другими читателями и автором. Однако это не означает, что автор обязан как-то участвовать в обсуждениях; его присутствие на форуме остается чисто добровольным (и не оплачивается). Мы советуем задавать автору какие-нибудь хитроумные вопросы, чтобы его интерес к форуму не угасал!

Форум автора в сети и архивы будут доступны на сайте издательства до тех пор, пока книга не перестанет печататься.

Об иллюстрации на обложке

Рисунок на обложке книги называется «The Venetian», или «Житель Венеции». Он взят из книги Жака Грассе де Сен-Совера «Энциклопедия путешествий», опубликованной в 1796 году. Путешествие ради удовольствия в те времена было в новинку, а путеводители, подобные этому, были весьма популярны – они знакомили настоящих туристов и путешественников, не покидающих своего кресла, с жителями других регионов Франции и всего мира.

Разнообразие иллюстраций в «Энциклопедии путешествий» красноречиво свидетельствует об уникальности и индивидуальности городков и провинций, которое можно было наблюдать каких-то 200 лет назад. То было время, когда по манере одеваться, отличающейся в местах, разделенных всего несколькими десятками миль, можно было сказать, откуда человек родом. Этот путеводитель возвращает к жизни ощущение изолированности и удаленности, свойственное тому периоду, да и всем остальным историческим периодам, кроме нашего гиперактивного настоящего.

С тех пор манера одеваться сильно изменилась, и различия между областями, когда-то столь разительные, сгладились. Теперь трудно отличить друг от друга даже выходцев с разных континентов, что уж говорить о странах или областях. Но можно взглянуть на это и с оптимизмом – мы обменяли культурное и визуальное разнообразие на иное устройство личной жизни – основанное на многостороннем и стремительном технологическом и интеллектуальном развитии.

Издательство Manning откликается на новации и инициативы в компьютерной отрасли обложками своих книг, на которых представлено широкое разнообразие местных укладов быта в позапрошлом веке. Мы возвращаем его в том виде, в каком оно запечатлено на рисунках из разных собраний, в том числе из этого путеводителя.

Часть I

ВВЕДЕНИЕ В ВЕРОЯТНОСТНОЕ ПРОГРАММИРОВАНИЕ И СИСТЕМУ FIGARO

Что такое вероятностное программирование? Почему оно полезно? Как им пользоваться? Ответы на эти вопросы составляют основное содержание первой части. В главе 1 излагаются основные идеи вероятностного программирования. Прежде всего, мы познакомимся с концепцией системы вероятностных рассуждений и покажем, как вероятностное программирование соединяет традиционные системы вероятностных рассуждений с технологией языков программирования.

В этой книге мы будем использовать систему вероятностного программирования Figaro. В главе 1 дается краткое введение в Figaro, а глава 2 представляет собой пособие по основным понятиям Figaro, позволяющее быстро приступить к написанию вероятностных программ. Глава 3 содержит законченное вероятностное приложение, из которого вы поймете, как реальная программа собирается из различных частей. Эта глава находится в начале книги, чтобы можно было составить общую картину, но имеет смысл возвращаться к ней и позже, после более глубокого знакомства с различными концепциями.

ГЛАВА 1. О вероятностном программировании в двух словах

В этой главе.

- Что такое вероятностное программирование?
- Какое мне до него дело? А моему начальнику?
- Как оно работает?
- Figaro – система вероятностного программирования.
- Сравнение вероятностных приложений, написанных с применением и без применения вероятностного программирования.

Из этой главы вы узнаете, как принимать решения с помощью вероятностной модели и алгоритма вывода – двух главных составных частей системы вероятностных рассуждений. Вы также узнаете, как современные языки вероятностного программирования упрощают создание подобных систем по сравнению с такими универсальными языками, как Java или Python. Здесь же вы познакомитесь с *Figaro*, основанным на Scala языком вероятностного программирования, которым мы будем пользоваться в этой книге.

1.1. Что такое вероятностное программирование?

Вероятностное программирование – это способ создания систем, помогающих принимать решения в условиях неопределенности. Многие решения, принимаемые нами каждодневно, подразумевают учет релевантных факторов, которые мы не наблюдаем непосредственно. Исторически одним из способов принять решение в таких условиях неопределенности стали системы вероятностного рассужде-

ния. *Вероятностное рассуждение* позволяет объединить наши знания о ситуации с вероятностными законами и таким образом учесть ненаблюдаемые факторы, важные для принятия решения. До недавнего времени область применения систем вероятностного рассуждения была ограничена, и ко многим возникающим на практике задачам их удавалось применить с большим трудом. Вероятностное программирование – это новый подход, позволивший упростить построение систем вероятностного рассуждения и расширить их применимость.

Чтобы понять, в чем смысл вероятностного программирования, начнем с рассмотрения того, как принимается решение в условиях неопределенности и какую роль в этом играют субъективные суждения. Затем посмотрим, как может помочь система вероятностного рассуждения. Мы познакомимся с тремя видами рассуждений, присущих таким системам. После этого станет понятно, что такое вероятностное программирование и как оно позволяет использовать всю мощь языков программирования для построения систем вероятностного рассуждения.

1.1.1. Как мы высказываем субъективное суждение?

В реальном мире интересующие нас вопросы редко допускают недвусмысленный ответ: да или нет. Например, при выводе на рынок нового продукта хотелось бы знать, хорошо ли он будет продаваться. Вы полагаете, что его ждет успех, потому что продукт хорошо спроектирован и изучение рынка показало, что в нем есть потребность, но полной уверенности нет. Быть может, ваш конкурент представит лучший продукт, а, быть может, в вашем продукте обнаружится фатальный изъян, из-за которого рынок от него отвернется. Или экономическая ситуация повернется к худшему. Если вы хотите стопроцентной уверенности, то никогда не сможете принять решение о начале продаж (см. рис. 1.1).

Для принятия решений такого рода помогает язык вероятностей. Запуская продукт, мы можем воспользоваться прошлым опытом продажи похожих продуктов для оценки вероятности того, что этот окажется успешным. А зная эту вероятность, мы сможем решить, выводить продукт на рынок или нет. Возможно, нас интересует не только сам факт успешности продукта, но и размер ожидаемого дохода или, напротив, размер убытков, которые мы понесем в случае неудачи. Знание вероятностей различных исходов помогает принимать более обоснованные решения.

Ну хорошо, вероятностное мышление может помочь в принятии трудных решений с элементами субъективности. Но как это выглядит на практике? Общий принцип выражен в виде следующего факта.

Факт. Субъективное суждение основано на *знании + логике*.

Вы располагаете некоторыми знаниями об интересующей вас проблеме. Например, вы много знаете о своем продукте и, возможно, провели исследование рынка, чтобы понять, кто будет его покупать. Возможно, вы также кое-что знаете о конкурентах и знакомы с прогнозом развития экономики. Логика же позволяет получать ответы на вопросы с использованием имеющихся знаний.

Рис. 1.1. В прошлом году мой продукт всем нравился, но как будет в следующем?

Нам необходим способ описания своих знаний и логика получения ответов на вопросы с использованием этих знаний. Вероятностное программирование дает то и другое. Но прежде чем переходить к описанию системы вероятностного программирования, я опишу более общую концепцию системы вероятностных рассуждений, которая предоставляет базовые средства для спецификации знаний и реализации логики.

1.1.2. Системы вероятностных рассуждений помогают принимать решения

Вероятностные рассуждения – это подход, в котором модель предметной области используется для принятия решений в условиях неопределенности. Возьмем пример из области футбола. Предположим, что по статистике 9 % угловых завершаются голом. Требуется предсказать исход конкретного углового удара. Рост центрального нападающего атакующей команды равен 193 см, и известно, что он отлично играет головой. Основного вратаря защищающейся команды только что унесли на носилках, и на замену ему вышел вратарь, который проводит свой пер-

вый матч. Кроме того, дует сильный ветер, который затрудняет контроль над полетом мяча. И как при таких условиях вычислить вероятность?

Рис. 1.2. Как система вероятностных рассуждений предсказывает исход углового удара

На рис. 1.2 показано, как получать ответ с помощью системы вероятностных рассуждений. Все свои знания об угловых ударах и релевантных факторах мы кодируем в виде модели углового. Затем мы предоставляем факты о конкретном угловом ударе: что центральный нападающий высокий, что вратарь неопытный и что дует сильный ветер. Мы говорим системе, что хотели бы узнать, будет ли забит гол. Алгоритм вывода возвращает ответ: гол будет забит с вероятностью 20 %.

Основные определения

Общие знания – что известно о предметной области в общем, без деталей, описывающих конкретную ситуацию.

Вероятностная модель – представление общих знаний о предметной области в количественном виде, в терминах вероятностей.

Факты – имеющаяся информация о конкретной ситуации.

Запрос – свойство ситуации, о котором мы хотели бы узнать.

Вывод – процесс использования вероятностной модели для получения ответа на запрос при известных фактах.

В системе вероятностных рассуждений мы создаем *модель*, в которой отражены все релевантные общие знания о предметной области в виде вероятностей. В нашем примере это может быть описание ситуации углового удара и всех релевант-

ных характеристик игроков и окружающих условий, которые могут повлиять на исход. Затем модель применяется к конкретной ситуации, для которой мы хотим получить заключение. Для этого мы передаем модели имеющуюся информацию, которая называется *фактами*. В данном случае есть три факта: центральный нападающий высокий, вратарь неопытный и дует сильный ветер. Полученное от модели заключение поможет принять решение – например, что на следующую игру нужно поставить другого вратаря. Сами заключения предстают в виде вероятностей, например, вероятность других игровых навыков вратаря.

Соотношение между моделью, предоставляемой вами информацией и ответами на запросы математически строго определено законами теории вероятностей. Процесс использования модели для получения ответов на запросы при известных фактах называется *вероятностным выводом*, или просто *выводом*. По счастью, разработаны алгоритмы, которые выполняют необходимые вычисления, скрывая от вас всю математику. Они называются *алгоритмами вывода*.

На рис. 1.3 схематически изображены описанные выше понятия.

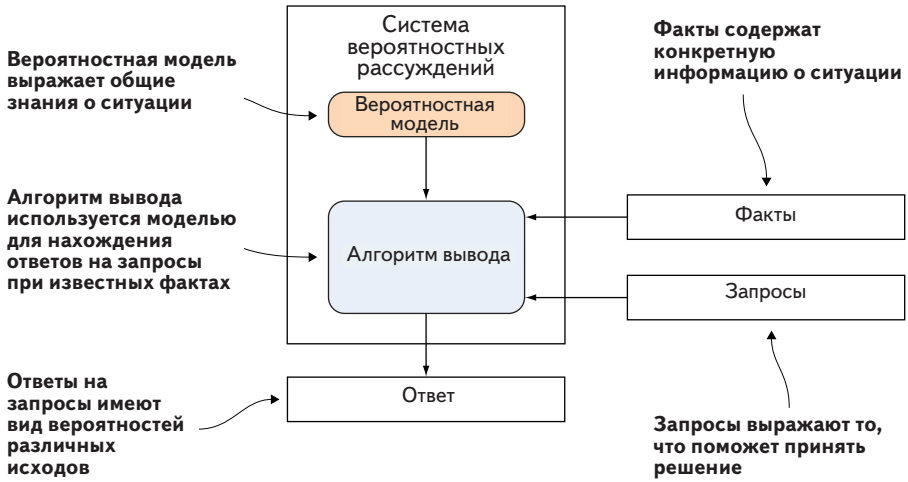

Рис. 1.3. Основные компоненты системы вероятностных рассуждений

Итак, мы вкратце описали основные компоненты системы вероятностных рассуждений и способ взаимодействия с ней. Но что можно делать с такой системой? Как она помогает принимать решения? В следующем разделе описаны три вида рассуждений, которые может выполнять такая система.

1.1.3. Система вероятностных рассуждений может рассуждать тремя способами

Системы вероятностных рассуждений обладают гибкостью. Они могут отвечать на запросы о любом аспекте ситуации, если заданы факты, относящиеся к любым другим аспектам. На практике система выполняет рассуждения трех видов.

- *Предсказание будущих событий*. Мы уже видели это на рис. 1.2, где система предсказывает, будет ли в данной ситуации забит гол. Факты обычно включают информацию о текущей ситуации, например рост центрального нападающего, опытность вратаря и силу ветра.
- *Вывод причины событий*. Прокрутим пленку вперед на 10 секунд. Высокий центральный нападающий только что забил гол головой, протолкнув мяч под корпусом вратаря. Что можно сказать об этом вратаре-новобранце при таких фактах? Можно ли заключить, что ему не хватает умения? На рис. 1.4 показано, как использовать систему вероятностных рассуждений для ответа на этот вопрос. Берется та же модель углового удара, что и для прогноза гола. (Это полезное свойство вероятностных рассуждений: одну и ту же модель можно использовать как для предсказания будущего результата, так и для объяснения причин, приведших к известному результату.) Факты те же, что и прежде, плюс тот факт, что гол забит. Запрос касается квалификации вратаря, а ответ дает вероятности различных уровней квалификации.

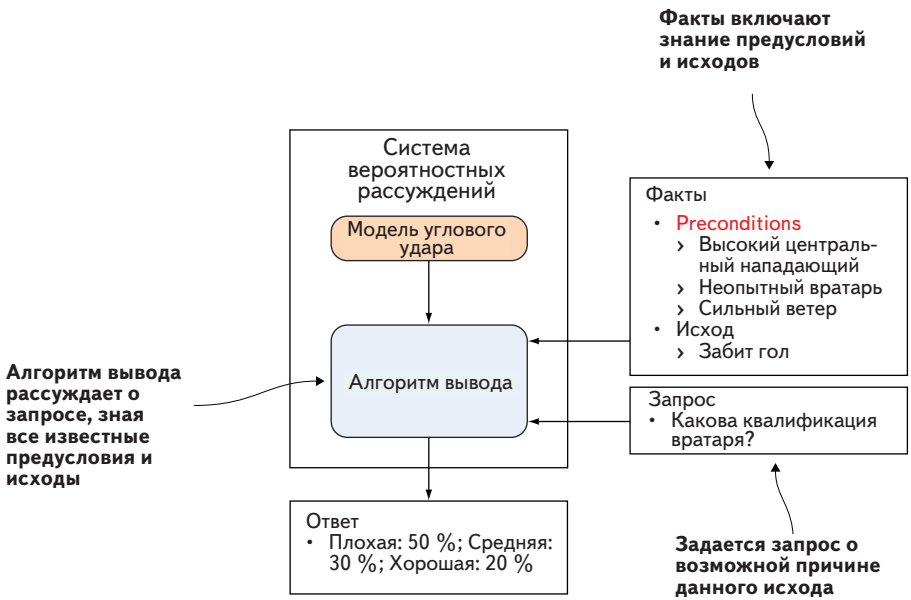

Рис. 1.4. Если изменить запрос и факты, то система сможет сделать вывод о том, почему был забит гол

Немного поразмыслив, вы поймете, что первый способ – рассуждения с прямым ходом времени, т. е. предсказание будущих событий на основе того, что известно о текущей ситуации, а второй – рассуждения с обратным ходом времени, когда требуется вывести прошлые условия из известных результатов. Типичная вероятностная модель следует естественному ходу времени.

Игрок пробивает угловой, затем ветер воздействует на летящий мяч, затем центральный нападающий прыгает, пытаясь достать мяч головой, и, наконец, вратарь пытается спасти ворота. Но процесс рассуждения может происходить как в прямом, так и в обратном порядке. Это ключевая особенность вероятностного рассуждения, которую я не раз буду подчеркивать: направление рассуждения необязательно совпадает с направлением модели.

- *Обучение на прошлых событиях, чтобы лучше предсказывать будущие.* Прокрутим пленку еще на 10 минут вперед. Та же команда заработала еще один угловой. Все так же, как и раньше – высокий центральный нападающий, неопытный вратарь – только ветер теперь стих. Вероятностное рассуждение позволяет использовать информацию о том, что случилось при подаче предыдущего углового, чтобы лучше предсказать исход следующего. На рис. 1.5 показано, как это делается. В состав фактов входит все то, что и в прошлый раз (с пометкой, что это было в прошлый раз), а также новая информация о текущей ситуации. Отвечая на вопрос, будет ли забит гол на этот раз, алгоритм вывода сначала определяет свойства ситуации, которые привели к голу в первый раз, например, квалификацию нападающего и вратаря. А затем эти свойства используются для предсказания исхода в новой ситуации.

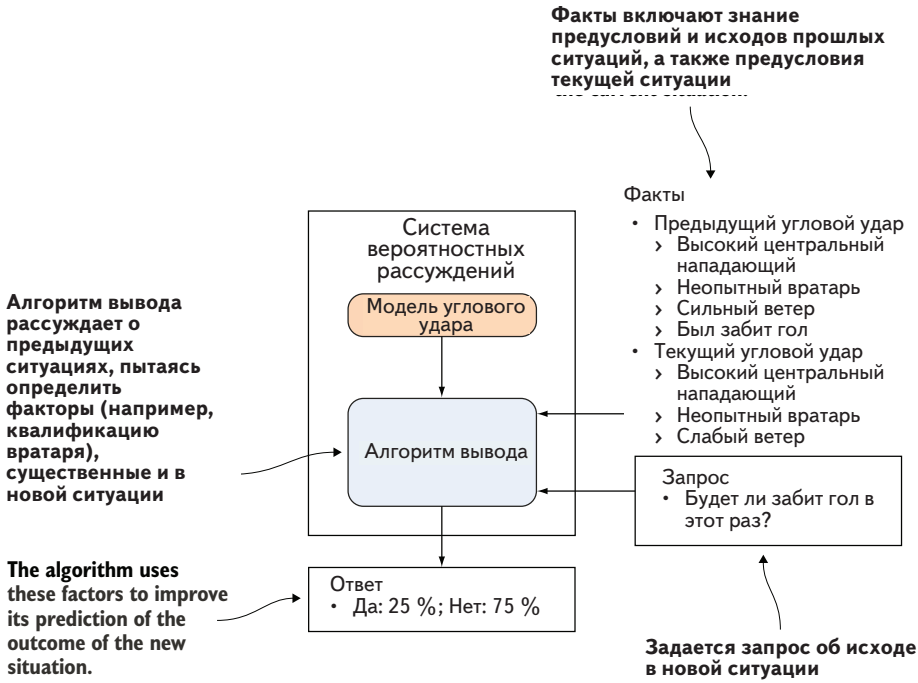

Рис. 1.5. Принимая во внимание факты, относящиеся к результату прошлого углового, система вероятностного рассуждения может лучше предсказать результат следующего

Запросы перечисленных видов могут помочь в принятии различных решений.

- Зная вероятность гола при наличии или отсутствии дополнительного защитника, можно решить, не стоит ли заменить атакующего защитником.
- На основе оценке квалификации вратаря можно решить, какую зарплату предложить ему в следующем контракте.
- Получив информацию о вратаре, можно решить, стоит ли ставить его на следующую игру.

Обучение улучшенной модели

Выше были описаны способы рассуждения о конкретных ситуациях при известных фактах. Но система вероятностных рассуждений позволяет сделать еще одну вещь: обучаясь на прошлом опыте, улучшить общие знания. Третий способ рассуждений – это использование прошлых результатов для лучшего предсказания исхода в конкретной новой ситуации. А сейчас мы говорим об улучшении самой модели. Если имеется обширный прошлый опыт, т. е. информация о многих угловых ударах, то почему бы не обучить новую модель, представляющую общие знания о том, что обычно происходит при подаче углового? На рис. 1.6 показано, что это достигается с помощью алгоритма обучения. В отличие от алгоритма вывода, его задача состоит не в том, чтобы отвечать на вопросы, а в том, чтобы породить новую модель. На вход алгоритма обучения подается исходная модель, а он обновляет ее на основе опыта. Затем новую модель можно использовать для ответа на будущие запросы. Предположительно ответы новой модели будут более обоснованы, чем ответы исходной.

Системы вероятностных рассуждений и точные предсказания

Как и в любой системе машинного обучения, чем больше данных у системы вероятностных рассуждений, тем точнее ее ответы. Качество предсказания зависит от двух факторов: верности отражения реального мира в исходной модели и объема предоставленных данных. Вообще говоря, чем больше данных, тем менее существенно качество исходной модели. Причина в том, что новая модель – комбинация исходной и информации, содержащейся в данных. Если данных очень мало, то доминирует исходная модель, поэтому чем она точнее, тем лучше. Если же данных много, то доминируют именно они, так что в новой модели остается очень мало от исходной, поэтому ее точность не слишком важна. Например, если мы обучаем модель по данным за весь футбольный сезон, то факторы, от которых зависит результативность углового, будут учтены весьма точно. Если же имеются данные только об одном матче, то придется изначально задать хорошие значения факторов, иначе на точность предсказаний можно не рассчитывать. Системы вероятностного программирования умеют пользоваться моделью и имеющимися данными так, что предсказания получаются максимально точными.

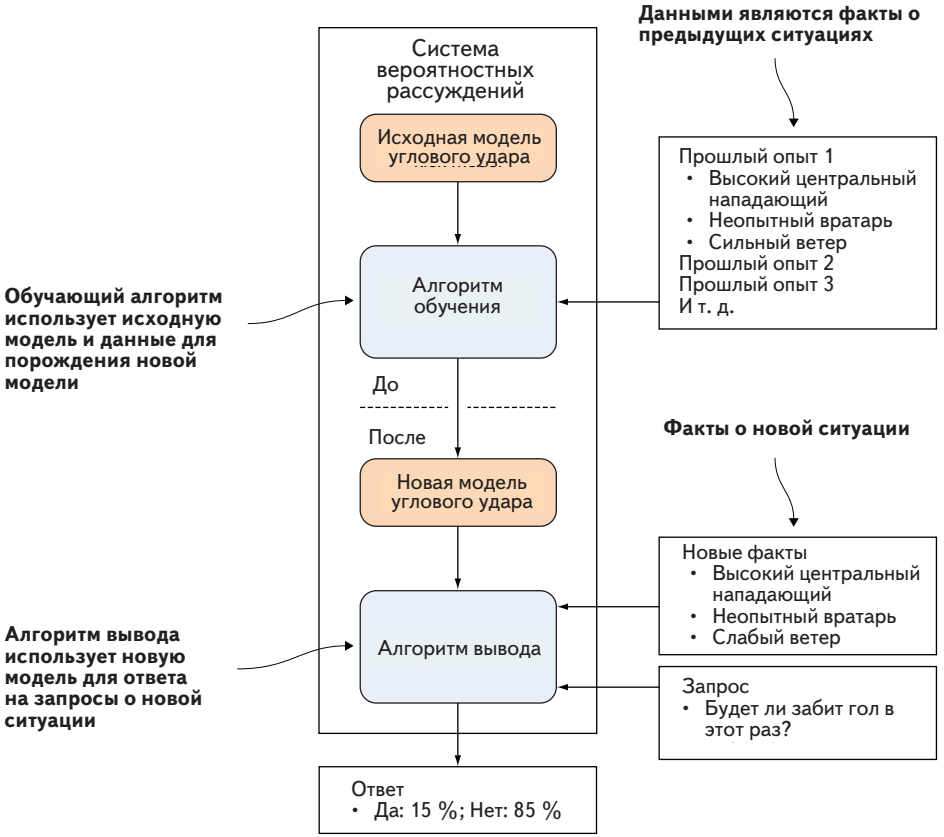

Рис. 1.6. С помощью алгоритма обучения можно обучить новую модель на основе прошлого опыта, а затем использовать ее для будущих выводов

Теперь вы знаете, что такое вероятностное рассуждение. Ну а что же тогда такое вероятностное программирование?

1.1.4. Система вероятностного программирования: система вероятностных рассуждений, выраженная на языке программирования

В любой системе вероятностных рассуждений используется *язык представления*, на котором выражаются вероятностные модели. Таких языков много. Возможно, о некоторых вы слыхали, например, о байесовских сетях (их еще называют сетями доверия) или скрытых марковских моделях. От языка представления зависит, какие модели сможет обработать система и как они выглядят. Множество моделей,

представимых языком, называется *выразительной силой* языка. Для разработки приложений требуется, чтобы выразительная сила была как можно больше.

Система *вероятностного программирования* – это просто система вероятностных рассуждений, для которой языком представления является язык программирования. Говоря «*язык программирования*», я имею в виду, что он обладает всеми возможностями, которые принято ожидать от типичного языка программирования: переменными, развитой системой типов данных, средствами управления потоком выполнения, функциями и т. д. Как вы вскоре увидите, языки вероятностного программирования способны выразить широкий спектр вероятностных моделей, далеко выходящий за рамки большинства традиционных систем вероятностных рассуждений. Выразительная сила языков вероятностного программирования очень высока.

На рис. 1.7 иллюстрируется соотношение между системами вероятностного программирования и вероятностных рассуждений в общем случае. Сравните этот рисунок с рис. 1.3, чтобы стали яснее различия между двумя системами. Основное изменение состоит в том, что модели выражаются в виде программ на некотором языке программирования, а не в виде математических конструкций типа байесовской сети. Поэтому факты, запросы и ответы становятся переменными программы. Факты можно представить конкретными значениями переменных, запрос – это получение значения переменной, а ответ – вероятности того, что переменные принимают те или иные значения. Кроме того, система вероятностного программирования обычно включает набор алгоритмов вывода. Эти алгоритмы применяются к моделям, написанным на языке системы.

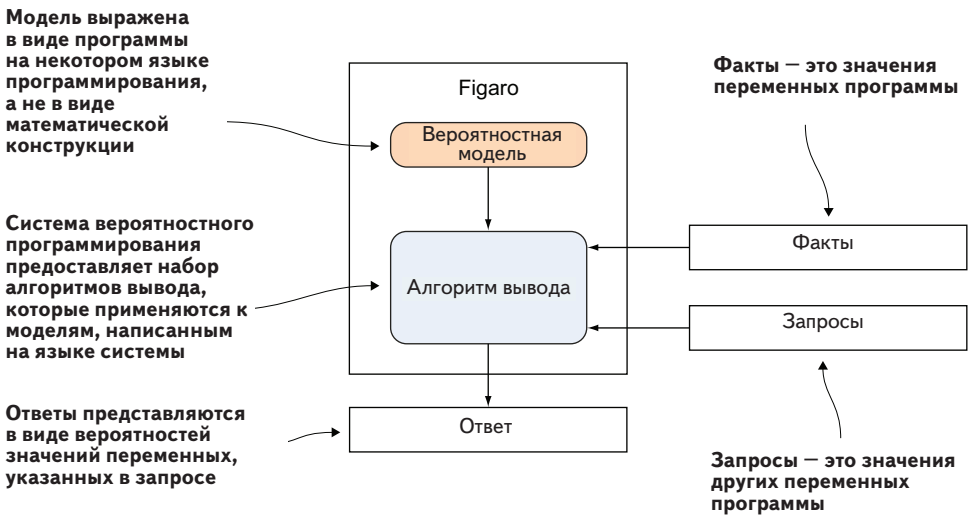

Рис. 1.7. Система вероятностного программирования – это система вероятностных рассуждений, в которой для представления вероятностных моделей применяется язык программирования

Существует много систем вероятностного программирования (см. обзор в приложении B), но в этой книге рассматриваются только функциональные, полные по Тьюрингу системы. *Функциональные* означает, что они основаны на функциональном языке программирования, но пусть это вас не пугает – чтобы пользоваться функциональной системой вероятностного программирования, не нужно знать, что такое лямбда-функция. Просто функциональное программирование составляет теоретическое основание, благодаря которому язык имеет возможность представлять вероятностные модели. А *полным по Тьюрингу* называется язык, на котором можно закодировать любое вычисление, которое способен выполнить цифровой компьютер. Если нечто вообще можно сделать с помощью компьютера, то это можно сделать на любом полном по Тьюрингу языке. Большинство языков программирования, с которыми вы знакомы, например C, Java и Python, являются полными по Тьюрингу. Поскольку системы вероятностного программирования основаны на языках программирования, полных по Тьюрингу, то они обладают выдающейся гибкостью в части типов моделей, которые можно построить с их помощью.

Основные определения

Язык представления – язык для кодирования знаний о предметной области в модели.

Выразительная сила – способность языка представления кодировать различные виды знаний.

Полный по Тьюрингу – язык, позволяющий выразить любое вычисление, которое можно выполнить на цифровом компьютере.

Язык вероятностного программирования – язык представления системы вероятностных рассуждений, в котором для представления знаний используется полный по Тьюрингу язык программирования.

В приложении B приведен обзор некоторых систем вероятностного программирования, помимо используемой в этой книге системы Figaro. В большинстве из них используются полные по Тьюрингу языки. В некоторых, например BUGS и Dimple, это не так, но они все равно полезны для тех приложений, на которые рассчитаны. В этой книге мы будем заниматься исключительно возможностями полных по Тьюрингу языков вероятностного программирования.

Представление вероятностных моделей в виде программ

Но каким образом язык программирования становится языком вероятностного моделирования? Как представить вероятностную модель в виде программы? Сейчас я дам очень краткий ответ на этот вопрос, а более глубокое обсуждение отложу на потом, когда мы станем лучше понимать, как выглядит вероятностная программа.

Главная идея любого языка программирования – *выполнение*. Мы выполняем программу, порождающую некоторый выход. Вероятностная программа в этом смысле аналогична, только она может иметь не один, а несколько путей выполнения, каждый из которых порождает свой выход. По какому пути следовать, определяется случайным выбором, совершаемым внутри программы. Каждый случайный выбор

имеет несколько возможных исходов, и в программе закодирована вероятность каждого исхода. Поэтому можно считать, что вероятностная программа – это программа, при выполнении которой случайным образом генерируются выходные данные.

Эта идея иллюстрируется на рис. 1.8. Здесь система вероятностного программирования содержит программу углового удара, которая описывает случайный процесс генерации исхода углового. Программа принимает ряд входных данных; в нашем примере это рост центрального нападающего, опытность вратаря и сила ветра. На основе этих данных программа случайным образом выполняется и генерирует выходные данные. Каждое случайное выполнение дает на выходе конкретный результат. Поскольку каждый случайный выбор может иметь несколько исходов, то существует много возможных путей выполнения, дающих различные результаты. Любой заданный результат, например взятие ворот, может быть сгенерирован на различных путях выполнения.

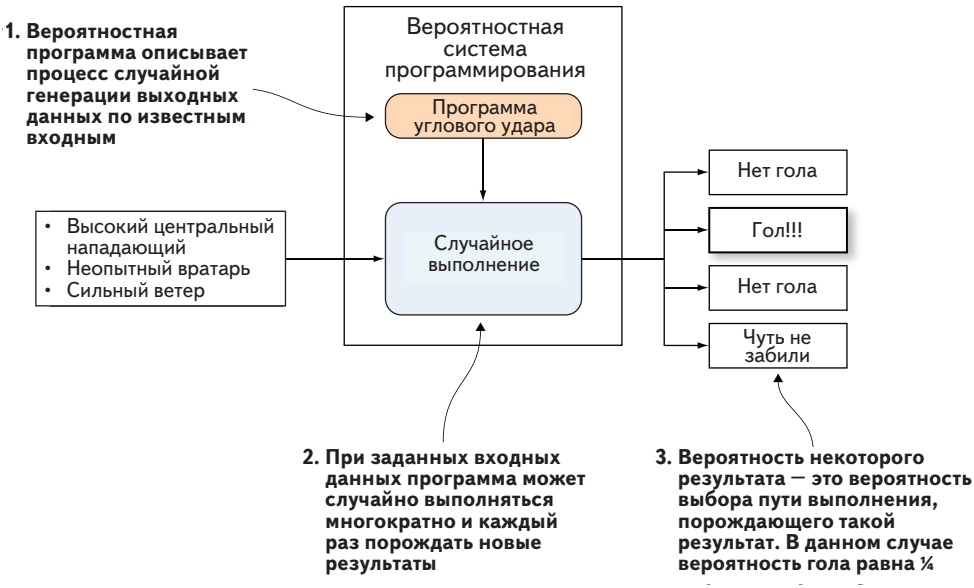

Рис. 1.8. Вероятностная программа определяет процесс случайной генерации выходных данных по известным входным

Рассмотрим, как эта программа определяет вероятностную модель. Любое конкретное выполнение программы является результатом последовательности случайных выборов. Каждый случайный выбор происходит с некоторой вероятностью. Если перемножить все эти вероятности, то получится вероятность пути выполнения. Таким образом, программа определяет вероятность каждого пути выполнения. Если представить себе, что программа прогоняется многократно, то доля прогонов, на которых генерируется заданный путь выполнения, равна его вероятности. Вероятность некоторого результата равна доле прогонов, на которых

был получен этот результат. На рис. 1.8 гол получается в 1/4 всех прогонов, поэтому вероятность гола равна 1/4.

Примечание. Возможно, вы недоумеваете, почему один из блоков на рис. 1.8 назван «Случайное выполнение», а не «Алгоритм вывода», как на других рисунках. На рис. 1.8 показан смысл вероятностной программы – определение процесса случайного выполнения, а не как использовать систему вероятностного программирования – выполнить алгоритм вывода для получения ответа на запрос при заданных фактах. Поэтому хотя структурно рисунки похожи, они иллюстрируют разные концепции. На самом деле, случайное выполнение лежит в основе и некоторых алгоритмов вывода, но есть много алгоритмов, которые не основаны на простом случайном выполнении.

Принятие решений с помощью вероятностного программирования

Легко видеть, как можно использовать вероятностную программу для предсказания будущего. Нужно просто выполнить ее случайное число раз, подавая на вход известные данные о настоящем, и посмотреть, сколько раз порождается каждый результат. В примере с угловым ударом на рис. 1.8 мы выполняли программу много раз, подавая на вход такие данные: высокий центральный нападающий, неопытный вратарь и сильный ветер. Поскольку в 1/4 прогонов получился гол, можно сказать, что при этих входных данных вероятность гола равна 25 %.

Однако прелесть вероятностного программирования заключается в том, что его можно с тем же успехом использовать для любых видов вероятностных рассуждений, описанных в разделе 1.3.1. Оно позволяет не только предсказывать будущее, но и выводить факты, приведшие к наблюдаемым результатам; можно «открутить» программу назад и посмотреть, какие причины породили данный исход. Можно также применить программу в одной ситуации, обучиться на полученном результате и использовать новую программу для получения более точных предсказаний в будущем. Вероятностное программирование помогает принимать любые решения в рамках вероятностных рассуждений.

Как все это работает? Вероятностное программирование стало применяться на практике, когда было осознано, что алгоритмы вывода, работающие для более простых языков представления, например байесовских сетей, можно обобщить и на программы. В части 3 описаны разнообразные алгоритмы вывода, благодаря которым это возможно. К счастью, в состав вероятностных систем программирования входит ряд встроенных алгоритмов вывода, которые применяются к программам автоматически. Вам нужно лишь предоставить знания о предметной области в виде вероятностной программы и описать факты, а система сама позаботится о выводе и обучении.

В этой книге мы будем изучать вероятностные рассуждения через призму вероятностного программирования. Прежде всего, мы узнаем, что такое вероятностная модель и как ее использовать для вывода заключений. Мы также научимся производить некоторые простые манипуляции для вывода таких заключений из модели, составленной из простых компонентов. Мы рассмотрим различные приемы

моделирования и покажем, как реализовать их с помощью вероятностного программирования. Мы разберемся в том, как работают вероятностные алгоритмы вывода, и это позволит эффективно проектировать и использовать модели. К концу книги вы сможете уверенно пользоваться вероятностным программированием для вывода полезных заключений и принятия обоснованных решений в условиях неопределенности.

1.2. Зачем нужно вероятностное программирование?

Вероятностные рассуждения – одна из фундаментальных технологий машинного обучения. Такие системы используются компаниями Google, Amazon и Microsoft для извлечения смысла из имеющихся данных. Вероятностные рассуждения применялись в таких разных приложениях, как прогнозирование цены акций, рекомендование фильмов, диагностика компьютеров и обнаружение вторжений в вычислительные системы. Во многих из этих приложений используются методы, описанные в этой книге.

В предыдущем разделе следует выделить два важнейших положения:

- вероятностные рассуждения можно использовать для предсказания будущего, объяснения прошлого и обучения на прошлом опыте для лучшего предсказания будущего;
- вероятностное программирование – это вероятностное рассуждение, в котором для представления используется полный по Тьюрингу язык программирования.

В сочетании эти два положения позволяют отчеканить следующую формулу.

Факт. Вероятностное рассуждение + полнота по Тьюрингу = вероятностное программирование.

Обоснованием вероятностного программирования является тот факт, что из двух концепций, которые сами по себе являются достаточно мощными, составляется новая. В результате получается более простой и гибкий способ применения компьютеров для принятия решений в условиях неопределенности.

1.2.1. Улучшенные вероятностные рассуждения

У большинства современных вероятностных языков представления имеются ограничения на сложность представляемых систем. В таких относительно простых языках, как байесовские сети, предполагается фиксированный набор переменных, поэтому они не обладают достаточной гибкостью для моделирования предметных областей, в которых состав переменных может изменяться. В последние годы были разработаны более гибкие языки. В некоторых (например, BUGS) даже имеются возможности, характерные для языков программирования, напри-

мер итерация и массивы, хотя полными по Тьюрингу они все же не являются. Успех языков, подобных BUGS, ясно говорит о необходимости более развитых и структурированных представлений. Но движение в сторону полноценных полных по Тьюрингу языков открывает целый мир новых возможностей для вероятностных рассуждений. Теперь стало возможно моделировать длительные процессы с большим числом событий и взаимодействующих сущностей.

Снова рассмотрим пример на футбольную тематику, но теперь представьте, что вы занимаетесь спортивной аналитикой и хотите порекомендовать решения по подбору игроков в команду. Можно было бы воспользоваться для этой цели накопленной статистикой, но статистика не улавливает контекст, в котором была собрана. Можно провести более тонкий контекстный анализ, построив детальную модель футбольного сезона. Для этого придется моделировать много взаимосвязанных событий и взаимодействие игроков и команд. Трудно представить себе, как построить такую модель без структур данных и управляющих конструкций, имеющихся в полноценном языке программирования.

Вернемся к вопросу о выводе продукта на рынок и взглянем на процесс принятия решений по развитию бизнеса во всей его полноте. Запуск продукта – не изолированное событие, ему предшествуют этапы анализа рынка, исследований и разработок, и исход каждого этапа не предрешен. Результаты запуска продукта зависят от всех предшествующих этапов, а также от анализа того, что еще имеется на рынке. Полный анализ должен также учитывать, как отреагируют на наш продукт конкуренты и какие новые продукты они смогут предложить. Это трудная задача, т. к. приходится выдвигать гипотезы о конкурирующих продуктах. Возможно даже, что существуют конкуренты, о которых вы еще не знаете. В этом примере продукты оказываются структурами данных, порождаемыми сложными процессами. Поэтому наличие полноценного языка программирования было бы крайне полезно для создания модели.

У вероятностного программирования есть одна приятная особенность: если вы хотите использовать более простой каркас вероятностных рассуждений – пожалуйста. Системы вероятностного программирования позволяют представить широкий спектр существующих каркасов, а также такие системы, которые ни один из существующих каркасов представить не в состоянии. В этой книге вы узнаете о многих каркасах, в которых используется вероятностное программирование. Поэтому, изучая вероятностное программирование, вы заодно освоите многие современные каркасы вероятностных рассуждений.

1.2.2. Улучшенные языки имитационного моделирования

Полные по Тьюрингу языки вероятностного моделирования уже существуют. Обычно они называются *языками имитационного моделирования* (simulation languages). Мы знаем, что можно построить имитационную модель такого сложного процесса, как футбольный сезон, пользуясь языками программирования. В этом контексте я буду использовать термин *язык имитационного моделирова-*

ния для описания языка, который позволяет представить выполнение сложного процесса с элементами случайности. Как и вероятностные программы, такие имитационные модели выполняются случайным образом и порождают различные результаты. Имитационное моделирование широко используется как вероятностное рассуждение и применяется в самых разных областях, например: военном планировании, проектировании компонентов, здравоохранении и прогнозировании спортивных результатов. На самом деле, широкое распространение изощренных имитационных моделей говорит о потребности в развитых языках вероятностного моделирования.

Но вероятностная программа – нечто гораздо большее, чем имитационная модель. Имитационное моделирование позволяет реализовать лишь один аспект вероятностного программирования: предсказание будущего. Его нельзя использовать для объяснения причин наблюдаемых результатов. И хотя модель можно улучшить, дополнив ее новой актуальной информацией, трудно включить неизвестную информацию, которую нужно вывести. Поэтому способность к обучению на прошлом опыте с целью улучшения будущих прогнозов и анализа оказывается ограниченной. Использовать имитационные модели в машинном обучении невозможно.

Вероятностную программу можно уподобить имитационной модели, которая допускает не только выполнение, но и анализ. При разработке вероятностного программирования пришло осознание того, что многие алгоритмы вывода, используемые в более простых системах моделирования, можно использовать и для имитационного моделирования. Поэтому открывается возможность создать вероятностную модель, написав имитационную модель и применив к ней алгоритмы вывода.

И последнее. Системы вероятностных рассуждений уже существуют, например программы Hugin, Netica и BayesiaLab предлагают реализацию байесовских сетей. Но более выразительные языки представления, применяемые в вероятностном программировании, настолько новые, что мы лишь начинаем открывать для себя таящуюся в них мощь. Положа руку на сердце, не могу сказать, что вероятностное программирование уже используется во многих прикладных системах. Но некоторые нетривиальные приложения все же существуют. Microsoft с помощью вероятностного программирования научилась определять истинный уровень умения игроков в онлайновые игры. Стюарт Рассел из Калифорнийского университета в Беркли написал программу, которая следит за соблюдением Договора о всеобъемлющем запрещении ядерных испытаний, определяя сейсмическую активность, которая могла бы указывать на ядерный взрыв. Джош Тененбаум из Массачусетского технологического института и Ноа Гудман из Стэнфордского университета создали вероятностные программы для моделирования когнитивных способностей человека, они обладают очень неплохими объяснительными возможностями. В компании Charles River Analytics мы использовали вероятностное программирование для вывода заключений о компонентах вредоносного ПО и определения их эволюции. Но я полагаю, что все это – только вершина айсберга. Системы вероятностного программирования достигли такого уровня, что могут использоваться

гораздо большим числом специалистов для принятия решений в своих предметных областях. Прочитав эту книгу, вы получите шанс присоединиться к новой технологии, пока не стало слишком поздно.

1.3. Введение в Figaro, язык вероятностного программирования

В этой книге мы будем работать с системой вероятностного программирования Figaro. (Я назвал ее в память персонажа оперы Моцарта «Женитьба Фигаро». Я люблю Моцарта и играл доктора Бартоло в Бостонской постановке этой оперы.) Главная цель книги – научить читателя принципам вероятностного программирования, так чтобы изученные методы можно было перенести на другие системы, некоторые из которых перечислены в приложении В. Но есть и другая цель – приобрести практический опыт создания вероятностных программ и предоставить инструменты, которыми можно воспользоваться немедленно. Поэтому многие примеры конкретизированы с помощью кода на Figaro.

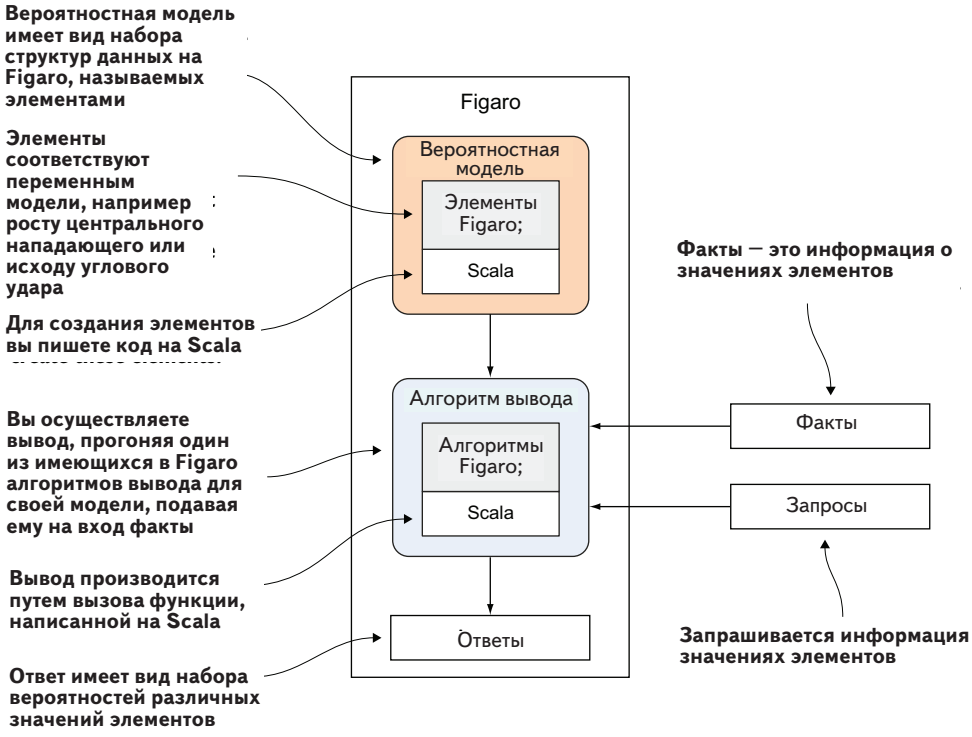

Рис. 1.9. Как в Figaro используется язык Scala для построения системы вероятностного программирования

Исходный код Figaro открыт, поддержка осуществляется с помощью GitHub, а сама система разрабатывается с 2009 года. Она реализована в виде библиотеки, написанной на Scala. На рис. 1.9 показано, как в Figaro используется Scala для реализации системы вероятностного программирования. Это уточнение рис. 1.7, где описаны основные компоненты такой системы. Начнем с вероятностной модели. В Figaro модель состоит из структур данных, называемых *элементами*. Каждый элемент представляет переменную, которая может принимать произвольное число значений. Структуры данных реализованы на Scala, и для создания модели, в которой они используются, вы пишете программу на Scala. Программе можно подать на вход факты, содержащие информацию о значениях элементов, а в запросе указать, какие выходные элементы вас интересуют. Вы выбираете один из встроенных в Figaro алгоритмов выбора и применяете его к своей модели, чтобы получить ответ на запрос при заданных фактах. Алгоритмы вывода реализованы на Scala, а применение алгоритм сводится к вызову функции Scala. Результатом вывода являются вероятности различных значений элементов, указанных в запросе.

Тот факт, что Figaro написана на Scala, дает ряд преимуществ. Некоторые из них связаны с использованием языка общего назначения вместо специализированного, другие – со спецификой именно Scala. Сначала остановимся на преимуществах вложения в язык общего назначения.

- Факты можно представлять, используя программу на объемлющем языке. Например, программа может прочитать данные из файла, каким-то образом обработать их и представить в виде фактов для модели Figaro. На специализированном независимом языке сделать это было бы гораздо труднее.
- Аналогично ответы, возвращенные Figaro, можно использовать в программе. Например, если менеджер футбольной команды работает с некоторой программой, то эта программа может запросить вероятность гола с тем, чтобы порекомендовать менеджеру варианты действий.
- Код на языке общего назначения можно включить в вероятностную программу. Рассмотрим, к примеру, физическую модель траектории мяча после удара головой. Эту модель можно было бы сделать частью элемента Figaro.
- Для построения модели Figaro можно использовать стандартные приемы программирования. Например, можно завести словарь, который содержит элементы Figaro, соответствующие всем игрокам команды, и выбирать из него элементы, исходя из ситуации, складывающейся на поле.

Теперь перечислим причины, по которым Scala особенно хорошо подходит на роль объемлющего языка для системы вероятностного программирования.

Поскольку Scala – функциональный язык программирования, Figaro также получает все преимущества функционального программирования, что позволяет естественно записывать многие модели. Я продемонстрирую этого во второй части книги.

Scala – объектно-ориентированный язык, и то, что он сочетает в себе функциональные и объектно-ориентированные черты, является дополнительным преимуществом. Figaro также является объектно-ориентированным. Во второй части

я покажу, что это полезно для выражения некоторых паттернов проектирования в вероятностном программировании.

Наконец, ряд преимуществ Figaro не связан с вложением в Scala.

- На Figaro можно представить чрезвычайно широкий спектр вероятностных моделей. Элементы Figaro могут принимать значения любого типа, в том числе: булевы, целые, с двойной точностью, массивы, деревья, графы и т. д. Связи между элементами можно определить с помощью произвольной функции.
- Figaro предоставляет развитые средства задания фактов с помощью условий и ограничений.
- В Figaro имеется обширный набор алгоритмов вывода.
- Figaro позволяет представлять динамические модели ситуаций, изменяющихся во времени, и рассуждать о них.
- Figaro дает возможность включать в модель явные решения и поддерживает вывод оптимальных решений.

Использование Scala

Поскольку система Figaro реализована в виде библиотеки на Scala, для работы с ней необходимы практические навыки работы со Scala. Эта книга посвящена вероятностному программированию, поэтому я не ставил себе задачу научить Scala. Для этой цели есть немало замечательных ресурсов, например Школа Scala в Твиттере (http://twitter.github.io/scala_school). Но на случай, если вы пока не уверены в своих познаниях, я буду объяснять используемые средства Scala по ходу дела. Вы сможете читать книгу, даже если совсем не знаете Scala.

Чтобы пользоваться всеми преимуществами вероятностного программирования и Figaro, не нужно знать Scala в совершенстве. В этой книге я избегал продвинутых и малопонятных возможностей Scala. С другой стороны, чем лучше вы знаете Scala, тем больших успехов достигнете в работе с Figaro. Возможно даже, что после прочтения этой книги вы станете лучше владеть Scala.

Есть несколько причин, по которым Figaro – удобный язык для изучения вероятностного программирования.

- Будучи реализован в виде библиотеки на Scala, Figaro может быть использован в программах, написанных на Java и Scala, что упрощает его интеграцию с приложениями.
- По той же причине Figaro позволяет использовать для построения моделей все богатство объемлющего языка. Scala – современный передовой язык программирования, обладающий множеством полезных средств для организации программ, и все это вы автоматически получаете в свое распоряжение, работая с Figaro.
- Figaro располагает полным набором алгоритмов.

В этой книге упор сделан на практические приемы и примеры. Всюду, где возможно, я объясняю общие принципы моделирования и описываю, как они реали-

зованы в Figaro. Это сослужит вам добрую службу, если вы решите обратиться к другой системе вероятностного программирования. Не во всех системах описанные здесь приемы реализуются с одинаковой легкостью. Так, в настоящее время еще мало объектно-ориентированных систем вероятностного программирования. Но обладая фундаментальными знаниями, вы найдете способ выразить свои потребности на выбранном языке.

1.3.1. Figaro и Java: построение простой системы вероятностного программирования

Для иллюстрации достоинств вероятностного программирования и языка Figaro я покажу два способа написания простого приложения: сначала на языке Java, с которым вы, возможно, знакомы, а потом на Figaro. Хотя у Scala есть некоторые преимущества по сравнению с Java, не в этом состоит различие, которое я хочу подчеркнуть. Основная идея в том, что *Figaro содержит такие средства для представления вероятностных моделей и выполнения вывода из них, которые недоступны без вероятностного программирования.*

Наше скромное приложение будет играть роль примера «Hello World» для Figaro. Представьте, что человек просыпается утром, смотрит, ясно ли на улице, и произносит приветствие, зависящее от погоды. Так происходит два дня подряд. Кроме того, погода во второй день зависит от первого: если в первый день было ясно, то вероятность, что и на завтра будет солнечно, повышается. Эти предложения на естественном языке можно записать количественно, как показано в табл. 1.1.

Таблица 1.1. Количественные вероятности в примере «Hello World»

Погода сегодня		
Ясно	0.2	
Пасмурно	0.8	
Приветствие сегодня		
Если сегодня на улице ясно	«Здравствуй, мир!»	0.6
	«Здравствуй, вселенная!»	0.4
Если сегодня на улице пасмурно	«Здравствуй, мир!»	0.2
	«О нет, только не это»	0.8
Погода завтра		
Если сегодня на улице ясно	Ясно	0.8
	Пасмурно	0.2
Если сегодня на улице пасмурно	Ясно	0.05
	Пасмурно	0.95

Приветствие завтра		
Если завтра на улице ясно	«Здравствуй, мир!»	0.6
	«Здравствуй, вселенная!»	0.4
Если завтра на улице пасмурно	«Здравствуй, мир!»	0.2
	«О нет, только не это»	0.8

В следующих главах будет точно объяснено, как интерпретировать эти числа. А пока достаточно интуитивного понимания того, что означает фраза «сегодня будет ясно с вероятностью 0.2». Аналогично, если завтра на улице будет ясно, то с вероятностью 0.6 будет произнесено приветствие «Здравствуй, мир!» и с вероятностью 0.4 – «Здравствуй, вселенная!».

Наметим три задачи, которые должна решать эта модель. В разделе 1.1.3 мы видели три типа рассуждений, доступных вероятностной модели: *предсказание* будущего, *вывод* прошлых событий, приведших к наблюдаемому результату, и обучение на прошлом опыте для лучшего предсказания будущего. Все это мы сделаем с помощью нашей простой модели. Точнее, задачи формулируются следующим образом.

1. Предсказать сегодняшнее приветствие.
2. Зная, что сегодня было произнесено приветствие «Здравствуй, мир!», сделать вывод о том, ясно ли на улице.
3. Зная, что сегодня было произнесено приветствие «Здравствуй, мир!», научиться предсказывать завтрашнее приветствие.

Посмотрим, как решить эти задачи на Java.

Листинг 1.1. Программа Hello World на Java

```
class HelloWorldJava {
   static String greeting1 = "Здравствуй, мир!";
   static String greeting2 = "Здравствуй, вселенная!";
   static String greeting3 = "О нет, только не это";

   static Double pSunnyToday = 0.2;
   static Double pNotSunnyToday = 0.8;
   static Double pSunnyTomorrowIfSunnyToday = 0.8;
   static Double pNotSunnyTomorrowIfSunnyToday = 0.2;
   static Double pSunnyTomorrowIfNotSunnyToday = 0.05;
   static Double pNotSunnyTomorrowIfNotSunnyToday = 0.95;
   static Double pGreeting1TodayIfSunnyToday = 0.6;
   static Double pGreeting2TodayIfSunnyToday = 0.4;
   static Double pGreeting1TodayIfNotSunnyToday = 0.2;
   static Double pGreeting3IfNotSunnyToday = 0.8;
   static Double pGreeting1TomorrowIfSunnyTomorrow = 0.5;
   static Double pGreeting2TomorrowIfSunnyTomorrow = 0.5;
   static Double pGreeting1TomorrowIfNotSunnyTomorrow = 0.1;
   static Double pGreeting3TomorrowIfNotSunnyTomorrow = 0.95;
```

```
  static void predict() {
    Double pGreeting1Today =
           pSunnyToday * pGreeting1TodayIfSunnyToday +
           pNotSunnyToday * pGreeting1TodayIfNotSunnyToday;
    System.out.println("Сегодня будет приветствие " +
       greeting1 + " с вероятностью " + pGreeting1Today + ".");
  }

  static void infer() {
    Double pSunnyTodayAndGreeting1Today =
           pSunnyToday * pGreeting1TodayIfSunnyToday;
    Double pNotSunnyTodayAndGreeting1Today =
           pNotSunnyToday * pGreeting1TodayIfNotSunnyToday;
    Double pSunnyTodayGivenGreeting1Today =
           pSunnyTodayAndGreeting1Today /
           (pSunnyTodayAndGreeting1Today +
            pNotSunnyTodayAndGreeting1Today);
    System.out.println("Если сегодня произнесено приветствие " +
       greeting1 + ", то сегодня ясно с вероятностью  " +
       pSunnyTodayGivenGreeting1Today + ".");
  }

  static void learnAndPredict() {
    Double pSunnyTodayAndGreeting1Today =
           pSunnyToday * pGreeting1TodayIfSunnyToday;
    Double pNotSunnyTodayAndGreeting1Today =
           pNotSunnyToday * pGreeting1TodayIfNotSunnyToday;
    Double pSunnyTodayGivenGreeting1Today =
           pSunnyTodayAndGreeting1Today /
           (pSunnyTodayAndGreeting1Today +
            pNotSunnyTodayAndGreeting1Today);
    Double pNotSunnyTodayGivenGreeting1Today =
           1 - pSunnyTodayGivenGreeting1Today;
    Double pSunnyTomorrowGivenGreeting1Today =
           pSunnyTodayGivenGreeting1Today *
              pSunnyTomorrowIfSunnyToday +
           pNotSunnyTodayGivenGreeting1Today *
              pSunnyTomorrowIfNotSunnyToday;
    Double pNotSunnyTomorrowGivenGreeting1Today =
           1 - pSunnyTomorrowGivenGreeting1Today;
    Double pGreeting1TomorrowGivenGreeting1Today =
           pSunnyTomorrowGivenGreeting1Today *
              pGreeting1TomorrowIfSunnyTomorrow +
           pNotSunnyTomorrowGivenGreeting1Today *
              pGreeting1TomorrowIfNotSunnyTomorrow;
    System.out.println("Если сегодня произнесено приветствие " +
       greeting1 + ", то завтра будет сказано " + greeting1 +
       " с вероятностью " +
       pGreeting1TomorrowGivenGreeting1Today);
  }

  public static void main(String[] args) {
    predict();
    infer();
    learnAndPredict();
  }
}
```

❸ ❹ ❺ ❻

❶ — Определяем приветствия
❷ — Задаем числовые параметры модели
❸ — Предсказываем сегодняшнее приветствие, применяя правила вероятностного вывода
❹ — Из того, что было произнесено приветствие «Здравствуй, мир!», выводим сегодняшнюю погоду, применяя правила вероятностного вывода
❺ — Из того, что было произнесено приветствие «Здравствуй, мир!», обучаемся предсказывать завтрашнее приветствие, применяя правила вероятностного вывода
❻ — Метод main, который выполняет все задачи

Не буду здесь описывать, как выполняются вычисления с применением правил вероятностного вывода. В коде используются три правила вывода: цепное правило, правило полной вероятности и правило Байеса. Все они будут объяснены в главе 9. А пока выделим две серьезные проблемы в этой программе.

- *Не существует способа определить структуру модели.*
  Определение модели содержится в списке имен переменных, принимающих значения типа double. Когда в начале этого раздела я описывал модель и приводил числа в табл. 1.1, структура модели была очевидна и более-менее понятна, пусть и на интуитивном уровне. У списка же переменных нет вообще никакой структуры. Назначение переменных закодировано только в их именах, что нельзя назвать хорошей идеей. Следовательно, написать модель таким образом трудно, и этот путь чреват ошибками. К тому же, написанный код трудно читать и сопровождать. Если потребуется модифицировать модель (например, сделать так, чтобы приветствие зависело еще и от того, хорошо ли вы спали), то, скорее всего, придется переписать большие куски кода.
- *Самостоятельно кодировать правила вывода трудно и чревато ошибками.*
  Вторая серьезная проблема связана с кодом, в котором для ответа на запросы используются правила вероятностного вывода. Чтобы написать такой код, необходимо хорошо знать правила вывода. Даже если такие знания имеются, написать код правильно все равно тяжело. И проверить, получен ли правильный ответ, тоже трудно. А ведь это очень простой пример. В сложном приложении написать код рассуждений подобным способом вообще нереально.

А теперь познакомимся с кодом, написанным на Scala/Figaro.

Листинг 1.2. Программа Hello World на Figaro

```
import com.cra.figaro.language.{Flip, Select}                    <-❶
import com.cra.figaro.library.compound.If
import com.cra.figaro.algorithm.factored.VariableElimination

object HelloWorld {
   val sunnyToday = Flip(0.2)                                    ❷->
   val greetingToday = If(sunnyToday,
        Select(0.6 -> "Здравствуй, мир!", 0.4 -> "Здравствуй, вселенная!"),
```

```
        Select(0.2 -> "Здравствуй, мир!", 0.8 -> "О нет, только не это"))
    val sunnyTomorrow = If(sunnyToday, Flip(0.8), Flip(0.05))
    val greetingTomorrow = If(sunnyTomorrow,
        Select(0.6 -> "Hello, world!", 0.4 -> "Howdy, universe!"),
        Select(0.2 -> "Hello, world!", 0.8 -> "Oh no, not again"))

    def predict() {
      val result = VariableElimination.probability(greetingToday,
                   "Здравствуй, мир!")
      println("Сегодня будет приветствие \"Здравствуй, мир!\" " +
              "с вероятностью " + result + ".")
    }

    def infer() {
      greetingToday.observe("Здравствуй, мир!")
      val result = VariableElimination.probability(sunnyToday, true)
      println("Если сегодня произнесено приветствие  \"Здравствуй, мир\", " +
              "то будет солнечно с вероятностью " + result + ".")
    }

    def learnAndPredict() {
      greetingToday.observe("Здравствуй, мир!")
      val result = VariableElimination.probability(greetingTomorrow,
                   "Hello, world!")
      println("Если сегодня произнесено приветствие  \"Здравствуй, мир!\", " +
              "то завтра будет сказано \"Здравствуй, мир!\" " +
              "с вероятностью " + result + ".")
    }

    def main(args: Array[String]) {
      predict()
      infer()
      learnAndPredict()
    }
}
```

❶ — Импортируем конструкции Figaro
❷ — Определяем модель
❸ — Предсказываем сегодняшнее приветствие, применяя алгоритм вывода
❹ — Применяем алгоритм вывода, чтобы вывести сегодняшнюю погоду из того, что сегодня было произнесено приветствие «Здравствуй, мир!»
❺ — Применяя алгоритм вывода, обучаемся предсказывать завтрашнее приветствие, зная, что сегодня было произнесено «Здравствуй, мир!»
❻ — Метод main, который выполняет все задачи

Я подробно объясню этот код в следующей главе. А пока хочу отметить, что он решает обе проблемы, встретившиеся в программе на Java. Во-первых, определение модели точно описывает ее структуру – в полном соответствии с табл. 1.1. Мы определяем четыре переменные: `sunnyToday`, `greetingToday`, `sunnyTomorrow` и `greetingTomorrow`, как описано в табл. 1.1. Вот, например, определение `greetingToday`:

```
    val greetingToday = If(sunnyToday,
        Select(0.6 -> "Здравствуй, мир!", 0.4 -> "Здравствуй, вселенная!"),
        Select(0.2 -> "Здравствуй, мир!", 0.8 -> "О нет, только не это"))
```

Здесь говорится, что если сегодня ясно, то с вероятностью 0.6 будет произнесено приветствие «Здравствуй, мир!» и с вероятностью 0.4 – «Здравствуй, вселенная!». Если же сегодня пасмурно, то с вероятностью 0.2 будет сказано «Здравствуй, мир!» и с вероятностью 0.8 – «О нет, только не это». Это в точности то же самое, что говорит табл. 1.1 о сегодняшнем приветствии. Поскольку код явно описывает модель, то писать, читать и сопровождать его гораздо проще. А если потребуется изменить модель (например, добавить переменную `sleepQuality`), то это можно будет сделать модульно.

Теперь рассмотрим код задач рассуждения. В нем нет никаких вычислений, а просто создается экземпляр объекта-алгоритма (в данном случае алгоритма исключения переменных, одного из встроенных в Figaro), и у него запрашивается искомая вероятность. В части 3 будет показано, что этот алгоритм основан на тех же правилах вывода, что и алгоритм из программы на Java. Всю сложную работу по организации и применению правил вывода берет на себя алгоритм. Даже если модель велика и сложна, можно прогнать алгоритм, и он отлично справится с выводом.

1.4. Резюме

- Для субъективных суждений нужны знания + логика.
- В системах вероятностных рассуждений вероятностная модель выражает знания, а алгоритм вывода инкапсулирует логику.
- Вероятностные рассуждения можно использовать для предсказания будущих событий, для вывода причин произошедших событий и обучения на прошлом опыте с целью улучшения предсказаний.
- Вероятностное программирование – это вероятностное рассуждение, в котором вероятностная модель выражена на языке программирования.
- В системе вероятностного программирования используется полный по Тьюрингу язык программирования для представления моделей и включены алгоритмы вывода для работы с этими моделями.
- Figaro – это система вероятностного программирования, реализованная на языке Scala, предлагающем функциональный и объектно-ориентированный стили программирования.

1.5. Упражнения

Решения избранных упражнений имеются на сайте www.manning.com/books/practical-probabilistic-programming.

1. Пусть требуется использовать систему вероятностных рассуждений для рассуждения об исходе сдачи карт в покере.
    a. Какие общие знания вы закодировали бы в этой модели?
    b. Опишите, как можно воспользоваться системой для предсказания будущего. Что в этом случае является фактами? А что – запросом?

c. Опишите, как бы вы использовали систему для вывода причин наблюдаемого результата. Что является фактами и что – запросом?
d. Опишите, как выведенные причины произошедшего события могут улучшить будущие предсказания.

2. В примере «Hello World» измените вероятность ясной погоды сегодня, как показано в таблице ниже. Как изменится результат работы программы? Почему вы так думаете?

Погода сегодня	
Ясно	0.9
Пасмурно	0.1

3. Добавьте в пример «Hello World» еще одно приветствие «Здравствуй, галактика!». Назначьте ему некоторую вероятность при ясной погоде, не забыв уменьшить вероятности других приветствий, так чтобы сумма вероятностей осталась равной 1. Кроме того, модифицируйте программу, так чтобы там, где раньше ответом на запрос было приветствие «Здравствуй, мир!», теперь печаталось «Здравствуй, галактика!». Попробуйте сделать это для обеих версий программы: на Java и на Figaro. Сравните трудоемкость.