PROBABILISTIC GRAPHICAL MODELS

PRINCIPLES AND TECHNIQUES

DAPHNE KOLLER AND NIR FRIEDMAN

Probabilistic Graphical Models

Adaptive Computation and Machine Learning

Thomas Dietterich, Editor

Christopher Bishop, David Heckerman, Michael Jordan, and Michael Kearns, Associate Editors

Bioinformatics: The Machine Learning Approach, Pierre Baldi and Søren Brunak

Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto

Graphical Models for Machine Learning and Digital Communication, Brendan J. Frey

Learning in Graphical Models, Michael I. Jordan

Causation, Prediction, and Search, 2nd ed., Peter Spirtes, Clark Glymour, and Richard Scheines

Principles of Data Mining, David Hand, Heikki Mannila, and Padhraic Smyth

Bioinformatics: The Machine Learning Approach, 2nd ed., Pierre Baldi and Søren Brunak

Learning Kernel Classifiers: Theory and Algorithms, Ralf Herbrich

Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, Bernhard Schölkopf and Alexander J. Smola

Introduction to Machine Learning, Ethem Alpaydin

Gaussian Processes for Machine Learning, Carl Edward Rasmussen and Christopher K. I. Williams

Semi-Supervised Learning, Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, eds.

The Minimum Description Length Principle, Peter D. Grünwald

Introduction to Statistical Relational Learning, Lise Getoor and Ben Taskar, eds.

Probabilistic Graphical Models: Principles and Techniques, Daphne Koller and Nir Friedman

Probabilistic Graphical Models

Principles and Techniques

Daphne Koller

Nir Friedman

The MIT Press
Cambridge, Massachusetts
London, England

This book was set by the authors in LaTeX2$_\epsilon$.
Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Koller, Daphne.
Probabilistic Graphical Models: Principles and Techniques / Daphne Koller and Nir Friedman.
p. cm. – (Adaptive computation and machine learning)
Includes bibliographical references and index.
ISBN 978-0-262-01319-2 (hardcover : alk. paper)
1. Graphical modeling (Statistics) 2. Bayesian statistical decision theory—Graphic methods. I. Koller, Daphne. II. Friedman, Nir.
QA279.5.K65 2010
519.5'420285–dc22
2009008615

10 9 8 7 6 5

To our families

my parents Dov and Ditza
my husband Dan
my daughters Natalie and Maya
D.K.

my parents Noga and Gad
my wife Yael
my children Roy and Lior
N.F.

As far as the laws of mathematics refer to reality, they are not certain, as far as they are certain, they do not refer to reality.

Albert Einstein, 1921

When we try to pick out anything by itself, we find that it is bound fast by a thousand invisible cords that cannot be broken, to everything in the universe.

John Muir, 1869

The actual science of logic is conversant at present only with things either certain, impossible, or entirely doubtful . . . Therefore the true logic for this world is the calculus of probabilities, which takes account of the magnitude of the probability which is, or ought to be, in a reasonable man's mind.

James Clerk Maxwell, 1850

The theory of probabilities is at bottom nothing but common sense reduced to calculus; it enables us to appreciate with exactness that which accurate minds feel with a sort of instinct for which ofttimes they are unable to account.

Pierre Simon Laplace, 1819

Misunderstanding of probability may be the greatest of all impediments to scientific literacy.

Stephen Jay Gould

Contents

Acknowledgments

This book owes a considerable debt of gratitude to the many people who contributed to its creation, and to those who have influenced our work and our thinking over the years.

First and foremost, we want to thank our students, who, by asking the right questions, and forcing us to formulate clear and precise answers, were directly responsible for the inception of this book and for any clarity of presentation.

We have been fortunate to share the same mentors, who have had a significant impact on our development as researchers and as teachers: Joe Halpern, Stuart Russell. Much of our core views on probabilistic models have been influenced by Judea Pearl. Judea through his persuasive writing and vivid presentations inspired us, and many other researchers of our generation, to plunge into research in this field.

There are many people whose conversations with us have helped us in thinking through some of the more difficult concepts in the book: Nando de Freitas, Gal Elidan, Dan Geiger, Amir Globerson, Uri Lerner, Chris Meek, David Sontag, Yair Weiss, and Ramin Zabih. Others, in conversations and collaborations over the year, have also influenced our thinking and the presentation of the material: Pieter Abbeel, Jeff Bilmes, Craig Boutilier, Moises Goldszmidt, Carlos Guestrin, David Heckerman, Eric Horvitz, Tommi Jaakkola, Michael Jordan, Kevin Murphy, Andrew Ng, Ben Taskar, and Sebastian Thrun.

We especially want to acknowledge Gal Elidan for constant encouragement, valuable feedback, and logistic support at many critical junctions, throughout the long years of writing this book.

Over the course of the years of work on this book, many people have contributed to it by providing insights, engaging in enlightening discussions, and giving valuable feedback. It is impossible to individually acknowledge all of the people who made such contributions. However, we specifically wish to express our gratitude to those people who read large parts of the book and gave detailed feedback: Rahul Biswas, James Cussens, James Diebel, Yoni Donner, Tal El-Hay, Gal Elidan, Stanislav Funiak, Amir Globerson, Russ Greiner, Carlos Guestrin, Tim Heilman, Geremy Heitz, Maureen Hillenmeyer, Ariel Jaimovich, Tommy Kaplan, Jonathan Laserson, Ken Levine, Brian Milch, Kevin Murphy, Ben Packer, Ronald Parr, Dana Pe'er, and Christian Shelton.

We are deeply grateful to the following people, who contributed specific text and/or figures, mostly to the case studies and concept boxes without which this book would be far less interesting: Gal Elidan, to chapter 11, chapter 18, and chapter 19; Stephen Gould, to chapter 4 and chapter 13; Vladimir Jojic, to chapter 12; Jonathan Laserson, to chapter 19; Uri Lerner, to chapter 14; Andrew McCallum and Charles Sutton, to chapter 4; Brian Milch, to chapter 6; Kevin

Murphy, to chapter 15; and Benjamin Packer, to many of the exercises used throughout the book. In addition, we are very grateful to Amir Globerson, David Sontag and Yair Weiss whose insights on chapter 13 played a key role in the development of the material in that chapter.

Special thanks are due to Bob Prior at MIT Press who convinced us to go ahead with this project and was constantly supportive, enthusiastic and patient in the face of the recurring delays and missed deadlines. We thank Greg McNamee, our copy editor, and Mary Reilly, our artist, for their help in improving this book considerably. We thank Chris Manning, for allowing us to use his LaTeX macros for typesetting this book, and for providing useful advice on how to use them. And we thank Miles Davis for invaluable technical support.

We also wish to thank the many colleagues who used drafts of this book in teaching provided enthusiastic feedback that encouraged us to continue this project at times where it seemed unending. Sebastian Thrun deserves a special note of thanks, for forcing us to set a deadline for completion of this book and to stick to it.

We also want to thank the past and present members of the DAGS group at Stanford, and the Computational Biology group at the Hebrew University, many of whom also contributed ideas, insights, and useful comments. We specifically want to thank them for bearing with us while we devoted far too much of our time to working on this book.

Finally, noone deserves our thanks more than our long-suffering families — Natalie Anna Koller Avida, Maya Rika Koller Avida, and Dan Avida; Lior, Roy, and Yael Friedman — for their continued love, support, and patience, as they watched us work evenings and weekends to complete this book. We could never have done this without you.

List of Figures

List of Algorithms

List of Boxes

1 *Introduction*

1.1 Motivation

Most tasks require a person or an automated system to *reason*: to take the available information and reach conclusions, both about what might be true in the world and about how to act. For example, a doctor needs to take information about a patient — his symptoms, test results, personal characteristics (gender, weight) — and reach conclusions about what diseases he may have and what course of treatment to undertake. A mobile robot needs to synthesize data from its sonars, cameras, and other sensors to conclude where in the environment it is and how to move so as to reach its goal without hitting anything. A speech-recognition system needs to take a noisy acoustic signal and infer the words spoken that gave rise to it.

In this book, we describe a general framework that can be used to allow a computer system to answer questions of this type. In principle, one could write a special-purpose computer program for every domain one encounters and every type of question that one may wish to answer. The resulting system, although possibly quite successful at its particular task, is often very brittle: If our application changes, significant changes may be required to the program. Moreover, this general approach is quite limiting, in that it is hard to extract lessons from one successful solution and apply it to one which is very different.

declarative representation

model

We focus on a different approach, based on the concept of a *declarative representation*. In this approach, we construct, within the computer, a *model* of the system about which we would like to reason. This model encodes our knowledge of how the system works in a computer-readable form. This representation can be manipulated by various algorithms that can answer questions based on the model. For example, a model for medical diagnosis might represent our knowledge about different diseases and how they relate to a variety of symptoms and test results. A reasoning algorithm can take this model, as well as observations relating to a particular patient, and answer questions relating to the patient's diagnosis. **The key property of a declarative representation is the separation of knowledge and reasoning. The representation has its own clear semantics, separate from the algorithms that one can apply to it. Thus, we can develop a general suite of algorithms that apply any model within a broad class, whether in the domain of medical diagnosis or speech recognition. Conversely, we can improve our model for a specific application domain without having to modify our reasoning algorithms constantly.**

Declarative representations, or model-based methods, are a fundamental component in many fields, and models come in many flavors. Our focus in this book is on models for complex sys-

uncertainty

tems that involve a significant amount of *uncertainty*. Uncertainty appears to be an inescapable aspect of most real-world applications. It is a consequence of several factors. We are often uncertain about the true state of the system because our observations about it are partial: only some aspects of the world are observed; for example, the patient's true disease is often not directly observable, and his future prognosis is never observed. Our observations are also noisy — even those aspects that are observed are often observed with some error. The true state of the world is rarely determined with certainty by our limited observations, as most relationships are simply not deterministic, at least relative to our ability to model them. For example, there are few (if any) diseases where we have a clear, universally true relationship between the disease and its symptoms, and even fewer such relationships between the disease and its prognosis. Indeed, while it is not clear whether the universe (quantum mechanics aside) is deterministic when modeled at a sufficiently fine level of granularity, it is quite clear that it is not deterministic relative to our current understanding of it. To summarize, uncertainty arises because of limitations in our ability to observe the world, limitations in our ability to model it, and possibly even because of innate nondeterminism.

Because of this ubiquitous and fundamental uncertainty about the true state of world, we need to allow our reasoning system to consider different possibilities. One approach is simply to consider any state of the world that is possible. Unfortunately, it is only rarely the case that we can completely eliminate a state as being impossible given our observations. In our medical diagnosis example, there is usually a huge number of diseases that are *possible* given a particular set of observations. Most of them, however, are highly unlikely. If we simply list all of the possibilities, our answers will often be vacuous of meaningful content (e.g., "the patient can have any of the following 573 diseases"). **Thus, to obtain meaningful conclusions, we need to reason not just about what is possible, but also about what is *probable*.**

probability theory

The calculus of *probability theory* (see section 2.1) provides us with a formal framework for considering multiple possible outcomes and their likelihood. It defines a set of mutually exclusive and exhaustive possibilities, and associates each of them with a *probability* — a number between 0 and 1, so that the total probability of all possibilities is 1. This framework allows us to consider options that are unlikely, yet not impossible, without reducing our conclusions to content-free lists of every possibility.

Furthermore, one finds that probabilistic models are very liberating. Where in a more rigid formalism we might find it necessary to enumerate every possibility, here we can often sweep a multitude of annoying exceptions and special cases under the "probabilistic rug," by introducing outcomes that roughly correspond to "something unusual happens." In fact, as we discussed, this type of approximation is often inevitable, as we can only rarely (if ever) provide a deterministic specification of the behavior of a complex system. Probabilistic models allow us to make this fact explicit, and therefore often provide a model which is more faithful to reality.

1.2 Structured Probabilistic Models

This book describes a general-purpose framework for constructing and using probabilistic models of complex systems. We begin by providing some intuition for the principles underlying this framework, and for the models it encompasses. This section requires some knowledge of

basic concepts in probability theory; a reader unfamiliar with these concepts might wish to read section 2.1 first.

Complex systems are characterized by the presence of multiple interrelated aspects, many of which relate to the reasoning task. For example, in our medical diagnosis application, there are multiple possible diseases that the patient might have, dozens or hundreds of symptoms and diagnostic tests, personal characteristics that often form predisposing factors for disease, and many more matters to consider. These domains can be characterized in terms of a set of *random variables*, where the value of each variable defines an important property of the world. For example, a particular disease, such as *Flu*, may be one variable in our domain, which takes on two values, for example, *present* or *absent*; a symptom, such as *Fever*, may be a variable in our domain, one that perhaps takes on continuous values. The set of possible variables and their values is an important design decision, and it depends strongly on the questions we may wish to answer about the domain.

random variable

Our task is to reason probabilistically about the values of one or more of the variables, possibly given observations about some others. In order to do so using principled probabilistic reasoning, we need to construct a *joint distribution* over the space of possible assignments to some set of random variables $\mathcal{X}$. This type of model allows us to answer a broad range of interesting queries. For example, we can make the observation that a variable X_i takes on the specific value x_i, and ask, in the resulting *posterior distribution*, what the probability distribution is over values of another variable X_j.

joint probability distribution

posterior distribution

Example 1.1

Consider a very simple medical diagnosis setting, where we focus on two diseases — flu and hayfever; these are not mutually exclusive, as a patient can have either, both, or none. Thus, we might have two binary-valued random variables, Flu and Hayfever. We also have a 4-valued random variable Season, which is correlated both with flu and hayfever. We may also have two symptoms, Congestion and Muscle Pain, each of which is also binary-valued. Overall, our probability space has $2 \times 2 \times 4 \times 2 \times 2 = 64$ values, corresponding to the possible assignments to these five variables. Given a joint distribution over this space, we can, for example, ask questions such as how likely the patient is to have the flu given that it is fall, and that she has sinus congestion but no muscle pain; as a probability expression, this query would be denoted

$$P(\textit{Flu} = \textit{true} \mid \textit{Season} = \textit{fall}, \textit{Congestion} = \textit{true}, \textit{Muscle Pain} = \textit{false}).$$

■

1.2.1 Probabilistic Graphical Models

Specifying a joint distribution over 64 possible values, as in example 1.1, already seems fairly daunting. When we consider the fact that a typical medical- diagnosis problem has dozens or even hundreds of relevant attributes, the problem appears completely intractable. This book describes the framework of probabilistic graphical models, which provides a mechanism for exploiting structure in complex distributions to describe them compactly, and in a way that allows them to be constructed and utilized effectively.

Probabilistic graphical models use a graph-based representation as the basis for compactly encoding a complex distribution over a high-dimensional space. In this graphical representation, illustrated in figure 1.1, the nodes (or ovals) correspond to the variables in our domain, and the edges correspond to direct probabilistic interactions between them. For example, figure 1.1a (top)

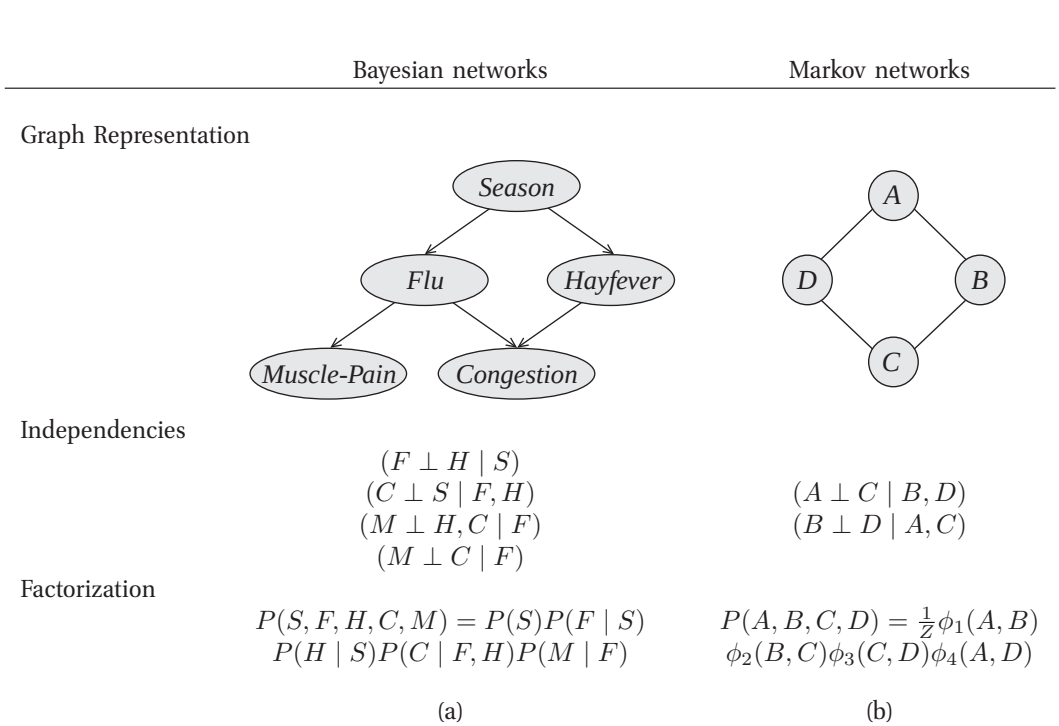


Figure 1.1 Different perspectives on probabilistic graphical models: top — the graphical representation; middle — the independencies induced by the graph structure; bottom — the factorization induced by the graph structure. (a) A sample Bayesian network. (b) A sample Markov network.

illustrates one possible graph structure for our flu example. In this graph, we see that there is no direct interaction between *Muscle Pain* and *Season*, but both interact directly with *Flu*.

There is a dual perspective that one can use to interpret the structure of this graph. From one perspective, the graph is a compact representation of a set of *independencies* that hold in the distribution; these properties take the form $\boldsymbol{X}$ is independent of $\boldsymbol{Y}$ given $\boldsymbol{Z}$, denoted $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$, for some subsets of variables $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$. For example, our "target" distribution P for the preceding example — the distribution encoding our beliefs about this particular situation — may satisfy the conditional independence $(\textit{Congestion} \perp \textit{Season} \mid \textit{Flu}, \textit{Hayfever})$. This statement asserts that

$$P(\textit{Congestion} \mid \textit{Flu}, \textit{Hayfever}, \textit{Season}) = P(\textit{Congestion} \mid \textit{Flu}, \textit{Hayfever});$$

that is, if we are interested in the distribution over the patient having congestion, and we know whether he has the flu and whether he has hayfever, the season is no longer informative. Note that this assertion does *not* imply that *Season* is independent of *Congestion*; only that all of the information we may obtain from the season on the chances of having congestion we already obtain by knowing whether the patient has the flu and has hayfever. Figure 1.1a (middle) shows the set of independence assumptions associated with the graph in figure 1.1a (top).

factor

The other perspective is that the graph defines a skeleton for compactly representing a high-dimensional distribution: Rather than encode the probability of every possible assignment to all of the variables in our domain, we can "break up" the distribution into smaller *factors*, each over a much smaller space of possibilities. We can then define the overall joint distribution as a product of these factors. For example, figure 1.1(a-bottom) shows the factorization of the distribution associated with the graph in figure 1.1 (top). It asserts, for example, that the probability of the event "spring, no flu, hayfever, sinus congestion, muscle pain" can be obtained by multiplying five numbers: $P(\textit{Season} = \textit{spring})$, $P(\textit{Flu} = \textit{false} \mid \textit{Season} = \textit{spring})$, $P(\textit{Hayfever} = \textit{true} \mid \textit{Season} = \textit{spring})$, $P(\textit{Congestion} = \textit{true} \mid \textit{Hayfever} = \textit{true}, \textit{Flu} = \textit{false})$, and $P(\textit{Muscle Pain} = \textit{true} \mid \textit{Flu} = \textit{false})$. This parameterization is significantly more compact, requiring only $3+4+4+4+2 = 17$ nonredundant parameters, as opposed to 63 nonredundant parameters for the original joint distribution (the 64th parameter is fully determined by the others, as the sum over all entries in the joint distribution must sum to 1). The graph structure defines the factorization of a distribution P associated with it — the set of factors and the variables that they encompass.

It turns out that these two perspectives — the graph as a representation of a set of independencies, and the graph as a skeleton for factorizing a distribution — are, in a deep sense, equivalent. The independence properties of the distribution are precisely what allow it to be represented compactly in a factorized form. Conversely, a particular factorization of the distribution guarantees that certain independencies hold.

Bayesian network

Markov network

We describe two families of graphical representations of distributions. One, called *Bayesian networks*, uses a directed graph (where the edges have a source and a target), as shown in figure 1.1a (top). The second, called *Markov networks*, uses an undirected graph, as illustrated in figure 1.1b (top). It too can be viewed as defining a set of independence assertions (figure 1.1b [middle] or as encoding a compact factorization of the distribution (figure 1.1b [bottom]). Both representations provide the duality of independencies and factorization, but they differ in the set of independencies they can encode and in the factorization of the distribution that they induce.

1.2.2 Representation, Inference, Learning

The graphical language exploits structure that appears present in many distributions that we want to encode in practice: the property that variables tend to interact *directly* only with very few others. Distributions that exhibit this type of structure can generally be encoded naturally and compactly using a graphical model.

This framework has many advantages. First, it often allows the distribution to be written down tractably, even in cases where the explicit representation of the joint distribution is astronomically large. Importantly, the type of representation provided by this framework is *transparent*, in that a human expert can understand and evaluate its semantics and properties. This property is important for constructing models that provide an accurate reflection of our understanding of a domain. Models that are opaque can easily give rise to unexplained, and even undesirable, answers.

inference

Second, as we show, the same structure often also allows the distribution to be used effectively for *inference* — answering queries using the distribution as our model of the world. In particular, we provide algorithms for computing the posterior probability of some variables given evidence

on others. For example, we might observe that it is spring and the patient has muscle pain, and we wish to know how likely he is to have the flu, a query that can formally be written as $P(Flu = true \mid Season = spring, Muscle\ Pain = true)$. These inference algorithms work directly on the graph structure and are generally orders of magnitude faster than manipulating the joint distribution explicitly.

Third, this framework facilitates the effective construction of these models, whether by a human expert or automatically, by *learning* from data a model that provides a good approximation to our past experience. For example, we may have a set of patient records from a doctor's office and wish to learn a probabilistic model encoding a distribution consistent with our aggregate experience. Probabilistic graphical models support a *data-driven approach* to model construction that is very effective in practice. In this approach, a human expert provides some rough guidelines on how to model a given domain. For example, the human usually specifies the attributes that the model should contain, often some of the main dependencies that it should encode, and perhaps other aspects. The details, however, are usually filled in automatically, by fitting the model to data. The models produced by this process are usually much better reflections of the domain than models that are purely hand-constructed. Moreover, they can sometimes reveal surprising connections between variables and provide novel insights about a domain.

data-driven approach

These three components — representation, inference, and learning — are critical components in constructing an intelligent system. We need a declarative representation that is a reasonable encoding of our world model. We need to be able to use this representation effectively to answer a broad range of questions that are of interest. And we need to be able to acquire this distribution, combining expert knowledge and accumulated data. Probabilistic graphical models are one of a small handful of frameworks that support all three capabilities for a broad range of problems.

1.3 Overview and Roadmap

1.3.1 Overview of Chapters

The framework of probabilistic graphical models is quite broad, and it encompasses both a variety of different types of models and a range of methods relating to them. This book describes several types of models. For each one, we describe the three fundamental cornerstones: representation, inference, and learning.

We begin in part I, by describing the most basic type of graphical models, which are the focus of most of the book. These models encode distributions over a fixed set $\mathcal{X}$ of random variables. We describe how graphs can be used to encode distributions over such spaces, and what the properties of such distributions are.

Specifically, in chapter 3, we describe the Bayesian network representation, based on directed graphs. We describe how a Bayesian network can encode a probability distribution. We also analyze the independence properties induced by the graph structure.

In chapter 4, we move to Markov networks, the other main category of probabilistic graphical models. Here also we describe the independencies defined by the graph and the induced factorization of the distribution. We also discuss the relationship between Markov networks and Bayesian networks, and briefly describe a framework that unifies both.

In chapter 5, we delve a little deeper into the representation of the parameters in probabilistic

models, focusing mostly on Bayesian networks, whose parameterization is more constrained. We describe representations that capture some of the finer-grained structure of the distribution, and show that, here also, capturing structure can provide significant gains.

In chapter 6, we turn to formalisms that extend the basic framework of probabilistic graphical models to settings where the set of variables is no longer rigidly circumscribed in advance. One such setting is a *temporal* one, where we wish to model a system whose state evolves over time, requiring us to consider distributions over entire trajectories, We describe a compact representation — a *dynamic Bayesian network* — that allows us to represent structured systems that evolve over time. We then describe a family of extensions that introduce various forms of higher level structure into the framework of probabilistic graphical models. Specifically, we focus on domains containing *objects* (whether concrete or abstract), characterized by attributes, and related to each other in various ways. Such domains can include repeated structure, since different objects of the same type share the same probabilistic model. These languages provide a significant extension to the expressive power of the standard graphical models.

In chapter 7, we take a deeper look at models that include continuous variables. Specifically, we explore the properties of the multivariate Gaussian distribution and the representation of such distributions as both directed and undirected graphical models. Although the class of Gaussian distributions is a limited one and not suitable for all applications, it turns out to play a critical role even when dealing with distributions that are not Gaussian.

In chapter 8, we take a deeper, more technical look at probabilistic models, defining a general framework called the *exponential family*, that encompasses a broad range of distributions. This chapter provides some basic concepts and tools that will turn out to play an important role in later development.

We then turn, in part II, to a discussion of the inference task. In chapter 9, we describe the basic ideas underlying exact inference in probabilistic graphical models. We first analyze the fundamental difficulty of the exact inference task, separately from any particular inference algorithm we might develop. We then present two basic algorithms for exact inference — variable elimination and conditioning — both of which are equally applicable to both directed and undirected models. Both of these algorithms can be viewed as operating over the graph structure defined by the probabilistic model. They build on basic concepts, such as graph properties and dynamic programming algorithms, to provide efficient solutions to the inference task. We also provide an analysis of their computational cost in terms of the graph structure, and we discuss where exact inference is feasible.

In chapter 10, we describe an alternative view of exact inference, leading to a somewhat different algorithm. The benefit of this alternative algorithm is twofold. First, it uses dynamic programming to avoid repeated computations in settings where we wish to answer more than a single query using the same network. Second, it defines a natural algorithm that uses message passing on a graph structure; this algorithm forms the basis for approximate inference algorithms developed in later chapters.

Because exact inference is computationally intractable for many models of interest, we then proceed to describe approximate inference algorithms, which trade off accuracy with computational cost. We present two main classes of such algorithms. In chapter 11, we describe a class of methods that can be viewed from two very different perspectives: On one hand, they are direct generalizations of the graph-based message-passing approach developed for the case of exact inference in chapter 10. On the other hand, they can be viewed as solving an optimization

problem: one where we approximate the distribution of interest using a simpler representation that allows for feasible inference. The equivalence of these views provides important insights and suggests a broad family of algorithms that one can apply to approximate inference.

In chapter 12, we describe a very different class of methods: *particle-based methods*, which approximate a complex joint distribution by considering samples from it (also known as particles). We describe several methods from this general family. These methods are generally based on core techniques from statistics, such as importance sampling and Markov-chain Monte Carlo methods. Once again, the connection to this general class of methods suggests multiple opportunities for new algorithms.

While the representation of probabilistic graphical models applies, to a great extent, to models including both discrete and continuous-valued random variables, inference in models involving continuous variables is significantly more challenging than the purely discrete case. In chapter 14, we consider the task of inference in continuous and *hybrid* (continuous/discrete) networks, and we discuss whether and how the exact and approximate inference methods developed in earlier chapters can be applied in this setting.

The representation that we discussed in chapter 6 allows a compact encoding of networks whose size can be unboundedly large. Such networks pose particular challenges to inference algorithms. In this chapter, we discuss some special-purpose methods that have been developed for the particular settings of networks that model dynamical systems.

We then turn, in part III, to the third of our main topics — learning probabilistic models from data. We begin in chapter 16 by reviewing some of the fundamental concepts underlying the general task of learning models from data. We then present the spectrum of learning problems that we address in this part of the book. These problems vary along two main axes: the extent to which we are given prior knowledge specifying the model, and whether the data from which we learn contain complete observations of all of the relevant variables. In contrast to the inference task, where the same algorithms apply equally to Bayesian networks and Markov networks, the learning task is quite different for these two classes of models. We begin with studying the learning task for Bayesian networks.

In chapter 17, we focus on the most basic learning task: learning parameters for a Bayesian network with a given structure, from fully observable data. Although this setting may appear somewhat restrictive, it turns out to form the basis for our entire development of Bayesian network learning. As we show, the factorization of the distribution, which was central both to representation and to inference, also plays a key role in making inference feasible.

We then move, in chapter 18, to the harder problem of learning both Bayesian network structure and the parameters, still from fully observed data. The learning algorithms we present trade off the accuracy with which the learned network represents the empirical distribution for the complexity of the resulting structure. As we show, the type of independence assumptions underlying the Bayesian network representation often hold, at least approximately, in real-world distributions. Thus, these learning algorithms often result in reasonably compact structures that capture much of the signal in the distribution.

In chapter 19, we address the Bayesian network learning task in a setting where we have access only to partial observations of the relevant variables (for example, when the available patient records have missing entries). This type of situation occurs often in real-world settings. Unfortunately, the resulting learning task is considerably harder, and the resulting algorithms are both more complex and less satisfactory in terms of their performance.

We conclude the discussion of learning in chapter 20 by considering the problem of learning Markov networks from data. It turns out that the learning tasks for Markov networks are significantly harder than the corresponding problem for Bayesian networks. We explain the difficulties and discuss the existing solutions.

Finally, in part IV, we turn to a different type of extension, where we consider the use of this framework for other forms of reasoning. Specifically, we consider cases where we can act, or intervene, in the world.

In chapter 21, we focus on the semantics of intervention and its relation to causality. We present the notion of a *causal model*, which allows us to answer not only queries of the form "if I observe X, what do I learn about Y," but also *intervention queries*, of the form "if I manipulate X, what effect does it have on Y."

We then turn to the task of *decision making* under uncertainty. Here, we must consider not only the distribution over different states of the world, but also the preferences of the agent regarding these outcomes. In chapter 22, we discuss the notion of *utility functions* and how they can encode an agent's preferences about complex situations involving multiple variables. As we show, the same ideas that we used to provide compact representations of probability distribution can also be used for utility functions.

In chapter 23, we describe a unified representation for decision making, called *influence diagrams*. Influence diagrams extend Bayesian networks by introducing actions and utilities. We present algorithms that use influence diagrams for making decisions that optimize the agent's expected utility. These algorithms utilize many of the same ideas that formed the basis for exact inference in Bayesian networks.

We conclude with a high-level synthesis of the techniques covered in this book, and with some guidance on how to use them in tackling a new problem.

1.3.2 Reader's Guide

As we mentioned, the topics described in this book relate to multiple fields, and techniques from other disciplines — probability theory, computer science, information theory, optimization, statistics, and more — are used in various places throughout it. While it is impossible to present all of the relevant material within the scope of this book, we have attempted to make the book somewhat self-contained by providing a very brief review of the key concepts from these related disciplines in chapter 2.

Some of this material, specifically the review of probability theory and of graph-related concepts, is very basic yet central to most of the development in this book. Readers who are less familiar with these topics may wish to read these sections carefully, and even knowledgeable readers may wish to briefly review them to gain familiarity with the notations used. Other background material, covering such topics as information theory, optimization, and algorithmic concepts, can be found in the appendix.

The chapters in the book are structured as follows. The main text in each chapter provides the detailed technical development of the key ideas. Beyond the main text, most chapters contain boxes that contain interesting material that augments these ideas. These boxes come in three types: *Skill boxes* describe "hands-on" tricks and techniques, which, while often heuristic in nature, are important for getting the basic algorithms described in the text to work in practice. *Case study boxes* describe empirical case studies relating to the techniques described in the text.

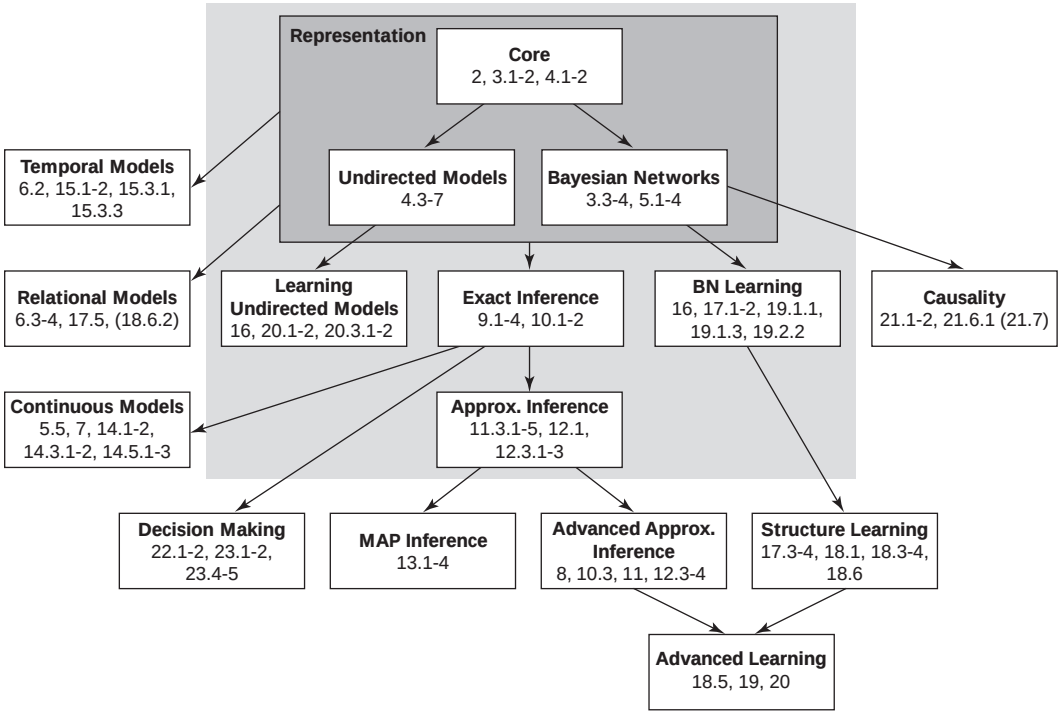


Figure 1.2 A reader's guide to the structure and dependencies in this book

These case studies include both empirical results on how the algorithms perform in practice and descriptions of applications of these algorithms to interesting domains, illustrating some of the issues encountered in practice. Finally, *concept boxes* present particular instantiations of the material described in the text, which have had significant impact in their own right.

This textbook is clearly too long to be used in its entirety in a one-semester class. Figure 1.2 tries to delineate some coherent subsets of the book that can be used for teaching and other purposes. The small, labeled boxes represent "units" of material on particular topics. Arrows between the boxes represent dependencies between these units. The first enclosing box (solid line) represents material that is fundamental to everything else, and that should be read by anyone using this book. One can then use the dependencies between the boxes to expand or reduce the depth of the coverage on any given topic. The material in the larger box (dashed line) forms a good basis for a one-semester (or even one-quarter) overview class. Some of the sections in the book are marked with an asterisk, denoting the fact that they contain more technically advanced material. In most cases, these sections are self-contained, and they can be skipped without harming the reader's ability to understand the rest of the text.

We have attempted in this book to present a synthesis of ideas, most of which have been developed over many years by multiple researchers. To avoid futile attempts to divide up the credit precisely, we have omitted all bibliographical references from the technical presentation

in the chapters. Rather, each chapter ends with a section called "Relevant Literature," which describes the historical evolution of the material in the chapter, acknowledges the papers and books that developed the key concepts, and provides some additional readings on material relevant to the chapter. We encourage the reader who is interested in a topic to follow up on some of these additional readings, since there are many interesting developments that we could not cover in this book.

Finally, each chapter includes a set of exercises that explore in additional depth some of the material described in the text and present some extensions to it. The exercises are annotated with an asterisk for exercises that are somewhat more difficult, and with two asterisks for ones that are truly challenging.

Additional material related to this book, including slides and figures, solutions to some of the exercises, and errata, can be found online at `http://pgm.stanford.edu`.

1.3.3 Connection to Other Disciplines

The ideas we describe in this book are connected to many fields. From probability theory, we inherit the basic concept of a probability distribution, as well as many of the operations we can use to manipulate it. From computer science, we exploit the key idea of using a graph as a data structure, as well as a variety of algorithms for manipulating graphs and other data structures. These algorithmic ideas and the ability to manipulate probability distributions using discrete data structures are some of the key elements that make the probabilistic manipulations tractable. Decision theory extends these basic ideas to the task of decision making under uncertainty and provides the formal foundation for this task.

From computer science, and specifically from artificial intelligence, these models inherit the idea of using a declarative representation of the world to separate procedural reasoning from our domain knowledge. This idea is of key importance to the generality of this framework and its applicability to such a broad range of tasks.

Various ideas from other disciplines also arise in this field. Statistics plays an important role both in certain aspects of the representation and in some of the work on learning models from data. Optimization plays a role in providing algorithms both for approximate inference and for learning models from data. Bayesian networks first arose, albeit in a restricted way, in the setting of modeling genetic inheritance in human family trees; in fact, restricted version of some of the exact inference algorithms we discuss were first developed in this context. Similarly, undirected graphical models first arose in physics as a model for systems of electrons, and some of the basic concepts that underlie recent work on approximate inference developed from that setting.

Information theory plays a dual role in its interaction with this field. Information-theoretic concepts such as entropy and information arise naturally in various settings in this framework, such as evaluating the quality of a learned model. Thus, tools from this discipline are a key component in our analytic toolkit. On the other side, the recent successes in coding theory, based on the relationship between inference in probabilistic models and the task of decoding messages sent over a noisy channel, have led to a resurgence of work on approximate inference in graphical models. The resulting developments have revolutionized both the development of error-correcting codes and the theory and practice of approximate message-passing algorithms in graphical models.

1.3.3.1 What Have We Gained?

Although the framework we describe here shares common elements with a broad range of other topics, it has a coherent common core: the use of structure to allow a compact representation, effective reasoning, and feasible learning of general-purpose, factored, probabilistic models. These elements provide us with a general infrastructure for reasoning and learning about complex domains.

As we discussed earlier, by using a declarative representation, we essentially separate out the description of the model for the particular application, and the general-purpose algorithms used for inference and learning. Thus, this framework provides a general algorithmic toolkit that can be applied to many different domains.

Indeed, probabilistic graphical models have made a significant impact on a broad spectrum of real-world applications. For example, these models have been used for medical and fault diagnosis, for modeling human genetic inheritance of disease, for segmenting and denoising images, for decoding messages sent over a noisy channel, for revealing genetic regulatory processes, for robot localization and mapping, and more. Throughout this book, we will describe how probabilistic graphical models were used to address these applications and what issues arise in the application of these models in practice.

In addition to practical applications, these models provide a formal framework for a variety of fundamental problems. For example, the notion of conditional independence and its explicit graph-based representation provide a clear formal semantics for irrelevance of information. This framework also provides a general methodology for handling data fusion — we can introduce *sensor variables* that are noisy versions of the true measured quantity, and use Bayesian conditioning to combine the different measurements. The use of a probabilistic model allows us to provide a formal measure for model quality, in terms of a numerical fit of the model to observed data; this measure underlies much of our work on learning models from data. The temporal models we define provide a formal framework for defining a general trend toward persistence of state over time, in a way that does not raise inconsistencies when change does occur.

In general, part of the rich development in this field is due to the close and continuous interaction between theory and practice. In this field, unlike many others, the distance between theory and practice is quite small, and there is a constant flow of ideas and problems between them. Problems or ideas arise in practical applications and are analyzed and subsequently developed in more theoretical papers. Algorithms for which no theoretical analysis exists are tried out in practice, and the profile of where they succeed and fail often provides the basis for subsequent analysis. This rich synergy leads to a continuous and vibrant development, and it is a key factor in the success of this area.

1.4 Historical Notes

The foundations of probability theory go back to the sixteenth century, when Gerolamo Cardano began a formal analysis of games of chance, followed by additional key developments by Pierre de Fermat and Blaise Pascal in the seventeenth century. The initial development involved only discrete probability spaces, and the analysis methods were purely combinatorial. The foundations of modern probability theory, with its measure-theoretic underpinnings, were laid by Andrey Kolmogorov in the 1930s.

Particularly central to the topics of this book is the so-called *Bayes theorem*, shown in the eighteenth century by the Reverend Thomas Bayes (Bayes 1763). This theorem allows us to use a model that tells us the conditional probability of event a given event b (say, a symptom given a disease) in order to compute the contrapositive: the conditional probability of event b given event a (the disease given the symptom). This type of reasoning is central to the use of graphical models, and it explains the choice of the name *Bayesian network*.

The notion of representing the interactions between variables in a multidimensional distribution using a graph structure originates in several communities, with very different motivations. In the area of statistical physics, this idea can be traced back to Gibbs (1902), who used an undirected graph to represent the distribution over a system of interacting particles. In the area of genetics, this idea dates back to the work on path analysis of Sewal Wright (Wright 1921, 1934). Wright proposed the use of a directed graph to study inheritance in natural species. This idea, although largely rejected by statisticians at the time, was subsequently adopted by economists and social scientists (Wold 1954; Blalock, Jr. 1971). In the field of statistics, the idea of analyzing interactions between variables was first proposed by Bartlett (1935), in the study of *contingency tables*, also known as *log-linear models*. This idea became more accepted by the statistics community in the 1960s and 70s (Vorobev 1962; Goodman 1970; Haberman 1974).

In the field of computer science, probabilistic methods lie primarily in the realm of Artificial Intelligence (AI). The AI community first encountered these methods in the endeavor of building
expert systems *expert systems*, computerized systems designed to perform difficult tasks, such as oil-well location or medical diagnosis, at an expert level. Researchers in this field quickly realized the need for methods that allow the integration of multiple pieces of evidence, and that provide support for making decisions under uncertainty. Some early systems (de Bombal et al. 1972; Gorry and Barnett 1968; Warner et al. 1961) used probabilistic methods, based on the very restricted *naive Bayes model*. This model restricts itself to a small set of possible hypotheses (e.g., diseases) and assumes that the different evidence variables (e.g., symptoms or test results) are independent given each hypothesis. These systems were surprisingly successful, performing (within their area of expertise) at a level comparable to or better than that of experts. For example, the system of de Bombal et al. (1972) averaged over 90 percent correct diagnoses of acute abdominal pain, whereas expert physicians were averaging around 65 percent.

Despite these successes, this approach fell into disfavor in the AI community, owing to a combination of several factors. One was the belief, prevalent at the time, that artificial intelligence should be based on similar methods to human intelligence, combined with a strong impression that people do not manipulate numbers when reasoning. A second issue was the belief that the strong independence assumptions made in the existing expert systems were fundamental to the approach. Thus, the lack of a flexible, scalable mechanism to represent interactions between variables in a distribution was a key factor in the rejection of the probabilistic framework.

The rejection of probabilistic methods was accompanied by the invention of a range of alternative formalisms for reasoning under uncertainty, and the construction of expert systems based on these formalisms (notably Prospector by Duda, Gaschnig, and Hart 1979 and Mycin by Buchanan and Shortliffe 1984). Most of these formalisms used the production rule framework, where each rule is augmented with some number(s) defining a measure of "confidence" in its validity. These frameworks largely lacked formal semantics, and many exhibited significant problems in key reasoning patterns. Other frameworks for handling uncertainty proposed at the time included fuzzy logic, possibility theory, and Dempster-Shafer belief functions. For a

discussion of some of these alternative frameworks see Shafer and Pearl (1990); Horvitz et al. (1988); Halpern (2003).

The widespread acceptance of probabilistic methods began in the late 1980s, driven forward by two major factors. The first was a series of seminal theoretical developments. The most influential among these was the development of the Bayesian network framework by Judea Pearl and his colleagues in a series of paper that culminated in Pearl's highly influential textbook *Probabilistic Reasoning in Intelligent Systems* (Pearl 1988). In parallel, the key paper by S.L. Lauritzen and D.J. Spiegelhalter 1988 set forth the foundations for efficient reasoning using probabilistic graphical models. The second major factor was the construction of large-scale, highly successful expert systems based on this framework that avoided the unrealistically strong assumptions made by early probabilistic expert systems. The most visible of these applications was the Pathfinder expert system, constructed by Heckerman and colleagues (Heckerman et al. 1992; Heckerman and Nathwani 1992b), which used a Bayesian network for diagnosis of pathology samples.

At this time, although work on other approaches to uncertain reasoning continues, probabilistic methods in general, and probabilistic graphical models in particular, have gained almost universal acceptance in a wide range of communities. They are in common use in fields as diverse as medical diagnosis, fault diagnosis, analysis of genetic and genomic data, communication and coding, analysis of marketing data, speech recognition, natural language understanding, and many more. Several other books cover aspects of this growing area; examples include Pearl (1988); Lauritzen (1996); Jensen (1996); Castillo et al. (1997a); Jordan (1998); Cowell et al. (1999); Neapolitan (2003); Korb and Nicholson (2003). The Artificial Intelligence textbook of Russell and Norvig (2003) places this field within the broader endeavor of constructing an intelligent agent.

2 *Foundations*

In this chapter, we review some important background material regarding key concepts from probability theory, information theory, and graph theory. This material is included in a separate introductory chapter, since it forms the basis for much of the development in the remainder of the book. Other background material — such as discrete and continuous optimization, algorithmic complexity analysis, and basic algorithmic concepts — is more localized to particular topics in the book. Many of these concepts are presented in the appendix; others are presented in concept boxes in the appropriate places in the text. All of this material is intended to focus only on the minimal subset of ideas required to understand most of the discussion in the remainder of the book, rather than to provide a comprehensive overview of the field it surveys. We encourage the reader to explore additional sources for more details about these areas.

2.1 Probability Theory

The main focus of this book is on complex probability distributions. In this section we briefly review basic concepts from probability theory.

2.1.1 Probability Distributions

When we use the word "probability" in day-to-day life, we refer to a degree of confidence that an event of an uncertain nature will occur. For example, the weather report might say "there is a low probability of light rain in the afternoon." Probability theory deals with the formal foundations for discussing such estimates and the rules they should obey.

Before we discuss the representation of probability, we need to define what the events are to which we want to assign a probability. These events might be different outcomes of throwing a die, the outcome of a horse race, the weather configurations in California, or the possible failures of a piece of machinery.

2.1.1.1 Event Spaces

event
outcome space

Formally, we define *events* by assuming that there is an agreed upon *space* of possible outcomes, which we denote by Ω. For example, if we consider dice, we might set $\Omega = \{1, 2, 3, 4, 5, 6\}$. In the case of a horse race, the space might be all possible orders of arrivals at the finish line, a much larger space.

measurable event

In addition, we assume that there is a set of *measurable events* $\mathcal{S}$ to which we are willing to assign probabilities. Formally, each event $\alpha \in \mathcal{S}$ is a subset of Ω. In our die example, the event $\{6\}$ represents the case where the die shows 6, and the event $\{1, 3, 5\}$ represents the case of an odd outcome. In the horse-race example, we might consider the event "Lucky Strike wins," which contains all the outcomes in which the horse Lucky Strike is first.

Probability theory requires that the event space satisfy three basic properties:

- It contains the *empty event* $\emptyset$, and the *trivial event* Ω.
- It is closed under union. That is, if $\alpha, \beta \in \mathcal{S}$, then so is $\alpha \cup \beta$.
- It is closed under complementation. That is, if $\alpha \in \mathcal{S}$, then so is $\Omega - \alpha$.

The requirement that the event space is closed under union and complementation implies that it is also closed under other Boolean operations, such as intersection and set difference.

2.1.1.2 Probability Distributions

Definition 2.1 probability distribution

A probability distribution P *over* $(\Omega, \mathcal{S})$ *is a mapping from events in* $\mathcal{S}$ *to real values that satisfies the following conditions:*

- $P(\alpha) \geq 0$ *for all* $\alpha \in \mathcal{S}$.
- $P(\Omega) = 1$.
- *If* $\alpha, \beta \in \mathcal{S}$ *and* $\alpha \cap \beta = \emptyset$, *then* $P(\alpha \cup \beta) = P(\alpha) + P(\beta)$. ■

The first condition states that probabilities are not negative. The second states that the "trivial event," which allows all possible outcomes, has the maximal possible probability of 1. The third condition states that the probability that one of two mutually disjoint events will occur is the sum of the probabilities of each event. These two conditions imply many other conditions. Of particular interest are $P(\emptyset) = 0$, and $P(\alpha \cup \beta) = P(\alpha) + P(\beta) - P(\alpha \cap \beta)$.

2.1.1.3 Interpretations of Probability

Before we continue to discuss probability distributions, we need to consider the interpretations that we might assign to them. Intuitively, the probability $P(\alpha)$ of an event α quantifies the degree of confidence that α will occur. If $P(\alpha) = 1$, we are certain that one of the outcomes in α occurs, and if $P(\alpha) = 0$, we consider all of them impossible. Other probability values represent options that lie between these two extremes.

This description, however, does not provide an answer to what the numbers mean. There are two common interpretations for probabilities.

frequentist interpretation

The *frequentist* interpretation views probabilities as frequencies of events. More precisely, the probability of an event is the fraction of times the event occurs if we repeat the experiment indefinitely. For example, suppose we consider the outcome of a particular die roll. In this case, the statement $P(\alpha) = 0.3$, for $\alpha = \{1, 3, 5\}$, states that if we repeatedly roll this die and record the outcome, then the fraction of times the outcomes in α will occur is 0.3. More precisely, the limit of the sequence of fractions of outcomes in α in the first roll, the first two rolls, the first three rolls, ..., the first n rolls, ... is 0.3.

The frequentist interpretation gives probabilities a tangible semantics. When we discuss concrete physical systems (for example, dice, coin flips, and card games) we can envision how these frequencies are defined. It is also relatively straightforward to check that frequencies must satisfy the requirements of proper distributions.

The frequentist interpretation fails, however, when we consider events such as "It will rain tomorrow afternoon." Although the time span of "Tomorrow afternoon" is somewhat ill defined, we expect it to occur exactly once. It is not clear how we define the frequencies of such events. Several attempts have been made to define the probability for such an event by finding a *reference class* of similar events for which frequencies are well defined; however, none of them has proved entirely satisfactory. Thus, the frequentist approach does not provide a satisfactory interpretation for a statement such as "the probability of rain tomorrow afternoon is 0.3."

reference class

An alternative interpretation views probabilities as *subjective* degrees of belief. Under this interpretation, the statement $P(\alpha) = 0.3$ represents a subjective statement about one's own degree of belief that the event α will come about. Thus, the statement "the probability of rain tomorrow afternoon is 50 percent" tells us that in the opinion of the speaker, the chances of rain and no rain tomorrow afternoon are the same. Although tomorrow afternoon will occur only once, we can still have uncertainty about its outcome, and represent it using numbers (that is, probabilities).

subjective interpretation

This description still does not resolve what exactly it means to hold a particular degree of belief. What stops a person from stating that the probability that Bush will win the election is 0.6 and the probability that he will lose is 0.8? The source of the problem is that we need to explain how subjective degrees of beliefs (something that is internal to each one of us) are reflected in our actions.

This issue is a major concern in subjective probabilities. One possible way of attributing degrees of beliefs is by a betting game. Suppose you believe that $P(\alpha) = 0.8$. Then you would be willing to place a bet of \$1 against \$3. To see this, note that with probability 0.8 you gain a dollar, and with probability 0.2 you lose \$3, so on average this bet is a good deal with expected gain of 20 cents. In fact, you might be even tempted to place a bet of \$1 against \$4. Under this bet the average gain is 0, so you should not mind. However, you would not consider it worthwhile to place a bet \$1 against \$4 and 10 cents, since that would have negative expected gain. Thus, by finding which bets you are willing to place, we can assess your degrees of beliefs.

The key point of this mental game is the following. If you hold degrees of belief that do not satisfy the rule of probability, then by a clever construction we can find a series of bets that would result in a sure negative outcome for you. Thus, the argument goes, a rational person must hold degrees of belief that satisfy the rules of probability.[1]

In the remainder of the book we discuss probabilities, but we usually do not explicitly state their interpretation. Since both interpretations lead to the same mathematical rules, the technical definitions hold for both interpretations.

1. As stated, this argument assumes as that people's preferences are directly proportional to their expected earnings. For small amounts of money, this assumption is quite reasonable. We return to this topic in chapter 22.

2.1.2 Basic Concepts in Probability

2.1.2.1 Conditional Probability

To use a concrete example, suppose we consider a distribution over a population of students taking a certain course. The space of outcomes is simply the set of all students in the population. Now, suppose that we want to reason about the students' intelligence and their final grade. We can define the event α to denote "all students with grade A," and the event β to denote "all students with high intelligence." Using our distribution, we can consider the probability of these events, as well as the probability of $\alpha \cap \beta$ (the set of intelligent students who got grade A). This, however, does not directly tell us how to update our beliefs given new evidence. Suppose we learn that a student has received the grade A; what does that tell us about her intelligence?

This kind of question arises every time we want to use distributions to reason about the real world. More precisely, after learning that an event α is true, how do we change our probability about β occurring? The answer is via the notion of *conditional probability*. Formally, the conditional probability of β given α is defined as

conditional probability

$$P(\beta \mid \alpha) = \frac{P(\alpha \cap \beta)}{P(\alpha)} \tag{2.1}$$

That is, the probability that β is true given that we know α is the relative proportion of outcomes satisfying β among these that satisfy α. (Note that the conditional probability is not defined when $P(\alpha) = 0$.)

The conditional probability given an event (say α) satisfies the properties of definition 2.1 (see exercise 2.4), and thus it is a probability distribution by its own right. Hence, we can think of the conditioning operation as taking one distribution and returning another over the same probability space.

2.1.2.2 Chain Rule and Bayes Rule

From the definition of the conditional distribution, we immediately see that

$$P(\alpha \cap \beta) = P(\alpha)P(\beta \mid \alpha). \tag{2.2}$$

chain rule

This equality is known as the *chain rule* of conditional probabilities. More generally, if $\alpha_1, \ldots, \alpha_k$ are events, then we can write

$$P(\alpha_1 \cap \ldots \cap \alpha_k) = P(\alpha_1)P(\alpha_2 \mid \alpha_1) \cdots P(\alpha_k \mid \alpha_1 \cap \ldots \cap \alpha_{k-1}). \tag{2.3}$$

In other words, we can express the probability of a combination of several events in terms of the probability of the first, the probability of the second given the first, and so on. It is important to notice that we can expand this expression using any order of events — the result will remain the same.

Bayes' rule

Another immediate consequence of the definition of conditional probability is *Bayes' rule*

$$P(\alpha \mid \beta) = \frac{P(\beta \mid \alpha)P(\alpha)}{P(\beta)}. \tag{2.4}$$

A more general conditional version of Bayes' rule, where all our probabilities are conditioned on some background event γ, also holds:

$$P(\alpha \mid \beta \cap \gamma) = \frac{P(\beta \mid \alpha \cap \gamma)P(\alpha \mid \gamma)}{P(\beta \mid \gamma)}.$$

Bayes' rule is important in that it allows us to compute the conditional probability $P(\alpha \mid \beta)$ from the "inverse" conditional probability $P(\beta \mid \alpha)$.

Example 2.1

Consider the student population, and let Smart denote smart students and GradeA denote students who got grade A. Assume we believe (perhaps based on estimates from past statistics) that $P(\textit{GradeA} \mid \textit{Smart}) = 0.6$, *and now we learn that a particular student received grade A. Can we estimate the probability that the student is smart? According to Bayes' rule, this depends on our* prior *probability for students being smart (before we learn anything about them) and the prior probability of students receiving high grades. For example, suppose that* $P(\textit{Smart}) = 0.3$ *and* $P(\textit{GradeA}) = 0.2$, *then we have that* $P(\textit{Smart} \mid \textit{GradeA}) = 0.6 * 0.3/0.2 = 0.9$. *That is, an A grade strongly suggests that the student is smart. On the other hand, if the test was easier and high grades were more common, say,* $P(\textit{GradeA}) = 0.4$ *then we would get that* $P(\textit{Smart} \mid \textit{GradeA}) = 0.6 * 0.3/0.4 = 0.45$, *which is much less conclusive about the student.* ■

prior

Another classic example that shows the importance of this reasoning is in disease screening. To see this, consider the following hypothetical example (none of the mentioned figures are related to real statistics).

Example 2.2

Suppose that a tuberculosis (TB) skin test is 95 percent accurate. That is, if the patient is TB-infected, then the test will be positive with probability 0.95, *and if the patient is not infected, then the test will be negative with probability* 0.95. *Now suppose that a person gets a positive test result. What is the probability that he is infected? Naive reasoning suggests that if the test result is wrong 5 percent of the time, then the probability that the subject is infected is* 0.95. *That is, 95 percent of subjects with positive results have TB.*

If we consider the problem by applying Bayes' rule, we see that we need to consider the prior probability of TB infection, and the probability of getting positive test result. Suppose that 1 in 1000 of the subjects who get tested is infected. That is, $P(\textit{TB}) = 0.001$. *What is the probability of getting a positive test result? From our description, we see that* $0.001 \cdot 0.95$ *infected subjects get a positive result, and* $0.999 \cdot 0.05$ *uninfected subjects get a positive result. Thus,* $P(\textit{Positive}) = 0.0509$. *Applying Bayes' rule, we get that* $P(\textit{TB} \mid \textit{Positive}) = 0.001 \cdot 0.95/0.0509 \approx 0.0187$. *Thus, although a subject with a positive test is much more probable to be TB-infected than is a random subject, fewer than 2 percent of these subjects are TB-infected.* ■

2.1.3 Random Variables and Joint Distributions

2.1.3.1 Motivation

Our discussion of probability distributions deals with events. Formally, we can consider any event from the set of measurable events. The description of events is in terms of sets of outcomes. In many cases, however, it would be more natural to consider *attributes* of the outcome. For example, if we consider a patient, we might consider attributes such as "age,"

"gender," and "smoking history" that are relevant for assigning probability over possible diseases and symptoms. We would like then consider events such as "age > 55, heavy smoking history, and suffers from repeated cough."

To use a concrete example, consider again a distribution over a population of students in a course. Suppose that we want to reason about the intelligence of students, their final grades, and so forth. We can use an event such as *GradeA* to denote the subset of students that received the grade A and use it in our formulation. However, this discussion becomes rather cumbersome if we also want to consider students with grade B, students with grade C, and so on. Instead, we would like to consider a way of directly referring to a student's grade in a clean, mathematical way.

random variable

The formal machinery for discussing attributes and their values in different outcomes are *random variables*. A random variable is a way of reporting an attribute of the outcome. For example, suppose we have a random variable *Grade* that reports the final grade of a student, then the statement $P(\mathit{Grade} = A)$ is another notation for $P(\mathit{GradeA})$.

2.1.3.2 What Is a Random Variable?

Formally, a random variable, such as *Grade*, is defined by a function that associates with each outcome in Ω a value. For example, *Grade* is defined by a function f_{Grade} that maps each person in Ω to his or her grade (say, one of A, B, or C). The event $\mathit{Grade} = A$ is a shorthand for the event $\{\omega \in \Omega : f_{\mathit{Grade}}(\omega) = A\}$. In our example, we might also have a random variable *Intelligence* that (for simplicity) takes as values either "high" or "low." In this case, the event "*Intelligence* = *high*" refers, as can be expected, to the set of smart (high intelligence) students.

Random variables can take different sets of values. We can think of *categorical* (or *discrete*) random variables that take one of a few values, as in our two preceding examples. We can also talk about random variables that can take infinitely many values (for example, integer or real values), such as *Height* that denotes a student's height. We use $\mathit{Val}(X)$ to denote the set of values that a random variable X can take.

In most of the discussion in this book we examine either categorical random variables or random variables that take real values. We will usually use uppercase roman letters X, Y, Z to denote random variables. In discussing generic random variables, we often use a lowercase letter to refer to a value of a random variable. Thus, we use x to refer to a generic value of X. For example, in statements such as "$P(X = x) \geq 0$ for all $x \in \mathit{Val}(X)$." When we discuss categorical random variables, we use the notation $x^1, \ldots, x^k$, for $k = |\mathit{Val}(X)|$ (the number of elements in $\mathit{Val}(X)$), when we need to enumerate the specific values of X, for example, in statements such as

$$\sum_{i=1}^{k} P(X = x^i) = 1.$$

multinomial distribution

The distribution over such a variable is called a *multinomial*. In the case of a binary-valued random variable X, where $\mathit{Val}(X) = \{\mathit{false}, \mathit{true}\}$, we often use x^1 to denote the value *true* for X, and x^0 to denote the value *false*. The distribution of such a random variable is called a *Bernoulli distribution*.

Bernoulli distribution

We also use boldface type to denote sets of random variables. Thus, $\boldsymbol{X}$, $\boldsymbol{Y}$, or $\boldsymbol{Z}$ are typically used to denote a set of random variables, while $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ denote assignments of values to the

variables in these sets. We extend the definition of $Val(\boldsymbol{X})$ to refer to sets of variables in the obvious way. Thus, $\boldsymbol{x}$ is always a member of $Val(\boldsymbol{X})$. For $\boldsymbol{Y} \subseteq \boldsymbol{X}$, we use $\boldsymbol{x}\langle\boldsymbol{Y}\rangle$ to refer to the assignment within $\boldsymbol{x}$ to the variables in $\boldsymbol{Y}$. For two assignments $\boldsymbol{x}$ (to $\boldsymbol{X}$) and $\boldsymbol{y}$ (to $\boldsymbol{Y}$), we say that $\boldsymbol{x} \sim \boldsymbol{y}$ if they agree on the variables in their intersection, that is, $\boldsymbol{x}\langle\boldsymbol{X} \cap \boldsymbol{Y}\rangle = \boldsymbol{y}\langle\boldsymbol{X} \cap \boldsymbol{Y}\rangle$.

In many cases, the notation $P(X = x)$ is redundant, since the fact that x is a value of X is already reported by our choice of letter. Thus, in many texts on probability, the identity of a random variable is not explicitly mentioned, but can be inferred through the notation used for its value. Thus, we use $P(x)$ as a shorthand for $P(X = x)$ when the identity of the random variable is clear from the context. Another shorthand notation is that $\sum_x$ refers to a sum over all possible values that X can take. Thus, the preceding statement will often appear as $\sum_x P(x) = 1$. Finally, another standard notation has to do with conjunction. Rather than write $P((X = x) \cap (Y = y))$, we write $P(X = x, Y = y)$, or just $P(x, y)$.

2.1.3.3 Marginal and Joint Distributions

marginal distribution

Once we define a random variable X, we can consider the distribution over events that can be described using X. This distribution is often referred to as the *marginal distribution* over the random variable X. We denote this distribution by $P(X)$.

Returning to our population example, consider the random variable *Intelligence*. The marginal distribution over *Intelligence* assigns probability to specific events such as $P(\textit{Intelligence} = \textit{high})$ and $P(\textit{Intelligence} = \textit{low})$, as well as to the trivial event $P(\textit{Intelligence} \in \{\textit{high}, \textit{low}\})$. Note that these probabilities are defined by the probability distribution over the original space. For concreteness, suppose that $P(\textit{Intelligence} = \textit{high}) = 0.3$, $P(\textit{Intelligence} = \textit{low}) = 0.7$.

If we consider the random variable *Grade*, we can also define a marginal distribution. This is a distribution over all events that can be described in terms of the *Grade* variable. In our example, we have that $P(\textit{Grade} = A) = 0.25$, $P(\textit{Grade} = B) = 0.37$, and $P(\textit{Grade} = C) = 0.38$.

It should be fairly obvious that the marginal distribution is a probability distribution satisfying the properties of definition 2.1. In fact, the only change is that we restrict our attention to the subsets of $\mathcal{S}$ that can be described with the random variable X.

joint distribution

In many situations, we are interested in questions that involve the values of several random variables. For example, we might be interested in the event "*Intelligence* = *high* and *Grade* = *A*." To discuss such events, we need to consider the *joint distribution* over these two random variables. In general, the joint distribution over a set $\mathcal{X} = \{X_1, \ldots, X_n\}$ of random variables is denoted by $P(X_1, \ldots, X_n)$ and is the distribution that assigns probabilities to events that are specified in terms of these random variables. We use ξ to refer to a full assignment to the variables in $\mathcal{X}$, that is, $\xi \in Val(\mathcal{X})$.

The joint distribution of two random variables has to be consistent with the marginal distribution, in that $P(x) = \sum_y P(x, y)$. This relationship is shown in figure 2.1, where we compute the marginal distribution over *Grade* by summing the probabilities along each row. Similarly, we find the marginal distribution over *Intelligence* by summing out along each column. The resulting sums are typically written in the row or column margins, whence the term "marginal distribution."

Suppose we have a joint distribution over the variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. The most fine-grained events we can discuss using these variables are ones of the form "$X_1 = x_1$ and $X_2 = x_2$, ..., and $X_n = x_n$" for a choice of values $x_1, \ldots, x_n$ for all the variables. Moreover,

		Intelligence		
		low	*high*	
Grade	*A*	0.07	0.18	0.25
	B	0.28	0.09	0.37
	C	0.35	0.03	0.38
		0.7	0.3	1

Figure 2.1 Example of a joint distribution $P(\textbf{\textit{Intelligence}}, \textbf{\textit{Grade}})$: Values of *Intelligence* (columns) and *Grade* (rows) with the associated marginal distribution on each variable.

any two such events must be either identical or disjoint, since they both assign values to all the variables in $\mathcal{X}$. In addition, any event defined using variables in $\mathcal{X}$ must be a union of a set of such events. Thus, we are effectively working in a *canonical outcome space*: a space where each outcome corresponds to a joint assignment to $X_1, \ldots, X_n$. More precisely, all our probability computations remain the same whether we consider the original outcome space (for example, all students), or the canonical space (for example, all combinations of intelligence and grade). We use ξ to denote these *atomic outcomes*: those assigning a value to each variable in $\mathcal{X}$. For example, if we let $\mathcal{X} = \{\textit{Intelligence}, \textit{Grade}\}$, there are six atomic outcomes, shown in figure 2.1. The figure also shows one possible joint distribution over these six outcomes.

canonical outcome space

atomic outcome

Based on this discussion, from now on we will not explicitly specify the set of outcomes and measurable events, and instead implicitly assume the canonical outcome space.

2.1.3.4 Conditional Probability

The notion of conditional probability extends to induced distributions over random variables. For example, we use the notation $P(\textit{Intelligence} \mid \textit{Grade} = a)$ to denote the *conditional distribution* over the events describable by *Intelligence* given the knowledge that the student's grade is A. Note that the conditional distribution over a random variable given an observation of the value of another one is not the same as the marginal distribution. In our example, $P(\textit{Intelligence} = \textit{high}) = 0.3$, and $P(\textit{Intelligence} = \textit{high} \mid \textit{Grade} = a) = 0.18/0.25 = 0.72$. Thus, clearly $P(\textit{Intelligence} \mid \textit{Grade} = a)$ is different from the marginal distribution $P(\textit{Intelligence})$. The latter distribution represents our *prior* knowledge about students before learning anything else about a particular student, while the conditional distribution represents our more informed distribution after learning her grade.

conditional distribution

We will often use the notation $P(X \mid Y)$ to represent a set of conditional probability distributions. Intuitively, for each value of Y, this object assigns a probability over values of X using the conditional probability. This notation allows us to write the shorthand version of the chain rule: $P(X, Y) = P(X)P(Y \mid X)$, which can be extended to multiple variables as

$$P(X_1, \ldots, X_k) = P(X_1)P(X_2 \mid X_1) \cdots P(X_k \mid X_1, \ldots, X_{k-1}). \tag{2.5}$$

Similarly, we can state Bayes' rule in terms of conditional probability distributions:

$$P(X \mid Y) = \frac{P(X)P(Y \mid X)}{P(Y)}. \tag{2.6}$$

2.1.4 Independence and Conditional Independence

2.1.4.1 Independence

As we mentioned, we usually expect $P(\alpha \mid \beta)$ to be different from $P(\alpha)$. That is, learning that β is true changes our probability over α. However, in some situations equality can occur, so that $P(\alpha \mid \beta) = P(\alpha)$. That is, learning that β occurs did not change our probability of α.

Definition 2.2 independent events

We say that an event α is independent *of event β in P, denoted $P \models (\alpha \perp \beta)$, if $P(\alpha \mid \beta) = P(\alpha)$ or if $P(\beta) = 0$.* ■

We can also provide an alternative definition for the concept of independence:

Proposition 2.1

A distribution P satisfies $(\alpha \perp \beta)$ if and only if $P(\alpha \cap \beta) = P(\alpha)P(\beta)$.

PROOF Consider first the case where $P(\beta) = 0$; here, we also have $P(\alpha \cap \beta) = 0$, and so the equivalence immediately holds. When $P(\beta) \neq 0$, we can use the chain rule; we write $P(\alpha \cap \beta) = P(\alpha \mid \beta)P(\beta)$. Since α is independent of β, we have that $P(\alpha \mid \beta) = P(\alpha)$. Thus, $P(\alpha \cap \beta) = P(\alpha)P(\beta)$. Conversely, suppose that $P(\alpha \cap \beta) = P(\alpha)P(\beta)$. Then, by definition, we have that

$$P(\alpha \mid \beta) = \frac{P(\alpha \cap \beta)}{P(\beta)} = \frac{P(\alpha)P(\beta)}{P(\beta)} = P(\alpha).$$

■

As an immediate consequence of this alternative definition, we see that independence is a symmetric notion. That is, $(\alpha \perp \beta)$ implies $(\beta \perp \alpha)$.

Example 2.3

For example, suppose that we toss two coins, and let α be the event "the first toss results in a head" and β the event "the second toss results in a head." It is not hard to convince ourselves that we expect that these two events to be independent. Learning that β is true would not change our probability of α. In this case, we see two different physical processes (that is, coin tosses) leading to the events, which makes it intuitive that the probabilities of the two are independent. In certain cases, the same process can lead to independent events. For example, consider the event α denoting "the die outcome is even" and the event β denoting "the die outcome is 1 or 2." It is easy to check that if the die is fair (each of the six possible outcomes has probability $\frac{1}{6}$), then these two events are independent. ■

2.1.4.2 Conditional Independence

☞ **While independence is a useful property, it is not often that we encounter two independent events. A more common situation is when two events are independent given an additional event.** For example, suppose we want to reason about the chance that our student is accepted to graduate studies at Stanford or MIT. Denote by *Stanford* the event "admitted to Stanford" and by *MIT* the event "admitted to MIT." In most reasonable distributions, these two events are not independent. If we learn that a student was admitted to Stanford, then our estimate of her probability of being accepted at MIT is now higher, since it is a sign that she is a promising student.

Now, suppose that both universities base their decisions only on the student's grade point average (GPA), and we know that our student has a GPA of A. In this case, we might argue that learning that the student was admitted to Stanford should not change the probability that she will be admitted to MIT: Her GPA already tells us the information relevant to her chances of admission to MIT, and finding out about her admission to Stanford does not change that. Formally, the statement is

$$P(\mathit{MIT} \mid \mathit{Stanford}, \mathit{GradeA}) = P(\mathit{MIT} \mid \mathit{GradeA}).$$

In this case, we say that *MIT* is *conditionally independent* of *Stanford given GradeA*.

Definition 2.3 conditional independence

We say that an event α is conditionally independent *of event β given event γ in P, denoted $P \models (\alpha \perp \beta \mid \gamma)$, if $P(\alpha \mid \beta \cap \gamma) = P(\alpha \mid \gamma)$ or if $P(\beta \cap \gamma) = 0$.* ■

It is easy to extend the arguments we have seen in the case of (unconditional) independencies to give an alternative definition.

Proposition 2.2

P satisfies $(\alpha \perp \beta \mid \gamma)$ if and only if $P(\alpha \cap \beta \mid \gamma) = P(\alpha \mid \gamma)P(\beta \mid \gamma)$.

2.1.4.3 Independence of Random Variables

Until now, we have focused on independence between events. Thus, we can say that two events, such as one toss landing heads and a second also landing heads, are independent. However, we would like to say that any pair of outcomes of the coin tosses is independent. To capture such statements, we can examine the generalization of independence to sets of random variables.

Definition 2.4 conditional independence; observed variable; marginal independence

Let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ be sets of random variables. We say that $\boldsymbol{X}$ is conditionally independent *of $\boldsymbol{Y}$ given $\boldsymbol{Z}$ in a distribution P if P satisfies $(\boldsymbol{X} = \boldsymbol{x} \perp \boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{Z} = \boldsymbol{z})$ for all values $\boldsymbol{x} \in Val(\boldsymbol{X})$, $\boldsymbol{y} \in Val(\boldsymbol{Y})$, and $\boldsymbol{z} \in Val(\boldsymbol{Z})$. The variables in the set $\boldsymbol{Z}$ are often said to be* observed. *If the set $\boldsymbol{Z}$ is empty, then instead of writing $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \emptyset)$, we write $(\boldsymbol{X} \perp \boldsymbol{Y})$ and say that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are* marginally independent. ■

Thus, an independence statement over random variables is a universal quantification over all possible values of the random variables.

The alternative characterization of conditional independence follows immediately:

Proposition 2.3

The distribution P satisfies $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ if and only if $P(\boldsymbol{X}, \boldsymbol{Y} \mid \boldsymbol{Z}) = P(\boldsymbol{X} \mid \boldsymbol{Z})P(\boldsymbol{Y} \mid \boldsymbol{Z})$.

Suppose we learn about a conditional independence. Can we conclude other independence properties that must hold in the distribution? We have already seen one such example:

symmetry

- **Symmetry**:

$$(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \Longrightarrow (\boldsymbol{Y} \perp \boldsymbol{X} \mid \boldsymbol{Z}). \tag{2.7}$$

There are several other properties that hold for conditional independence, and that often provide a very clean method for proving important properties about distributions. Some key properties are:

decomposition

- **Decomposition**:

$$(\boldsymbol{X} \perp \boldsymbol{Y}, \boldsymbol{W} \mid \boldsymbol{Z}) \Longrightarrow (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}). \tag{2.8}$$

weak union

- **Weak union**:

$$(\boldsymbol{X} \perp \boldsymbol{Y}, \boldsymbol{W} \mid \boldsymbol{Z}) \Longrightarrow (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{W}). \tag{2.9}$$

contraction

- **Contraction**:

$$(\boldsymbol{X} \perp \boldsymbol{W} \mid \boldsymbol{Z}, \boldsymbol{Y}) \,\&\, (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \Longrightarrow (\boldsymbol{X} \perp \boldsymbol{Y}, \boldsymbol{W} \mid \boldsymbol{Z}). \tag{2.10}$$

An additional important property does not hold in general, but it does hold in an important subclass of distributions.

Definition 2.5 positive distribution

A distribution P is said to be positive *if for all events $\alpha \in \mathcal{S}$ such that $\alpha \neq \emptyset$, we have that $P(\alpha) > 0$.* ■

For positive distributions, we also have the following property:

intersection

- **Intersection**: For positive distributions, and for mutually disjoint sets $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{W}$:

$$(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{W}) \,\&\, (\boldsymbol{X} \perp \boldsymbol{W} \mid \boldsymbol{Z}, \boldsymbol{Y}) \Longrightarrow (\boldsymbol{X} \perp \boldsymbol{Y}, \boldsymbol{W} \mid \boldsymbol{Z}). \tag{2.11}$$

The proof of these properties is not difficult. For example, to prove Decomposition, assume that $(X \perp Y, W \mid Z)$ holds. Then, from the definition of conditional independence, we have that $P(X, Y, W \mid Z) = P(X \mid Z)P(Y, W \mid Z)$. Now, using basic rules of probability and arithmetic, we can show

$$\begin{aligned} P(X, Y \mid Z) &= \sum_{w} P(X, Y, w \mid Z) \\ &= \sum_{w} P(X \mid Z)P(Y, w \mid Z) \\ &= P(X \mid Z) \sum_{w} P(Y, w \mid Z) \\ &= P(X \mid Z)P(Y \mid Z). \end{aligned}$$

The only property we used here is called "reasoning by cases" (see exercise 2.6). We conclude that $(X \perp Y \mid Z)$.

2.1.5 Querying a Distribution

Our focus throughout this book is on using a joint probability distribution over multiple random variables to answer queries of interest.

2.1.5.1 Probability Queries

probability query

Perhaps the most common query type is the *probability query*. Such a query consists of two parts:

evidence

- The *evidence*: a subset $\boldsymbol{E}$ of random variables in the model, and an instantiation $\boldsymbol{e}$ to these variables;

query variables

- the *query variables*: a subset $\boldsymbol{Y}$ of random variables in the network.

Our task is to compute

$$P(\boldsymbol{Y} \mid \boldsymbol{E} = \boldsymbol{e}),$$

posterior distribution

that is, the *posterior probability distribution* over the values $\boldsymbol{y}$ of $\boldsymbol{Y}$, conditioned on the fact that $\boldsymbol{E} = \boldsymbol{e}$. This expression can also be viewed as the marginal over $\boldsymbol{Y}$, in the distribution we obtain by conditioning on $\boldsymbol{e}$.

2.1.5.2 MAP Queries

A second important type of task is that of finding a high-probability joint assignment to some subset of variables. The simplest variant of this type of task is the *MAP* query (also called *most probable explanation (MPE)*), whose aim is to find the *MAP assignment* — the most likely assignment to all of the (non-evidence) variables. More precisely, if we let $\boldsymbol{W} = \mathcal{X} - \boldsymbol{E}$, our task is to find the most likely assignment to the variables in $\boldsymbol{W}$ given the evidence $\boldsymbol{E} = \boldsymbol{e}$:

MAP assignment

$$\text{MAP}(\boldsymbol{W} \mid \boldsymbol{e}) = \arg\max_{\boldsymbol{w}} P(\boldsymbol{w}, \boldsymbol{e}), \tag{2.12}$$

where, in general, $\arg\max_x f(x)$ represents the value of x for which $f(x)$ is maximal. Note that there might be more than one assignment that has the highest posterior probability. In this case, we can either decide that the MAP task is to return the set of possible assignments, or to return an arbitrary member of that set.

It is important to understand the difference between MAP queries and probability queries. In a MAP query, we are finding the most likely *joint* assignment to $\boldsymbol{W}$. To find the most likely assignment to a single variable A, we could simply compute $P(A \mid \boldsymbol{e})$ and then pick the most likely value. **However, the assignment where each variable individually picks its most likely value can be quite different from the most likely joint assignment to all variables simultaneously.** This phenomenon can occur even in the simplest case, where we have no evidence.

Example 2.4

Consider a two node chain $A \rightarrow B$ where A and B are both binary-valued. Assume that:

a^0	a^1
0.4	0.6

A	b^0	b^1
a^0	0.1	0.9
a^1	0.5	0.5

(2.13)

We can see that $P(a^1) > P(a^0)$, so that $\text{MAP}(A) = a^1$. However, $\text{MAP}(A, B) = (a^0, b^1)$: Both values of B have the same probability given a^1. Thus, the most likely assignment containing a^1 has probability $0.6 \times 0.5 = 0.3$. On the other hand, the distribution over values of B is more skewed given a^0, and the most likely assignment (a^0, b^1) has the probability $0.4 \times 0.9 = 0.36$. Thus, we have that $\arg\max_{a,b} P(a, b) \neq (\arg\max_a P(a), \arg\max_b P(b))$. ■

2.1.5.3 Marginal MAP Queries

To motivate our second query type, let us return to the phenomenon demonstrated in example 2.4. Now, consider a medical diagnosis problem, where the most likely disease has multiple possible symptoms, each of which occurs with some probability, but not an overwhelming probability. On the other hand, a somewhat rarer disease might have only a few symptoms, each of which is very likely given the disease. As in our simple example, the MAP assignment to the data and the symptoms might be higher for the second disease than for the first one. The solution here is to look for the most likely assignment to the disease variable(s) only, rather than the most likely assignment to both the disease and symptom variables. This approach suggests the use of a more general query type. In the *marginal MAP* query, we have a subset of variables $\boldsymbol{Y}$ that forms our query. The task is to find the most likely assignment to the variables in $\boldsymbol{Y}$ given the evidence $\boldsymbol{E} = \boldsymbol{e}$:

marginal MAP

$$\text{MAP}(\boldsymbol{Y} \mid \boldsymbol{e}) = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y} \mid \boldsymbol{e}).$$

If we let $\boldsymbol{Z} = \mathcal{X} - \boldsymbol{Y} - \boldsymbol{E}$, the marginal MAP task is to compute:

$$\text{MAP}(\boldsymbol{Y} \mid \boldsymbol{e}) = \arg\max_{\boldsymbol{Y}} \sum_{\boldsymbol{Z}} P(\boldsymbol{Y}, \boldsymbol{Z} \mid \boldsymbol{e}).$$

Thus, marginal MAP queries contain both summations and maximizations; in a way, it contains elements of both a conditional probability query and a MAP query.

Note that example 2.4 shows that marginal MAP assignments are not monotonic: the most likely assignment $\text{MAP}(Y_1 \mid \boldsymbol{e})$ might be completely different from the assignment to Y_1 in $\text{MAP}(\{Y_1, Y_2\} \mid \boldsymbol{e})$. Thus, in particular, we cannot use a MAP query to give us the correct answer to a marginal MAP query.

2.1.6 Continuous Spaces

In the previous section, we focused on random variables that have a finite set of possible values. In many situations, we also want to reason about continuous quantities such as weight, height, duration, or cost that take real numbers in $\mathbb{R}$.

When dealing with probabilities over continuous random variables, we have to deal with some technical issues. For example, suppose that we want to reason about a random variable X that can take values in the range between 0 and 1. That is, $Val(X)$ is the interval $[0, 1]$. Moreover, assume that we want to assign each number in this range equal probability. What would be the probability of a number x? Clearly, since each x has the same probability, and there are infinite number of values, we must have that $P(X = x) = 0$. This problem appears even if we do not require uniform probability.

2.1.6.1 Probability Density Functions

How do we define probability over a continuous random variable? We say that a function $p : \mathbb{R} \mapsto \mathbb{R}$ is a *probability density function* or (*PDF*) for X if it is a nonnegative integrable

density function

function such that

$$\int_{Val(X)} p(x)dx = 1.$$

That is, the integral over the set of possible values of X is 1. The PDF defines a distribution for X as follows: for any x in our event space:

$$P(X \leq a) = \int_{-\infty}^{a} p(x)dx.$$

cumulative distribution

The function P is the *cumulative distribution* for X. We can easily employ the rules of probability to see that by using the density function we can evaluate the probability of other events. For example,

$$P(a \leq X \leq b) = \int_{a}^{b} p(x)dx.$$

Intuitively, the value of a PDF $p(x)$ at a point x is the incremental amount that x adds to the cumulative distribution in the integration process. The higher the value of p at and around x, the more mass is added to the cumulative distribution as it passes x.

The simplest PDF is the uniform distribution.

Definition 2.6

uniform distribution

A variable X has a uniform distribution *over $[a, b]$, denoted $X \sim \mathrm{Unif[a, b]}$ if it has the PDF*

$$p(x) = \begin{cases} \frac{1}{b-a} & b \geq x \geq a \\ 0 & \text{otherwise.} \end{cases}$$

■

Thus, the probability of any subinterval of $[a, b]$ is proportional its size relative to the size of $[a, b]$. Note that, if $b - a < 1$, then the density can be greater than 1. Although this looks unintuitive, this situation can occur even in a legal PDF, if the interval over which the value is greater than 1 is not too large. We have only to satisfy the constraint that the total area under the PDF is 1.

As a more complex example, consider the Gaussian distribution.

Definition 2.7

Gaussian distribution

A random variable X has a Gaussian distribution *with* mean μ *and* variance σ^2*, denoted $X \sim \mathcal{N}(\mu; \sigma^2)$, if it has the PDF*

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

standard Gaussian

A standard Gaussian *is one with mean* 0 *and variance* 1. ■

A Gaussian distribution has a bell-like curve, where the mean parameter μ controls the location of the peak, that is, the value for which the Gaussian gets its maximum value. The variance parameter σ^2 determines how peaked the Gaussian is: the smaller the variance, the

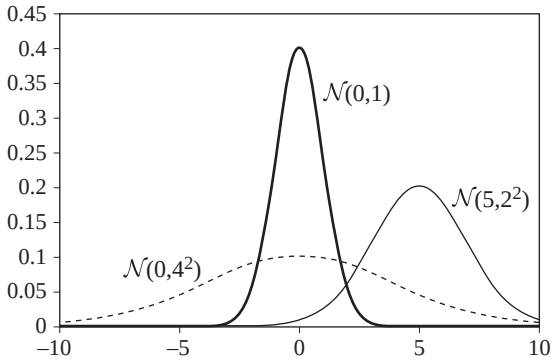


Figure 2.2 Example PDF of three Gaussian distributions

more peaked the Gaussian. Figure 2.2 shows the probability density function of a few different Gaussian distributions.

More technically, the probability density function is specified as an exponential, where the expression in the exponent corresponds to the square of the number of standard deviations σ that x is away from the mean μ. The probability of x decreases exponentially with the square of its deviation from the mean, as measured in units of its standard deviation.

2.1.6.2 Joint Density Functions

The discussion of density functions for a single variable naturally extends for joint distributions of continuous random variables.

Definition 2.8 joint density

Let P be a joint distribution over continuous random variables $X_1, \ldots, X_n$. A function $p(x_1, \ldots, x_n)$ is a joint density function *of $X_1, \ldots, X_n$ if*

- $p(x_1, \ldots, x_n) \geq 0$ *for all values $x_1, \ldots, x_n$ of $X_1, \ldots, X_n$.*
- *p is an integrable function.*
- *For any choice of $a_1, \ldots, a_n$, and $b_1, \ldots, b_n$,*

$$P(a_1 \leq X_1 \leq b_1, \ldots, a_n \leq X_n \leq b_n) = \int_{a_1}^{b_1} \cdots \int_{a_n}^{b_n} p(x_1, \ldots, x_n) dx_1 \ldots dx_n.$$

■

Thus, a joint density specifies the probability of any joint event over the variables of interest.

Both the uniform distribution and the Gaussian distribution have natural extensions to the multivariate case. The definition of a multivariate uniform distribution is straightforward. We defer the definition of the multivariate Gaussian to section 7.1.

From the joint density we can derive the marginal density of any random variable by integrating out the other variables. Thus, for example, if $p(x, y)$ is the joint density of X and Y,

then

$$p(x) = \int_{-\infty}^{\infty} p(x, y)dy.$$

To see why this equality holds, note that the event $a \leq X \leq b$ is, by definition, equal to the event "$a \leq X \leq b$ and $-\infty \leq Y \leq \infty$." This rule is the direct analogue of marginalization for discrete variables. Note that, as with discrete probability distributions, we abuse notation a bit and use p to denote both the joint density of X and Y and the marginal density of X. In cases where the distinction is not clear, we use subscripts, so that p_X will be the marginal density, of X, and $p_{X,Y}$ the joint density.

2.1.6.3 Conditional Density Functions

As with discrete random variables, we want to be able to describe conditional distributions of continuous variables. Suppose, for example, we want to define $P(Y \mid X = x)$. Applying the definition of conditional distribution (equation (2.1)), we run into a problem, since $P(X = x) = 0$. Thus, the ratio of $P(Y, X = x)$ and $P(X = x)$ is undefined.

To avoid this problem, we might consider conditioning on the event $x - \epsilon \leq X \leq x + \epsilon$, which can have a positive probability. Now, the conditional probability is well defined. Thus, we might consider the limit of this quantity when $\epsilon \to 0$. We define

$$P(Y \mid x) = \lim_{\epsilon \to 0} P(Y \mid x - \epsilon \leq X \leq x + \epsilon).$$

When does this limit exist? If there is a continuous joint density function $p(x, y)$, then we can derive the form for this term. To do so, consider some event on Y, say $a \leq Y \leq b$. Recall that

$$\begin{aligned} P(a \leq Y \leq b \mid x - \epsilon \leq X \leq x + \epsilon) &= \frac{P(a \leq Y \leq b, x - \epsilon \leq X \leq x + \epsilon)}{P(x - \epsilon \leq X \leq x + \epsilon)} \\ &= \frac{\int_a^b \int_{x-\epsilon}^{x+\epsilon} p(x', y)dydx'}{\int_{x-\epsilon}^{x+\epsilon} p(x')dx'}. \end{aligned}$$

When ϵ is sufficiently small, we can approximate

$$\int_{x-\epsilon}^{x+\epsilon} p(x')dx' \approx 2\epsilon p(x).$$

Using a similar approximation for $p(x', y)$, we get

$$\begin{aligned} P(a \leq Y \leq b \mid x - \epsilon \leq X \leq x + \epsilon) &\approx \frac{\int_a^b 2\epsilon p(x, y)dy}{2\epsilon p(x)} \\ &= \int_a^b \frac{p(x, y)}{p(x)} dy. \end{aligned}$$

We conclude that $\frac{p(x,y)}{p(x)}$ is the density of $P(Y \mid X = x)$.

Definition 2.9 conditional density function

Let $p(x, y)$ be the joint density of X and Y. The conditional density function *of Y given X is defined as*

$$p(y \mid x) = \frac{p(x, y)}{p(x)}$$

When $p(x) = 0$, the conditional density is undefined. ■

The conditional density $p(y \mid x)$ characterizes the conditional distribution $P(Y \mid X = x)$ we defined earlier.

The properties of joint distributions and conditional distributions carry over to joint and conditional density functions. In particular, we have the chain rule

$$p(x, y) = p(x)p(y \mid x) \tag{2.14}$$

and Bayes' rule

$$p(x \mid y) = \frac{p(x)p(y \mid x)}{p(y)}. \tag{2.15}$$

As a general statement, whenever we discuss joint distributions of continuous random variables, we discuss properties with respect to the joint density function instead of the joint distribution, as we do in the case of discrete variables. Of particular interest is the notion of (conditional) independence of continuous random variables.

Definition 2.10 conditional independence

Let $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ be sets of continuous random variables with joint density $p(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$. We say that $\boldsymbol{X}$ is conditionally independent *of $\boldsymbol{Y}$ given $\boldsymbol{Z}$ if*

$$p(\boldsymbol{x} \mid \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) \text{ for all } \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \text{ such that } p(\boldsymbol{z}) > 0.$$ ■

2.1.7 Expectation and Variance

2.1.7.1 Expectation

expectation

Let X be a discrete random variable that takes numerical values; then the *expectation* of X under the distribution P is

$$\boldsymbol{E}_P[X] = \sum_x x \cdot P(x).$$

If X is a continuous variable, then we use the density function

$$\boldsymbol{E}_P[X] = \int x \cdot p(x)dx.$$

For example, if we consider X to be the outcome of rolling a fair die with probability $1/6$ for each outcome, then $\boldsymbol{E}[X] = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + \cdots + 6 \cdot \frac{1}{6} = 3.5$. On the other hand, if we consider a biased die where $P(X = 6) = 0.5$ and $P(X = x) = 0.1$ for $x < 6$, then $\boldsymbol{E}[X] = 1 \cdot 0.1 + \cdots + 5 \cdot 0.1 + \cdots + 6 \cdot 0.5 = 4.5$.

Often we are interested in expectations of a function of a random variable (or several random variables). Thus, we might consider extending the definition to consider the expectation of a functional term such as $X^2 + 0.5X$. Note, however, that any function g of a set of random variables $X_1, \ldots, X_k$ is essentially defining a new random variable Y: For any outcome $\omega \in \Omega$, we define the value of Y as $g(f_{X_1}(\omega), \ldots, f_{X_k}(\omega))$.

Based on this discussion, we often define new random variables by a functional term. For example $Y = X^2$, or $Y = e^X$. We can also consider functions that map values of one or more categorical random variables to numerical values. One such function that we use quite often is the *indicator function*, which we denote $\mathbf{1}\{\boldsymbol{X} = \boldsymbol{x}\}$. This function takes value 1 when $\boldsymbol{X} = \boldsymbol{x}$, and 0 otherwise.

indicator function

In addition, we often consider expectations of functions of random variables without bothering to name the random variables they define. For example $\boldsymbol{E}_P[X + Y]$. Nonetheless, we should keep in mind that such a term does refer to an expectation of a random variable.

We now turn to examine properties of the expectation of a random variable.

First, as can be easily seen, the expectation of a random variable is a linear function in that random variable. Thus,

$$\boldsymbol{E}[a \cdot X + b] = a\boldsymbol{E}[X] + b.$$

A more complex situation is when we consider the expectation of a function of several random variables that have some joint behavior. An important property of expectation is that the expectation of a sum of two random variables is the sum of the expectations.

Proposition 2.4

$$\boldsymbol{E}[X + Y] = \boldsymbol{E}[X] + \boldsymbol{E}[Y].$$

linearity of expectation

This property is called *linearity of expectation*. It is important to stress that this identity is true even when the variables are not independent. As we will see, this property is key in simplifying many seemingly complex problems.

Finally, what can we say about the expectation of a product of two random variables? In general, very little:

Example 2.5

Consider two random variables X and Y, each of which takes the value $+1$ with probability $1/2$, and the value -1 with probability $1/2$. If X and Y are independent, then $\boldsymbol{E}[X \cdot Y] = 0$. On the other hand, if X and Y are correlated in that they always take the same value, then $\boldsymbol{E}[X \cdot Y] = 1$. ■

However, when X and Y are independent, then, as in our example, we can compute the expectation simply as a product of their individual expectations:

Proposition 2.5

If X and Y are independent, then

$$\boldsymbol{E}[X \cdot Y] = \boldsymbol{E}[X] \cdot \boldsymbol{E}[Y].$$

conditional expectation

We often also use the expectation given some evidence. The *conditional expectation* of X given $\boldsymbol{y}$ is

$$\boldsymbol{E}_P[X \mid \boldsymbol{y}] = \sum_x x \cdot P(x \mid \boldsymbol{y}).$$

2.1.7.2 Variance

variance

The expectation of X tells us the mean value of X. However, It does not indicate how far X deviates from this value. A measure of this deviation is the *variance* of X.

$$\mathbf{Var}_P[X] = \mathbf{E}_P\left[(X - \mathbf{E}_P[X])^2\right].$$

Thus, the variance is the expectation of the squared difference between X and its expected value. It gives us an indication of the spread of values of X around the expected value.

An alternative formulation of the variance is

$$\mathbf{Var}[X] = \mathbf{E}\left[X^2\right] - (\mathbf{E}[X])^2 . \tag{2.16}$$

(see exercise 2.11).

Similar to the expectation, we can consider the expectation of a functions of random variables.

Proposition 2.6 *If X and Y are independent, then*

$$\mathbf{Var}[X + Y] = \mathbf{Var}[X] + \mathbf{Var}[Y].$$

It is straightforward to show that the variance scales as a quadratic function of X. In particular, we have:

$$\mathbf{Var}[a \cdot X + b] = a^2 \mathbf{Var}[X].$$

standard deviation

For this reason, we are often interested in the square root of the variance, which is called the *standard deviation* of the random variable. We define

$$\sigma_X = \sqrt{\mathbf{Var}[X]}.$$

The intuition is that it is improbable to encounter values of X that are farther than several standard deviations from the expected value of X. Thus, σ_X is a normalized measure of "distance" from the expected value of X.

As an example consider the Gaussian distribution of definition 2.7.

Proposition 2.7 *Let X be a random variable with Gaussian distribution $N(\mu, \sigma^2)$, then $\mathbf{E}[X] = \mu$ and $\mathbf{Var}[X] = \sigma^2$.*

Thus, the parameters of the Gaussian distribution specify the expectation and the variance of the distribution. As we can see from the form of the distribution, the density of values of X drops exponentially fast in the distance $\frac{x-\mu}{\sigma}$.

Not all distributions show such a rapid decline in the probability of outcomes that are distant from the expectation. However, even for arbitrary distributions, one can show that there is a decline.

Theorem 2.1 **(Chebyshev inequality)**:

Chebyshev's inequality

$$P(|X - \mathbf{E}_P[X]| \geq t) \leq \frac{\mathbf{Var}_P[X]}{t^2}.$$

We can restate this inequality in terms of standard deviations: We write $t = k\sigma_X$ to get

$$P(|X - \boldsymbol{E}_P[X]| \geq k\sigma_X) \leq \frac{1}{k^2}.$$

Thus, for example, the probability of X being more than two standard deviations away from $\boldsymbol{E}[X]$ is less than $1/4$.

2.2 Graphs

Perhaps the most pervasive concept in this book is the representation of a probability distribution using a graph as a data structure. In this section, we survey some of the basic concepts in graph theory used in the book.

2.2.1 Nodes and Edges

A graph is a data structure $\mathcal{K}$ consisting of a set of nodes and a set of edges. Throughout most this book, we will assume that the set of nodes is $\mathcal{X} = \{X_1, \ldots, X_n\}$. A pair of nodes X_i, X_j can be connected by a *directed edge* $X_i \to X_j$ or an *undirected edge* X_i—X_j. Thus, the set of edges $\mathcal{E}$ is a set of pairs, where each pair is one of $X_i \to X_j$, $X_j \to X_i$, or X_i—X_j, for $X_i, X_j \in \mathcal{X}$, $i < j$. We assume throughout the book that, for each pair of nodes X_i, X_j, at most one type of edge exists; thus, we cannot have both $X_i \to X_j$ and $X_j \to X_i$, nor can we have $X_i \to X_j$ and X_i—X_j.[2] The notation $X_i \leftarrow X_j$ is equivalent to $X_j \to X_i$, and the notation X_j—X_i is equivalent to X_i—X_j. We use $X_i \rightleftharpoons X_j$ to represent the case where X_i and X_j are connected via some edge, whether directed (in any direction) or undirected.

directed edge

undirected edge

In many cases, we want to restrict attention to graphs that contain only edges of one kind or another. We say that a graph is *directed* if all edges are either $X_i \to X_j$ or $X_j \to X_i$. We usually denote directed graphs as $\mathcal{G}$. We say that a graph is *undirected* if all edges are X_i—X_j. We denote undirected graphs as $\mathcal{H}$. We sometimes convert a general graph to an undirected graph by ignoring the directions on the edges.

directed graph

undirected graph

Definition 2.11 *Given a graph $\mathcal{K} = (\mathcal{X}, \mathcal{E})$, its* undirected version *is a graph $\mathcal{H} = (\mathcal{X}, \mathcal{E}')$ where $\mathcal{E}' = \{X$—$Y : X \rightleftharpoons Y \in \mathcal{E}\}$.* ■

graph's undirected version

Whenever we have that $X_i \to X_j \in \mathcal{E}$, we say that X_j is the *child* of X_i in $\mathcal{K}$, and that X_i is the *parent* of X_j in $\mathcal{K}$. When we have X_i—$X_j \in \mathcal{E}$, we say that X_i is a *neighbor* of X_j in $\mathcal{K}$ (and vice versa). We say that X and Y are adjacent whenever $X \rightleftharpoons Y \in \mathcal{E}$. We use Pa_X to denote the parents of X, Ch_X to denote its children, and Nb_X to denote its neighbors. We define the *boundary* of X, denoted $\mathrm{Boundary}_X$, to be $\mathrm{Pa}_X \cup \mathrm{Nb}_X$; for DAGs, this set is simply X's parents, and for undirected graphs X's neighbors.[3] Figure 2.3 shows an example of a graph $\mathcal{K}$. There, we have that A is the only parent of C, and F, I are the children of C. The only neighbor of C is D, but its adjacent nodes are A, D, F, I. The *degree* of a node X is the number of edges in which it participates. Its *indegree* is the number of directed edges $Y \to X$. The *degree* of a graph is the maximal degree of a node in the graph.

child

parent

neighbor

boundary

degree

indegree

2. Note that our definition is somewhat restricted, in that it disallows cycles of length two, where $X_i \to X_j \to X_i$, and allows self-loops where $X_i \to X_i$.

3. When the graph is not clear from context, we often add the graph as an additional argument.

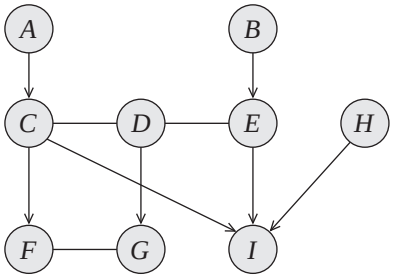


Figure 2.3 An example of a partially directed graph $\mathcal{K}$

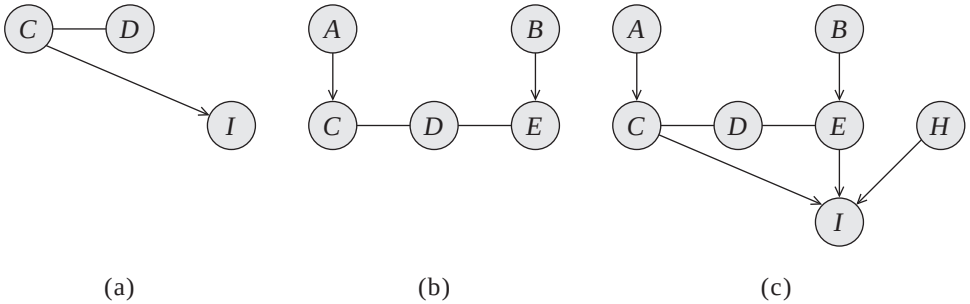


Figure 2.4 Induced graphs and their upward closure: (a) The induced subgraph $\mathcal{K}[C, D, I]$. (b) The upwardly closed subgraph $\mathcal{K}^+[C]$. (c) The upwardly closed subgraph $\mathcal{K}^+[C, D, I]$.

2.2.2 Subgraphs

In many cases, we want to consider only the part of the graph that is associated with a particular subset of the nodes.

Definition 2.12 induced subgraph

Let $\mathcal{K} = (\mathcal{X}, \mathcal{E})$, and let $\boldsymbol{X} \subset \mathcal{X}$. We define the induced subgraph $\mathcal{K}[\boldsymbol{X}]$ *to be the graph $(\boldsymbol{X}, \mathcal{E}')$ where $\mathcal{E}'$ are all the edges $X \rightleftharpoons Y \in \mathcal{E}$ such that $X, Y \in \boldsymbol{X}$.* ■

For example, figure 2.4a shows the induced subgraph $\mathcal{K}[C, D, I]$.

A type of subgraph that is often of particular interest is one that contains all possible edges.

Definition 2.13 complete subgraph, clique

A subgraph over $\boldsymbol{X}$ is complete *if every two nodes in $\boldsymbol{X}$ are connected by some edge. The set $\boldsymbol{X}$ is often called a* clique*; we say that a clique $\boldsymbol{X}$ is* maximal *if for any superset of nodes $\boldsymbol{Y} \supset \boldsymbol{X}$, $\boldsymbol{Y}$ is not a clique.* ■

Although the subset of nodes $\boldsymbol{X}$ can be arbitrary, we are often interested in sets of nodes that preserve certain aspects of the graph structure.

Definition 2.14 upward closure

We say that a subset of nodes $\boldsymbol{X} \in \mathcal{X}$ is upwardly closed *in $\mathcal{K}$ if, for any $X \in \boldsymbol{X}$, we have that* $\text{Boundary}_X \subset \boldsymbol{X}$. *We define the* upward closure *of $\boldsymbol{X}$ to be the minimal upwardly closed subset*

$\boldsymbol{Y}$ that contains $\boldsymbol{X}$. We define the upwardly closed subgraph *of $\boldsymbol{X}$, denoted $\mathcal{K}^+[\boldsymbol{X}]$, to be the induced subgraph over $\boldsymbol{Y}$, $\mathcal{K}[\boldsymbol{Y}]$.* ■

For example, the set A, B, C, D, E is the upward closure of the set $\{C\}$ in $\mathcal{K}$. The upwardly closed subgraph of $\{C\}$ is shown in figure 2.4b. The upwardly closed subgraph of $\{C, D, I\}$ is shown in figure 2.4c.

2.2.3 Paths and Trails

Using the basic notion of edges, we can define different types of longer-range connections in the graph.

Definition 2.15 path

We say that $X_1, \ldots, X_k$ form a path *in the graph $\mathcal{K} = (\mathcal{X}, \mathcal{E})$ if, for every $i = 1, \ldots, k-1$, we have that either $X_i \rightarrow X_{i+1}$ or X_i—X_{i+1}. A path is* directed *if, for at least one i, we have $X_i \rightarrow X_{i+1}$.* ■

Definition 2.16 trail

We say that $X_1, \ldots, X_k$ form a trail *in the graph $\mathcal{K} = (\mathcal{X}, \mathcal{E})$ if, for every $i = 1, \ldots, k-1$, we have that $X_i \rightleftharpoons X_{i+1}$.* ■

In the graph $\mathcal{K}$ of figure 2.3, A, C, D, E, I is a path, and hence also a trail. On the other hand, A, C, F, G, D is a trail, which is not a path.

Definition 2.17 connected graph

A graph is connected *if for every X_i, X_j there is a trail between X_i and X_j.* ■

We can now define longer-range relationships in the graph.

Definition 2.18 ancestor descendant

We say that X is an ancestor *of Y in $\mathcal{K} = (\mathcal{X}, \mathcal{E})$, and that Y is a* descendant *of X, if there exists a directed path $X_1, \ldots, X_k$ with $X_1 = X$ and $X_k = Y$. We use $\mathrm{Descendants}_X$ to denote X's descendants, $\mathrm{Ancestors}_X$ to denote X's ancestors, and $\mathrm{NonDescendants}_X$ to denote the set of nodes in $\mathcal{X} - \mathrm{Descendants}_X$.* ■

In our example graph $\mathcal{K}$, we have that F, G, I are descendants of C. The ancestors of C are A, via the path A, C, and B, via the path B, E, D, C.

A final useful notion is that of an ordering of the nodes in a directed graph that is consistent with the directionality its edges.

Definition 2.19 topological ordering

Let $\mathcal{G} = (\mathcal{X}, \mathcal{E})$ be a graph. An ordering of the nodes $X_1, \ldots, X_n$ is a topological ordering *relative to $\mathcal{K}$ if, whenever we have $X_i \rightarrow X_j \in \mathcal{E}$, then $i < j$.* ■

Appendix A.3.1 presents an algorithm for finding such a topological ordering.

2.2.4 Cycles and Loops

Note that, in general, we can have a cyclic path that leads from a node to itself, making that node its own descendant.

Definition 2.20

cycle

acyclic

A cycle *in $\mathcal{K}$ is a directed path $X_1, \ldots, X_k$ where $X_1 = X_k$. A graph is* acyclic *if it contains no cycles.* ■

For most of this book, we will restrict attention to graphs that do not allow such cycles, since it is quite difficult to define a coherent probabilistic model over graphs with directed cycles.

DAG

A *directed acyclic graph (DAG)* is one of the central concepts in this book, as DAGs are the basic graphical representation that underlies Bayesian networks. For some of this book, we also use acyclic graphs that are partially directed. The graph $\mathcal{K}$ of figure 2.3 is acyclic. However, if we add the undirected edge A—E to $\mathcal{K}$, we have a path A, C, D, E, A from A to itself. Clearly, adding a directed edge $E \rightarrow A$ would also lead to a cycle. Note that prohibiting cycles does not imply that there is no *trail* from a node to itself. For example, $\mathcal{K}$ contains several trails: C, D, E, I, C as well as C, D, G, F, C.

PDAG

chain component

An acyclic graph containing both directed and undirected edges is called a *partially directed acyclic graph* or *PDAG*. The acyclicity requirement on a PDAG implies that the graph can be decomposed into a directed graph of *chain components*, where the nodes within each chain component are connected to each other only with undirected edges. The acyclicity of a PDAG guarantees us that we can order the components so that all edges point from lower-numbered components to higher-numbered ones.

Definition 2.21

Let $\mathcal{K}$ be a PDAG over $\mathcal{X}$. Let $\boldsymbol{K}_1, \ldots, \boldsymbol{K}_\ell$ be a disjoint partition of $\mathcal{X}$ such that:

- *the induced subgraph over $\boldsymbol{K}_i$ contains no directed edges;*
- *for any pair of nodes $X \in \boldsymbol{K}_i$ and $Y \in \boldsymbol{K}_j$ for $i < j$, an edge between X and Y can only be a directed edge $X \rightarrow Y$.*

chain component

Each component $\boldsymbol{K}_i$ is called a chain component. ■

chain graph

Because of its chain structure, a PDAG is also called a *chain graph*.

Example 2.6

In the PDAG of figure 2.3, we have six chain components: $\{A\}$, $\{B\}$, $\{C, D, E\}$, $\{F, G\}$, $\{H\}$, and $\{I\}$. This ordering of the chain components is one of several possible legal orderings. ■

Note that when the PDAG is an undirected graph, the entire graph forms a single chain component. Conversely, when the PDAG is a directed graph (and therefore acyclic), each node in the graph is its own chain component.

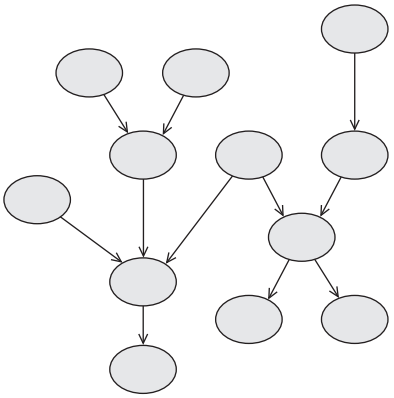


Figure 2.5 An example of a polytree

Different from a cycle is the notion of a loop:

Definition 2.22

loop

singly connected

leaf

polytree

forest

tree

A loop *in $\mathcal{K}$ is a trail $X_1, \ldots, X_k$ where $X_1 = X_k$. A graph is* singly connected *if it contains no loops. A node in a singly connected graph is called a* leaf *if it has exactly one adjacent node. A singly connected directed graph is also called a* polytree. *A singly connected undirected graph is called a* forest*; if it is also connected, it is called a* tree. ■

We can also define a notion of a forest, or of a tree, for directed graphs.

Definition 2.23

A directed graph is a forest *if each node has at most one parent. A directed forest is a* tree *if it is also connected.* ■

Note that polytrees are very different from trees. For example, figure 2.5 shows a graph that is a polytree but is not a tree, because several nodes have more than one parent. As we will discuss later in the book, loops in the graph increase the computational cost of various tasks.

We conclude this section with a final definition relating to loops in the graph. This definition will play an important role in evaluating the cost of reasoning using graph-based representations.

Definition 2.24

chordal graph

Let X_1—X_2—$\cdots$—X_k—X_1 be a loop in the graph; a chord *in the loop is an edge connecting X_i and X_j for two nonconsecutive nodes X_i, X_j. An undirected graph $\mathcal{H}$ is said to be* chordal *if any loop X_1—X_2—$\cdots$—X_k—X_1 for $k \geq 4$ has a chord.* ■

Thus, for example, a loop A—B—C—D—A (as in figure 1.1b) is nonchordal, but adding an edge A—C would render it chordal. In other words, in a chordal graph, the longest "minimal loop" (one that has no shortcut) is a triangle. Thus, chordal graphs are often also called *triangulated*.

triangulated graph

We can extend the notion of chordal graphs to graphs that contain directed edges.

Definition 2.25

A graph $\mathcal{K}$ is said to be chordal *if its underlying undirected graph is chordal.* ■

2.3 Relevant Literature

Section 1.4 provides some history on the development of of probabilistic methods. There are many good textbooks about probability theory; see, for example, DeGroot (1989), Ross (1988) or Feller (1970). The distinction between the frequentist and subjective view of probability was a major issue during much of the late nineteenth and early twentieth centuries. Some references that touch on this discussion include Cox (2001) and Jaynes (2003) on the Bayesian side, and Feller (1970) on the frequentist side; these books also contain much useful general material about probabilistic reasoning.

Dawid (1979, 1980) was the first to propose the axiomatization of conditional independence properties, and he showed how they can help unify a variety of topics within probability and statistics. These axioms were studied in great detail by Pearl and colleagues, work that is presented in detail in Pearl (1988).

2.4 Exercises

Exercise 2.1

Prove the following properties using basic properties of definition 2.1:

a. $P(\emptyset) = 0$.
b. If $\alpha \subseteq \beta$, then $P(\alpha) \leq P(\beta)$.
c. $P(\alpha \cup \beta) = P(\alpha) + P(\beta) - P(\alpha \cap \beta)$.

Exercise 2.2

a. Show that for binary random variables X, Y, the event-level independence $(x^0 \perp y^0)$ implies random-variable independence $(X \perp Y)$.
b. Show a counterexample for nonbinary variables.
c. Is it the case that, for a binary-valued variable Z, we have that $(X \perp Y \mid z^0)$ implies $(X \perp Y \mid Z)$?

Exercise 2.3

Consider two events α and β such that $P(\alpha) = p_a$ and $P(\beta) = p_b$. Given only that knowledge, what is the maximum and minimum values of the probability of the events $\alpha \cap \beta$ and $\alpha \cup \beta$. Can you characterize the situations in which each of these extreme values occurs?

Exercise 2.4⋆

Let P be a distribution over $(\Omega, \mathcal{S})$, and let $a \in \mathcal{S}$ be an event such that $P(\alpha) > 0$. The conditional probability $P(\cdot \mid \alpha)$ assigns a value to each event in $\mathcal{S}$. Show that it satisfies the properties of definition 2.1.

Exercise 2.5

Let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ be three disjoint subsets of variables such that $\mathcal{X} = \boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$. Prove that $P \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ if and only if we can write P in the form:

$$P(\mathcal{X}) = \phi_1(\boldsymbol{X}, \boldsymbol{Z})\phi_2(\boldsymbol{Y}, \boldsymbol{Z}).$$

Exercise 2.6

An often useful rule in dealing with probabilities is known as *reasoning by cases*. Let X, Y, and Z be random variables, then

$$P(X \mid Y) = \sum_z P(X, z \mid Y).$$

Prove this equality using the chain rule of probabilities and basic properties of (conditional) distributions.

Exercise 2.7⋆

In this exercise, you will prove the properties of conditional independence discussed in section 2.1.4.3.

a. Prove that the weak union and contraction properties hold for any probability distribution P.
b. Prove that the intersection property holds for any positive probability distribution P.
c. Provide a counterexample to the intersection property in cases where the distribution P is not positive.

Exercise 2.8

a. Show that for binary random variables X and Y, $(x^1 \perp y^1)$ implies $(X \perp Y)$.
b. Provide a counterexample to this property for nonbinary variables.
c. Is it the case that, for binary Z, $(X \perp Y \mid z^1)$ implies $(X \perp Y \mid Z)$? Prove or provide a counterexample.

Exercise 2.9

Show how you can use breadth-first search to determine whether a graph $\mathcal{K}$ is cyclic.

Exercise 2.10⋆

In appendix A.3.1, we describe an algorithm for finding a topological ordering for a directed graph. Extend this algorithm to one that finds a topological ordering for the chain components in a PDAG. Your algorithm should construct both the chain components of the PDAG, as well as an ordering over them that satisfies the conditions of definition 2.21. Analyze the asymptotic complexity of your algorithm.

Exercise 2.11

Use the properties of expectation to show that we can rewrite the variance of a random variable X as

$$\mathbf{Var}[X] = \mathbf{E}\left[X^2\right] - (\mathbf{E}[X])^2 .$$

Exercise 2.12⋆

Prove the following property of expectations

Theorem 2.2 Markov inequality

(Markov inequality) *Let X be a random variable such that $P(X \geq 0) = 1$, then for any $t \geq 0$,*

$$P(X \geq t) \leq \frac{\mathbf{E}_P[X]}{t}.$$

You may assume in your proof that X is a discrete random variable with a finite number of values.

Exercise 2.13

Prove Chebyshev's inequality using the Markov inequality shown in exercise 2.12. (Hint: define a new random variable Y, so that the application of the Markov inequality with respect to this random variable gives the required result.)

Exercise 2.14⋆

Let $X \sim \mathcal{N}\left(\mu; \sigma^2\right)$, and define a new variable $Y = a \cdot X + b$. Show that $Y \sim \mathcal{N}\left(a \cdot \mu + b; a^2\sigma^2\right)$.

Exercise 2.15⋆

concave function
convex function

A function f is *concave* if for any $0 \leq \alpha \leq 1$ and any x and y, we have that $f(\alpha x + (1-\alpha)y) \geq \alpha f(x) + (1-\alpha)f(y)$. A function is *convex* if the opposite holds, that is, $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$.

a. Prove that a continuous and differentiable function f is concave if and only if $f''(x) \leq 0$ for all x.
b. Show that $\log(x)$ is concave (over the positive real numbers).

Exercise 2.16⋆

Proposition 2.8

Jensen inequality

Jensen's inequality: Let f be a concave function and P a distribution over a random variable X. Then

$$\boldsymbol{E}_P[f(X)] \leq f(\boldsymbol{E}_P[X])$$

Use this inequality to show that:

a. $\boldsymbol{H}_P(X) \leq \log |Val(X)|$.
b. $\boldsymbol{H}_P(X) \geq 0$.
c. $\boldsymbol{D}(P \| Q) \geq 0$.

See appendix A.1 for the basic definitions.

Exercise 2.17

Show that, for any probability distribution $P(X)$, we have that

$$\boldsymbol{H}_P(X) = \log K - \boldsymbol{D}(P(X) \| P_u(X))$$

where $P_u(X)$ is the uniform distribution over $Val(X)$ and $K = |Val(X)|$.

Exercise 2.18⋆

Prove proposition A.3, and use it to show that $\boldsymbol{I}(X;Y) \geq 0$.

Exercise 2.19

conditional mutual information

As with entropies, we can define the notion of *conditional mutual information*

$$\boldsymbol{I}_P(X;Y \mid Z) = \boldsymbol{E}_P\left[\log \frac{P(X \mid Y,Z)}{P(X \mid Z)}\right].$$

Prove that:

a. $\boldsymbol{I}_P(X;Y \mid Z) = \boldsymbol{H}_P(X \mid Z) - \boldsymbol{H}_P(X,Y \mid Z)$.
b. The *chain rule of mutual information*:

chain rule of mutual information

$$\boldsymbol{I}_P(X;Y,Z) = \boldsymbol{I}_P(X;Y) + \boldsymbol{I}_P(X;Z \mid Y).$$

Exercise 2.20

Use the chain law of mutual information to prove that

$$\boldsymbol{I}_P(X;Y) \leq \boldsymbol{I}_P(X;Y,Z).$$

That is, the information that Y and Z together convey about X cannot be less than what Y alone conveys about X.

Exercise 2.21⋆

Consider a sequence of N independent samples from a binary random variable X whose distribution is $P(x^1) = p$, $P(x^0) = 1 - p$. As in appendix A.2, let S_N be the number of trials whose outcome is x^1. Show that

$$P(S_N = r) \approx \exp[-N \cdot \boldsymbol{D}((p, 1-p) \| (r/N, 1 - r/N))].$$

Your proof should use Stirling's approximation to the factorial function:

$$m! \approx \frac{1}{2\pi m} m^m e^{-m}.$$

Part I

Representation

3 The Bayesian Network Representation

Our goal is to represent a joint distribution P over some set of random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. Even in the simplest case where these variables are binary-valued, a joint distribution requires the specification of $2^n - 1$ numbers — the probabilities of the 2^n different assignments of values $x_1, \ldots, x_n$. For all but the smallest n, **the explicit representation of the joint distribution is unmanageable from every perspective. Computationally, it is very expensive to manipulate and generally too large to store in memory. Cognitively, it is impossible to acquire so many numbers from a human expert; moreover, the numbers are very small and do not correspond to events that people can reasonably contemplate. Statistically, if we want to learn the distribution from data, we would need ridiculously large amounts of data to estimate this many parameters robustly. These problems were the main barrier to the adoption of probabilistic methods for expert systems until the development of the methodologies described in this book.**

In this chapter, we first show how independence properties in the distribution can be used to represent such high-dimensional distributions much more compactly. We then show how a combinatorial data structure — a directed acyclic graph — can provide us with a general-purpose modeling language for exploiting this type of structure in our representation.

3.1 Exploiting Independence Properties

The compact representations we explore in this chapter are based on two key ideas: the representation of independence properties of the distribution, and the use of an alternative parameterization that allows us to exploit these finer-grained independencies.

3.1.1 Independent Random Variables

To motivate our discussion, consider a simple setting where we know that each X_i represents the outcome of a toss of coin i. In this case, we typically assume that the different coin tosses are marginally independent (definition 2.4), so that our distribution P will satisfy $(X_i \perp X_j)$ for any i, j. More generally (strictly more generally — see exercise 3.1), we assume that the distribution satisfies $(\boldsymbol{X} \perp \boldsymbol{Y})$ for any disjoint subsets of the variables $\boldsymbol{X}$ and $\boldsymbol{Y}$. Therefore, we have that:

$$P(X_1, \ldots, X_n) = P(X_1)P(X_2) \cdots P(X_n).$$

If we use the standard parameterization of the joint distribution, this independence structure is obscured, and the representation of the distribution requires 2^n parameters. However, we can use a more natural set of parameters for specifying this distribution: If θ_i is the probability with which coin i lands heads, the joint distribution P can be specified using the n *parameters* $\theta_1, \ldots, \theta_n$. These parameters implicitly specify the 2^n probabilities in the joint distribution. For example, the probability that all of the coin tosses land heads is simply $\theta_1 \cdot \theta_2 \cdot \ldots \cdot \theta_n$. More generally, letting $\theta_{x_i} = \theta_i$ when $x_i = x_i^1$ and $\theta_{x_i} = 1 - \theta_i$ when $x_i = x_i^0$, we can define:

parameters

$$P(x_1, \ldots, x_n) = \prod_i \theta_{x_i}. \tag{3.1}$$

This representation is limited, and there are many distributions that we cannot capture by choosing values for $\theta_1, \ldots, \theta_n$. This fact is obvious not only from intuition, but also from a somewhat more formal perspective. The space of all joint distributions is a $2^n - 1$ dimensional subspace of $\mathbb{R}^{2^n}$ — the set $\{(p_1, \ldots, p_{2^n}) \in \mathbb{R}^{2^n} : p_1 + \ldots + p_{2^n} = 1\}$. On the other hand, the space of all joint distributions specified in a factorized way as in equation (3.1) is an n-dimensional manifold in $\mathbb{R}^{2^n}$.

A key concept here is the notion of *independent parameters* — parameters whose values are not determined by others. For example, when specifying an arbitrary multinomial distribution over a k dimensional space, we have $k - 1$ independent parameters: the last probability is fully determined by the first $k - 1$. In the case where we have an arbitrary joint distribution over n binary random variables, the number of independent parameters is $2^n - 1$. On the other hand, the number of independent parameters for distributions represented as n independent binomial coin tosses is n. Therefore, the two spaces of distributions cannot be the same. (While this argument might seem trivial in this simple case, it turns out to be an important tool for comparing the expressive power of different representations.)

independent parameters

As this simple example shows, certain families of distributions — in this case, the distributions generated by n independent random variables — permit an alternative parameterization that is substantially more compact than the naive representation as an explicit joint distribution. Of course, in most real-world applications, the random variables are not marginally independent. However, a generalization of this approach will be the basis for our solution.

3.1.2 The Conditional Parameterization

Let us begin with a simple example that illustrates the basic intuition. Consider the problem faced by a company trying to hire a recent college graduate. The company's goal is to hire intelligent employees, but there is no way to test intelligence directly. However, the company has access to the student's SAT scores, which are informative but not fully indicative. Thus, our probability space is induced by the two random variables *Intelligence* (I) and *SAT* (S). For simplicity, we assume that each of these takes two values: $Val(I) = \{i^1, i^0\}$, which represent the values high intelligence (i^1) and low intelligence (i^0); similarly $Val(S) = \{s^1, s^0\}$, which also represent the values *high* (score) and *low* (score), respectively.

Thus, our joint distribution in this case has four entries. For example, one possible joint

distribution P would be

I	S	$P(I,S)$
i^0	s^0	0.665
i^0	s^1	0.035
i^1	s^0	0.06
i^1	s^1	0.24.

(3.2)

There is, however, an alternative, and even more natural way of representing the same joint distribution. Using the chain rule of conditional probabilities (see equation (2.5)), we have that

$$P(I,S) = P(I)P(S \mid I).$$

Intuitively, we are representing the process in a way that is more compatible with causality. Various factors (genetics, upbringing, ...) first determined (stochastically) the student's intelligence. His performance on the SAT is determined (stochastically) by his intelligence. We note that the models we construct are not required to follow causal intuitions, but they often do. We return to this issue later on.

From a mathematical perspective, this equation leads to the following alternative way of representing the joint distribution. Instead of specifying the various joint entries $P(I,S)$, we would specify it in the form of $P(I)$ and $P(S \mid I)$. Thus, for example, we can represent the joint distribution of equation (3.2) using the following two tables, one representing the *prior distribution* over I and the other the *conditional probability distribution (CPD)* of S given I:

prior distribution

CPD

i^0	i^1
0.7	0.3

I	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

(3.3)

The CPD $P(S \mid I)$ represents the probability that the student will succeed on his SATs in the two possible cases: the case where the student's intelligence is low, and the case where it is high. The CPD asserts that a student of low intelligence is extremely unlikely to get a high SAT score ($P(s^1 \mid i^0) = 0.05$); on the other hand, a student of high intelligence is likely, but far from certain, to get a high SAT score ($P(s^1 \mid i^1) = 0.8$).

It is instructive to consider how we could parameterize this alternative representation. Here, we are using three binomial distributions, one for $P(I)$, and two for $P(S \mid i^0)$ and $P(S \mid i^1)$. Hence, we can parameterize this representation using three independent parameters, say θ_{i^1}, $\theta_{s^1|i^1}$, and $\theta_{s^1|i^0}$. Our representation of the joint distribution as a four-outcome multinomial also required three parameters. Thus, although the conditional representation is more natural than the explicit representation of the joint, it is not more compact. However, as we will soon see, the conditional parameterization provides a basis for our compact representations of more complex distributions.

Although we will only define Bayesian networks formally in section 3.2.2, it is instructive to see how this example would be represented as one. The Bayesian network, as shown in figure 3.1a, would have a node for each of the two random variables I and S, with an edge from I to S representing the direction of the dependence in this model.

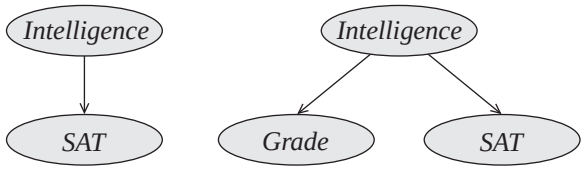


Figure 3.1 Simple Bayesian networks for the student **example**

3.1.3 The Naive Bayes Model

We now describe perhaps the simplest example where a conditional parameterization is combined with conditional independence assumptions to produce a very compact representation of a high-dimensional probability distribution. Importantly, unlike the previous example of fully independent random variables, none of the variables in this distribution are (marginally) independent.

3.1.3.1 The Student Example

Elaborating our example, we now assume that the company also has access to the student's grade G in some course. In this case, our probability space is the joint distribution over the three relevant random variables I, S, and G. Assuming that I and S are as before, and that G takes on three values g^1, g^2, g^3, representing the grades A, B, and C, respectively, then the joint distribution has twelve entries.

Before we even consider the specific numerical aspects of our distribution P in this example, we can see that independence does not help us: for any reasonable P, there are no independencies that hold. The student's intelligence is clearly correlated both with his SAT score and with his grade. The SAT score and grade are also not independent: if we condition on the fact that the student received a high score on his SAT, the chances that he gets a high grade in his class are also likely to increase. Thus, we may assume that, for our particular distribution P, $P(g^1 \mid s^1) > P(g^1 \mid s^0)$.

However, it is quite plausible that our distribution P in this case satisfies a conditional independence property. If we know that the student has high intelligence, a high grade on the SAT no longer gives us information about the student's performance in the class. More formally:

$$P(g \mid i^1, s^1) = P(g \mid i^1).$$

More generally, we may well assume that

$$P \models (S \perp G \mid I). \tag{3.4}$$

Note that this independence statement holds only if we assume that the student's intelligence is the only reason why his grade and SAT score might be correlated. In other words, it assumes that there are no correlations due to other factors, such as the student's ability to take timed exams. These assumptions are also not "true" in any formal sense of the word, and they are often only approximations of our true beliefs. (See box 3.C for some further discussion.)

As in the case of marginal independence, conditional independence allows us to provide a compact specification of the joint distribution. Again, the compact representation is based on a very natural alternative parameterization. By simple probabilistic reasoning (as in equation (2.5)), we have that

$$P(I, S, G) = P(S, G \mid I)P(I).$$

But now, the conditional independence assumption of equation (3.4) implies that

$$P(S, G \mid I) = P(S \mid I)P(G \mid I).$$

Hence, we have that

$$P(I, S, G) = P(S \mid I)P(G \mid I)P(I). \tag{3.5}$$

Thus, we have factorized the joint distribution $P(I, S, G)$ as a product of three conditional probability distributions (CPDs). This factorization immediately leads us to the desired alternative parameterization. In order to specify fully a joint distribution satisfying equation (3.4), we need the following three CPDs: $P(I)$, $P(S \mid I)$, and $P(G \mid I)$. The first two might be the same as in equation (3.3). The latter might be

I	g^1	g^2	g^3
i^0	0.2	0.34	0.46
i^1	0.74	0.17	0.09

Together, these three CPDs fully specify the joint distribution (assuming the conditional independence of equation (3.4)). For example,

$$\begin{aligned} P(i^1, s^1, g^2) &= P(i^1)P(s^1 \mid i^1)P(g^2 \mid i^1) \\ &= 0.3 \cdot 0.8 \cdot 0.17 = 0.0408. \end{aligned}$$

Once again, we note that this probabilistic model would be represented using the Bayesian network shown in figure 3.1b.

In this case, the alternative parameterization is more compact than the joint. We now have three binomial distributions — $P(I)$, $P(S \mid i^1)$ and $P(S \mid i^0)$, and two three-valued multinomial distributions — $P(G \mid i^1)$ and $P(G \mid i^0)$. Each of the binomials requires one independent parameter, and each three-valued multinomial requires two independent parameters, for a total of seven. By contrast, our joint distribution has twelve entries, so that eleven independent parameters are required to specify an arbitrary joint distribution over these three variables.

It is important to note another advantage of this way of representing the joint: modularity. When we added the new variable G, the joint distribution changed entirely. Had we used the explicit representation of the joint, we would have had to write down twelve new numbers. In the factored representation, we could reuse our local probability models for the variables I and S, and specify only the probability model for G — the CPD $P(G \mid I)$. This property will turn out to be invaluable in modeling real-world systems.

3.1.3.2 The General Model

naive Bayes

This example is an instance of a much more general model commonly called the *naive Bayes*

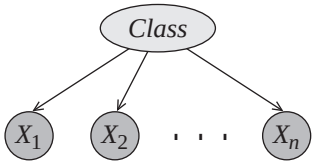


Figure 3.2 The Bayesian network graph for a naive Bayes model

model (also known as the *Idiot Bayes model*). The naive Bayes model assumes that instances fall into one of a number of mutually exclusive and exhaustive *classes*. Thus, we have a class variable C that takes on values in some set $\{c^1, \ldots, c^k\}$. In our example, the class variable is the student's intelligence I, and there are two classes of instances — students with high intelligence and students with low intelligence.

features

The model also includes some number of *features* $X_1, \ldots, X_n$ whose values are typically observed. The *naive Bayes assumption* is that the features are conditionally independent given the instance's class. In other words, within each class of instances, the different properties can be determined independently. More formally, we have that

$$(X_i \perp \boldsymbol{X}_{-i} \mid C) \quad \text{for all } i, \tag{3.6}$$

where $\boldsymbol{X}_{-i} = \{X_1, \ldots, X_n\} - \{X_i\}$. This model can be represented using the Bayesian network of figure 3.2. In this example, and later on in the book, we use a darker oval to represent variables that are always observed when the network is used.

factorization

Based on these independence assumptions, we can show that the model *factorizes* as:

$$P(C, X_1, \ldots, X_n) = P(C) \prod_{i=1}^{n} P(X_i \mid C). \tag{3.7}$$

(See exercise 3.2.) Thus, in this model, we can represent the joint distribution using a small set of factors: a prior distribution $P(C)$, specifying how likely an instance is to belong to different classes a priori, and a set of CPDs $P(X_j \mid C)$, one for each of the n finding variables. These factors can be encoded using a very small number of parameters. For example, if all of the variables are binary, the number of independent parameters required to specify the distribution is $2n + 1$ (see exercise 3.6). Thus, the number of parameters is linear in the number of variables, as opposed to exponential for the explicit representation of the joint.

Box 3.A — Concept: The Naive Bayes Model. *The naive Bayes model, despite the strong assumptions that it makes, is often used in practice, because of its simplicity and the small number of parameters required. The model is generally used for* classification *— deciding, based on the values of the evidence variables for a given instance, the class to which the instance is most likely to belong. We might also want to compute our confidence in this decision, that is, the extent to which our model favors one class c^1 over another c^2. Both queries can be addressed by the following ratio:*

classification

$$\frac{P(C = c^1 \mid x_1, \ldots, x_n)}{P(C = c^2 \mid x_1, \ldots, x_n)} = \frac{P(C = c^1)}{P(C = c^2)} \prod_{i=1}^{n} \frac{P(x_i \mid C = c^1)}{P(x_i \mid C = c^2)}; \tag{3.8}$$

see exercise 3.2). This formula is very natural, since it computes the posterior probability ratio of c^1 *versus* c^2 *as a product of their prior probability ratio (the first term), multiplied by a set of terms* $\frac{P(x_i|C=c^1)}{P(x_i|C=c^2)}$ *that measure the relative support of the finding* x_i *for the two classes.*

medical diagnosis

This model was used in the early days of medical diagnosis, *where the different values of the class variable represented different diseases that the patient could have. The evidence variables represented different symptoms, test results, and the like. Note that the model makes several strong assumptions that are not generally true, specifically that the patient can have at most one disease, and that, given the patient's disease, the presence or absence of different symptoms, and the values of different tests, are all independent. This model was used for medical diagnosis because the small number of interpretable parameters made it easy to elicit from experts. For example, it is quite natural to ask of an expert physician what the probability is that a patient with pneumonia has high fever. Indeed, several early medical diagnosis systems were based on this technology, and some were shown to provide better diagnoses than those made by expert physicians.*

However, later experience showed that the strong assumptions underlying this model decrease its diagnostic accuracy. In particular, the model tends to overestimate the impact of certain evidence by "overcounting" it. For example, both hypertension (high blood pressure) and obesity are strong indicators of heart disease. However, because these two symptoms are themselves highly correlated, equation (3.8), which contains a multiplicative term for each of them, double-counts the evidence they provide about the disease. Indeed, some studies showed that the diagnostic performance of a naive Bayes model degraded as the number of features increased; this degradation was often traced to violations of the strong conditional independence assumption. This phenomenon led to the use of more complex Bayesian networks, with more realistic independence assumptions, for this application (see box 3.D).

Nevertheless, the naive Bayes model is still useful in a variety of applications, particularly in the context of models learned from data in domains with a large number of features and a relatively small number of instances, such as classifying documents into topics using the words in the documents as features; see box 17.E).

3.2 Bayesian Networks

Bayesian networks build on the same intuitions as the naive Bayes model by exploiting conditional independence properties of the distribution in order to allow a compact and natural representation. However, they are not restricted to representing distributions satisfying the strong independence assumptions implicit in the naive Bayes model. They allow us the flexibility to tailor our representation of the distribution to the independence properties that appear reasonable in the current setting.

The core of the Bayesian network representation is a directed acyclic graph (DAG) $\mathcal{G}$, whose nodes are the random variables in our domain and whose edges correspond, intuitively, to direct influence of one node on another. **This graph $\mathcal{G}$ can be viewed in two very different ways:**

- **as a data structure that provides the skeleton for representing a joint distribution compactly in a factorized way;**

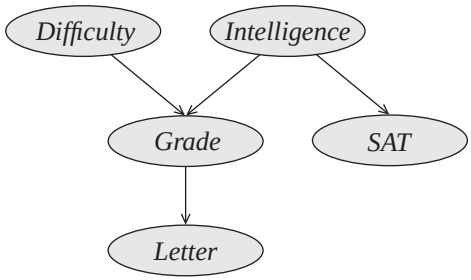


Figure 3.3 The Bayesian Network graph for the Student **example**

- **as a compact representation for a set of conditional independence assumptions about a distribution.**

As we will see, these two views are, in a strong sense, equivalent.

3.2.1 The Student Example Revisited

We begin our discussion with a simple toy example, which will accompany us, in various versions, throughout much of this book.

3.2.1.1 The Model

Consider our student from before, but now consider a slightly more complex scenario. The student's grade, in this case, depends not only on his intelligence but also on the difficulty of the course, represented by a random variable D whose domain is $Val(D) = \{easy, hard\}$. Our student asks his professor for a recommendation letter. The professor is absentminded and never remembers the names of her students. She can only look at his grade, and she writes her letter for him based on that information alone. The quality of her letter is a random variable L, whose domain is $Val(L) = \{strong, weak\}$. The actual quality of the letter depends stochastically on the grade. (It can vary depending on how stressed the professor is and the quality of the coffee she had that morning.)

We therefore have five random variables in this domain: the student's intelligence (I), the course difficulty (D), the grade (G), the student's SAT score (S), and the quality of the recommendation letter (L). All of the variables except G are binary-valued, and G is ternary-valued. Hence, the joint distribution has 48 entries.

As we saw in our simple illustrations of figure 3.1, a Bayesian network is represented using a directed graph whose nodes represent the random variables and whose edges represent direct influence of one variable on another. We can view the graph as encoding a generative sampling process executed by nature, where the value for each variable is selected by nature using a distribution that depends only on its parents. In other words, each variable is a stochastic function of its parents.

Based on this intuition, perhaps the most natural network structure for the distribution in this example is the one presented in figure 3.3. The edges encode our intuition about

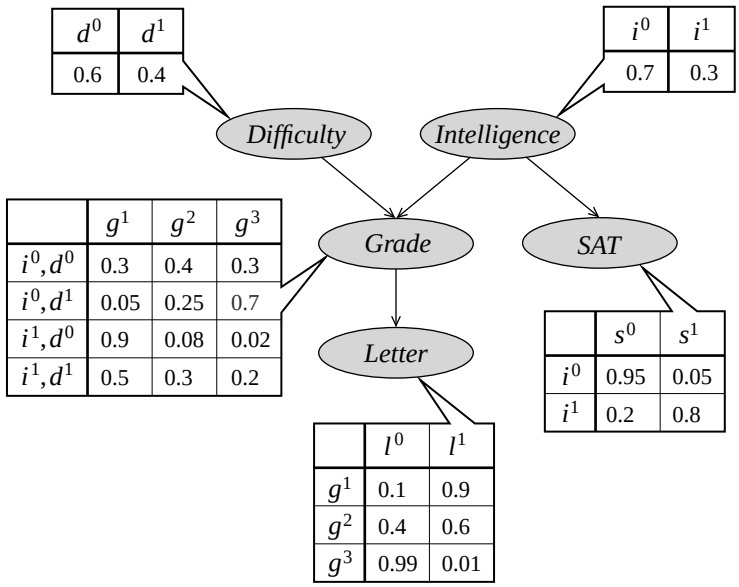


Figure 3.4 Student **Bayesian network $\mathcal{B}^{student}$ with CPDs**

the way the world works. The course difficulty and the student's intelligence are determined independently, and before any of the variables in the model. The student's grade depends on both of these factors. The student's SAT score depends only on his intelligence. The quality of the professor's recommendation letter depends (by assumption) only on the student's grade in the class. Intuitively, each variable in the model depends directly only on its parents in the network. We formalize this intuition later.

local probability model

The second component of the Bayesian network representation is a set of *local probability models* that represent the nature of the dependence of each variable on its parents. One such model, $P(I)$, represents the distribution in the population of intelligent versus less intelligent student. Another, $P(D)$, represents the distribution of difficult and easy classes. The distribution over the student's grade is a conditional distribution $P(G \mid I, D)$. It specifies the distribution over the student's grade, inasmuch as it depends on the student's intelligence and the course difficulty. Specifically, we would have a different distribution for each assignment of values i, d. For example, we might believe that a smart student in an easy class is 90 percent likely to get an A, 8 percent likely to get a B, and 2 percent likely to get a C. Conversely, a smart student in a hard class may only be 50 percent likely to get an A. In general, each variable X in the model is associated with a *conditional probability distribution (CPD)* that specifies a distribution over the values of X given each possible joint assignment of values to its parents in the model. For a node with no parents, the CPD is conditioned on the empty set of variables. Hence, the CPD turns into a marginal distribution, such as $P(D)$ or $P(I)$. One possible choice of CPDs for this domain is shown in figure 3.4. The network structure together with its CPDs is a *Bayesian network* $\mathcal{B}$; we use $\mathcal{B}^{student}$ to refer to the Bayesian network for our student example.

CPD

How do we use this data structure to specify the joint distribution? Consider some particular state in this space, for example, i^1, d^0, g^2, s^1, l^0. Intuitively, the probability of this event can be computed from the probabilities of the basic events that comprise it: the probability that the student is intelligent; the probability that the course is easy; the probability that a smart student gets a B in an easy class; the probability that a smart student gets a high score on his SAT; and the probability that a student who got a B in the class gets a weak letter. The total probability of this state is:

$$\begin{aligned} P(i^1, d^0, g^2, s^1, l^0) &= P(i^1)P(d^0)P(g^2 \mid i^1, d^0)P(s^1 \mid i^1)P(l^0 \mid g^2) \\ &= 0.3 \cdot 0.6 \cdot 0.08 \cdot 0.8 \cdot 0.4 = 0.004608. \end{aligned}$$

Clearly, we can use the same process for any state in the joint probability space. In general, we will have that

$$P(I, D, G, S, L) = P(I)P(D)P(G \mid I, D)P(S \mid I)P(L \mid G). \tag{3.9}$$

chain rule for Bayesian networks

This equation is our first example of the *chain rule for Bayesian networks* which we will define in a general setting in section 3.2.3.2.

3.2.1.2 Reasoning Patterns

A joint distribution $P_{\mathcal{B}}$ specifies (albeit implicitly) the probability $P_{\mathcal{B}}(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{E} = \boldsymbol{e})$ of any event $\boldsymbol{y}$ given any observations $\boldsymbol{e}$, as discussed in section 2.1.3.3: We condition the joint distribution on the event $\boldsymbol{E} = \boldsymbol{e}$ by eliminating the entries in the joint inconsistent with our observation $\boldsymbol{e}$, and renormalizing the resulting entries to sum to 1; we compute the probability of the event $\boldsymbol{y}$ by summing the probabilities of all of the entries in the resulting posterior distribution that are consistent with $\boldsymbol{y}$. To illustrate this process, let us consider our $\mathcal{B}^{student}$ network and see how the probabilities of various events change as evidence is obtained.

Consider a particular student, George, about whom we would like to reason using our model. We might ask how likely George is to get a strong recommendation (l^1) from his professor in Econ101. Knowing nothing else about George or Econ101, this probability is about 50.2 percent. More precisely, let $P_{\mathcal{B}^{student}}$ be the joint distribution defined by the preceding BN; then we have that $P_{\mathcal{B}^{student}}(l^1) \approx 0.502$. We now find out that George is not so intelligent (i^0); the probability that he gets a strong letter from the professor of Econ101 goes down to around 38.9 percent; that is, $P_{\mathcal{B}^{student}}(l^1 \mid i^0) \approx 0.389$. We now further discover that Econ101 is an easy class (d^0). The probability that George gets a strong letter from the professor is now $P_{\mathcal{B}^{student}}(l^1 \mid i^0, d^0) \approx 0.513$. Queries such as these, where we predict the "downstream" effects of various factors (such as George's intelligence), are instances of *causal reasoning* or *prediction*.

causal reasoning

Now, consider a recruiter for Acme Consulting, trying to decide whether to hire George based on our previous model. A priori, the recruiter believes that George is 30 percent likely to be intelligent. He obtains George's grade record for a particular class Econ101 and sees that George received a C in the class (g^3). His probability that George has high intelligence goes down significantly, to about 7.9 percent; that is, $P_{\mathcal{B}^{student}}(i^1 \mid g^3) \approx 0.079$. We note that the probability that the class is a difficult one also goes up, from 40 percent to 62.9 percent.

Now, assume that the recruiter fortunately (for George) lost George's transcript, and has only the recommendation letter from George's professor in Econ101, which (not surprisingly) is

weak. The probability that George has high intelligence still goes down, but only to 14 percent: $P_{\mathcal{B}^{student}}(i^1 \mid l^0) \approx 0.14$. Note that if the recruiter has both the grade and the letter, we have the same probability as if he had only the grade: $P_{\mathcal{B}^{student}}(i^1 \mid g^3, l^0) \approx 0.079$; we will revisit this issue. Queries such as this, where we reason from effects to causes, are instances of *evidential reasoning* or *explanation*.

evidential reasoning

Finally, George submits his SAT scores to the recruiter, and astonishingly, his SAT score is high. The probability that George has high intelligence goes up dramatically, from 7.9 percent to 57.8 percent: $P_{\mathcal{B}^{student}}(i^1 \mid g^3, s^1) \approx 0.578$. Intuitively, the reason that the high SAT score outweighs the poor grade is that students with low intelligence are extremely unlikely to get good scores on their SAT, whereas students with high intelligence can still get C's. However, smart students are much more likely to get C's in hard classes. Indeed, we see that the probability that Econ101 is a difficult class goes up from the 62.9 percent we saw before to around 76 percent.

This last pattern of reasoning is a particularly interesting one. The information about the SAT gave us information about the student's intelligence, which, in conjunction with the student's grade in the course, told us something about the difficulty of the course. In effect, we have one causal factor for the *Grade* variable — *Intelligence* — giving us information about another — *Difficulty*.

Let us examine this pattern in its pure form. As we said, $P_{\mathcal{B}^{student}}(i^1 \mid g^3) \approx 0.079$. On the other hand, if we now discover that Econ101 is a hard class, we have that $P_{\mathcal{B}^{student}}(i^1 \mid g^3, d^1) \approx 0.11$. In effect, we have provided at least a partial explanation for George's grade in Econ101. To take an even more striking example, if George gets a B in Econ 101, we have that $P_{\mathcal{B}^{student}}(i^1 \mid g^2) \approx 0.175$. On the other hand, if Econ101 is a hard class, we get $P_{\mathcal{B}^{student}}(i^1 \mid g^2, d^1) \approx 0.34$. In effect we have *explained away* the poor grade via the difficulty of the class. **Explaining away is an instance of a general reasoning pattern called *intercausal reasoning*, where different causes of the same effect can interact. This type of reasoning is a very common pattern in human reasoning.** For example, when we have fever and a sore throat, and are concerned about mononucleosis, we are greatly relieved to be told we have the flu. Clearly, having the flu does not prohibit us from having mononucleosis. Yet, having the flu provides an alternative explanation of our symptoms, thereby reducing substantially the probability of mononucleosis.

explaining away

intercausal reasoning

This intuition of providing an alternative explanation for the evidence can be made very precise. As shown in exercise 3.3, if the flu deterministically causes the symptoms, the probability of mononucleosis goes down to its prior probability (the one prior to the observations of any symptoms). On the other hand, if the flu might occur without causing these symptoms, the probability of mononucleosis goes down, but it still remains somewhat higher than its base level. Explaining away, however, is not the only form of intercausal reasoning. The influence can go in any direction. Consider, for example, a situation where someone is found dead and may have been murdered. The probabilities that a suspect has motive and opportunity both go up. If we now discover that the suspect has motive, the probability that he has opportunity goes up. (See exercise 3.4.)

It is important to emphasize that, although our explanations used intuitive concepts such as cause and evidence, there is nothing mysterious about the probability computations we performed. They can be replicated simply by generating the joint distribution, as defined in equation (3.9), and computing the probabilities of the various events directly from that.

3.2.2 Basic Independencies in Bayesian Networks

As we discussed, a Bayesian network graph $\mathcal{G}$ can be viewed in two ways. In the previous section, we showed, by example, how it can be used as a skeleton data structure to which we can attach local probability models that together define a joint distribution. In this section, we provide a formal semantics for a Bayesian network, starting from the perspective that the graph encodes a set of conditional independence assumptions. We begin by understanding, intuitively, the basic conditional independence assumptions that we want a directed graph to encode. We then formalize these desired assumptions in a definition.

3.2.2.1 Independencies in the Student Example

In the Student example, we used the intuition that edges represent direct dependence. For example, we made intuitive statements such as "the professor's recommendation letter depends only on the student's grade in the class"; this statement was encoded in the graph by the fact that there are no direct edges into the L node except from G. This intuition, that "a node depends directly only on its parents," lies at the heart of the semantics of Bayesian networks.

We give formal semantics to this assertion using conditional independence statements. For example, the previous assertion can be stated formally as the assumption that L is conditionally independent of all other nodes in the network given its parent G:

$$(L \perp I, D, S \mid G). \tag{3.10}$$

In other words, once we know the student's grade, our beliefs about the quality of his recommendation letter are not influenced by information about any other variable. Similarly, to formalize our intuition that the student's SAT score depends only on his intelligence, we can say that S is conditionally independent of all other nodes in the network given its parent I:

$$(S \perp D, G, L \mid I). \tag{3.11}$$

Now, let us consider the G node. Following the pattern blindly, we may be tempted to assert that G is conditionally independent of all other variables in the network given its parents. However, this assumption is false both at an intuitive level and for the specific example distribution we used earlier. Assume, for example, that we condition on i^1, d^1; that is, we have a smart student in a difficult class. In this setting, is G independent of L? Clearly, the answer is no: if we observe l^1 (the student got a strong letter), then our probability in g^1 (the student received an A in the course) should go up; that is, we would expect

$$P(g^1 \mid i^1, d^1, l^1) > P(g^1 \mid i^1, d^1).$$

Indeed, if we examine our distribution, the latter probability is 0.5 (as specified in the CPD), whereas the former is a much higher 0.712.

Thus, we see that we do not expect a node to be conditionally independent of all other nodes given its parents. In particular, even given its parents, it can still depend on its descendants. Can it depend on other nodes? For example, do we expect G to depend on S given I and D? Intuitively, the answer is no. Once we know, say, that the student has high intelligence, his SAT score gives us no additional information that is relevant toward predicting his grade. Thus, we

would want the property that:

$$(G \perp S \mid I, D). \tag{3.12}$$

It remains only to consider the variables I and D, which have no parents in the graph. Thus, in our search for independencies given a node's parents, we are now looking for marginal independencies. As the preceding discussion shows, in our distribution $P_{\mathcal{B}^{student}}$, I is not independent of its descendants G, L, or S. Indeed, the only nondescendant of I is D. Indeed, we assumed implicitly that *Intelligence* and *Difficulty* are independent. Thus, we expect that:

$$(I \perp D). \tag{3.13}$$

This analysis might seem somewhat surprising in light of our earlier examples, where learning something about the course difficulty drastically changed our beliefs about the student's intelligence. In that situation, however, we were reasoning in the presence of information about the student's grade. In other words, we were demonstrating the dependence of I and D *given* G. This phenomenon is a very important one, and we will return to it.

For the variable D, both I and S are nondescendants. Recall that, if $(I \perp D)$ then $(D \perp I)$. The variable S increases our beliefs in the student's intelligence, but knowing that the student is smart (or not) does not influence our beliefs in the difficulty of the course. Thus, we have that

$$(D \perp I, S). \tag{3.14}$$

We can see a pattern emerging. Our intuition tells us that the parents of a variable "shield" it from probabilistic influence that is causal in nature. In other words, once I know the value of the parents, no information relating directly or indirectly to its parents or other ancestors can influence my beliefs about it. However, information about its descendants *can* change my beliefs about it, via an evidential reasoning process.

3.2.2.2 Bayesian Network Semantics

We are now ready to provide the formal definition of the semantics of a Bayesian network structure. We would like the formal definition to match the intuitions developed in our example.

Definition 3.1 Bayesian network structure

local independencies

A Bayesian network structure $\mathcal{G}$ *is a directed acyclic graph whose nodes represent random variables* $X_1, \ldots, X_n$. *Let* $\mathrm{Pa}_{X_i}^{\mathcal{G}}$ *denote the parents of* X_i *in* $\mathcal{G}$, *and* $\mathrm{NonDescendants}_{X_i}$ *denote the variables in the graph that are not descendants of* X_i. *Then* $\mathcal{G}$ *encodes the following set of conditional independence assumptions, called the* local independencies, *and denoted by* $\mathcal{I}_\ell(\mathcal{G})$*:*

For each variable X_i*:* $(X_i \perp \mathrm{NonDescendants}_{X_i} \mid \mathrm{Pa}_{X_i}^{\mathcal{G}})$. ■

In other words, the local independencies state that each node X_i is conditionally independent of its nondescendants given its parents.

Returning to the Student network $G_{student}$, the local Markov independencies are precisely the ones dictated by our intuition, and specified in equation (3.10) – equation (3.14).

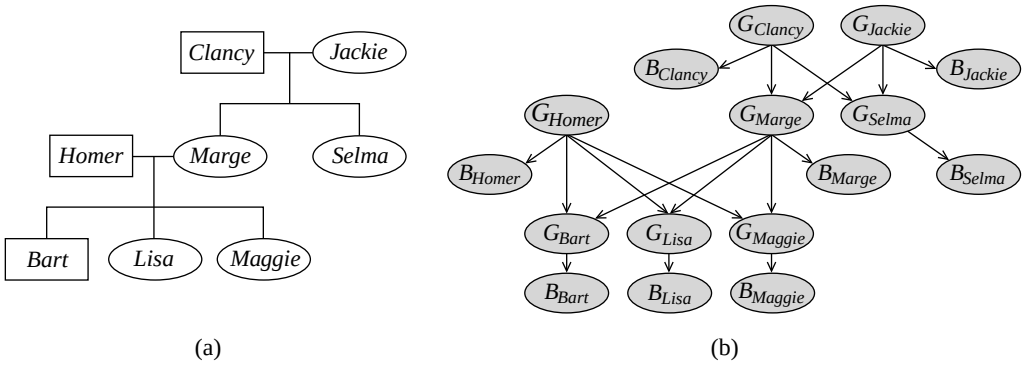


Figure 3.B.1 — Modeling Genetic Inheritance (a) A small family tree. (b) A simple BN for genetic inheritance in this domain. The G variables represent a person's genotype, and the B variables the result of a blood-type test.

Box 3.B — Case Study: The Genetics **Example.** *One of the very earliest uses of a Bayesian network model (long before the general framework was defined) is in the area of genetic pedigrees. In this setting, the local independencies are particularly intuitive. In this application, we want to model the transmission of a certain property, say blood type, from parent to child. The blood type of a person is an observable quantity that depends on her genetic makeup. Such properties are called* phenotypes. *The genetic makeup of a person is called* genotype.

To model this scenario properly, we need to introduce some background on genetics. The human genetic material consists of 22 pairs of autosomal *chromosomes and a pair of the* sex *chromosomes (X and Y). Each chromosome contains a set of genetic material, consisting (among other things) of genes that determine a person's properties. A region of the chromosome that is of interest is called a* locus*; a locus can have several variants, called* alleles.

For concreteness, we focus on autosomal chromosome pairs. In each autosomal pair, one chromosome is the paternal *chromosome, inherited from the father, and the other is the* maternal *chromosome, inherited from the mother. For genes in an autosomal pair, a person has two copies of the gene, one on each copy of the chromosome. Thus, one of the gene's alleles is inherited from the person's mother, and the other from the person's father. For example, the region containing the gene that encodes a person's blood type is a locus. This gene comes in three variants, or alleles: A, B, and O. Thus, a person's genotype is denoted by an ordered pair, such as $\langle A, B \rangle$; with three choices for each entry in the pair, there are 9 possible genotypes. The blood type phenotype is a function of both copies of the gene. For example, if the person has an A allele and an O allele, her observed blood type is "A." If she has two O alleles, her observed blood type is "O."*

To represent this domain, we would have, for each person, two variables: one representing the person's genotype, and the other her phenotype. We use the name $G(p)$ to represent person p's genotype, and $B(p)$ to represent her blood type.

In this example, the independence assumptions arise immediately from the biology. Since the

blood type is a function of the genotype, once we know the genotype of a person, additional evidence about other members of the family will not provide new information about the blood type. Similarly, the process of genetic inheritance implies independence assumption. Once we know the genotype of both parents, we know what each of them can pass on to the offspring. Thus, learning new information about ancestors (or nondescendants) does not provide new information about the genotype of the offspring. These are precisely the local independencies in the resulting network structure, shown for a simple family tree in figure 3.B.1. The intuition here is clear; for example, Bart's blood type is correlated with that of his aunt Selma, but once we know Homer's and Marge's genotype, the two become independent.

To define the probabilistic model fully, we need to specify the CPDs. There are three types of CPDs in this model:

- *The* penetrance model $P(B(c) \mid G(c))$*, which describes the probability of different variants of a particular phenotype (say different blood types) given the person's genotype. In the case of the blood type, this CPD is a deterministic function, but in other cases, the dependence can be more complex.*
- *The* transmission model $P(G(c) \mid G(p), G(m))$*, where c is a person and p, m her father and mother, respectively. Each parent is equally likely to transmit either of his or her two alleles to the child.*
- Genotype priors $P(G(c))$*, used when person c has no parents in the pedigree. These are the general genotype frequencies within the population.*

Our discussion of blood type is simplified for several reasons. First, some phenotypes, such as late-onset diseases, are not a deterministic function of the genotype. Rather, an individual with a particular genotype might be more likely to have the disease than an individual with other genotypes. Second, the genetic makeup of an individual is defined by many genes. Some phenotypes might depend on multiple genes. In other settings, we might be interested in multiple phenotypes, which (naturally) implies a dependence on several genes. Finally, as we now discuss, the inheritance patterns of different genes are not independent of each other.

Recall that each of the person's autosomal chromosomes is inherited from one of her parents. However, each of the parents also has two copies of each autosomal chromosome. These two copies, within each parent, recombine to produce the chromosome that is transmitted to the child. Thus, the maternal chromosome inherited by Bart is a combination of the chromosomes inherited by his mother Marge from her mother Jackie and her father Clancy. The recombination process is stochastic, but only a handful recombination events take place within a chromosome in a single generation. Thus, if Bart inherited the allele for some locus from the chromosome his mother inherited from her mother Jackie, he is also much more likely to inherit Jackie's copy for a nearby locus. Thus, to construct an appropriate model for multilocus inheritance, we must take into consideration the probability of a recombination taking place between pairs of adjacent loci.

We can facilitate this modeling by introducing selector variables *that capture the inheritance pattern along the chromosome. In particular, for each locus ℓ and each child c, we have a variable $S(\ell, c, m)$ that takes the value 1 if the locus ℓ in c's maternal chromosome was inherited from c's maternal grandmother, and 2 if this locus was inherited from c's maternal grandfather. We have a similar selector variable $S(\ell, c, p)$ for c's paternal chromosome. We can now model correlations induced by low recombination frequency by correlating the variables $S(\ell, c, m)$ and $S(\ell', c, m)$ for adjacent loci ℓ, ℓ'.*

This type of model has been used extensively for many applications. In genetic counseling and prediction, one takes a phenotype with known loci and a set of observed phenotype and genotype data for some individuals in the pedigree to infer the genotype and phenotype for another person in the pedigree (say, a planned child). The genetic data can consist of direct measurements of the relevant disease loci (for some individuals) or measurements of nearby loci, which are correlated with the disease loci.

In linkage analysis, the task is a harder one: identifying the location of disease genes from pedigree data using some number of pedigrees where a large fraction of the individuals exhibit a disease phenotype. Here, the available data includes phenotype information for many individuals in the pedigree, as well as genotype information for loci whose location in the chromosome is known. Using the inheritance model, the researchers can evaluate the likelihood of these observations under different hypotheses about the location of the disease gene relative to the known loci. By repeated calculation of the probabilities in the network for different hypotheses, researchers can pinpoint the area that is "linked" to the disease. This much smaller region can then be used as the starting point for more detailed examination of genes in that area. This process is crucial, for it can allow the researchers to focus on a small area (for example, $1/10,000$ *of the genome).*

As we will see in later chapters, the ability to describe the genetic inheritance process using a sparse Bayesian network provides us the capability to use sophisticated inference algorithms that allow us to reason about large pedigrees and multiple loci. It also allows us to use algorithms for model learning to obtain a deeper understanding of the genetic inheritance process, such as recombination rates in different regions or penetrance probabilities for different diseases.

3.2.3 Graphs and Distributions

The formal semantics of a Bayesian network graph is as a set of independence assertions. On the other hand, our Student BN was a graph annotated with CPDs, which defined a joint distribution via the chain rule for Bayesian networks. In this section, we show that these two definitions are, in fact, equivalent. A distribution P satisfies the local independencies associated with a graph $\mathcal{G}$ if and only if P is representable as a set of CPDs associated with the graph $\mathcal{G}$. We begin by formalizing the basic concepts.

3.2.3.1 I-Maps

We first define the set of independencies associated with a distribution P.

independencies in P

Definition 3.2 *Let P be a distribution over $\mathcal{X}$. We define $\mathcal{I}(P)$ to be the set of* independence assertions *of the form $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ that hold in P.* ■

We can now rewrite the statement that "P satisfies the local independencies associated with $\mathcal{G}$" simply as $\mathcal{I}_\ell(\mathcal{G}) \subseteq \mathcal{I}(P)$. In this case, we say that $\mathcal{G}$ is an *I-map* (independency map) for P. However, it is useful to define this concept more broadly, since different variants of it will be used throughout the book.

I-map

Definition 3.3 *Let $\mathcal{K}$ be any graph object associated with a set of independencies $\mathcal{I}(\mathcal{K})$. We say that $\mathcal{K}$ is an* I-map *for a set of independencies $\mathcal{I}$ if $\mathcal{I}(\mathcal{K}) \subseteq \mathcal{I}$.* ■

We now say that $\mathcal{G}$ is an I-map for P if $\mathcal{G}$ is an I-map for $\mathcal{I}(P)$.

As we can see from the direction of the inclusion, for $\mathcal{G}$ to be an I-map of P, it is necessary that $\mathcal{G}$ does not mislead us regarding independencies in P: any independence that $\mathcal{G}$ asserts must also hold in P. Conversely, P may have additional independencies that are not reflected in $\mathcal{G}$.

Let us illustrate the concept of an I-map on a very simple example.

Example 3.1

Consider a joint probability space over two independent random variables X and Y. There are three possible graphs over these two nodes: $\mathcal{G}_\emptyset$, which is a disconnected pair $X \quad Y$; $\mathcal{G}_{X \to Y}$, which has the edge $X \to Y$; and $\mathcal{G}_{Y \to X}$, which contains $Y \to X$. The graph $\mathcal{G}_\emptyset$ encodes the assumption that $(X \perp Y)$. The latter two encode no independence assumptions.

Consider the following two distributions:

X	Y	$P(X,Y)$
x^0	y^0	0.08
x^0	y^1	0.32
x^1	y^0	0.12
x^1	y^1	0.48

X	Y	$P(X,Y)$
x^0	y^0	0.4
x^0	y^1	0.3
x^1	y^0	0.2
x^1	y^1	0.1

In the example on the left, X and Y are independent in P; for example, $P(x^1) = 0.48 + 0.12 = 0.6$, $P(y^1) = 0.8$, and $P(x^1, y^1) = 0.48 = 0.6 \cdot 0.8$. Thus, $(X \perp Y) \in \mathcal{I}(P)$, and we have that $\mathcal{G}_\emptyset$ is an I-map of P. In fact, all three graphs are I-maps of P: $\mathcal{I}_\ell(\mathcal{G}_{X \to Y})$ is empty, so that trivially P satisfies all the independencies in it (similarly for $\mathcal{G}_{Y \to X}$). In the example on the right, $(X \perp Y) \notin \mathcal{I}(P)$, so that $\mathcal{G}_\emptyset$ is not an I-map of P. Both other graphs are I-maps of P. ■

3.2.3.2 I-Map to Factorization

A BN structure $\mathcal{G}$ encodes a set of conditional independence assumptions; every distribution for which $\mathcal{G}$ is an I-map must satisfy these assumptions. This property is the key to allowing the compact factorized representation that we saw in the Student example in section 3.2.1. The basic principle is the same as the one we used in the naive Bayes decomposition in section 3.1.3.

Consider any distribution P for which our Student BN $G_{student}$ is an I-map. We will decompose the joint distribution and show that it factorizes into local probabilistic models, as in section 3.2.1. Consider the joint distribution $P(I, D, G, L, S)$; from the chain rule for probabilities (equation (2.5)), we can decompose this joint distribution in the following way:

$$P(I, D, G, L, S) = P(I)P(D \mid I)P(G \mid I, D)P(L \mid I, D, G)P(S \mid I, D, G, L). \tag{3.15}$$

This transformation relies on no assumptions; it holds for any joint distribution P. However, it is also not very helpful, since the conditional probabilities in the factorization on the right-hand side are neither natural nor compact. For example, the last factor requires the specification of 24 conditional probabilities: $P(s^1 \mid i, d, g, l)$ for every assignment of values i, d, g, l.

This form, however, allows us to apply the conditional independence assumptions induced from the BN. Let us assume that $G_{student}$ is an I-map for our distribution P. In particular, from equation (3.13), we have that $(D \perp I) \in \mathcal{I}(P)$. From that, we can conclude that $P(D \mid I) = P(D)$, allowing us to simplify the second factor on the right-hand side. Similarly, we know from

equation (3.10) that $(L \perp I, D \mid G) \in \mathcal{I}(P)$. Hence, $P(L \mid I, D, G) = P(L \mid G)$, allowing us to simplify the third term. Using equation (3.11) in a similar way, we obtain that

$$P(I, D, G, L, S) = P(I)P(D)P(G \mid I, D)P(L \mid G)P(S \mid I). \tag{3.16}$$

This factorization is precisely the one we used in section 3.2.1.

This result tells us that any entry in the joint distribution can be computed as a product of factors, one for each variable. Each factor represents a conditional probability of the variable given its parents in the network. This factorization applies to *any* distribution P for which $G_{student}$ is an I-map.

We now state and prove this fundamental result more formally.

Definition 3.4 factorization

Let $\mathcal{G}$ be a BN graph over the variables $X_1, \ldots, X_n$. We say that a distribution P over the same space factorizes *according to $\mathcal{G}$ if P can be expressed as a product*

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathrm{Pa}_{X_i}^{\mathcal{G}}). \tag{3.17}$$

chain rule for Bayesian networks

This equation is called the chain rule for Bayesian networks. *The individual factors $P(X_i \mid \mathrm{Pa}_{X_i}^{\mathcal{G}})$ are called* conditional probability distributions (CPDs) *or* local probabilistic models. ■

CPD

Definition 3.5 Bayesian network

A Bayesian network *is a pair $\mathcal{B} = (\mathcal{G}, P)$ where P factorizes over $\mathcal{G}$, and where P is specified as a set of CPDs associated with $\mathcal{G}$'s nodes. The distribution P is often annotated $P_{\mathcal{B}}$.* ■

We can now prove that the phenomenon we observed for $G_{student}$ holds more generally.

Theorem 3.1

Let $\mathcal{G}$ be a BN structure over a set of random variables $\mathcal{X}$, and let P be a joint distribution over the same space. If $\mathcal{G}$ is an I-map for P, then P factorizes according to $\mathcal{G}$.

topological ordering

Proof Assume, without loss of generality, that $X_1, \ldots, X_n$ is a *topological ordering* of the variables in $\mathcal{X}$ relative to $\mathcal{G}$ (see definition 2.19). As in our example, we first use the chain rule for probabilities:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid X_1, \ldots, X_{i-1}).$$

Now, consider one of the factors $P(X_i \mid X_1, \ldots, X_{i-1})$. As $\mathcal{G}$ is an I-map for P, we have that $(X_i \perp \mathrm{NonDescendants}_{X_i} \mid \mathrm{Pa}_{X_i}^{\mathcal{G}}) \in \mathcal{I}(P)$. By assumption, all of X_i's parents are in the set $X_1, \ldots, X_{i-1}$. Furthermore, none of X_i's descendants can possibly be in the set. Hence,

$$\{X_1, \ldots, X_{i-1}\} = \mathrm{Pa}_{X_i} \cup \boldsymbol{Z}$$

where $\boldsymbol{Z} \subseteq \mathrm{NonDescendants}_{X_i}$. From the local independencies for X_i and from the decomposition property (equation (2.8)) it follows that $(X_i \perp \boldsymbol{Z} \mid \mathrm{Pa}_{X_i})$. Hence, we have that

$$P(X_i \mid X_1, \ldots, X_{i-1}) = P(X_i \mid \mathrm{Pa}_{X_i}).$$

Applying this transformation to all of the factors in the chain rule decomposition, the result follows. ■

Thus, the conditional independence assumptions implied by a BN structure $\mathcal{G}$ allow us to factorize a distribution P for which $\mathcal{G}$ is an I-map into small CPDs. Note that the proof is constructive, providing a precise algorithm for constructing the factorization given the distribution P and the graph $\mathcal{G}$.

The resulting factorized representation can be substantially more compact, particularly for sparse structures.

Example 3.2

In our Student *example, the number of independent parameters is fifteen: we have two binomial distributions $P(I)$ and $P(D)$, with one independent parameter each; we have four multinomial distributions over G — one for each assignment of values to I and D — each with two independent parameters; we have three binomial distributions over L, each with one independent parameter; and similarly two binomial distributions over S, each with an independent parameter. The specification of the full joint distribution would require $48 - 1 = 47$ independent parameters.* ■

More generally, in a distribution over n binary random variables, the specification of the joint distribution requires $2^n - 1$ independent parameters. If the distribution factorizes according to a graph $\mathcal{G}$ where each node has at most k parents, the total number of independent parameters required is less than $n \cdot 2^k$ (see exercise 3.6). In many applications, we can assume a certain locality of influence between variables: although each variable is generally correlated with many of the others, it often depends *directly* on only a small number of other variables. Thus, in many cases, k will be very small, even though n is large. As a consequence, the number of parameters in the Bayesian network representation is typically exponentially smaller than the number of parameters of a joint distribution. This property is one of the main benefits of the Bayesian network representation.

3.2.3.3 Factorization to I-Map

Theorem 3.1 shows one direction of the fundamental connection between the conditional independencies encoded by the BN structure and the factorization of the distribution into local probability models: that the conditional independencies imply factorization. The converse also holds: factorization according to $\mathcal{G}$ implies the associated conditional independencies.

Theorem 3.2

Let $\mathcal{G}$ be a BN structure over a set of random variables $\mathcal{X}$ and let P be a joint distribution over the same space. If P factorizes according to $\mathcal{G}$, then $\mathcal{G}$ is an I-map for P.

We illustrate this theorem by example, leaving the proof as an exercise (exercise 3.9). Let P be some distribution that factorizes according to $G_{\textit{student}}$. We need to show that $\mathcal{I}_\ell(G_{\textit{student}})$ holds in P. Consider the independence assumption for the random variable S — $(S \perp D, G, L \mid I)$. To prove that it holds for P, we need to show that

$$P(S \mid I, D, G, L) = P(S \mid I).$$

By definition,

$$P(S \mid I, D, G, L) = \frac{P(S, I, D, G, L)}{P(I, D, G, L)}.$$

By the chain rule for BNs equation (3.16), the numerator is equal to $P(I)P(D)P(G \mid I, D)P(L \mid G)P(S \mid I)$. By the process of marginalizing over a joint distribution, we have that the denominator is:

$$\begin{aligned} P(I, D, G, L) &= \sum_S P(I, D, G, L, S) \\ &= \sum_S P(I)P(D)P(G \mid I, D)P(L \mid G)P(S \mid I) \\ &= P(I)P(D)P(G \mid I, D)P(L \mid G) \sum_S P(S \mid I) \\ &= P(I)P(D)P(G \mid I, D)P(L \mid G), \end{aligned}$$

where the last step is a consequence of the fact that $P(S \mid I)$ is a distribution over values of S, and therefore it sums to 1. We therefore have that

$$\begin{aligned} P(S \mid I, D, G, L) &= \frac{P(S, I, D, G, L)}{P(I, D, G, L)} \\ &= \frac{P(I)P(D)P(G \mid I, D)P(L \mid G)P(S \mid I)}{P(I)P(D)P(G \mid I, D)P(L \mid G)} \\ &= P(S \mid I). \end{aligned}$$

Box 3.C — Skill: Knowledge Engineering. *Our discussion of Bayesian network construction focuses on the process of going from a given distribution to a Bayesian network. Real life is not like that. We have a vague model of the world, and we need to crystallize it into a network structure and parameters. This task breaks down into several components, each of which can be quite subtle. Unfortunately, modeling mistakes can have significant consequences for the quality of the answers obtained from the network, or to the cost of using the network in practice.*

Picking variables *When we model a domain, there are many possible ways to describe the relevant entities and their attributes. Choosing which random variables to use in the model is often one of the hardest tasks, and this decision has implications throughout the model. A common problem is using ill-defined variables. For example, deciding to include the variable Fever to describe a patient in a medical domain seems fairly innocuous. However, does this random variable relate to the internal temperature of the patient? To the thermometer reading (if one is taken by the medical staff)? Does it refer to the temperature of the patient at a specific moment (for example, the time of admission to the hospital) or to occurrence of a fever over a prolonged period? Clearly, each of these might be a reasonable attribute to model, but the interaction of Fever with other variables depends on the specific interpretation we use.*

clarity test

As this example shows, we must be precise in defining the variables in the model. The clarity test *is a good way of evaluating whether they are sufficiently well defined. Assume that we are a million years after the events described in the domain; can an omniscient being, one who saw everything, determine the value of the variable? For example, consider a Weather variable with a value sunny. To be absolutely precise, we must define where we check the weather, at what time,*

and what fraction of the sky must be clear in order for it to be sunny. For a variable such as Heart-attack, we must specify how large the heart attack has to be, during what period of time it has to happen, and so on. By contrast, a variable such as Risk-of-heart-attack is meaningless, as even an omniscient being cannot evaluate whether a person had high risk or low risk, only whether the heart attack occurred or not. Introducing variables such as this confounds actual events and their probability. Note, however, that we can *use a notion of "risk group," as long as it is defined in terms of clearly specified attributes such as age or lifestyle.*

If we are not careful in our choice of variables, we will have a hard time making sure that evidence observed and conclusions made are coherent.

hidden variable

Generally speaking, we want our model to contain variables that we can potentially observe or that we may want to query. However, sometimes we want to put in a hidden variable *that is neither observed nor directly of interest. Why would we want to do that? Let us consider an example relating to a cholesterol test. Assume that, for the answers to be accurate, the subject has to have eaten nothing after 10:00 PM the previous evening. If the person eats (having no willpower), the results are consistently off. We do not really care about a Willpower variable, nor can we observe it. However, without it, all of the different cholesterol tests become correlated. To avoid graphs where all the tests are correlated, it is better to put in this additional hidden variable, rendering them conditionally independent given the true cholesterol level and the person's willpower.*

On the other hand, it is not necessary to add every variable that might be relevant. In our Student *example, the student's SAT score may be affected by whether he goes out for drinks on the night before the exam. Is this variable important to represent? The probabilities already account for the fact that he may achieve a poor score despite being intelligent. It might not be worthwhile to include this variable if it cannot be observed.*

It is also important to specify a reasonable domain of values for our variables. In particular, if our partition is not fine enough, conditional independence assumptions may be false. For example, we might want to construct a model where we have a person's cholesterol level, and two cholesterol tests that are conditionally independent given the person's true cholesterol level. We might choose to define the value normal to correspond to levels up to 200, and high to levels above 200. But it may be the case that both tests are more likely to fail if the person's cholesterol is marginal (200–240). In this case, the assumption of conditional independence given the value (high/normal) of the cholesterol test is false. It is only true if we add a marginal value.

Picking structure *As we saw, there are many structures that are consistent with the same set of independencies. One successful approach is to choose a structure that reflects the causal order and dependencies, so that causes are parents of the effect. Such structures tend to work well. Either because of some real locality of influence in the world, or because of the way people perceive the world, causal graphs tend to be sparser. It is important to stress that the causality is in the world, not in our inference process. For example, in an automobile insurance network, it is tempting to put Previous-accident as a parent of Good-driver, because that is how the insurance company thinks about the problem. This is not the causal order in the world, because being a bad driver causes previous (and future) accidents. In principle, there is nothing to prevent us from directing the edges in this way. However, a noncausal ordering often requires that we introduce many additional edges to account for induced dependencies (see section 3.4.1).*

One common approach to constructing a structure is a backward construction process. We begin with a variable of interest, say Lung-Cancer. We then try to elicit a prior probability for that

variable. If our expert responds that this probability is not determinable, because it depends on other factors, that is a good indication that these other factors should be added as parents for that variable (and as variables into the network). For example, we might conclude using this process that Lung-Cancer really should have Smoking as a parent, and (perhaps not as obvious) that Smoking should have Gender and Age as a parent. This approach, called extending the conversation, *avoids probability estimates that result from an average over a heterogeneous population, and therefore leads to more precise probability estimates.*

When determining the structure, however, we must also keep in mind that approximations are inevitable. For many pairs of variables, we can construct a scenario where one depends on the other. For example, perhaps Difficulty depends on Intelligence, because the professor is more likely to make a class difficult if intelligent students are registered. In general, **there are many weak influences that we might choose to model, but if we put in all of them, the network can become very complex.** *Such networks are problematic from a representational perspective: they are hard to understand and hard to debug, and eliciting (or learning) parameters can get very difficult. Moreover, as reasoning in Bayesian networks depends strongly on their connectivity (see section 9.4), adding such edges can make the network too expensive to use.*

This final consideration may lead us, in fact, to make approximations that we know to be wrong. For example, in networks for fault or medical diagnosis, the correct approach is usually to model each possible fault as a separate random variable, allowing for multiple failures. However, such networks might be too complex to perform effective inference in certain settings, and so we may sometimes resort to a single fault approximation, *where we have a single random variable encoding the primary fault or disease.*

Picking probabilities *One of the most challenging tasks in constructing a network manually is eliciting probabilities from people. This task is somewhat easier in the context of causal models, since the parameters tend to be natural and more interpretable. Nevertheless, people generally dislike committing to an exact estimate of probability.*

One approach is to elicit estimates qualitatively, using abstract terms such as "common," "rare," and "surprising," and then assign these to numbers using a predefined scale. This approach is fairly crude, and often can lead to misinterpretation. There are several approaches developed for assisting in eliciting probabilities from people. For example, one can visualize the probability of the event as an area (slice of a pie), or ask people how they would compare the probability in question to certain predefined lotteries. Nevertheless, probability elicitation is a long, difficult process, and one whose outcomes are not always reliable: the elicitation method can often influence the results, and asking the same question using different phrasing can often lead to significant differences in the answer. For example, studies show that people's estimates for an event such as "Death by disease" are significantly lower than their estimates for this event when it is broken down into different possibilities such as "Death from cancer," "Death from heart disease," and so on.

How important is it that we get our probability estimates exactly right? In some cases, small errors have very little effect. For example, changing a conditional probability of 0.7 *to* 0.75 *generally does not have a significant effect. Other errors, however, can have a significant effect:*

- **Zero probabilities***: A common mistake is to assign a probability of zero to an event that is extremely unlikely, but not impossible. The problem is that* **one can never condition away a zero probability, no matter how much evidence we get. When an event is unlikely**

but not impossible, giving it probability zero is guaranteed to lead to irrecoverable errors. *For example, in one of the early versions of the the Pathfinder system (box 3.D), 10 percent of the misdiagnoses were due to zero probability estimates given by the expert to events that were unlikely but not impossible. As a general rule, very few things (except definitions) have probability zero, and we must be careful in assigning zeros.*

- **Orders of magnitude:** *Small differences in very low probability events can make a large difference to the network conclusions. Thus, a (conditional) probability of* 10^{-4} *is very different from* 10^{-5}.
- **Relative values:** *The qualitative behavior of the conclusions reached by the network — the value that has the highest probability — is fairly sensitive to the relative sizes of* $P(x \mid \boldsymbol{y})$ *for different values* $\boldsymbol{y}$ *of* Pa_X. *For example, it is important that the network encode correctly that the probability of having a high fever is greater when the patient has pneumonia than when he has the flu.*

sensitivity analysis

A very useful tool for estimating network parameters is sensitivity analysis, *which allows us to determine the extent to which a given probability parameter affects the outcome. This process allows us to evaluate whether it is important to get a particular CPD entry right. It also helps us figure out which CPD entries are responsible for an answer to some query that does not match our intuitions.*

Box 3.D — Case Study: Medical Diagnosis Systems. *One of the earliest applications of Bayesian networks was to the task of* medical diagnosis. *In the 1980s, a very active area of research was the construction of* expert systems *— computer-based systems that replace or assist an expert in performing a complex task. One such task that was tackled in several ways was medical diagnosis. This task, more than many others, required a treatment of uncertainty, due to the complex, nondeterministic relationships between findings and diseases. Thus, it formed the basis for experimentation with various formalisms for uncertain reasoning.*

medical diagnosis

expert system

Pathfinder

The Pathfinder *expert system was designed by Heckerman and colleagues (Heckerman and Nathwani 1992a; Heckerman et al. 1992; Heckerman and Nathwani 1992b) to help a pathologist diagnose diseases in lymph nodes. Ultimately, the model contained more than sixty different diseases and around a hundred different features. It evolved through several versions, including some based on nonprobabilistic formalisms, and several that used variants of Bayesian networks. Its diagnostic ability was evaluated over real pathological cases and compared to the diagnoses of pathological experts.*

One of the first models used was a simple naive Bayes model, which was compared to the models based on alternative uncertainty formalisms, and judged to be superior in its diagnostic ability. It therefore formed the basis for subsequent development of the system.

The same evaluation pointed out important problems in the way in which parameters were elicited from the expert. First, it was shown that 10 percent of the cases were diagnosed incorrectly, because the correct disease was ruled out by a finding that was unlikely, but not impossible, to manifest in that disease. Second, in the original construction, the expert estimated the probabilities $P(\mathit{Finding} \mid \mathit{Disease})$ *by fixing a single disease and evaluating the probabilities of all its findings.*

It was found that the expert was more comfortable considering a single finding and evaluating its probability across all diseases. This approach allows the expert to compare the relative values of the same finding across multiple diseases, as described in box 3.C.

With these two lessons in mind, another version of Pathfinder — Pathfinder III — was constructed, still using the naive Bayes model. Finally, Pathfinder IV used a full Bayesian network, with a single disease hypothesis but with dependencies between the features. Pathfinder IV was constructed using a similarity network (see box 5.B), significantly reducing the number of parameters that must be elicited. Pathfinder IV, viewed as a Bayesian network, had a total of around 75,000 parameters, but the use of similarity networks allowed the model to be constructed with fewer than 14,000 distinct parameters. Overall, the structure of Pathfinder IV took about 35 hours to define, and the parameters 40 hours.

A comprehensive evaluation of the performance of the two models revealed some important insights. First, the Bayesian network performed as well or better on most cases than the naive Bayes model. In most of the cases where the Bayesian network performed better, the use of richer dependency models was a contributing factor. As expected, these models were useful because they address the strong conditional independence assumptions of the naive Bayes model, as described in box 3.A. Somewhat more surprising, they also helped in allowing the expert to condition the probabilities on relevant factors other than the disease, using the process of extending the conversation described in box 3.C, leading to more accurate elicited probabilities. Finally, the use of similarity networks led to more accurate models, for the smaller number of elicited parameters reduced irrelevant fluctuations in parameter values (due to expert inconsistency) that can lead to spurious dependencies.

Overall, the Bayesian network model agreed with the predictions of an expert pathologist in $50/53$ *cases, as compared with* $47/53$ *cases for the naive Bayes model, with significant therapeutic implications. A later evaluation showed that the diagnostic accuracy of Pathfinder IV was at least as good as that of the expert used to design the system. When used with less expert pathologists, the system significantly improved the diagnostic accuracy of the physicians alone. Moreover, the system showed greater ability to identify important findings and to integrate these findings into a correct diagnosis.*

Unfortunately, multiple reasons prevent the widespread adoption of Bayesian networks as an aid for medical diagnosis, including legal liability issues for misdiagnoses and incompatibility with the physicians' workflow. However, several such systems have been fielded, with significant success. Moreover, similar technology is being used successfully in a variety of other diagnosis applications (see box 23.C).

3.3 Independencies in Graphs

Dependencies and independencies are key properties of a distribution and are crucial for understanding its behavior. As we will see, independence properties are also important for answering queries: they can be exploited to reduce substantially the computation cost of inference. Therefore, it is important that our representations make these properties clearly visible both to a user and to algorithms that manipulate the BN data structure.

As we discussed, a graph structure $\mathcal{G}$ encodes a certain set of conditional independence assumptions $\mathcal{I}_\ell(\mathcal{G})$. Knowing only that a distribution P factorizes over $\mathcal{G}$, we can conclude that it satisfies $\mathcal{I}_\ell(\mathcal{G})$. An immediate question is whether there are other independencies that we can "read off" directly from $\mathcal{G}$. That is, are there other independencies that hold for *every* distribution P that factorizes over $\mathcal{G}$?

3.3.1 D-separation

Our aim in this section is to understand when we can *guarantee* that an independence $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ holds in a distribution associated with a BN structure $\mathcal{G}$. To understand when a property is guaranteed to hold, it helps to consider its converse: "Can we imagine a case where it does not?" Thus, we focus our discussion on analyzing when it is *possible* that $\boldsymbol{X}$ can influence $\boldsymbol{Y}$ given $\boldsymbol{Z}$. If we construct an example where this influence occurs, then the converse property $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ cannot hold for all of the distributions that factorize over $\mathcal{G}$, and hence the independence property $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ cannot follow from $\mathcal{I}_\ell(\mathcal{G})$.

We therefore begin with an intuitive case analysis: Here, we try to understand when an observation regarding a variable X can possibly change our beliefs about Y, in the presence of evidence about the variables $\boldsymbol{Z}$. Although this analysis will be purely intuitive, we will show later that our conclusions are actually provably correct.

Direct connection We begin with the simple case, when X and Y are directly connected via an edge, say $X \rightarrow Y$. For any network structure $\mathcal{G}$ that contains the edge $X \rightarrow Y$, it is possible to construct a distribution where X and Y are correlated regardless of any evidence about any of the other variables in the network. In other words, if X and Y are directly connected, we can always get examples where they influence each other, regardless of $\boldsymbol{Z}$.

In particular, assume that $Val(X) = Val(Y)$; we can simply set $X = Y$. That, by itself, however, is not enough; if (given the evidence $\boldsymbol{Z}$) X deterministically takes some particular value, say 0, then X and Y both deterministically take that value, and are uncorrelated. We therefore set the network so that X is (for example) uniformly distributed, regardless of the values of any of its parents. This construction suffices to induce a correlation between X and Y, regardless of the evidence.

Indirect connection Now consider the more complicated case when X and Y are not directly connected, but there is a trail between them in the graph. We begin by considering the simplest such case: a three-node network, where X and Y are not directly connected, but where there is a trail between them via Z. It turns out that this simple case is the key to understanding the whole notion of indirect interaction in Bayesian networks.

There are four cases where X and Y are connected via Z, as shown in figure 3.5. The first two correspond to causal chains (in either direction), the third to a common cause, and the fourth to a common effect. We analyze each in turn.

Indirect causal effect (figure 3.5a). To gain intuition, let us return to the Student example, where we had a causal trail $I \rightarrow G \rightarrow L$. Let us begin with the case where G is not observed. Intuitively, if we observe that the student is intelligent, we are more inclined to believe that he gets an A, and therefore that his recommendation letter is strong. In other words, the probability of these latter events is higher conditioned on the observation that the student is intelligent.

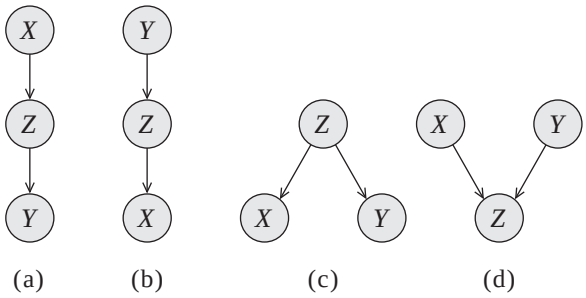


Figure 3.5 The four possible two-edge trails from X to Y via Z: (a) An indirect causal effect; (b) An indirect evidential effect; (c) A common cause; (d) A common effect.

In fact, we saw precisely this behavior in the distribution of figure 3.4. Thus, in this case, we believe that X can influence Y via Z.

Now assume that Z is observed, that is, $Z \in \boldsymbol{Z}$. As we saw in our analysis of the Student example, if we observe the student's grade, then (as we assumed) his intelligence no longer influences his letter. In fact, the local independencies for this network tell us that $(L \perp I \mid G)$. Thus, we conclude that X cannot influence Y via Z if Z is observed.

Indirect evidential effect (figure 3.5b). Returning to the Student example, we have a chain $I \rightarrow G \rightarrow L$. We have already seen that observing a strong recommendation letter for the student changes our beliefs in his intelligence. Conversely, once the grade is observed, the letter gives no additional information about the student's intelligence. Thus, our analysis in the case $Y \rightarrow Z \rightarrow X$ here is identical to the causal case: X can influence Y via Z, but only if Z is not observed. The similarity is not surprising, as dependence is a symmetrical notion. Specifically, if $(X \perp Y)$ does not hold, then $(Y \perp X)$ does not hold either.

Common cause (figure 3.5c). This case is one that we have analyzed extensively, both within the simple naive Bayes model of section 3.1.3 and within our Student example. Our example has the student's intelligence I as a parent of his grade G and his SAT score S. As we discussed, S and G are correlated in this model, in that observing (say) a high SAT score gives us information about a student's intelligence and hence helps us predict his grade. However, once we observe I, this correlation disappears, and S gives us no additional information about G. Once again, for this network, this conclusion follows from the local independence assumption for the node G (or for S). Thus, our conclusion here is identical to the previous two cases: X can influence Y via Z if and only if Z is not observed.

Common effect (figure 3.5d). In all of the three previous cases, we have seen a common pattern: X can influence Y via Z if and only if Z is not observed. Therefore, we might expect that this pattern is universal, and will continue through this last case. Somewhat surprisingly, this is not the case. Let us return to the Student example and consider I and D, which are parents of G. When G is not observed, we have that I and D are independent. In fact, this conclusion follows (once again) from the local independencies from the network. Thus, in this case, influence cannot "flow" along the trail $X \rightarrow Z \leftarrow Y$ if the intermediate node Z is not observed.

On the other hand, consider the behavior when Z is observed. In our discussion of the

Student example, we analyzed precisely this case, which we called intercausal reasoning; we showed, for example, that the probability that the student has high intelligence goes down dramatically when we observe that his grade is a C ($G = g^3$), but then goes up when we observe that the class is a difficult one $D = d^1$. Thus, in presence of the evidence $G = g^3$, we have that I and D are correlated.

Let us consider a variant of this last case. Assume that we do not observe the student's grade, but we do observe that he received a weak recommendation letter ($L = l^0$). Intuitively, the same phenomenon happens. The weak letter is an indicator that he received a low grade, and therefore it suffices to correlate I and D.

When influence can flow from X to Y via Z, we say that the trail $X \rightleftharpoons Z \rightleftharpoons Y$ is *active*. The results of our analysis for active two-edge trails are summarized thus:

- **Causal trail** $X \rightarrow Z \rightarrow Y$**:** active if and only if Z is not observed.
- **Evidential trail** $X \leftarrow Z \leftarrow Y$**:** active if and only if Z is not observed.
- **Common cause** $X \leftarrow Z \rightarrow Y$**:** active if and only if Z is not observed.
- **Common effect** $X \rightarrow Z \leftarrow Y$**:** active if and only if either Z or one of Z's descendants is observed.

v-structure

A structure where $X \rightarrow Z \leftarrow Y$ (as in figure 3.5d) is also called a *v-structure*.

It is useful to view probabilistic influence as a flow in the graph. Our analysis here tells us when influence from X can "flow" through Z to affect our beliefs about Y.

General Case Now consider the case of a longer trail $X_1 \rightleftharpoons \cdots \rightleftharpoons X_n$. Intuitively, for influence to "flow" from X_1 to X_n, it needs to flow through every single node on the trail. In other words, X_1 can influence X_n if every two-edge trail $X_{i-1} \rightleftharpoons X_i \rightleftharpoons X_{i+1}$ along the trail allows influence to flow.

We can summarize this intuition in the following definition:

Definition 3.6

observed variable

active trail

Let $\mathcal{G}$ be a BN structure, and $X_1 \rightleftharpoons \ldots \rightleftharpoons X_n$ a trail in $\mathcal{G}$. Let $\boldsymbol{Z}$ be a subset of observed variables. *The trail $X_1 \rightleftharpoons \ldots \rightleftharpoons X_n$ is* active *given $\boldsymbol{Z}$ if*

- *Whenever we have a v-structure $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$, then X_i or one of its descendants are in $\boldsymbol{Z}$;*
- *no other node along the trail is in $\boldsymbol{Z}$.* ■

Note that if X_1 or X_n are in $\boldsymbol{Z}$ the trail is not active.

In our Student BN, we have that $D \rightarrow G \leftarrow I \rightarrow S$ is not an active trail for $\boldsymbol{Z} = \emptyset$, because the v-structure $D \rightarrow G \leftarrow I$ is not activated. That same trail is active when $\boldsymbol{Z} = \{L\}$, because observing the descendant of G activates the v-structure. On the other hand, when $\boldsymbol{Z} = \{L, I\}$, the trail is not active, because observing I blocks the trail $G \leftarrow I \rightarrow S$.

What about graphs where there is more than one trail between two nodes? Our flow intuition continues to carry through: one node can influence another if there is any trail along which influence can flow. Putting these intuitions together, we obtain the notion of *d-separation*, which provides us with a notion of separation between nodes in a directed graph (hence the term d-separation, for directed separation):

d-separation

Definition 3.7 *Let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ be three sets of nodes in $\mathcal{G}$. We say that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are* d-separated given $\boldsymbol{Z}$*, denoted* d-sep$_{\mathcal{G}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$*, if there is no active trail between any node $X \in \boldsymbol{X}$ and $Y \in \boldsymbol{Y}$ given $\boldsymbol{Z}$.*

We use $\mathcal{I}(\mathcal{G})$ to denote the set of independencies that correspond to d-separation:

$$\mathcal{I}(\mathcal{G}) = \{(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \ : \ \text{d-sep}_{\mathcal{G}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})\}. \quad \blacksquare$$

global Markov independencies

This set is also called the set of *global Markov independencies*. The similarity between the notation $\mathcal{I}(\mathcal{G})$ and our notation $\mathcal{I}(P)$ is not coincidental: As we discuss later, the independencies in $\mathcal{I}(\mathcal{G})$ are precisely those that are guaranteed to hold for every distribution over $\mathcal{G}$.

3.3.2 Soundness and Completeness

So far, our definition of d-separation has been based on our intuitions regarding flow of influence, and on our one example. As yet, we have no guarantee that this analysis is "correct." Perhaps there is a distribution over the BN where X can influence Y despite the fact that all trails between them are blocked.

soundness of d-separation

Hence, the first property we want to ensure for d-separation as a method for determining independence is *soundness*: if we find that two nodes X and Y are d-separated given some $\boldsymbol{Z}$, then we are guaranteed that they are, in fact, conditionally independent given $\boldsymbol{Z}$.

Theorem 3.3 *If a distribution P factorizes according to $\mathcal{G}$, then $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}(P)$.*

In other words, any independence reported by d-separation is satisfied by the underlying distribution. The proof of this theorem requires some additional machinery that we introduce in chapter 4, so we defer the proof to that chapter (see section 4.5.1.1).

completeness of d-separation

A second desirable property is the complementary one — *completeness*: d-separation detects *all* possible independencies. More precisely, if we have that two variables X and Y are independent given $\boldsymbol{Z}$, then they are d-separated. A careful examination of the completeness property reveals that it is ill defined, inasmuch as it does not specify the distribution in which X and Y are independent.

To formalize this property, we first define the following notion:

Definition 3.8 *A distribution P is* faithful *to $\mathcal{G}$ if, whenever $(X \perp Y \mid \boldsymbol{Z}) \in \mathcal{I}(P)$, then* d-sep$_{\mathcal{G}}(X; Y \mid \boldsymbol{Z})$*. In other words, any independence in P is reflected in the d-separation properties of the graph.* $\blacksquare$

faithful

We can now provide one candidate formalization of the completeness property is as follows:

- For any distribution P that factorizes over $\mathcal{G}$, we have that P is faithful to $\mathcal{G}$; that is, if X and Y are *not d-separated* given $\boldsymbol{Z}$ in $\mathcal{G}$, then X and Y are *dependent in all* distributions P that factorize over $\mathcal{G}$.

This property is the obvious converse to our notion of soundness: If true, the two together would imply that, for any P that factorizes over $\mathcal{G}$, we have that $\mathcal{I}(P) = \mathcal{I}(\mathcal{G})$. Unfortunately, this highly desirable property is easily shown to be false: Even if a distribution factorizes over $\mathcal{G}$, it can still contain additional independencies that are not reflected in the structure.

Example 3.3 *Consider a distribution P over two variables A and B, where A and B are independent. One possible I-map for P is the network $A \rightarrow B$. For example, we can set the CPD for B to be*

	b^0	b^1
a^0	0.4	0.6
a^1	0.4	0.6

This example clearly violates the first candidate definition of completeness, because the graph $\mathcal{G}$ is an I-map for the distribution P, yet there are independencies that hold for this distribution but do not follow from d-separation. In fact, these are not independencies that we can hope to discover by examining the network structure. ■

Thus, the completeness property does not hold for this candidate definition of completeness. We therefore adopt a weaker yet still useful definition:

- If $(X \perp Y \mid \boldsymbol{Z})$ in *all* distributions P that factorize over $\mathcal{G}$, then *d-sep*$_{\mathcal{G}}(X; Y \mid \boldsymbol{Z})$. And the contrapositive: If X and Y are *not d-separated* given $\boldsymbol{Z}$ in $\mathcal{G}$, then X and Y are *dependent in some* distribution P that factorizes over $\mathcal{G}$.

Using this definition, we can show:

Theorem 3.4 *Let $\mathcal{G}$ be a BN structure. If X and Y are* not d-separated *given $\boldsymbol{Z}$ in $\mathcal{G}$, then X and Y are* dependent *given $\boldsymbol{Z}$ in* some distribution P *that factorizes over $\mathcal{G}$.*

PROOF The proof constructs a distribution P that makes X and Y correlated. The construction is roughly as follows. As X and Y are not d-separated, there exists an active trail $U_1, \ldots, U_k$ between them. We define CPDs for the variables on the trail so as to make each pair U_i, U_{i+1} correlated; in the case of a v-structure $U_i \rightarrow U_{i+1} \leftarrow U_{i+2}$, we define the CPD of U_{i+1} so as to ensure correlation, and also define the CPDs of the path to some downstream evidence node, in a way that guarantees that the downstream evidence activates the correlation between U_i and U_{i+2}. All other CPDs in the graph are chosen to be uniform, and thus the construction guarantees that influence only flows along this single path, preventing cases where the influence of two (or more) paths cancel out. The details of the construction are quite technical and laborious, and we omit them. ■

We can view the completeness result as telling us that our definition of $\mathcal{I}(\mathcal{G})$ is the maximal one. For any independence assertion that is not a consequence of d-separation in $\mathcal{G}$, we can always find a counterexample distribution P that factorizes over $\mathcal{G}$. In fact, this result can be strengthened significantly:

Theorem 3.5 *For almost all distributions P that factorize over $\mathcal{G}$, that is, for all distributions except for a set of measure zero in the space of CPD parameterizations, we have that $\mathcal{I}(P) = \mathcal{I}(\mathcal{G})$.*[1]

1. A set has measure zero if it is infinitesimally small relative to the overall space. For example, the set of all rationals has measure zero within the interval $[0, 1]$. A straight line has measure zero in the plane. This intuition is defined formally in the field of *measure theory*.

This result strengthens theorem 3.4 in two distinct ways: First, whereas theorem 3.4 shows that any dependency in the graph can be found in some distribution, this new result shows that there exists a single distribution that is faithful to the graph, that is, where all of the dependencies in the graph hold simultaneously. Second, not only does this property hold for a single distribution, but it also holds for almost all distributions that factorize over $\mathcal{G}$.

PROOF At a high level, the proof is based on the following argument: Each conditional independence assertion is a set of polynomial equalities over the space of CPD parameters (see exercise 3.13). A basic property of polynomials is that a polynomial is either identically zero or it is nonzero almost everywhere (its set of roots has measure zero). Theorem 3.4 implies that polynomials corresponding to assertions outside $\mathcal{I}(\mathcal{G})$ cannot be identically zero, because they have at least one counterexample. Thus, the set of distributions P, which exhibit any one of these "spurious" independence assertions, has measure zero. The set of distributions that do not satisfy $\mathcal{I}(P) = \mathcal{I}(\mathcal{G})$ is the union of these separate sets, one for each spurious independence assertion. The union of a finite number of sets of measure zero is a set of measure zero, proving the result. ■

☞ **These results state that for almost all parameterizations P of the graph $\mathcal{G}$ (that is, for almost all possible choices of CPDs for the variables), the d-separation test precisely characterizes the independencies that hold for P.** In other words, even if we have a distribution P that satisfies more independencies than $\mathcal{I}(\mathcal{G})$, a slight perturbation of the CPDs of P will almost always eliminate these "extra" independencies. This guarantee seems to state that such independencies are always accidental, and we will never encounter them in practice. However, as we illustrate in example 3.7, there are cases where our CPDs have certain local structure that is not accidental, and that implies these additional independencies that are not detected by d-separation.

3.3.3 An Algorithm for d-Separation

The notion of d-separation allows us to infer independence properties of a distribution P that factorizes over $\mathcal{G}$ simply by examining the connectivity of $\mathcal{G}$. However, in order to be useful, we need to be able to determine d-separation effectively. Our definition gives us a constructive solution, but a very inefficient one: We can enumerate all trails between X and Y, and check each one to see whether it is active. The running time of this algorithm depends on the number of trails in the graph, which can be exponential in the size of the graph.

Fortunately, there is a much more efficient algorithm that requires only linear time in the size of the graph. The algorithm has two phases. We begin by traversing the graph bottom up, from the leaves to the roots, marking all nodes that are in $\boldsymbol{Z}$ or that have descendants in $\boldsymbol{Z}$. Intuitively, these nodes will serve to enable v-structures. In the second phase, we traverse breadth-first from X to Y, stopping the traversal along a trail when we get to a blocked node. A node is blocked if: (a) it is the "middle" node in a v-structure and unmarked in phase I, or (b) is not such a node and is in $\boldsymbol{Z}$. If our breadth-first search gets us from X to Y, then there is an active trail between them.

The precise algorithm is shown in algorithm 3.1. The first phase is straightforward. The second phase is more subtle. For efficiency, and to avoid infinite loops, the algorithm must keep track of all nodes that have been visited, so as to avoid visiting them again. However, in graphs

Algorithm 3.1 Algorithm for finding nodes reachable from X given $\boldsymbol{Z}$ via active trails

```
    Procedure Reachable (
      G,    // Bayesian network graph
      X,    // Source variable
      Z     // Observations
    )
1     // Phase I: Insert all ancestors of Z into A
2   L ← Z     // Nodes to be visited
3   A ← ∅     // Ancestors of Z
4   while L ≠ ∅
5     Select some Y from L
6     L ← L − {Y}
7     if Y ∉ A then
8       L ← L ∪ Pa_Y     // Y's parents need to be visited
9     A ← A ∪ {Y}     // Y is ancestor of evidence
10
11    // Phase II: traverse active trails starting from X
12  L ← {(X, ↑)}     // (Node,direction) to be visited
13  V ← ∅     // (Node,direction) marked as visited
14  R ← ∅     // Nodes reachable via active trail
15  while L ≠ ∅
16    Select some (Y, d) from L
17    L ← L − {(Y, d)}
18    if (Y, d) ∉ V then
19      if Y ∉ Z then
20        R ← R ∪ {Y}     // Y is reachable
21      V ← V ∪ {(Y, d)}     // Mark (Y, d) as visited
22      if d =↑ and Y ∉ Z then     // Trail up through Y active if Y not in Z
23        for each Z ∈ Pa_Y
24          L ← L ∪ {(Z, ↑)}     // Y's parents to be visited from bottom
25        for each Z ∈ Ch_Y
26          L ← L ∪ {(Z, ↓)}     // Y's children to be visited from top
27      else if d =↓ then     // Trails down through Y
28        if Y ∉ Z then
29          // Downward trails to Y's children are active
30          for each Z ∈ Ch_Y
31            L ← L∪{(Z, ↓)}     // Y's children to be visited from top
32        if Y ∈ A then     // v-structure trails are active
33          for each Z ∈ Pa_Y
34            L ← L∪{(Z, ↑)}     // Y's parents to be visited from bottom
35  return R
```

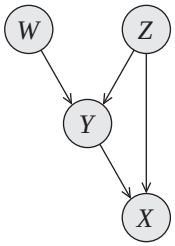


Figure 3.6 A simple example for the d-separation algorithm

with loops (multiple trails between a pair of nodes), an intermediate node Y might be involved in several trails, which may require different treatment within the algorithm:

Example 3.4 *Consider the Bayesian network of figure 3.6, where our task is to find all nodes reachable from X. Assume that Y is observed, that is, $Y \in \boldsymbol{Z}$. Assume that the algorithm first encounters Y via the direct edge $Y \rightarrow X$. Any extension of this trail is blocked by Y, and hence the algorithm stops the traversal along this trail. However, the trail $X \leftarrow Z \rightarrow Y \leftarrow W$ is not blocked by Y. Thus, when we encounter Y for the second time via the edge $Z \rightarrow Y$, we should not ignore it. Therefore, after the first visit to Y, we can mark it as visited for the purpose of trails coming in from children of Y, but not for the purpose of trails coming in from parents of Y.* ■

In general, we see that, for each node Y, we must keep track separately of whether it has been visited from the top and whether it has been visited from the bottom. Only when both directions have been explored is the node no longer useful for discovering new active trails.

Based on this intuition, we can now show that the algorithm achieves the desired result:

Theorem 3.6 *The algorithm* Reachable$(\mathcal{G}, X, \boldsymbol{Z})$ *returns the set of all nodes reachable from X via trails that are active in $\mathcal{G}$ given $\boldsymbol{Z}$.*

The proof is left as an exercise (exercise 3.14).

3.3.4 I-Equivalence

The notion of $\mathcal{I}(\mathcal{G})$ specifies a set of conditional independence assertions that are associated with a graph. This notion allows us to abstract away the details of the graph structure, viewing it purely as a specification of independence properties. In particular, one important implication of this perspective is the observation that very different BN structures can actually be equivalent, in that they encode precisely the same set of conditional independence assertions. Consider, for example, the three networks in figure 3.5a,(b),(c). All three of them encode precisely the same independence assumptions: $(X \perp Y \mid Z)$.

Definition 3.9 I-equivalence

Two graph structures $\mathcal{K}_1$ and $\mathcal{K}_2$ over $\mathcal{X}$ are I-equivalent *if $\mathcal{I}(\mathcal{K}_1) = \mathcal{I}(\mathcal{K}_2)$. The set of all graphs over $\mathcal{X}$ is partitioned into a set of mutually exclusive and exhaustive* I-equivalence classes, *which are the set of equivalence classes induced by the I-equivalence relation.* ■

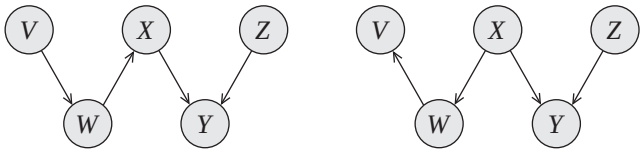


Figure 3.7 Skeletons and v-structures in a network. The two networks shown have the same skeleton and v-structures ($X \rightarrow Y \leftarrow Z$).

Note that the v-structure network in figure 3.5d induces a very different set of d-separation assertions, and hence it does not fall into the same I-equivalence class as the first three. Its I-equivalence class contains only that single network.

I-equivalence of two graphs immediately implies that any distribution P that can be factorized over one of these graphs can be factorized over the other. Furthermore, **there is no intrinsic property of P that would allow us to associate it with one graph rather than an equivalent one. This observation has important implications with respect to our ability to determine the directionality of influence.** In particular, although we can determine, for a distribution $P(X, Y)$, whether X and Y are correlated, there is nothing in the distribution that can help us determine whether the correct structure is $X \rightarrow Y$ or $Y \rightarrow X$. We return to this point when we discuss the causal interpretation of Bayesian networks in chapter 21.

The d-separation criterion allows us to test for I-equivalence using a very simple graph-based algorithm. We start by considering the trails in the networks.

Definition 3.10 skeleton

The skeleton *of a Bayesian network graph $\mathcal{G}$ over $\mathcal{X}$ is an undirected graph over $\mathcal{X}$ that contains an edge $\{X, Y\}$ for every edge (X, Y) in $\mathcal{G}$.* ■

In the networks of figure 3.7, the networks (a) and (b) have the same skeleton.

If two networks have a common skeleton, then the set of trails between two variables X and Y is same in both networks. If they do not have a common skeleton, we can find a trail in one network that does not exist in the other and use this trail to find a counterexample for the equivalence of the two networks.

Ensuring that the two networks have the same trails is clearly not enough. For example, the networks in figure 3.5 all have the same skeleton. Yet, as the preceding discussion shows, the network of figure 3.5d is not equivalent to the networks of figure 3.5a–(c). The difference, is of course, the v-structure in figure 3.5d. Thus, it seems that if the two networks have the same skeleton and exactly the same set of v-structures, they are equivalent. Indeed, this property provides a sufficient condition for I-equivalence:

Theorem 3.7

Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two graphs over $\mathcal{X}$. If $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same skeleton and the same set of v-structures then they are I-equivalent.

The proof is left as an exercise (see exercise 3.16).

Unfortunately, this characterization is not an equivalence: there are graphs that are I-equivalent but do not have the same set of v-structures. As a counterexample, consider *complete* graphs over a set of variables. Recall that a complete graph is one to which we cannot add

additional arcs without causing cycles. Such graphs encode the empty set of conditional independence assertions. Thus, any two complete graphs are I-equivalent. Although they have the same skeleton, they invariably have different v-structures. Thus, by using the criterion on theorem 3.7, we can conclude (in certain cases) only that two networks are I-equivalent, but we cannot use it to guarantee that they are not.

We can provide a stronger condition that does correspond exactly to I-equivalence. Intuitively, the unique independence pattern that we want to associate with a v-structure $X \rightarrow Z \leftarrow Y$ is that X and Y are independent (conditionally on their parents), but dependent given Z. If there is a direct edge between X and Y, as there was in our example of the complete graph, the first part of this pattern is eliminated.

Definition 3.11 *A v-structure $X \rightarrow Z \leftarrow Y$ is an* immorality *if there is no direct edge between X and Y. If there is such an edge, it is called a* covering edge *for the v-structure.* ■

immorality

covering edge

Note that not every v-structure is an immorality, so that two networks with the same immoralities do not necessarily have the same v-structures. For example, two different complete directed graphs always have the same immoralities (none) but different v-structures.

Theorem 3.8 *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two graphs over $\mathcal{X}$. Then $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same skeleton and the same set of immoralities if and only if they are I-equivalent.*

The proof of this (more difficult) result is also left as an exercise (see exercise 3.17).

We conclude with a final characterization of I-equivalence in terms of local operations on the graph structure.

Definition 3.12 *An edge $X \rightarrow Y$ in a graph $\mathcal{G}$ is said to be* covered *if $\mathrm{Pa}_Y^{\mathcal{G}} = \mathrm{Pa}_X^{\mathcal{G}} \cup \{X\}$.* ■

covered edge

Theorem 3.9 *Two graphs $\mathcal{G}$ and $\mathcal{G}'$ are I-equivalent if and only if there exists a sequence of networks $\mathcal{G} = \mathcal{G}_1, \ldots, \mathcal{G}_k = \mathcal{G}'$ that are all I-equivalent to $\mathcal{G}$ such that the only difference between $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$ is a single reversal of a covered edge.*

The proof of this theorem is left as an exercise (exercise 3.18).

3.4 From Distributions to Graphs

In the previous sections, we showed that, if P factorizes over $\mathcal{G}$, we can derive a rich set of independence assertions that hold for P by simply examining $\mathcal{G}$. This result immediately leads to the idea that we can use a graph as a way of revealing the structure in a distribution. In particular, we can test for independencies in P by constructing a graph $\mathcal{G}$ that represents P and testing d-separation in $\mathcal{G}$. As we will see, having a graph that reveals the structure in P has other important consequences, in terms of reducing the number of parameters required to specify or learn the distribution, and in terms of the complexity of performing inference on the network.

In this section, we examine the following question: Given a distribution P, to what extent can we construct a graph $\mathcal{G}$ whose independencies are a reasonable surrogate for the independencies

in P? It is important to emphasize that we will never actually take a fully specified distribution P and construct a graph $\mathcal{G}$ for it: As we discussed, a full joint distribution is much too large to represent explicitly. However, answering this question is an important conceptual exercise, which will help us later on when we try to understand the process of constructing a Bayesian network that represents our model of the world, whether manually or by learning from data.

3.4.1 Minimal I-Maps

One approach to finding a graph that represents a distribution P is simply to take any graph that is an I-map for P. The problem with this naive approach is clear: As we saw in example 3.3, the complete graph is an I-map for any distribution, yet it does not reveal any of the independence structure in the distribution. However, examples such as this one are not very interesting. The graph that we used as an I-map is clearly and trivially unrepresentative of the distribution, in that there are edges that are obviously redundant. This intuition leads to the following definition, which we also define more broadly:

Definition 3.13 minimal I-map

A graph $\mathcal{K}$ is a minimal I-map *for a set of independencies $\mathcal{I}$ if it is an I-map for $\mathcal{I}$, and if the removal of even a single edge from $\mathcal{K}$ renders it not an I-map.* ■

This notion of an I-map applies to multiple types of graphs, both Bayesian networks and other types of graphs that we will encounter later on. Moreover, because it refers to a set of independencies $\mathcal{I}$, it can be used to define an I-map for a distribution P, by taking $\mathcal{I} = \mathcal{I}(P)$, or to another graph $\mathcal{K}'$, by taking $\mathcal{I} = \mathcal{I}(\mathcal{K}')$.

Recall that definition 3.5 defines a Bayesian network to be a distribution P that factorizes over $\mathcal{G}$, thereby implying that $\mathcal{G}$ is an I-map for P. It is standard to restrict the definition even further, by requiring that $\mathcal{G}$ be a minimal I-map for P.

How do we obtain a minimal I-map for the set of independencies induced by a given distribution P? The proof of the factorization theorem (theorem 3.1) gives us a procedure, which is shown in algorithm 3.2. We assume we are given a predetermined *variable ordering*, say, $\{X_1, \ldots, X_n\}$. We now examine each variable X_i, $i = 1, \ldots, n$ in turn. For each X_i, we pick some minimal subset $\boldsymbol{U}$ of $\{X_1, \ldots, X_{i-1}\}$ to be X_i's parents in $\mathcal{G}$. More precisely, we require that $\boldsymbol{U}$ satisfy $(X_i \perp \{X_1, \ldots, X_{i-1}\} - \boldsymbol{U} \mid \boldsymbol{U})$, and that no node can be removed from $\boldsymbol{U}$ without violating this property. We then set $\boldsymbol{U}$ to be the parents of X_i.

variable ordering

The proof of theorem 3.1 tells us that, if each node X_i is independent of $X_1, \ldots, X_{i-1}$ given its parents in $\mathcal{G}$, then P factorizes over $\mathcal{G}$. We can then conclude from theorem 3.2 that $\mathcal{G}$ is an I-map for P. By construction, $\mathcal{G}$ is minimal, so that $\mathcal{G}$ is a minimal I-map for P.

Note that our choice of $\boldsymbol{U}$ may not be unique. Consider, for example, a case where two variables A and B are logically equivalent, that is, our distribution P only gives positive probability to instantiations where A and B have the same value. Now, consider a node C that is correlated with A. Clearly, we can choose either A or B to be a parent of C, but having chosen the one, we cannot choose the other without violating minimality. Hence, the minimal parent set $\boldsymbol{U}$ in our construction is not necessarily unique. However, one can show that, if the distribution is positive (see definition 2.5), that is, if for any instantiation ξ to all the network variables $\mathcal{X}$ we have that $P(\xi) > 0$, then the choice of parent set, given an ordering, is unique. Under this assumption, algorithm 3.2 can produce all minimal I-maps for P: Let $\mathcal{G}$ be any

Algorithm 3.2 Procedure to build a minimal I-map given an ordering

```
Procedure Build-Minimal-I-Map (
  X_1, ..., X_n    // an ordering of random variables in 𝒳
  ℐ    // Set of independencies
)
1   Set 𝒢 to an empty graph over 𝒳
2   for i = 1, ..., n
3     U ← {X_1, ..., X_{i-1}}    // U is the current candidate for parents of X_i
4     for U' ⊆ {X_1, ..., X_{i-1}}
5       if U' ⊂ U and (X_i ⊥ {X_1, ..., X_{i-1}} − U' | U') ∈ ℐ then
6         U ← U'
7       // At this stage U is a minimal set satisfying (X_i ⊥
          {X_1, ..., X_{i-1}} − U | U)
8       // Now set U to be the parents of X_i
9     for X_j ∈ U
10      Add X_j → X_i to 𝒢
11    return 𝒢
```

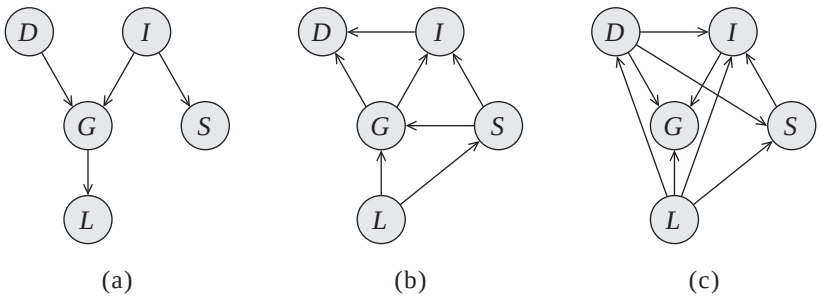


Figure 3.8 Three minimal I-maps for $P_{\mathcal{B}^{student}}$, induced by different orderings: (a) D, I, S, G, L; (b) L, S, G, I, D; (C) L, D, S, I, G.

minimal I-map for P. If we give call Build-Minimal-I-Map with an ordering $\prec$ that is topological for $\mathcal{G}$, then, due to the uniqueness argument, the algorithm must return $\mathcal{G}$.

At first glance, the minimal I-map seems to be a reasonable candidate for capturing the structure in the distribution: It seems that if $\mathcal{G}$ is a minimal I-map for a distribution P, then we should be able to "read off" all of the independencies in P directly from $\mathcal{G}$. Unfortunately, this intuition is false.

Example 3.5

Consider the distribution $P_{\mathcal{B}^{\text{student}}}$, as defined in figure 3.4, and let us go through the process of constructing a minimal I-map for $P_{\mathcal{B}^{\text{student}}}$. We note that the graph G_{student} precisely reflects the independencies in this distribution $P_{\mathcal{B}^{\text{student}}}$ (that is, $\mathcal{I}(P_{\mathcal{B}^{\text{student}}}) = \mathcal{I}(G_{\text{student}})$), so that we can use G_{student} to determine which independencies hold in $P_{\mathcal{B}^{\text{student}}}$.

Our construction process starts with an arbitrary ordering on the nodes; we will go through this

process for three different orderings. Throughout this process, it is important to remember that we are testing independencies relative to the distribution $P_{\mathcal{B}^{\text{student}}}$. We can use G_{student} (figure 3.4) to guide our intuition about which independencies hold in $P_{\mathcal{B}^{\text{student}}}$, but we can always resort to testing these independencies in the joint distribution $P_{\mathcal{B}^{\text{student}}}$.

The first ordering is a very natural one: D, I, S, G, L. We add one node at a time and see which of the possible edges from the preceding nodes are redundant. We start by adding D, then I. We can now remove the edge from D to I because this particular distribution satisfies $(I \perp D)$, so I is independent of D given its other parents (the empty set). Continuing on, we add S, but we can remove the edge from D to S because our distribution satisfies $(S \perp D \mid I)$. We then add G, but we can remove the edge from S to G, because the distribution satisfies $(G \perp S \mid I, D)$. Finally, we add L, but we can remove all edges from D, I, S. Thus, our final output is the graph in figure 3.8a, which is precisely our original network for this distribution.

Now, consider a somewhat less natural ordering: L, S, G, I, D. In this case, the resulting I-map is not quite as natural or as sparse. To see this, let us consider the sequence of steps. We start by adding L to the graph. Since it is the first variable in the ordering, it must be a root. Next, we consider S. The decision is whether to have L as a parent of S. Clearly, we need an edge from L to S, because the quality of the student's letter is correlated with his SAT score in this distribution, and S has no other parents that help render it independent of L. Formally, we have that $(S \perp L)$ does not *hold in the distribution. In the next iteration of the algorithm, we introduce G. Now, all possible subsets of $\{L, S\}$ are potential parents set for G. Clearly, G is dependent on L. Moreover, although G is independent of S given I, it is not independent of S given L. Hence, we must add the edge between S and G. Carrying out the procedure, we end up with the graph shown in figure 3.8b.*

Finally, consider the ordering: L, D, S, I, G. In this case, a similar analysis results in the graph shown in figure 3.8c, which is almost a complete graph, missing only the edge from S to G, which we can remove because G is independent of S given I. ■

Note that the graphs in figure 3.8b,c really are minimal I-maps for this distribution. However, they fail to capture some or all of the independencies that hold in the distribution. Thus, they show that the fact that $\mathcal{G}$ is a minimal I-map for P is far from a guarantee that $\mathcal{G}$ captures the independence structure in P.

3.4.2 Perfect Maps

We aim to find a graph $\mathcal{G}$ that precisely captures the independencies in a given distribution P.

Definition 3.14 perfect map

We say that a graph $\mathcal{K}$ is a perfect map (P-map) *for a set of independencies $\mathcal{I}$ if we have that $\mathcal{I}(\mathcal{K}) = \mathcal{I}$. We say that $\mathcal{K}$ is a perfect map for P if $\mathcal{I}(\mathcal{K}) = \mathcal{I}(P)$.* ■

If we obtain a graph $\mathcal{G}$ that is a P-map for a distribution P, then we can (by definition) read the independencies in P directly from $\mathcal{G}$. By construction, our original graph $G_{\textit{student}}$ is a P-map for $P_{\mathcal{B}^{\textit{student}}}$.

If our goal is to find a perfect map for a distribution, an immediate question is whether every distribution has a perfect map. Unfortunately, the answer is no, and for several reasons. The first type of counterexample involves regularity in the parameterization of the distribution that cannot be captured in the graph structure.

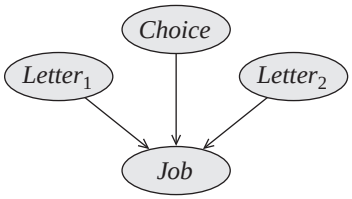


Figure 3.9 Network for the OneLetter **example**

Example 3.6 *Consider a joint distribution P over 3 random variables X,Y,Z such that:*

$$P(x,y,z) = \begin{cases} 1/12 & x \oplus y \oplus z = \textit{false} \\ 1/6 & x \oplus y \oplus z = \textit{true} \end{cases}$$

where $\oplus$ is the XOR (exclusive OR) function. A simple calculation shows that $(X \perp Y) \in \mathcal{I}(P)$, and that Z is not independent of X given Y or of Y given X. Hence, one minimal I-map for this distribution is the network $X \rightarrow Z \leftarrow Y$, using a deterministic XOR for the CPD of Z. However, this network is not a perfect map; a precisely analogous calculation shows that $(X \perp Z) \in \mathcal{I}(P)$, but this conclusion is not supported by a d-separation analysis. ■

Thus, we see that deterministic relationships can lead to distributions that do not have a P-map. Additional examples arise as a consequence of other regularities in the CPD.

Example 3.7 *Consider a slight elaboration of our* Student *example. During his academic career, our student George has taken both Econ101 and CS102. The professors of both classes have written him letters, but the recruiter at Acme Consulting asks for only a single recommendation. George's chance of getting the job depends on the quality of the letter he gives the recruiter. We thus have four random variables: $L1$ and $L2$, corresponding to the quality of the recommendation letters for Econ101 and CS102 respectively; C, whose value represents George's choice of which letter to use; and J, representing the event that George is hired by Acme Consulting.*

The obvious minimal I-map for this distribution is shown in figure 3.9. Is this a perfect map? Clearly, it does not reflect independencies that are not at the variable level. In particular, we have that $(L1 \perp J \mid C = 2)$. However, this limitation is not surprising; by definition, a BN structure makes independence assertions only at the level of variables. (We return to this issue in section 5.2.2.) However, our problems are not limited to these finer-grained independencies. Some thought reveals that, in our target distribution, we also have that $(L1 \perp L2 \mid C, J)$! This independence is not implied by d-separation, because the v-structure $L1 \rightarrow J \leftarrow L2$ is enabled. However, we can convince ourselves that the independence holds using reasoning by cases. If $C = 1$, then there is no dependence of J on $L2$. Intuitively, the edge from $L2$ to J disappears, eliminating the trail between $L1$ and $L2$, so that $L1$ and $L2$ are independent in this case. A symmetric analysis applies in the case that $C = 2$. Thus, in both cases, we have that $L1$ and $L2$ are independent. This independence assertion is not captured by our minimal I-map, which is therefore not a P-map. ■

A different class of examples is not based on structure within a CPD, but rather on symmetric variable-level independencies that are not naturally expressed within a Bayesian network.

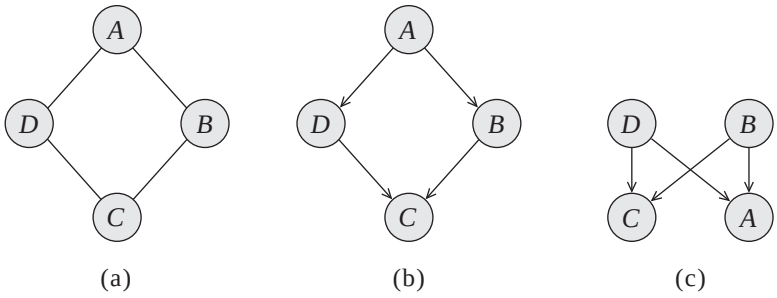


Figure 3.10 Attempted Bayesian network models for the Misconception **example:** (a) Study pairs over four students. (b) First attempt at a Bayesian network model. (c) Second attempt at a Bayesian network model.

A second class of distributions that do not have a perfect map are those for which the independence assumptions imposed by the structure of Bayesian networks is simply not appropriate.

Example 3.8

Consider a scenario where we have four students who get together in pairs to work on the homework for a class. For various reasons, only the following pairs meet: Alice and Bob; Bob and Charles; Charles and Debbie; and Debbie and Alice. (Alice and Charles just can't stand each other, and Bob and Debbie had a relationship that ended badly.) The study pairs are shown in figure 3.10a.

In this example, the professor accidentally misspoke in class, giving rise to a possible misconception among the students in the class. Each of the students in the class may subsequently have figured out the problem, perhaps by thinking about the issue or reading the textbook. In subsequent study pairs, he or she may transmit this newfound understanding to his or her study partners. We therefore have four binary random variables, representing whether the student has the misconception or not. We assume that for each $X \in \{A, B, C, D\}$, x^1 denotes the case where the student has the misconception, and x^0 denotes the case where he or she does not.

Because Alice and Charles never speak to each other directly, we have that A and C are conditionally independent given B and D. Similarly, B and D are conditionally independent given A and C. Can we represent this distribution (with these independence properties) using a BN? One attempt is shown in figure 3.10b. Indeed, it encodes the independence assumption that $(A \perp C \mid \{B, D\})$. However, it also implies that B and D are independent given only A, but dependent given both A and C. Hence, it fails to provide a perfect map for our target distribution. A second attempt, shown in figure 3.10c, is equally unsuccessful. It also implies that $(A \perp C \mid \{B, D\})$, but it also implies that B and D are marginally independent. It is clear that all other candidate BN structures are also flawed, so that this distribution does not have a perfect map. ■

3.4.3 Finding Perfect Maps ⋆

Earlier we discussed an algorithm for finding minimal I-maps. We now consider an algorithm for finding a perfect map (P-map) of a distribution. Because the requirements from a P-map are stronger than the ones we require from an I-map, the algorithm will be more involved.

Throughout the discussion in this section, we assume that P has a P-map. In other words, there is an unknown DAG $\mathcal{G}^*$ that is P-map of P. Since $\mathcal{G}^*$ is a P-map, we will interchangeably refer to independencies in P and in $\mathcal{G}^*$ (since these are the same). We note that the algorithms we describe do fail when they are given a distribution that does not have a P-map. We discuss this issue in more detail later.

Thus, our goal is to identify $\mathcal{G}^*$ from P. One obvious difficulty that arises when we consider this goal is that $\mathcal{G}^*$ is, in general, not uniquely identifiable from P. A P-map of a distribution, if one exists, is generally not unique: As we saw, for example, in figure 3.5, multiple graphs can encode precisely the same independence assumptions. However, the P-map of a distribution *is* unique up to I-equivalence between networks. That is, a distribution P can have many P-maps, but all of them are I-equivalent.

If we require that a P-map construction algorithm return a single network, the output we get may be some arbitrary member of the I-equivalence class of $\mathcal{G}^*$. A more correct answer would be to return the entire equivalence class, thus avoiding an arbitrary commitment to a possibly incorrect structure. Of course, we do not want our algorithm to return a (possibly very large) set of distinct networks as output. Thus, one of our tasks in this section is to develop a compact representation of an entire equivalence class of DAGs. As we will see later in the book, this representation plays a useful role in other contexts as well.

This formulation of the problem points us toward a solution. Recall that, according to theorem 3.8, two DAGs are I-equivalent if they share the same skeleton and the same set of immoralities. Thus, we can construct the I-equivalence class for $\mathcal{G}^*$ by determining its skeleton and its immoralities from the independence properties of the given distribution P. We then use both of these components to build a representation of the equivalence class.

3.4.3.1 Identifying the Undirected Skeleton

At this stage we want to construct an undirected graph S that contains an edge $X—Y$ if X and Y are adjacent in $\mathcal{G}^*$; that is, if either $X \to Y$ or $Y \to X$ is an edge in $\mathcal{G}^*$.

The basic idea is to use independence queries of the form $(X \perp Y \mid \boldsymbol{U})$ for different sets of variables $\boldsymbol{U}$. This idea is based on the observation that if X and Y are adjacent in $\mathcal{G}^*$, we cannot separate them with any set of variables.

Lemma 3.1 *Let $\mathcal{G}^*$ be a P-map of a distribution P, and let X and Y be two variables such that $X \to Y$ is in $\mathcal{G}^*$. Then, $P \not\models (X \perp Y \mid \boldsymbol{U})$ for any set $\boldsymbol{U}$ that does not include X and Y.*

PROOF Assume that that $X \to Y \in \mathcal{G}^*$, and let $\boldsymbol{U}$ be a set of variables. According to d-separation the trail $X \to Y$ cannot be blocked by the evidence set $\boldsymbol{U}$. Thus, X and Y are not d-separated by U. Since $\mathcal{G}^*$ is a P-map of P, we have that $P \not\models (X \perp Y \mid \boldsymbol{U})$. ■

This lemma implies that if X and Y are adjacent in $\mathcal{G}^*$, all conditional independence queries that involve both of them would fail. Conversely, if X and Y are not adjacent in $\mathcal{G}$, we would hope to be able to find a set of variables that makes these two variables conditionally independent. Indeed, as we now show, we can provide a precise characterization of such a set:

Lemma 3.2 *Let $\mathcal{G}^*$ be an I-map of a distribution P, and let X and Y be two variables that are not adjacent in $\mathcal{G}^*$. Then either $P \models (X \perp Y \mid \mathrm{Pa}_X^{\mathcal{G}^*})$ or $P \models (X \perp Y \mid \mathrm{Pa}_Y^{\mathcal{G}^*})$.*

The proof is left as an exercise (exercise 3.19).

witness

Thus, if X and Y are not adjacent in $\mathcal{G}^*$, then we can find a set $\boldsymbol{U}$ so that $\mathcal{P} \models (X \perp Y \mid \boldsymbol{U})$. We call this set $\boldsymbol{U}$ a *witness* of their independence. Moreover, the lemma shows that we can find a witness of bounded size. Thus, if we assume that $\mathcal{G}^*$ has bounded indegree, say less than or equal to d, then we do not need to consider witness sets larger than d.

Algorithm 3.3 Recovering the undirected skeleton for a distribution P that has a P-map

```
    Procedure Build-PMap-Skeleton (
      X = {X_1, ..., X_n},    // Set of random variables
      P,    // Distribution over X
      d     // Bound on witness set
    )
1     Let H be the complete undirected graph over X
2     for X_i, X_j in X
3       U_{X_i,X_j} ← ∅
4       for U ∈ Witnesses(X_i, X_j, H, d)
5             // Consider U as a witness set for X_i, X_j
6         if P ⊨ (X_i ⊥ X_j | U) then
7           U_{X_i,X_j} ← U
8           Remove X_i—X_j from H
9           break
10      return (H,{U_{X_i,X_j} : i, j ∈ {1, ..., n}})
```

With these tools in hand, we can now construct an algorithm for building a skeleton of $\mathcal{G}^*$, shown in algorithm 3.3. For each pair of variables, we consider all potential witness sets and test for independence. If we find a witness that separates the two variables, we record it (we will soon see why) and move on to the next pair of variables. If we do not find a witness, then we conclude that the two variables are adjacent in $\mathcal{G}^*$ and add them to the skeleton. The list Witnesses$(X_i, X_j, \mathcal{H}, d)$ in line 4 specifies the set of possible witness sets that we consider for separating X_i and X_j. From our earlier discussion, if we assume a bound d on the indegree, then we can restrict attention to sets $\boldsymbol{U}$ of size at most d. Moreover, using the same analysis, we saw that we have a witness that consists either of the parents of X_i or of the parents of X_j. In the first case, we can restrict attention to sets $\boldsymbol{U} \subseteq \mathrm{Nb}_{X_i}^{\mathcal{H}} - \{X_j\}$, where $\mathrm{Nb}_{X_i}^{\mathcal{H}}$ are the neighbors of X_i in the current graph $\mathcal{H}$; in the second, we can similarly restrict attention to sets $\boldsymbol{U} \subseteq \mathrm{Nb}_{X_j}^{\mathcal{H}} - \{X_i\}$. Finally, we note that if $\boldsymbol{U}$ separates X_i and X_j, then also many of $\boldsymbol{U}$'s supersets will separate X_i and X_j. Thus, we search the set of possible witnesses in order of increasing size.

This algorithm will recover the correct skeleton given that $\mathcal{G}^*$ is a P-map of P and has bounded indegree d. If P does not have a P-map, then the algorithm can fail; see exercise 3.22. This algorithm has complexity of $O(n^{d+2})$ since we consider $O(n^2)$ pairs, and for each we perform $O((n-2)^d)$ independence tests. We greatly reduce the number of independence tests by ordering potential witnesses accordingly, and by aborting the inner loop once we find a witness for a pair (after line 9). However, for pairs of variables that are directly connected in the skeleton, we still need to evaluate all potential witnesses.

Algorithm 3.4 Marking immoralities in the construction of a perfect map

```
Procedure Mark-Immoralities (
  X = {X_1, ..., X_n},
  S    // Skeleton
  {U_{X_i,X_j} : 1 ≤ i, j ≤ n}    // Witnesses found by Build-PMap-Skeleton
)
1   K ← S
2   for X_i, X_j, X_k such that X_i—X_j—X_k ∈ S and X_i—X_k ∉ S
3       // X_i—X_j—X_k is a potential immorality
4     if X_j ∉ U_{X_i,X_k} then
5       Add the orientations X_i → X_j and X_j ← X_k to K
6   return K
```

3.4.3.2 Identifying Immoralities

At this stage we have reconstructed the undirected skeleton S using Build-PMap-Skeleton. Now, we want to reconstruct edge direction. The main cue for learning about edge directions in $\mathcal{G}^*$ are immoralities. As shown in theorem 3.8, all DAGs in the equivalence class of $\mathcal{G}^*$ share the same set of immoralities. Thus, our goal is to consider *potential immoralities* in the skeleton and for each one determine whether it is indeed an immorality. A triplet of variables X, Z, Y is a *potential immorality* if the skeleton contains $X—Z—Y$ but does not contain an edge between X and Y. If such a triplet is indeed an immorality in $\mathcal{G}^*$, then X and Y cannot be independent given Z. Nor will they be independent given a set $\boldsymbol{U}$ that contains Z. More precisely,

potential immorality

Proposition 3.1 *Let $\mathcal{G}^*$ be a P-map of a distribution P, and let X, Y and Z be variables that form an immorality $X \to Z \leftarrow Y$. Then, $P \not\models (X \perp Y \mid \boldsymbol{U})$ for any set $\boldsymbol{U}$ that contains Z.*

Proof Let $\boldsymbol{U}$ be a set of variables that contains Z. Since Z is observed, the trail $X \to Z \leftarrow Y$ is active, and so X and Y are not d-separated in $\mathcal{G}^*$. Since $\mathcal{G}^*$ is a P-map of P, we have that $P^* \not\models (X \perp Y \mid \boldsymbol{U})$. ■

What happens in the complementary situation? Suppose $X—Z—Y$ in the skeleton, but is not an immorality. This means that one of the following three cases is in $\mathcal{G}^*$: $X \to Z \to Y$, $Y \to Z \to X$, or $X \leftarrow Z \to Y$. In all three cases, X and Y are d-separated only if Z is observed.

Proposition 3.2 *Let $\mathcal{G}^*$ be a P-map of a distribution P, and let the triplet X, Y, Z be a potential immorality in the skeleton of $\mathcal{G}^*$, such that $X \to Z \leftarrow Y$ is not in $\mathcal{G}^*$. If $\boldsymbol{U}$ is such that $P \models (X \perp Y \mid \boldsymbol{U})$, then $Z \in \boldsymbol{U}$.*

Proof Consider all three configurations of the trail $X \rightleftharpoons Z \rightleftharpoons Y$. In all three, Z must be observed in order to block the trail. Since $\mathcal{G}^*$ is a P-map of P, we have that if $P \models (X \perp Y \mid \boldsymbol{U})$, then $Z \in \boldsymbol{U}$. ■

Combining these two results, we see that a potential immorality $X—Z—Y$ is an immorality if and only if Z is not in the witness set(s) for X and Y. That is, if $X—Z—Y$ is an immorality,

then proposition 3.1 shows that Z is not in any witness set $\boldsymbol{U}$; conversely, if $X—Z—Y$ is not an immorality, the Z must be in every witness set $\boldsymbol{U}$. Thus, we can use the specific witness set $\boldsymbol{U}_{X,Y}$ that we recorded for X, Y in order to determine whether this triplet is an immorality or not: we simply check whether $Z \in \boldsymbol{U}_{X,Y}$. If $Z \notin \boldsymbol{U}_{X,Y}$, then we declare the triplet an immorality. Otherwise, we declare that it is not an immorality. The Mark-Immoralities procedure shown in algorithm 3.4 summarizes this process.

3.4.3.3 Representing Equivalence Classes

Once we have the skeleton and identified the immoralities, we have a specification of the equivalence class of $\mathcal{G}^*$. For example, to test if $\mathcal{G}$ is equivalent to $\mathcal{G}^*$ we can check whether it has the same skeleton as $\mathcal{G}^*$ and whether it agrees on the location of the immoralities.

The description of an equivalence class using only the skeleton and the set of immoralities is somewhat unsatisfying. For example, we might want to know whether the fact that our network is in the equivalence class implies that there is an arc $X \to Y$. Although the definition does tell us whether there is some edge between X and Y, it leaves the direction unresolved. In other cases, however, the direction of an edge is fully determined, for example, by the presence of an immorality. To encode both of these cases, we use a graph that allows both directed and undirected edges, as defined in section 2.2. Indeed, as we show, the chain graph, or PDAG, representation (definition 2.21) provides precisely the right framework.

Definition 3.15 class PDAG

Let $\mathcal{G}$ be a DAG. A chain graph $\mathcal{K}$ is a class PDAG *of the equivalence class of $\mathcal{G}$ if shares the same skeleton as $\mathcal{G}$, and contains a directed edge $X \to Y$ if and only if all $\mathcal{G}'$ that are I-equivalent to $\mathcal{G}$ contain the edge $X \to Y$.*[2] ■

In other words, a class PDAG represents potential edge orientations in the equivalence classes. If the edge is directed, then all the members of the equivalence class agree on the orientation of the edge. If the edge is undirected, there are two DAGs in the equivalence class that disagree on the orientation of the edge.

For example, the networks in figure 3.5a–(c) are I-equivalent. The class PDAG of this equivalence class is the graph $X—Z—Y$, since both edges can be oriented in either direction in some member of the equivalence class. Note that, although both edges in this PDAG are undirected, not all joint orientations of these edges are in the equivalence class. As discussed earlier, setting the orientations $X \to Z \leftarrow Y$ results in the network of figure 3.5d, which does not belong this equivalence class. More generally, if the class PDAG has k undirected edges, the equivalence class can contain at most 2^k networks, but the actual number can be much smaller.

Can we effectively construct the class PDAG $\mathcal{K}$ for $\mathcal{G}^*$ from the reconstructed skeleton and immoralities? Clearly, edges involved in immoralities must be directed in $\mathcal{K}$. The obvious question is whether $\mathcal{K}$ can contain directed edges that are not involved in immoralities. In other words, can there be additional edges whose direction is necessarily the same in every member of the equivalence class? To understand this issue better, consider the following example:

Example 3.9

Consider the DAG of figure 3.11a. This DAG has a single immorality $A \to C \leftarrow B$. This immorality implies that the class PDAG of this DAG must have the arcs $A \to C$ and $B \to C$ directed, as

2. For consistency with standard terminology, we use the PDAG terminology when referring to the chain graph representing an I-equivalence class.

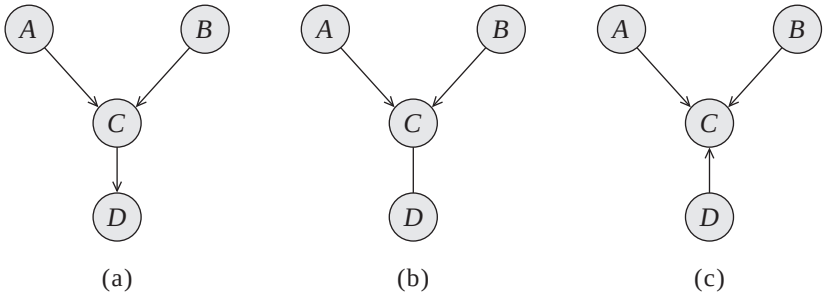


Figure 3.11 Simple example of compelled edges in the representation of an equivalence class. (a) Original DAG $\mathcal{G}^*$. (b) Skeleton of $\mathcal{G}^*$ annotated with immoralities. (c) a DAG that is not equivalent to $\mathcal{G}^*$.

shown in figure 3.11b. This PDAG representation suggests that the edge C—D can assume either orientation. Note, however, that the DAG of figure 3.11c, where we orient the edge between C and D as $D \rightarrow C$, contains additional immoralities (that is, $A \rightarrow C \leftarrow D$ and $B \rightarrow C \leftarrow D$). Thus, this DAG is not *equivalent to our original DAG.*

In this example, there is only one possible orientation of C—D that is consistent with the finding that A—C—D is not an immorality. Thus, we conclude that the class PDAG for the DAG of figure 3.11a is simply the DAG itself. In other words, the equivalence class of this DAG is a singleton.■

As this example shows, a negative result in an immorality test also provides information about edge orientation. In particular, in any case where the PDAG $\mathcal{K}$ contains a structure $X \rightarrow Y$—Z and there is no edge from X to Z, then we must orient the edge $Y \rightarrow Z$, for otherwise we would create an immorality $X \rightarrow Y \leftarrow Z$.

Some thought reveals that there are other local configurations of edges where some ways of orienting edges are inconsistent, forcing a particular direction for an edge. Each such configuration can be viewed as a local constraint on edge orientation, give rise to a rule that can be used to orient more edges in the PDAG. Three such rules are shown in figure 3.12.

Let us understand the intuition behind these rules. Rule R1 is precisely the one we discussed earlier. Rule R2 is derived from the standard acyclicity constraint: If we have the directed path $X \rightarrow Y \rightarrow Z$, and an undirected edge X—Z, we cannot direct the edge $X \leftarrow Z$ without creating a cycle. Hence, we can conclude that the edge must be directed $X \rightarrow Z$. The third rule seems a little more complex, but it is also easily motivated. Assume, by contradiction, that we direct the edge $Z \rightarrow X$. In this case, we cannot direct the edge X—Y_1 as $X \rightarrow Y_1$ without creating a cycle; thus, we must have $Y_1 \rightarrow X$. Similarly, we must have $Y_2 \rightarrow X$. But, in this case, $Y_1 \rightarrow X \leftarrow Y_2$ forms an immorality (as there is no edge between Y_1 and Y_2), which contradicts the fact that the edges X—Y_1 and X—Y_2 are undirected in the original PDAG.

These three rules can be applied constructively in an obvious way: A rule applies to a PDAG whenever the induced subgraph on a subset of variables exactly matches the graph on the left-hand side of the rule. In that case, we modify this subgraph to match the subgraph on the right-hand side of the rule. Note that, by applying one rule and orienting a previously undirected edge, we create a new graph. This might create a subgraph that matches the antecedent of a rule, enforcing the orientation of additional edges. This process, however, must terminate at

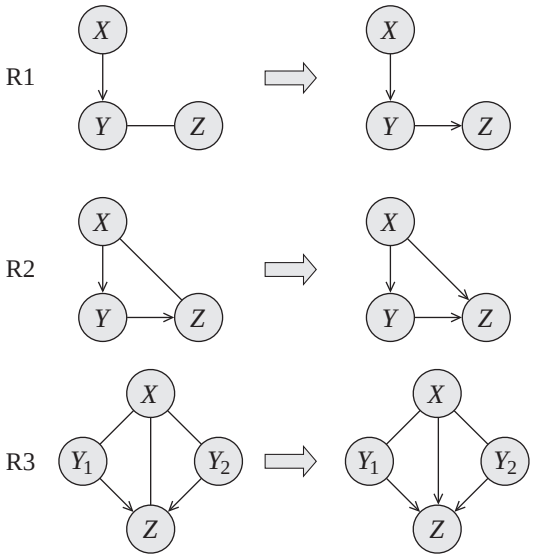


Figure 3.12 Rules for orienting edges in PDAG. Each rule lists a configuration of edges before and after an application of the rule.

some point (since we are only adding orientations at each step, and the number of edges is finite). This implies that iterated application of this local constraint to the graph (a process known as *constraint propagation*) is guaranteed to converge.

constraint propagation

Algorithm 3.5 Finding the class PDAG characterizing the P-map of a distribution P

```
    Procedure Build-PDAG (
      𝒳 = {X_1, ..., X_n}    // A specification of the random variables
      P    // Distribution of interest
    )
1     S, {U_{X_i,X_j}} ← Build-PMap-Skeleton(𝒳, P)
2     𝒦 ← Find-Immoralities(𝒳, S, {U_{X_i,X_j}})
3     while not converged
4       Find a subgraph in 𝒦 matching the left-hand side of a rule R1–R3
5       Replace the subgraph with the right-hand side of the rule
6     return K
```

Algorithm 3.5 implements this process. It builds an initial graph using Build-PMap-Skeleton and Mark-Immoralities, and then iteratively applies the three rules until convergence, that is, until we cannot find a subgraph that matches a left-hand side of any of the rules.

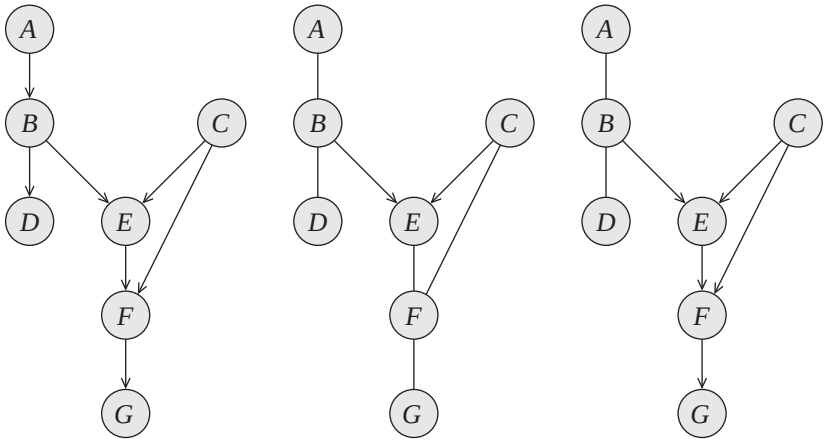


Figure 3.13 More complex example of compelled edges in the representation of an equivalence class. (a) Original DAG $\mathcal{G}^*$. (b) Skeleton of $\mathcal{G}^*$ annotated with immoralities. (c) Complete PDAG representation of the equivalence class of $\mathcal{G}^*$.

Example 3.10

Consider the DAG shown in figure 3.13a. After checking for immoralities, we find the graph shown in figure 3.13b. Now, we can start applying the preceding rules. For example, consider the variables B, E, and F. They induce a subgraph that matches the left-hand side of rule R1. Thus, we orient the edge between E and F to $E \rightarrow F$. Now, consider the variables C, E, and F. Their induced subgraph matches the left-hand side of rule R2, so we now orient the edge between C and F to $C \rightarrow F$. At this stage, if we consider the variables E, F, G, we can apply the rule R1, and orient the edge $F \rightarrow G$. (Alternatively, we could have arrived at the same orientation using C, F, and G.) The resulting PDAG is shown in figure 3.13c. ■

It seems fairly obvious that this algorithm is guaranteed to be sound: Any edge that is oriented by this procedure is, indeed, directed in exactly the same way in all of the members of the equivalence class. Much more surprising is the fact that it is also complete: Repeated application of these three local rules is guaranteed to capture all edge orientations in the equivalence class, without the need for additional global constraints. More precisely, we can prove that this algorithm produces the correct class PDAG for the distribution P:

Theorem 3.10

Let P be a distribution that has a P-map $\mathcal{G}^*$, and let $\mathcal{K}$ be the PDAG returned by Build-PDAG($\mathcal{X}$, P). *Then, $\mathcal{K}$ is a class PDAG of $\mathcal{G}^*$.*

The proof of this theorem can be decomposed into several aspects of correctness. We have already established the correctness of the skeleton found by Build-PMap-Skeleton. Thus, it remains to show that the directionality of the edges is correct. Specifically, we need to establish three basic facts:

- **Acyclicity:** The graph returned by Build-PDAG($\mathcal{X}$,P) is acyclic.

- **Soundness:** If $X \to Y \in \mathcal{K}$, then $X \to Y$ appears in all DAGs in $\mathcal{G}^*$'s I-equivalence class.
- **Completeness:** If $X — Y \in \mathcal{K}$, then we can find a DAG $\mathcal{G}$ that is I-equivalent to $\mathcal{G}^*$ such that $X \to Y \in \mathcal{G}$.

The last condition establishes completeness, since there is no constraint on the direction of the arc. In other words, the same condition can be used to prove the existence of a graph with $X \to Y$ and of a graph with $Y \to X$. Hence, it shows that either direction is possible within the equivalence class.

We begin with the soundness of the procedure.

Proposition 3.3 *Let P be a distribution that has a P-map $\mathcal{G}^*$, and let $\mathcal{K}$ be the graph returned by* Build-PDAG$(\mathcal{X}, P)$. *Then, if $X \to Y \in \mathcal{K}$, then $X \to Y$ appears in all DAGs in the I-equivalence class of $\mathcal{G}^*$.*

The proof is left as an exercise (exercise 3.23).

Next, we consider the acyclicity of the graph. We start by proving a property of graphs returned by the procedure. (Note that, once we prove that the graph returned by the procedure is the correct PDAG, it will follow that this property also holds for class PDAGs in general.)

Proposition 3.4 *Let $\mathcal{K}$ be the graph returned by* Build-PDAG. *Then, if $X \to Y \in \mathcal{K}$ and $Y — Z \in \mathcal{K}$, then $X \to Z \in \mathcal{K}$.*

The proof is left as an exercise (exercise 3.24).

Proposition 3.5 *Let $\mathcal{K}$ be the chain graph returned by* Build-PDAG. *Then $\mathcal{K}$ is acyclic.*

Proof Suppose, by way of contradiction, that $\mathcal{K}$ contains a cycle. That is, there is a (partially) directed path $X_1 \rightleftharpoons X_2 \rightleftharpoons \ldots \rightleftharpoons X_n \rightleftharpoons X_1$. Without loss of generality, assume that this path is the shortest cycle in $\mathcal{K}$. We claim that the path cannot contain an undirected edge. To see that, suppose that the the path contains the triplet $X_i \to X_{i+1} — X_{i+2}$. Then, invoking proposition 3.4, we have that $X_i \to X_{i+2} \in \mathcal{K}$, and thus, we can construct a shorter path without X_{i+1} that contains the edge $X_i \to X_{i+2}$. At this stage, we have a directed cycle $X_1 \to X_2 \to \ldots X_n \to X_1$. Using proposition 3.3, we conclude that this cycle appears in any DAG in the I-equivalence class, and in particular in $\mathcal{G}^*$. This conclusion contradicts the assumption that $\mathcal{G}^*$ is acyclic. It follows that $\mathcal{K}$ is acyclic. ■

The final step is the completeness proof. Again, we start by examining a property of the graph $\mathcal{K}$.

Proposition 3.6 *The PDAG $\mathcal{K}$ returned by* Build-PDAG *is necessarily chordal.*

The proof is left as an exercise (exercise 3.25).

This property allows us to characterize the structure of the PDAG $\mathcal{K}$ returned by Build-PDAG. Recall that, since $\mathcal{K}$ is an undirected chain graph, we can partition $\mathcal{X}$ into chain components $\boldsymbol{K}_1, \ldots, \boldsymbol{K}_\ell$, where each chain component contains variables that are connected by undirected edges (see definition 2.21). It turns out that, in an undirected chordal graph, we can orient any edge in any direction without creating an immorality.

Proposition 3.7 *Let $\mathcal{K}$ be a undirected chordal graph over $\mathcal{X}$, and let $X, Y \in \mathcal{X}$. Then, there is a DAG $\mathcal{G}$ such that*

(a) The skeleton of $\mathcal{G}$ is $\mathcal{K}$.

(b) $\mathcal{G}$ does not contain immoralities.

(c) $X \rightarrow Y \in \mathcal{G}$.

The proof of this proposition requires some additional machinery that we introduce in chapter 4, so we defer the proof to that chapter.

Using this proposition, we see that we can orient edges in the chain component $\boldsymbol{K}_j$ without introducing immoralities within the component. We still need to ensure that orienting an edge X—Y within a component cannot introduce an immorality involving edges from outside the component. To see why this situation cannot occur, suppose we orient the edge $X \rightarrow Y$, and suppose that $Z \rightarrow Y \in \mathcal{K}$. This seems like a potential immorality. However, applying proposition 3.4, we see that since $Z \rightarrow Y$ and Y—X are in $\mathcal{K}$, then so must be $Z \rightarrow X$. Since Z is a parent of both X and Y, we have that $X \rightarrow Y \leftarrow Z$ is not an immorality. This argument applies to any edge we orient within an undirected component, and thus no new immoralities are introduced.

With these tools, we can complete the completeness proof of Build-PDAG.

Proposition 3.8 *Let P be a distribution that has a P-map $\mathcal{G}^*$, and let $\mathcal{K}$ be the graph returned by* Build-PDAG$(\mathcal{X}, P)$. *If X—$Y \in \mathcal{K}$, then we can find a DAG $\mathcal{G}$ that is I-equivalent to $\mathcal{G}^*$ such that $X \rightarrow Y \in \mathcal{G}$.*

Proof Suppose we have an undirected edge X—$Y \in \mathcal{K}$. We want to show that there is a DAG $\mathcal{G}$ that has the same skeleton and immoralities as $\mathcal{K}$ such that $X \rightarrow Y \in \mathcal{G}$. If can build such a graph $\mathcal{G}$, then clearly it is in the I-equivalence class of $\mathcal{G}^*$.

The construction is simple. We start with the chain component that contains X—Y, and use proposition 3.7 to orient the edges in the component so that $X \rightarrow Y$ is in the resulting DAG. Then, we use the same construction to orient all other chain components. Since the chain components are ordered and acyclic, and our orientation of each chain component is acyclic, the resulting directed graph is acyclic. Moreover, as shown, the new orientation in each component does not introduce immoralities. Thus, the resulting DAG has exactly the same skeleton and immoralities as $\mathcal{K}$. ■

3.5 Summary

In this chapter, we discussed the issue of specifying a high-dimensional joint distribution compactly by exploiting its independence properties. We provided two complementary definitions of a Bayesian network. The first is as a directed graph $\mathcal{G}$, annotated with a set of conditional probability distributions $P(X_i \mid \mathrm{Pa}_{X_i})$. The network together with the CPDs define a distribution via the chain rule for Bayesian networks. In this case, we say that P factorizes over $\mathcal{G}$. We also defined the independence assumptions associated with the graph: the local independencies, the set of basic independence assumptions induced by the network structure; and the larger set of global independencies that are derived from the d-separation criterion. We showed the

equivalence of these three fundamental notions: P factorizes over $\mathcal{G}$ if and only if P satisfies the local independencies of $\mathcal{G}$, which holds if and only if P satisfies the global independencies derived from d-separation. This result shows the equivalence of our two views of a Bayesian network: as a scaffolding for factoring a probability distribution P, and as a representation of a set of independence assumptions that hold for P. We also showed that the set of independencies derived from d-separation is a complete characterization of the independence properties that are implied by the graph structure alone, rather than by properties of a specific distribution over $\mathcal{G}$.

We defined a set of basic notions that use the characterization of a graph as a set of independencies. We defined the notion of a *minimal I-map* and showed that almost every distribution has multiple minimal I-maps, but that a minimal I-map for P does not necessarily capture all of the independence properties in P. We then defined a more stringent notion of a *perfect map*, and showed that not every distribution has a perfect map. We defined *I-equivalence*, which captures an independence-equivalence relationship between two graphs, one where they specify precisely the same set of independencies.

Finally, we defined the notion of a *class PDAG*, a partially directed graph that provides a compact representation for an entire I-equivalence class, and we provided an algorithm for constructing this graph.

These definitions and results are fundamental properties of the Bayesian network representation and its semantics. Some of the algorithms that we discussed are never used as is; for example, we never directly use the procedure to find a minimal I-map given an explicit representation of the distribution. However, these results are crucial to understanding the cases where we can construct a Bayesian network that reflects our understanding of a given domain, and what the resulting network means.

3.6 Relevant Literature

The use of a directed graph as a framework for analyzing properties of distributions can be traced back to the path analysis of Wright (1921, 1934).

influence diagram

The use of a directed acyclic graph to encode a general probability distribution (not within a specific domain) was first proposed within the context of *influence diagrams*, a decision-theoretic framework for making decisions under uncertainty (see chapter 23). Within this setting, Howard and Matheson (1984b) and Smith (1989) both proved the equivalence between the ability to represent a distribution as a DAG and the local independencies (our theorem 3.1 and theorem 3.2).

The notion of Bayesian networks as a qualitative data structure encoding independence relationships was first proposed by Pearl and his colleagues in a series of papers (for example, Verma and Pearl 1988; Geiger and Pearl 1988; Geiger et al. 1989, 1990), and in Pearl's book *Probabilistic Reasoning in Intelligent Systems* (Pearl 1988). Our presentation of I-maps, P-maps, and Bayesian networks largely follows the trajectory laid forth in this body of work.

The definition of d-separation was first set forth by Pearl (1986b), although without formal justification. The soundness of d-separation was shown by Verma (1988), and its completeness for the case of Gaussian distributions by Geiger and Pearl (1993). The measure-theoretic notion of completeness of d-separation, stating that almost all distributions are faithful (theorem 3.5), was shown by Meek (1995b). Several papers have been written exploring the yet stronger notion

BayesBall

of completeness for d-separation (faithfulness for all distributions that are minimal I-maps), in various subclasses of models (for example, Becker et al. 2000). The *BayesBall* algorithm, an elegant and efficient algorithm for d-separation and a class of related problems, was proposed by (Shachter 1998).

The notion of I-equivalence was defined by Verma and Pearl (1990, 1992), who also provided and proved the graph-theoretic characterization of theorem 3.8. Chickering (1995) provided the alternative characterization of I-equivalence in terms of covered edge reversal. This definition provides an easy mechanism for proving important properties of I-equivalent networks. As we will see later in the book, the notion of I-equivalence class plays an important role in identifying networks, particularly when learning networks from data. The first algorithm for constructing a perfect map for a distribution, in the form of an I-equivalence class, was proposed by Pearl and Verma (1991); Verma and Pearl (1992). This algorithm was subsequently extended by Spirtes et al. (1993) and by Meek (1995a). Meek also provides an algorithm for finding all of the directed edges that occur in every member of the I-equivalence class.

inclusion

A notion related to I-equivalence is that of *inclusion*, where the set of independencies $\mathcal{I}(\mathcal{G}')$ is *included* in the set of independencies $\mathcal{I}(\mathcal{G})$ (so that $\mathcal{G}$ is an I-map for any distribution that factorizes over $\mathcal{G}'$). Shachter (1989) showed how to construct a graph $\mathcal{G}'$ that includes a graph $\mathcal{G}$, but with one edge reversed. Meek (1997) conjectured that inclusion holds if and only if one can transform $\mathcal{G}$ to $\mathcal{G}'$ using the operations of edge addition and covered edge reversal. A limited version of this conjecture was subsequently proved by Kočka, Bouckaert, and Studený (2001).

The naive Bayes model, although naturally represented as a graphical model, far predates this view. It was applied with great success within expert systems in the 1960s and 1970s (de Bombal et al. 1972; Gorry and Barnett 1968; Warner et al. 1961). It has also seen significant use as a simple yet highly effective method for classification tasks in machine learning, starting as early as the 1970s (for example, Duda and Hart 1973), and continuing to this day.

The general usefulness of the types of reasoning patterns supported by a Bayesian network, including the very important pattern of intercausal reasoning, was one of the key points raised by Pearl in his book (Pearl 1988). These qualitative patterns were subsequently formalized by Wellman (1990) in his framework of *qualitative probabilistic networks*, which explicitly annotate arcs with the direction of influence of one variable on another. This framework has been used to facilitate knowledge elicitation and knowledge-guided learning (Renooij and van der Gaag 2002; Hartemink et al. 2002) and to provide verbal explanations of probabilistic inference (Druzdzel 1993).

qualitative probabilistic networks

There have been many applications of the Bayesian network framework in the context of real-world problems. The idea of using directed graphs as a model for genetic inheritance appeared as far back as the work on path analysis of Wright (1921, 1934). A presentation much closer to modern-day Bayesian networks was proposed by Elston and colleagues in the 1970s (Elston and Stewart 1971; Lange and Elston 1975). More recent developments include the development of better algorithms for inference using these models (for example, Kong 1991; Becker et al. 1998; Friedman et al. 2000) and the construction of systems for genetic linkage analysis based on this technology (Szolovits and Pauker 1992; Schäffer 1996).

Many of the first applications of the Bayesian network framework were to medical expert systems. The Pathfinder system is largely the work of David Heckerman and his colleagues (Heckerman and Nathwani 1992a; Heckerman et al. 1992; Heckerman and Nathwani 1992b). The success of this system as a diagnostic tool, including its ability to outperform expert physicians, was one

of the major factors that led to the rise in popularity of probabilistic methods in the early 1990s. Several other large diagnostic networks were developed around the same period, including MUNIN (Andreassen et al. 1989), a network of over 1000 nodes used for interpreting electromyographic data, and QMR-DT (Shwe et al. 1991; Middleton et al. 1991), a probabilistic reconstruction of the QMR/INTERNIST system (Miller et al. 1982) for general medical diagnosis.

The problem of knowledge acquisition of network models has received some attention. Probability elicitation is a long-standing question in decision analysis; see, for example, Spetzler and von Holstein (1975); Chesley (1978). Unfortunately, elicitation of probabilities from humans is a difficult process, and one subject to numerous biases (Tversky and Kahneman 1974; Daneshkhah 2004). Shachter and Heckerman (1987) propose the "backward elicitation" approach for obtaining both the network structure and the parameters from an expert. *Similarity networks* (Heckerman and Nathwani 1992a; Geiger and Heckerman 1996) generalize this idea by allowing an expert to construct several small networks for differentiating between "competing" diagnoses, and then superimposing them to construct a single large network. Morgan and Henrion (1990) provide an overview of knowledge elicitation methods.

similarity network

The difficulties in eliciting accurate probability estimates from experts are well recognized across a wide range of disciplines. In the specific context of Bayesian networks, this issue has been tackled in several ways. First, there has been both empirical (Pradhan et al. 1996) and theoretical (Chan and Darwiche 2002) analysis of the extent to which the choice of parameters affects the conclusions of the inference. Overall, the results suggest that even fairly significant changes to network parameters cause only small degradations in performance, except when the changes relate to extreme parameters — those very close to 0 and 1. Second, the concept of *sensitivity analysis* (Morgan and Henrion 1990) is used to allow researchers to evaluate the sensitivity of their specific network to variations in parameters. Largely, sensitivity has been measured using the derivative of network queries relative to various parameters (Laskey 1995; Castillo et al. 1997b; Kjærulff and van der Gaag 2000; Chan and Darwiche 2002), with the focus of most of the work being on properties of sensitivity values and on efficient algorithms for estimating them.

sensitivity analysis

As pointed out by Pearl (1988), the notion of a Bayesian network structure as a representation of independence relationships is a fundamental one, which transcends the specifics of probabilistic representations. There have been many proposed variants of Bayesian networks that use a nonprobabilistic "parameterization" of the local dependency models. Examples include various logical calculi (Darwiche 1993), Dempster-Shafer belief functions (Shenoy 1989), possibility values (Dubois and Prade 1990), qualitative (order-of-magnitude) probabilities (known as kappa rankings; Darwiche and Goldszmidt 1994), and interval constraints on probabilities (Fertig and Breese 1989; Cozman 2000).

The acyclicity constraint of Bayesian networks has led to many concerns about its ability to express certain types of interactions. There have been many proposals intended to address this limitation. Markov networks, based on undirected graphs, present a solution for certain types of interactions; this class of probability models are described in chapter 4. Dynamic Bayesian networks "stretch out" the interactions over time, therefore providing an acyclic version of feedback loops; these models are are described in section 6.2.

There has also been some work on directed models that encode cyclic dependencies directly. *Cyclic graphical models* (Richardson 1994; Spirtes 1995; Koster 1996; Pearl and Dechter 1996) are based on distributions over systems of simultaneous linear equations. These models are a

cyclic graphical model

natural generalization of Gaussian Bayesian networks (see chapter 7), and are also associated with notions of d-separation or I-equivalence. Spirtes (1995) shows that this connection breaks down when the system of equations is nonlinear and provides a weaker version for the cyclic case.

dependency network

Dependency networks (Heckerman et al. 2000) encode a set of local dependency models, representing the conditional distribution of each variable on all of the others (which can be compactly represented by its dependence on its Markov blanket). A dependency network represents a probability distribution only indirectly, and is only guaranteed to be coherent under certain conditions. However, it provides a local model of dependencies that is very naturally interpreted by people.

3.7 Exercises

Exercise 3.1

Provide an example of a distribution $P(X_1, X_2, X_3)$ where for each $i \neq j$, we have that $(X_i \perp X_j) \in \mathcal{I}(P)$, but we also have that $(X_1, X_2 \perp X_3) \notin \mathcal{I}(P)$.

Exercise 3.2

a. Show that the naive Bayes factorization of equation (3.7) follows from the naive Bayes independence assumptions of equation (3.6).

b. Show that equation (3.8) follows from equation (3.7).

c. Show that, if all the variables $C, X_1, \ldots, X_n$ are binary-valued, then $\log \frac{P(C=c^1|x_1,\ldots,x_n)}{P(C=c^2|x_1,\ldots,x_n)}$ is a linear function of the value of the finding variables, that is, can be written as $\sum_{i=1}^{n} \alpha_i X_i + \alpha_0$ (where $X_i = 0$ if $X = x^0$ and 1 otherwise).

Exercise 3.3

Consider a simple example (due to Pearl), where a burglar alarm (A) can be set off by either a burglary (B) or an earthquake (E).

a. Define constraints on the CPD of $P(A \mid B, E)$ that imply the *explaining away* property.

b. Show that if our model is such that the alarm always (deterministically) goes off whenever there is a earthquake:

$$P(a^1 \mid b^1, e^1) = P(a^1 \mid b^0, e^1) = 1$$

then $P(b^1 \mid a^1, e^1) = P(b^1)$, that is, observing an earthquake provides a full explanation for the alarm.

Exercise 3.4

We have mentioned that explaining away is one type of intercausal reasoning, but that other type of intercausal interactions are also possible. Provide a realistic example that exhibits the opposite type of interaction. More precisely, consider a v-structure $X \rightarrow Z \leftarrow Y$ over three binary-valued variables. Construct a CPD $P(Z \mid X, Y)$ such that:

- X and Y both increase the probability of the effect, that is, $P(z^1 \mid x^1) > P(z^1)$ and $P(z^1 \mid y^1) > P(z^1)$,
- each of X and Y increases the probability of the other, that is, $P(x^1 \mid z^1) < P(x^1 \mid y^1, z^1)$, and similarly $P(y^1 \mid z^1) < P(y^1 \mid x^1, z^1)$.

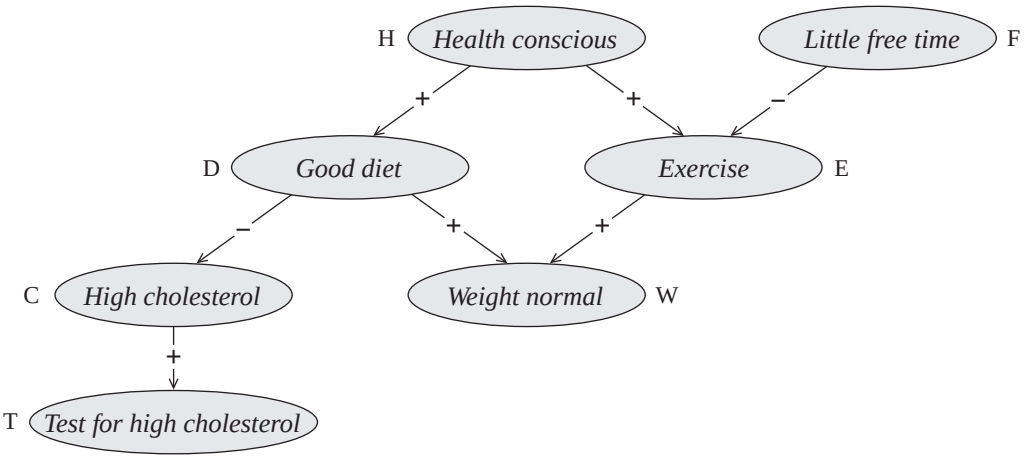


Figure 3.14 A Bayesian network with qualitative influences

Note that strong (rather than weak) inequality must hold in all cases.

Your example should be realistic, that is, X, Y, Z should correspond to real-world variables, and the CPD should be reasonable.

Exercise 3.5

Consider the Bayesian network of figure 3.14.

Assume that all variables are binary-valued. We do not know the CPDs, but do know how each random variable *qualitatively* affects its children. The influences, shown in the figure, have the following interpretation:

- $X \xrightarrow{+} Y$ means $P(y^1 \mid x^1, \boldsymbol{u}) > P(y^1 \mid x^0, \boldsymbol{u})$, for all values $\boldsymbol{u}$ of Y's other parents.
- $X \xrightarrow{-} Y$ means $P(y^1 \mid x^1, \boldsymbol{u}) < P(y^1 \mid x^0, \boldsymbol{u})$, for all values $\boldsymbol{u}$ of Y's other parents.

We also assume explaining away as the interaction for all cases of intercausal reasoning.

For each of the following pairs of conditional probability queries, use the information in the network to determine if one is larger than the other, if they are equal, or if they are incomparable. For each pair of queries, indicate all relevant active trails, and their direction of influence.

(a)	$P(t^1 \mid d^1)$	$P(t^1)$
(b)	$P(d^1 \mid t^0)$	$P(d^1)$
(c)	$P(h^1 \mid e^1, f^1)$	$P(h^1 \mid e^1)$
(d)	$P(c^1 \mid f^0)$	$P(c^1)$
(e)	$P(c^1 \mid h^0)$	$P(c^1)$
(f)	$P(c^1 \mid h^0, f^0)$	$P(c^1 \mid h^0)$
(g)	$P(d^1 \mid h^1, e^0)$	$P(d^1 \mid h^1)$
(h)	$P(d^1 \mid e^1, f^0, w^1)$	$P(d^1 \mid e^1, f^0)$
(i)	$P(t^1 \mid w^1, f^0)$	$P(t^1 \mid w^1)$

Exercise 3.6

Consider a set of variables $X_1, \ldots, X_n$ where each X_i has $|Val(X_i)| = \ell$.

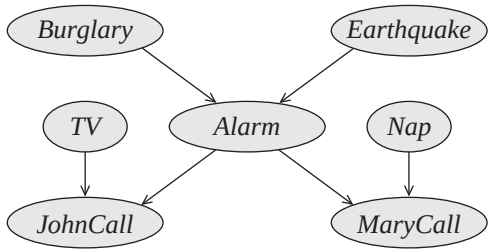


Figure 3.15 A simple network for a burglary alarm **domain**

a. Assume that we have a Bayesian network over $X_1, \ldots, X_n$, such that each node has at most k parents. What is a simple upper bound on the number of independent parameters in the Bayesian network? How many independent parameters are in the full joint distribution over $X_1, \ldots, X_n$?

b. Now, assume that each variable X_i has the parents $X_1, \ldots, X_{i-1}$. How many independent parameters are there in the Bayesian network? What can you conclude about the expressive power of this type of network?

c. Now, consider a naive Bayes model where $X_1, \ldots, X_n$ are evidence variables, and we have an additional class variable C, which has k possible values $c_1, \ldots, c_k$. How many independent parameters are required to specify the naive Bayes model? How many independent parameters are required for an explicit representation of the joint distribution?

Exercise 3.7

Show how you could efficiently compute the distribution over a variable X_i given some assignment to all the other variables in the network: $P(X_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. Your procedure should not require the construction of the entire joint distribution $P(X_1, \ldots, X_n)$. Specify the computational complexity of your procedure.

Exercise 3.8

Let $\mathcal{B} = (\mathcal{G}, P)$ be a Bayesian network over some set of variables $\mathcal{X}$. Consider some subset of evidence nodes $\boldsymbol{Z}$, and let $\boldsymbol{X}$ be all of the ancestors of the nodes in $\boldsymbol{Z}$. Let $\mathcal{B}'$ be a network over the induced subgraph over $\boldsymbol{X}$, where the CPD for every node $X \in \boldsymbol{X}$ is the same in $\mathcal{B}'$ as in $\mathcal{B}$. Prove that the joint distribution over $\boldsymbol{X}$ is the same in $\mathcal{B}$ and in $\mathcal{B}'$. The nodes in $\mathcal{X} - \boldsymbol{X}$ are called *barren nodes* relative to $\boldsymbol{X}$, because (when not instantiated) they are irrelevant to computations concerning $\boldsymbol{X}$.

barren node

Exercise 3.9⋆

Prove theorem 3.2 for a general BN structure $\mathcal{G}$. Your proof should not use the soundness of d-separation.

Exercise 3.10

Prove that the global independencies, derived from d-separation, imply the local independencies. In other words, prove that a node is d-separated from its nondescendants given its parents.

Exercise 3.11⋆

One operation on Bayesian networks that arises in many settings is the marginalization of some node in the network.

a. Consider the Burglary Alarm network $\mathcal{B}$ shown in figure 3.15. Construct a Bayesian network $\mathcal{B}'$ over all of the nodes except for *Alarm* that is a minimal I-map for the marginal distribution $P_{\mathcal{B}}(B, E, T, N, J, M)$. Be sure to get *all* dependencies that remain from the original network.

b. Generalize the procedure you used to solve the preceding problem into a node elimination algorithm. That is, define an algorithm that transforms the structure of $\mathcal{G}$ into $\mathcal{G}'$ such that one of the nodes X_i of $\mathcal{G}$ is not in $\mathcal{G}'$ and $\mathcal{G}'$ is an I-map of the marginal distribution over the remaining variables as defined by $\mathcal{G}$.

Exercise 3.12⋆⋆

edge reversal

Another operation on Bayesian networks that arises often is *edge reversal*. This involves transforming a Bayesian network $\mathcal{G}$ containing nodes X and Y as well as arc $X \rightarrow Y$ into another Bayesian network $\mathcal{G}'$ with reversed arc $Y \rightarrow X$. However, we want $\mathcal{G}'$ to represent the same distribution as $\mathcal{G}$; therefore, $\mathcal{G}'$ will need to be an I-map of the original distribution.

a. Consider the Bayesian network structure of figure 3.15. Suppose we wish to reverse the arc $B \rightarrow A$. What additional minimal modifications to the structure of the network are needed to ensure that the new network is an I-map of the original distribution? Your network should not reverse any additional edges, and it should differ only minimally from the original in terms of the number of edge additions or deletions. Justify your response.

b. Now consider a general Bayesian network $\mathcal{G}$. For simplicity, assume that the arc $X \rightarrow Y$ is the only directed trail from X to Y. Define a general procedure for reversing the arc $X \rightarrow Y$, that is, for constructing a graph $\mathcal{G}'$ is an I-map for the original distribution, but that contains an arc $Y \rightarrow X$ and otherwise differs minimally from $\mathcal{G}$ (in the same sense as before). Justify your response.

c. Suppose that we use the preceding method to transform $\mathcal{G}$ into a graph $\mathcal{G}'$ with a reversed arc between X and Y. Now, suppose we reverse that arc *back* to its original direction in $\mathcal{G}$ by repeating the preceding method, transforming $\mathcal{G}'$ into $\mathcal{G}''$. Are we guaranteed that the final network structure is equivalent to the original network structure ($\mathcal{G} = \mathcal{G}''$)?

Exercise 3.13⋆

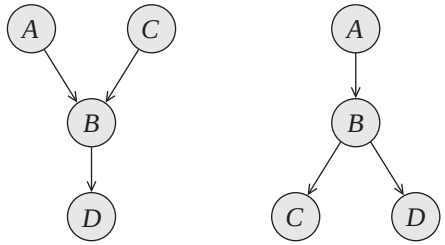
Let $\mathcal{B} = (\mathcal{G}, P)$ be a Bayesian network over $\mathcal{X}$. The Bayesian network is parameterized by a set of CPD parameters of the form $\theta_{x|\boldsymbol{u}}$ for $X \in \mathcal{X}$, $\boldsymbol{U} = \mathrm{Pa}_X^{\mathcal{G}}$, $x \in Val(X)$, $\boldsymbol{u} \in Val(\boldsymbol{U})$. Consider any conditional independence statement of the form $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$. Show how this statement translates into a set of polynomial equalities over the set of CPD parameters $\theta_{x|\boldsymbol{u}}$. (Note: A polynomial equality is an assertion of the form $a\theta_1^2 + b\theta_1\theta_2 + c\theta_2^3 + d = 0$.)

Exercise 3.14⋆

Prove theorem 3.6.

Exercise 3.15

Consider the two networks:



(a) (b)

For each of them, determine whether there can be any other Bayesian network that is I-equivalent to it.

Exercise 3.16⋆

Prove theorem 3.7.

Exercise 3.17⋆⋆

We proved earlier that two networks that have the same skeleton and v-structures imply the same conditional independence assumptions. As shown, this condition is not an if and only if. Two networks can have different v-structures, yet still imply the same conditional independence assumptions. In this problem, you will provide a condition that precisely relates I-equivalence and similarity of network structure.

minimal active trail

a. A key notion in this question is that of a *minimal active trail*. We define an active trail $X_1 \dots X_m$ to be *minimal* if there is no other active trail from X_1 to X_m that "shortcuts" some of the nodes, that is, there is no active trail $X_1 \rightleftharpoons X_{i_1} \rightleftharpoons \dots \rightleftharpoons X_{i_k} \rightleftharpoons X_m$ for $1 < i_1 < \dots < i_k < m$.

   Our first goal is to analyze the types of "triangles" that can occur in a minimal active trail, that is, cases where we have $X_{i-1} \rightleftharpoons X_i \rightleftharpoons X_{i+1}$ with a direct edge between X_{i-1} and X_{i+1}. Prove that the only possible triangle in a minimal active trail is one where $X_{i-1} \leftarrow X_i \rightarrow X_{i+1}$, with an edge between X_{i-1} and X_{i+1}, and where either X_{i-1} or X_{i+1} are the center of a v-structure in the trail.

b. Now, consider two networks $\mathcal{G}_1$ and $\mathcal{G}_2$ that have the same skeleton and same immoralities. Prove, using the notion of minimal active trail, that $\mathcal{G}_1$ and $\mathcal{G}_2$ imply precisely the same conditional independence assumptions, that is, that if X and Y are d-separated given $\boldsymbol{Z}$ in $\mathcal{G}_1$, then X and Y are also d-separated given $\boldsymbol{Z}$ in $\mathcal{G}_2$.

c. Finally, prove the other direction. That is, prove that two networks $\mathcal{G}_1$ and $\mathcal{G}_2$ that induce the same conditional independence assumptions must have the same skeleton and the same immoralities.

Exercise 3.18⋆

In this exercise, you will prove theorem 3.9. This result provides an alternative reformulation of I-equivalence in terms of local operations on the graph structure.

covered edge

a. Let $\mathcal{G}$ be a directed graph with a *covered edge* $X \rightarrow Y$ (as in definition 3.12), and $\mathcal{G}'$ the graph that results by reversing the edge $X \rightarrow Y$ to produce $Y \rightarrow X$, but leaving everything else unchanged. Prove that $\mathcal{G}$ and $\mathcal{G}'$ are I-equivalent.

b. Provide a counterexample to this result in the case where $X \rightarrow Y$ is not a covered edge.

c. Now, prove that for every pair of I-equivalent networks $\mathcal{G}$ and $\mathcal{G}'$, there exists a sequence of covered edge reversal operations that converts $\mathcal{G}$ to $\mathcal{G}'$. Your proof should show how to construct this sequence.

Exercise 3.19⋆

Prove lemma 3.2.

Exercise 3.20⋆

requisite CPD

In this question, we will consider the sensitivity of a particular query $P(X \mid \boldsymbol{Y})$ to the CPD of a particular node Z. Let X and Z be nodes, and $\boldsymbol{Y}$ be a set of nodes. We say that Z has a *requisite CPD* for answering the query $P(\boldsymbol{X} \mid \boldsymbol{Y})$ if there are two networks $\mathcal{B}_1$ and $\mathcal{B}_2$ that have identical graph structure $\mathcal{G}$ and identical CPDs everywhere except at the node Z, and where $P_{\mathcal{B}_1}(X \mid \boldsymbol{Y}) \neq P_{\mathcal{B}_2}(X \mid \boldsymbol{Y})$; in other words, the CPD of Z affects the answer to this query.

This type of analysis is useful in various settings, including determining which CPDs we need to acquire for a certain query (and others that we discuss later in the book).

Show that we can test whether Z is a requisite probability node for $P(X \mid \boldsymbol{Y})$ using the following procedure: We modify $\mathcal{G}$ into a graph $\mathcal{G}'$ that contains a new "dummy" parent $\widehat{Z}$, and then test whether $\widehat{Z}$ has an active trail to $\boldsymbol{X}$ given $\boldsymbol{Y}$.

- Show that this is a sound criterion for determining whether Z is a requisite probability node for $P(X \mid \boldsymbol{Y})$ in $\mathcal{G}$, that is, for all pairs of networks $\mathcal{B}_1, \mathcal{B}_2$ as before, $P_{\mathcal{B}_1}(X \mid \boldsymbol{Y}) = P_{\mathcal{B}_2}(X \mid \boldsymbol{Y})$.
- Show that this criterion is weakly complete (like d-separation), in the sense that, if it fails to identify Z as requisite in $\mathcal{G}$, there exists some pair of networks $\mathcal{B}_1, \mathcal{B}_2$ as before, $P_{\mathcal{B}_1}(X \mid \boldsymbol{Y}) \neq P_{\mathcal{B}_2}(X \mid \boldsymbol{Y})$.

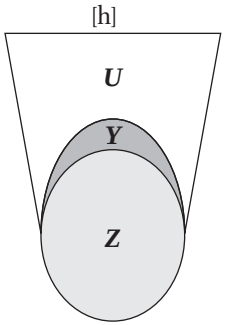


Figure 3.16 Illustration of the concept of a self-contained set

Exercise 3.21⋆

Define a set $\boldsymbol{Z}$ of nodes to be *self-contained* if, for every pair of nodes $A, B \in \boldsymbol{Z}$, and any *directed* path between A and B, all nodes along the trail are also in $\boldsymbol{Z}$.

a. Consider a self-contained set $\boldsymbol{Z}$, and let $\boldsymbol{Y}$ be the set of all nodes that are a parent of some node in $\boldsymbol{Z}$ but are not themselves in $\boldsymbol{Z}$. Let $\boldsymbol{U}$ be the set of nodes that are an ancestor of some node in $\boldsymbol{Z}$ but that are not already in $\boldsymbol{Y} \cup \boldsymbol{Z}$. (See figure 3.16.)

   Prove, based on the d-separation properties of the network, that $(\boldsymbol{Z} \perp \boldsymbol{U} \mid \boldsymbol{Y})$. Make sure that your proof covers all possible cases.

b. Provide a counterexample to this result if we retract the assumption that $\boldsymbol{Z}$ is self-contained. (Hint: 4 nodes are enough.)

Exercise 3.22⋆

We showed that the algorithm Build-PMap-Skeleton of algorithm 3.3 constructs the skeleton of the P-map of a distribution P if P has a P-map (and that P-map has indegrees bounded by the parameter d). In this question, we ask you consider what happens if P does not have a P-map.

There are two types of errors we might want to consider:

- **Missing edges**: The edge $X \rightleftharpoons Y$ appears in all the minimal I-maps of P, yet $X—Y$ is not in the skeleton S returned by Build-PMap-Skeleton.
- **Spurious edges**: The edge $X \rightleftharpoons Y$ does not appear in all of the minimal I-maps of P (but may appear in some of them), yet $X—Y$ is in the skeleton S returned by Build-PMap-Skeleton.

For each of these two types of errors, either prove that they cannot happen, or provide a counterexample (that is, a distribution P for which Build-PMap-Skeleton makes that type of an error).

Exercise 3.23⋆

In this exercise, we prove proposition 3.3. To help us with the proof, we need an auxiliary definition. We say that a partially directed graph $\mathcal{K}$ is a *partial class graph* for a DAG $\mathcal{G}^*$ if

a. $\mathcal{K}$ has the same skeleton as $\mathcal{G}^*$;
b. $\mathcal{K}$ has the same immoralities as $\mathcal{G}^*$;
c. if $X \rightarrow Y \in \mathcal{K}$, then $X \rightarrow Y \in \mathcal{G}$ for any DAG $\mathcal{G}$ that is I-equivalent to $\mathcal{G}^*$.

Clearly, the graph returned by by Mark-Immoralities is a partial class graph of $\mathcal{G}^*$.

Prove that if $\mathcal{K}$ is a partial class graph of $\mathcal{G}^*$, and we apply one of the rules R1–R3 of figure 3.12, then the resulting graph is also a partial class graph $\mathcal{G}^*$. Use this result to prove proposition 3.3 by induction.

Exercise 3.24⋆

Prove proposition 3.4. Hint: consider the different cases by which the edge $X \rightarrow Y$ was oriented during the procedure.

Exercise 3.25

Prove proposition 3.6. Hint: Show that this property is true of the graph returned by Mark-Immoralities.

Exercise 3.26

Implement an efficient algorithm that takes a Bayesian network over a set of variables $\mathcal{X}$ and a full instantiation ξ to $\mathcal{X}$, and computes the probability of ξ according to the network.

Exercise 3.27

Implement Reachable of algorithm 3.1.

Exercise 3.28⋆

Implement an efficient algorithm that determines, for a given set $\boldsymbol{Z}$ of observed variables and all pairs of nodes X and Y, whether X, Y are d-separated in $\mathcal{G}$ given $\boldsymbol{Z}$. Your algorithm should be significantly more efficient than simply running Reachable of algorithm 3.1 separately for each possible source variable X_i.

4 *Undirected Graphical Models*

So far, we have dealt only with directed graphical models, or Bayesian networks. These models are useful because both the structure and the parameters provide a natural representation for many types of real-world domains. In this chapter, we turn our attention to another important class of graphical models, defined on the basis of undirected graphs.

As we will see, these models are useful in modeling a variety of phenomena where one cannot naturally ascribe a directionality to the interaction between variables. Furthermore, the undirected models also offer a different and often simpler perspective on directed models, in terms of both the independence structure and the inference task. We also introduce a combined framework that allows both directed and undirected edges. We note that, unlike our results in the previous chapter, some of the results in this chapter require that we restrict attention to distributions over discrete state spaces.

4.1 The Misconception Example

To motivate our discussion of an alternative graphical representation, let us reexamine the Misconception example of section 3.4.2 (example 3.8). In this example, we have four students who get together in pairs to work on their homework for a class. The pairs that meet are shown via the edges in the undirected graph of figure 3.10a.

As we discussed, we intuitively want to model a distribution that satisfies $(A \perp C \mid \{B, D\})$ and $(B \perp D \mid \{A, C\})$, but no other independencies. As we showed, these independencies cannot be naturally captured in a Bayesian network: any Bayesian network I-map of such a distribution would necessarily have extraneous edges, and it would not capture at least one of the desired independence statements. More broadly, a Bayesian network requires that we ascribe a directionality to each influence. In this case, the interactions between the variables seem symmetrical, and we would like a model that allows us to represent these correlations without forcing a specific direction to the influence.

Markov network

A representation that implements this intuition is an undirected graph. As in a Bayesian network, the nodes in the graph of a *Markov network* represent the variables, and the edges correspond to a notion of direct probabilistic interaction between the neighboring variables — an interaction that is not mediated by any other variable in the network. In this case, the graph of figure 3.10, which captures the interacting pairs, is precisely the Markov network structure that captures our intuitions for this example. As we will see, this similarity is not an accident.

$\phi_1(A,B)$			$\phi_2(B,C)$			$\phi_3(C,D)$			$\phi_4(D,A)$		
a^0	b^0	30	b^0	c^0	100	c^0	d^0	1	d^0	a^0	100
a^0	b^1	5	b^0	c^1	1	c^0	d^1	100	d^0	a^1	1
a^1	b^0	1	b^1	c^0	1	c^1	d^0	100	d^1	a^0	1
a^1	b^1	10	b^1	c^1	100	c^1	d^1	1	d^1	a^1	100
(a)			(b)			(c)			(d)		

Figure 4.1 Factors for the Misconception **example**

The remaining question is how to parameterize this undirected graph. Because the interaction is not directed, there is no reason to use a standard CPD, where we represent the distribution over one node given others. Rather, we need a more symmetric parameterization. Intuitively, what we want to capture is the *affinities* between related variables. For example, we might want to represent the fact that Alice and Bob are more likely to agree than to disagree. We associate with A, B a general-purpose function, also called a *factor*:

Definition 4.1 factor

Let $\boldsymbol{D}$ be a set of random variables. We define a factor ϕ *to be a function from $Val(\boldsymbol{D})$ to $\mathbb{R}$. A factor is* nonnegative *if all its entries are nonnegative. The set of variables $\boldsymbol{D}$ is called the* scope *of the factor and denoted Scope$[\phi]$.* ■

scope

Unless stated otherwise, we restrict attention to nonnegative factors.

In our example, we have a factor $\phi_1(A, B) : Val(A, B) \mapsto \mathbb{R}^+$. The value associated with a particular assignment a, b denotes the affinity between these two values: the higher the value $\phi_1(a, b)$, the more compatible these two values are.

Figure 4.1a shows one possible compatibility factor for these variables. Note that this factor is not normalized; indeed, the entries are not even in $[0, 1]$. Roughly speaking, $\phi_1(A, B)$ asserts that it is more likely that Alice and Bob agree. It also adds more weight for the case where they are both right than for the case where they are both wrong. This factor function also has the property that $\phi_1(a^1, b^0) < \phi_1(a^0, b^1)$. Thus, if they disagree, there is less weight for the case where Alice has the misconception but Bob does not than for the converse case.

In a similar way, we define a compatibility factor for each other interacting pair: $\{B, C\}$, $\{C, D\}$, and $\{A, D\}$. Figure 4.1 shows one possible choice of factors for all four pairs. For example, the factor over C, D represents the compatibility of Charles and Debbie. It indicates that Charles and Debbie argue all the time, so that the most likely instantiations are those where they end up disagreeing.

As in a Bayesian network, the parameterization of the Markov network defines the local interactions between directly related variables. To define a global model, we need to combine these interactions. As in Bayesian networks, we combine the local models by multiplying them. Thus, we want $P(a, b, c, d)$ to be $\phi_1(a, b) \cdot \phi_2(b, c) \cdot \phi_3(c, d) \cdot \phi_4(d, a)$. In this case, however, we have no guarantees that the result of this process is a normalized joint distribution. Indeed, in this example, it definitely is not. Thus, we define the distribution by taking the product of

A	B	C	D	$Unnormalized$	$Normalized$
a^0	b^0	c^0	d^0	$300,000$	0.04
a^0	b^0	c^0	d^1	$300,000$	0.04
a^0	b^0	c^1	d^0	$300,000$	0.04
a^0	b^0	c^1	d^1	30	$4.1 \cdot 10^{-6}$
a^0	b^1	c^0	d^0	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^0	d^1	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^1	d^0	$5,000,000$	0.69
a^0	b^1	c^1	d^1	500	$6.9 \cdot 10^{-5}$
a^1	b^0	c^0	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^0	d^1	$1,000,000$	0.14
a^1	b^0	c^1	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^1	d^1	100	$1.4 \cdot 10^{-5}$
a^1	b^1	c^0	d^0	10	$1.4 \cdot 10^{-6}$
a^1	b^1	c^0	d^1	$100,000$	0.014
a^1	b^1	c^1	d^0	$100,000$	0.014
a^1	b^1	c^1	d^1	$100,000$	0.014

Figure 4.2 Joint distribution for the Misconception **example.** The unnormalized measure and the normalized joint distribution over A, B, C, D, obtained from the parameterization of figure 4.1. The value of the partition function in this example is $7,201,840$.

the local factors, and then normalizing it to define a legal distribution. Specifically, we define

$$P(a, b, c, d) = \frac{1}{Z}\phi_1(a, b) \cdot \phi_2(b, c) \cdot \phi_3(c, d) \cdot \phi_4(d, a),$$

where

$$Z = \sum_{a,b,c,d} \phi_1(a, b) \cdot \phi_2(b, c) \cdot \phi_3(c, d) \cdot \phi_4(d, a)$$

partition function

Markov random field

is a normalizing constant known as the *partition function*. The term "partition" originates from the early history of Markov networks, which originated from the concept of *Markov random field* (or *MRF*) in statistical physics (see box 4.C); the "function" is because the value of Z is a function of the parameters, a dependence that will play a significant role in our discussion of learning.

In our example, the unnormalized measure (the simple product of the four factors) is shown in the next-to-last column in figure 4.2. For example, the entry corresponding to a^1, b^1, c^0, d^1 is obtained by multiplying:

$$\phi_1(a^1, b^1) \cdot \phi_2(b^1, c^0) \cdot \phi_3(c^0, d^1) \cdot \phi_4(d^1, a^1) = 10 \cdot 1 \cdot 100 \cdot 100 = 100,000.$$

The last column shows the normalized distribution.

We can use this joint distribution to answer queries, as usual. For example, by summing out A, C, and D, we obtain $P(b^1) \approx 0.732$ and $P(b^0) \approx 0.268$; that is, Bob is 26 percent likely to have the misconception. On the other hand, if we now observe that Charles does not have the misconception (c^0), we obtain $P(b^1 \mid c^0) \approx 0.06$.

The benefit of this representation is that it allows us great flexibility in representing interactions between variables. For example, if we want to change the nature of the interaction between A and B, we can simply modify the entries in that factor, without having to deal with normalization constraints and the interaction with other factors. The flip side of this flexibility, as we will see later, is that the effects of these changes are not always intuitively understandable.

As in Bayesian networks, there is a tight connection between the factorization of the distribution and its independence properties. The key result here is stated in exercise 2.5: $P \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ if and only if we can write P in the form $P(\mathcal{X}) = \phi_1(\boldsymbol{X}, \boldsymbol{Z})\phi_2(\boldsymbol{Y}, \boldsymbol{Z})$. In our example, the structure of the factors allows us to decompose the distribution in several ways; for example:

$$P(A, B, C, D) = \left[\frac{1}{Z}\phi_1(A, B)\phi_2(B, C)\right] \phi_3(C, D)\phi_4(A, D).$$

From this decomposition, we can infer that $P \models (B \perp D \mid A, C)$. We can similarly infer that $P \models (A \perp C \mid B, D)$. These are precisely the two independencies that we tried, unsuccessfully, to achieve using a Bayesian network, in example 3.8. Moreover, these properties correspond to our intuition of "paths of influence" in the graph, where we have that B and D are separated given A, C, and that A and C are separated given B, D. Indeed, as in a Bayesian network, independence properties of the distribution P correspond directly to separation properties in the graph over which P factorizes.

4.2 Parameterization

We begin our formal discussion by describing the parameterization used in the class of undirected graphical models that are the focus of this chapter. In the next section, we make the connection to the graph structure and demonstrate how it captures the independence properties of the distribution.

To represent a distribution, we need to associate the graph structure with a set of parameters, in the same way that CPDs were used to parameterize the directed graph structure. However, the parameterization of Markov networks is not as intuitive as that of Bayesian networks, since the factors do not correspond either to probabilities or to conditional probabilities. As a consequence, the parameters are not intuitively understandable, making them hard to elicit from people. As we will see in chapter 20, they are also significantly harder to estimate from data.

4.2.1 Factors

A key issue in parameterizing a Markov network is that the representation is undirected, so that the parameterization cannot be directed in nature. We therefore use factors, as defined in definition 4.1. Note that a factor subsumes both the notion of a joint distribution and the notion of a CPD. A joint distribution over $\boldsymbol{D}$ is a factor over $\boldsymbol{D}$: it specifies a real number for every assignment of values of $\boldsymbol{D}$. A conditional distribution $P(X \mid \boldsymbol{U})$ is a factor over $\{X\} \cup \boldsymbol{U}$. However, both CPDs and joint distributions must satisfy certain normalization constraints (for example, in a joint distribution the numbers must sum to 1), whereas there are no constraints on the parameters in a factor.

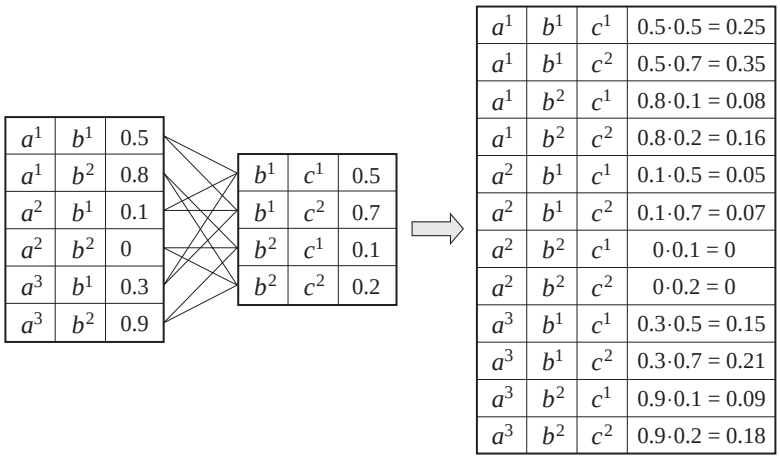


Figure 4.3 An example of factor product

As we discussed, we can view a factor as roughly describing the "compatibilities" between different values of the variables in its scope. We can now parameterize the graph by associating a set of a factors with it. One obvious idea might be to associate parameters directly with the edges in the graph. However, a simple calculation will convince us that this approach is insufficient to parameterize a full distribution.

Example 4.1

Consider a fully connected graph over $\mathcal{X}$; in this case, the graph specifies no conditional independence assumptions, so that we should be able to specify an arbitrary joint distribution over $\mathcal{X}$. If all of the variables are binary, each factor over an edge would have 4 parameters, and the total number of parameters in the graph would be $4\binom{n}{2}$. However, the number of parameters required to specify a joint distribution over n binary variables is $2^n - 1$. Thus, pairwise factors simply do not have enough parameters to encompass the space of joint distributions. More intuitively, such factors capture only the pairwise interactions, and not interactions that involve combinations of values of larger subsets of variables. ■

A more general representation can be obtained by allowing factors over arbitrary subsets of variables. To provide a formal definition, we first introduce the following important operation on factors.

Definition 4.2

factor product

Let $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ be three disjoint sets of variables, and let $\phi_1(\boldsymbol{X}, \boldsymbol{Y})$ and $\phi_2(\boldsymbol{Y}, \boldsymbol{Z})$ be two factors. We define the factor product *$\phi_1 \times \phi_2$ to be a factor $\psi : Val(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}) \mapsto \mathbb{R}$ as follows:*

$$\psi(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}) = \phi_1(\boldsymbol{X}, \boldsymbol{Y}) \cdot \phi_2(\boldsymbol{Y}, \boldsymbol{Z}).$$ ■

The key aspect to note about this definition is the fact that the two factors ϕ_1 and ϕ_2 are multiplied in a way that "matches up" the common part $\boldsymbol{Y}$. Figure 4.3 shows an example of the product of two factors. We have deliberately chosen factors that do not correspond either to probabilities or to conditional probabilities, in order to emphasize the generality of this operation.

As we have already observed, both CPDs and joint distributions are factors. Indeed, the chain rule for Bayesian networks defines the joint distribution factor as the product of the CPD factors. For example, when computing $P(A, B) = P(A)P(B \mid A)$, we always multiply entries in the $P(A)$ and $P(B \mid A)$ tables that have the same value for A. Thus, letting $\phi_{X_i}(X_i, \mathrm{Pa}_{X_i})$ represent $P(X_i \mid \mathrm{Pa}_{X_i})$, we have that

$$P(X_1, \ldots, X_n) = \prod_i \phi_{X_i}.$$

4.2.2 Gibbs Distributions and Markov Networks

We can now use the more general notion of factor product to define an undirected parameterization of a distribution.

Definition 4.3 Gibbs distribution

A distribution P_Φ is a Gibbs distribution *parameterized by a set of factors $\Phi = \{\phi_1(\boldsymbol{D}_1), \ldots, \phi_K(\boldsymbol{D}_K)\}$ if it is defined as follows:*

$$P_\Phi(X_1, \ldots, X_n) = \frac{1}{Z}\tilde{P}_\Phi(X_1, \ldots, X_n),$$

where

$$\tilde{P}_\Phi(X_1, \ldots, X_n) = \phi_1(\boldsymbol{D}_1) \times \phi_2(\boldsymbol{D}_2) \times \cdots \times \phi_m(\boldsymbol{D}_m)$$

is an unnormalized measure and

$$Z = \sum_{X_1, \ldots, X_n} \tilde{P}_\Phi(X_1, \ldots, X_n)$$

partition function

is a normalizing constant called the partition function. ■

It is tempting to think of the factors as representing the marginal probabilities of the variables in their scope. Thus, looking at any individual factor, we might be led to believe that the behavior of the distribution defined by the Markov network as a whole corresponds to the behavior defined by the factor. However, this intuition is overly simplistic. **A factor is only one contribution to the overall joint distribution. The distribution as a whole has to take into consideration the contributions from all of the factors involved.**

Example 4.2

Consider the distribution of figure 4.2. The marginal distribution over A, B, is

a^0	b^0	0.13
a^0	b^1	0.69
a^1	b^0	0.14
a^1	b^1	0.04

The most likely configuration is the one where Alice and Bob disagree. By contrast, the highest entry in the factor $\phi_1(A, B)$ in figure 4.1 corresponds to the assignment a^0, b^0. The reason for the discrepancy is the influence of the other factors on the distribution. In particular, $\phi_3(C, D)$ asserts that Charles and Debbie disagree, whereas $\phi_2(B, C)$ and $\phi_4(D, A)$ assert that Bob and Charles agree and that Debbie and Alice agree. Taking just these factors into consideration, we would conclude that Alice and Bob are likely to disagree. In this case, the "strength" of these other factors is much stronger than that of the $\phi_1(A, B)$ factor, so that the influence of the latter is overwhelmed. ■

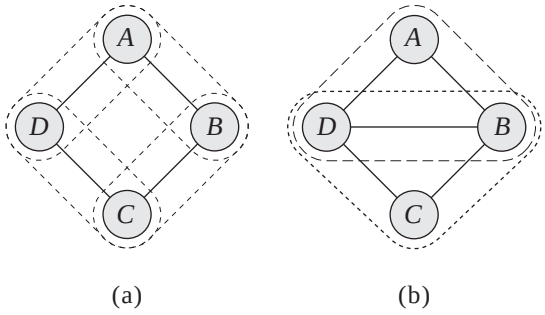


Figure 4.4 The cliques in two simple Markov networks. In (a), the cliques are the pairs $\{A, B\}$, $\{B, C\}$, $\{C, D\}$, and $\{D, A\}$. In (b), the cliques are $\{A, B, D\}$ and $\{B, C, D\}$.

We now want to relate the parameterization of a Gibbs distribution to a graph structure. If our parameterization contains a factor whose scope contains both X and Y, we are introducing a direct interaction between them. Intuitively, we would like these direct interactions to be represented in the graph structure. Thus, if our parameterization contains such a factor, we would like the associated Markov network structure $\mathcal{H}$ to contain an edge between X and Y.

Definition 4.4
Markov network factorization

We say that a distribution P_Φ with $\Phi = \{\phi_1(\boldsymbol{D}_1), \ldots, \phi_K(\boldsymbol{D}_K)\}$ factorizes *over a Markov network $\mathcal{H}$ if each $\boldsymbol{D}_k$ ($k = 1, \ldots, K$) is a complete subgraph of $\mathcal{H}$.* ■

clique potentials

The factors that parameterize a Markov network are often called *clique potentials*.

As we will see, if we associate factors only with complete subgraphs, as in this definition, we are not violating the independence assumptions induced by the network structure, as defined later in this chapter.

Note that, because every complete subgraph is a subset of some (maximal) clique, we can reduce the number of factors in our parameterization by allowing factors only for maximal cliques. More precisely, let $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_k$ be the cliques in $\mathcal{H}$. We can parameterize P using a set of factors $\phi_1(\boldsymbol{C}_1), \ldots, \phi_l(\boldsymbol{C}_l)$. Any factorization in terms of complete subgraphs can be converted into this form simply by assigning each factor to a clique that encompasses its scope and multiplying all of the factors assigned to each clique to produce a clique potential. In our Misconception example, we have four cliques: $\{A, B\}$, $\{B, C\}$, $\{C, D\}$, and $\{A, D\}$. Each of these cliques can have its own clique potential. One possible setting of the parameters in these clique potential is shown in figure 4.1. Figure 4.4 shows two examples of a Markov network and the (maximal) cliques in that network.

☞ **Although it can be used without loss of generality, the parameterization using maximal clique potentials generally obscures structure that is present in the original set of factors.** For example, consider the Gibbs distribution described in example 4.1. Here, we have a potential for every pair of variables, so the Markov network associated with this distribution is a single large clique containing all variables. If we associate a factor with this single clique, it would be exponentially large in the number of variables, whereas the original parameterization in terms of edges requires only a quadratic number of parameters. See section 4.4.1.1 for further discussion.

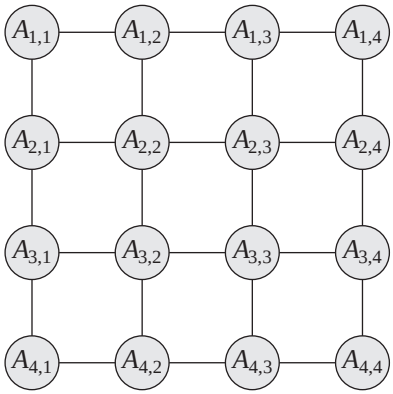


Figure 4.A.1 — A pairwise Markov network (MRF) structured as a grid.

pairwise Markov network

node potential

edge potential

Box 4.A — Concept: Pairwise Markov Networks. *A subclass of Markov networks that arises in many contexts is that of* pairwise Markov networks, *representing distributions where all of the factors are over single variables or pairs of variables. More precisely, a pairwise Markov network over a graph $\mathcal{H}$ is associated with a set of* node potentials $\{\phi(X_i) : i = 1, \ldots, n\}$ *and a set of* edge potentials $\{\phi(X_i, X_j) : (X_i, X_j) \in \mathcal{H}\}$. *The overall distribution is (as always) the normalized product of all of the potentials (both node and edge). Pairwise MRFs are attractive because of their simplicity, and because interactions on edges are an important special case that often arises in practice (see, for example, box 4.C and box 4.B).*

A class of pairwise Markov networks that often arises, and that is commonly used as a benchmark for inference, is the class of networks structured in the form of a grid, as shown in figure 4.A.1. As we discuss in the inference chapters of this book, although these networks have a simple and compact representation, they pose a significant challenge for inference algorithms.

4.2.3 Reduced Markov Networks

We end this section with one final concept that will prove very useful in later sections. Consider the process of conditioning a distribution on some assignment $\boldsymbol{u}$ to some subset of variables $\boldsymbol{U}$. Conditioning a distribution corresponds to eliminating all entries in the joint distribution that are inconsistent with the event $\boldsymbol{U} = \boldsymbol{u}$, and renormalizing the remaining entries to sum to 1. Now, consider the case where our distribution has the form P_Φ for some set of factors Φ. Each entry in the unnormalized measure $\tilde{P}_\Phi$ is a product of entries from the factors Φ, one entry from each factor. If, in some factor, we have an entry that is inconsistent with $\boldsymbol{U} = \boldsymbol{u}$, it will only contribute to entries in $\tilde{P}_\Phi$ that are also inconsistent with this event. Thus, we can eliminate all such entries from every factor in Φ.

More generally, we can define:

a^1	b^1	c^1	0.25
a^1	b^2	c^1	0.08
a^2	b^1	c^1	0.05
a^2	b^2	c^1	0
a^3	b^1	c^1	0.15
a^3	b^2	c^1	0.09

Figure 4.5 Factor reduction: The factor computed in figure 4.3, reduced to the context $C = c^1$.

Definition 4.5 factor reduction

Let $\phi(\boldsymbol{Y})$ be a factor, and $\boldsymbol{U} = \boldsymbol{u}$ an assignment for $\boldsymbol{U} \subseteq \boldsymbol{Y}$. We define the reduction *of the factor ϕ to the context $\boldsymbol{U} = \boldsymbol{u}$, denoted $\phi[\boldsymbol{U} = \boldsymbol{u}]$ (and abbreviated $\phi[\boldsymbol{u}]$), to be a factor over scope $\boldsymbol{Y}' = \boldsymbol{Y} - \boldsymbol{U}$, such that*

$$\phi[\boldsymbol{u}](\boldsymbol{y}') = \phi(\boldsymbol{y}', \boldsymbol{u}).$$

For $\boldsymbol{U} \not\subset \boldsymbol{Y}$, we define $\phi[\boldsymbol{u}]$ to be $\phi[\boldsymbol{U}' = \boldsymbol{u}']$, where $\boldsymbol{U}' = \boldsymbol{U} \cap \boldsymbol{Y}$, and $\boldsymbol{u}' = \boldsymbol{u}\langle \boldsymbol{U}' \rangle$, where $\boldsymbol{u}\langle \boldsymbol{U}' \rangle$ denotes the assignment in $\boldsymbol{u}$ to the variables in $\boldsymbol{U}'$. ■

Figure 4.5 illustrates this operation, reducing the of figure 4.3 to the context $C = c^1$.

Now, consider a product of factors. An entry in the product is consistent with $\boldsymbol{u}$ if and only if it is a product of entries that are all consistent with $\boldsymbol{u}$. We can therefore define:

Definition 4.6 reduced Gibbs distribution

Let P_Φ be a Gibbs distribution parameterized by $\Phi = \{\phi_1, \ldots, \phi_K\}$ and let $\boldsymbol{u}$ be a context. The reduced Gibbs distribution *$P_\Phi[\boldsymbol{u}]$ is the Gibbs distribution defined by the set of factors $\Phi[\boldsymbol{u}] = \{\phi_1[\boldsymbol{u}], \ldots, \phi_K[\boldsymbol{u}]\}$.* ■

Reducing the set of factors defining P_Φ to some context $\boldsymbol{u}$ corresponds directly to the operation of conditioning P_Φ on the observation $\boldsymbol{u}$. More formally:

Proposition 4.1

Let $P_\Phi(\boldsymbol{X})$ be a Gibbs distribution. Then $P_\Phi[\boldsymbol{u}] = P_\Phi(\boldsymbol{W} \mid \boldsymbol{u})$ where $\boldsymbol{W} = \boldsymbol{X} - \boldsymbol{U}$.

Thus, to condition a Gibbs distribution on a context $\boldsymbol{u}$, we simply reduce every one of its factors to that context. Intuitively, the renormalization step needed to account for $\boldsymbol{u}$ is simply folded into the standard renormalization of any Gibbs distribution. This result immediately provides us with a construction for the Markov network that we obtain when we condition the associated distribution on some observation $\boldsymbol{u}$.

Definition 4.7 reduced Markov network

Let $\mathcal{H}$ be a Markov network over $\boldsymbol{X}$ and $\boldsymbol{U} = \boldsymbol{u}$ a context. The reduced Markov network *$\mathcal{H}[\boldsymbol{u}]$ is a Markov network over the nodes $\boldsymbol{W} = \boldsymbol{X} - \boldsymbol{U}$, where we have an edge X—Y if there is an edge X—Y in $\mathcal{H}$.* ■

Proposition 4.2

Let $P_\Phi(\boldsymbol{X})$ be a Gibbs distribution that factorizes over $\mathcal{H}$, and $\boldsymbol{U} = \boldsymbol{u}$ a context. Then $P_\Phi[\boldsymbol{u}]$ factorizes over $\mathcal{H}[\boldsymbol{u}]$.

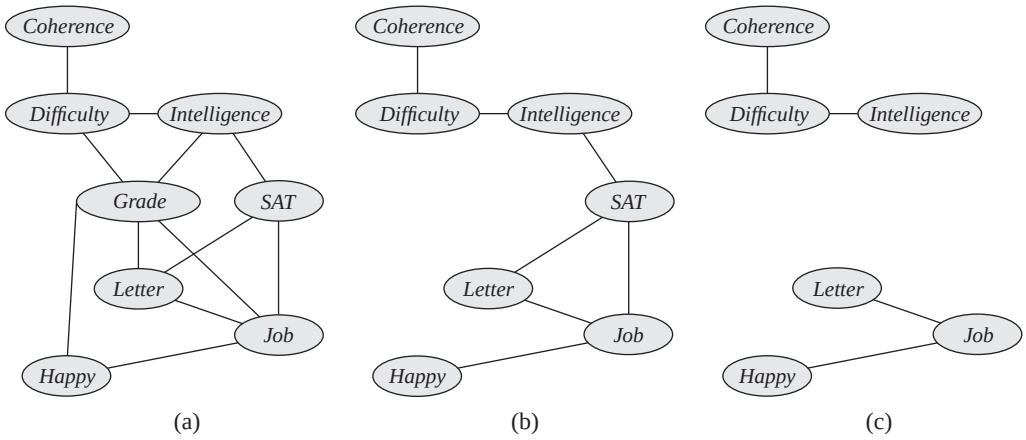


Figure 4.6 Markov networks for the factors in an extended Student **example:** (a) The initial set of factors; (b) Reduced to the context $G = g$; (c) Reduced to the context $G = g, S = s$.

Note the contrast to the effect of conditioning in a Bayesian network: Here, conditioning on a context $\boldsymbol{u}$ only eliminates edges from the graph; in a Bayesian network, conditioning on evidence can activate a v-structure, creating new dependencies. We return to this issue in section 4.5.1.1.

Example 4.3

Consider, for example, the Markov network shown in figure 4.6a; as we will see, this network is the Markov network required to capture the distribution encoded by an extended version of our Student *Bayesian network (see figure 9.8). Figure 4.6b shows the same Markov network reduced over a context of the form $G = g$, and (c) shows the network reduced over a context of the form $G = g, S = s$. As we can see, the network structures are considerably simplified.* ■

Box 4.B — Case Study: Markov Networks for Computer Vision. *One important application area for Markov networks is computer vision. Markov networks, typically called MRFs in this vision community, have been used for a wide variety of visual processing tasks, such as image segmentation, removal of blur or noise, stereo reconstruction, object recognition, and many more.*

In most of these applications, the network takes the structure of a pairwise MRF, where the variables correspond to pixels and the edges (factors) to interactions between adjacent pixels in the grid that represents the image; thus, each (interior) pixel has exactly four neighbors. The value space of the variables and the exact form of factors depend on the task. These models are usually formulated in terms of energies (negative log-potentials), so that values represent "penalties," and a lower value corresponds to a higher-probability configuration.

image denoising

In image denoising*, for example, the task is to restore the "true" value of all of the pixels given possibly noisy pixel values. Here, we have a node potential for each pixel X_i that penalizes large discrepancies from the observed pixel value y_i. The edge potential encodes a preference for continuity between adjacent pixel values, penalizing cases where the inferred value for X_i is too*

far from the inferred pixel value for one of its neighbors X_j. However, it is important not to overpenalize true disparities (such as edges between objects or regions), leading to oversmoothing of the image. Thus, we bound the penalty, using, for example, some truncated norm, as described in box 4.D: $\epsilon(x_i, x_j) = \min(c\|x_i - x_j\|_p, \text{dist}_{\max})$ (for $p \in \{1, 2\}$).

stereo reconstruction

Slight variants of the same model are used in many other applications. For example, in stereo reconstruction, *the goal is to reconstruct the depth disparity of each pixel in the image. Here, the values of the variables represent some discretized version of the depth dimension (usually more finely discretized for distances close to the camera and more coarsely discretized as the distance from the camera increases). The individual node potential for each pixel X_i uses standard techniques from computer vision to estimate, from a pair of stereo images, the individual depth disparity of this pixel. The edge potentials, precisely as before, often use a truncated metric to enforce continuity of the depth estimates, with the truncation avoiding an overpenalization of true depth disparities (for example, when one object is partially in front of the other). Here, it is also quite common to make the penalty inversely proportional to the image gradient between the two pixels, allowing a smaller penalty to be applied in cases where a large image gradient suggests an edge between the pixels, possibly corresponding to an occlusion boundary.*

image segmentation

In image segmentation, *the task is to partition the image pixels into regions corresponding to distinct parts of the scene. There are different variants of the segmentation task, many of which can be formulated as a Markov network. In one formulation, known as multiclass segmentation, each variable X_i has a domain $\{1, \ldots, K\}$, where the value of X_i represents a region assignment for pixel i (for example, grass, water, sky, car). Since classifying every pixel can be computationally expensive, some state-of-the-art methods for image segmentation and other tasks first oversegment the image into superpixels (or small coherent regions) and classify each region — all pixels within a region are assigned the same label. The oversegmented image induces a graph in which there is one node for each superpixel and an edge between two nodes if the superpixels are adjacent (share a boundary) in the underlying image. We can now define our distribution in terms of this graph.*

Features are extracted from the image for each pixel or superpixel. The appearance features depend on the specific task. In image segmentation, for example, features typically include statistics over color, texture, and location. Often the features are clustered or provided as input to local classifiers to reduce dimensionality. The features used in the model are then the soft cluster assignments or local classifier outputs for each superpixel. The node potential for a pixel or superpixel is then a function of these features. We note that the factors used in defining this model depend on the specific values of the pixels in the image, so that each image defines a different probability distribution over the segment labels for the pixels or superpixels. In effect, the model used here is a conditional random field, *a concept that we define more formally in section 4.6.1.*

conditional random field

The model contains an edge potential between every pair of neighboring superpixels X_i, X_j. Most simply, this potential encodes a contiguity preference, with a penalty of λ whenever $X_i \neq X_j$. Again, we can improve the model by making the penalty depend on the presence of an image gradient between the two pixels. An even better model does more than penalize discontinuities. We can have nondefault values for other class pairs, allowing us to encode the fact that we more often find tigers adjacent to vegetation than adjacent to water; we can even make the model depend on the relative pixel location, allowing us to encode the fact that we usually find water below vegetation, cars over roads, and sky above everything.

Figure 4.B.1 shows segmentation results in a model containing only potentials on single pixels (thereby labeling each of them independently) versus results obtained from a model also containing

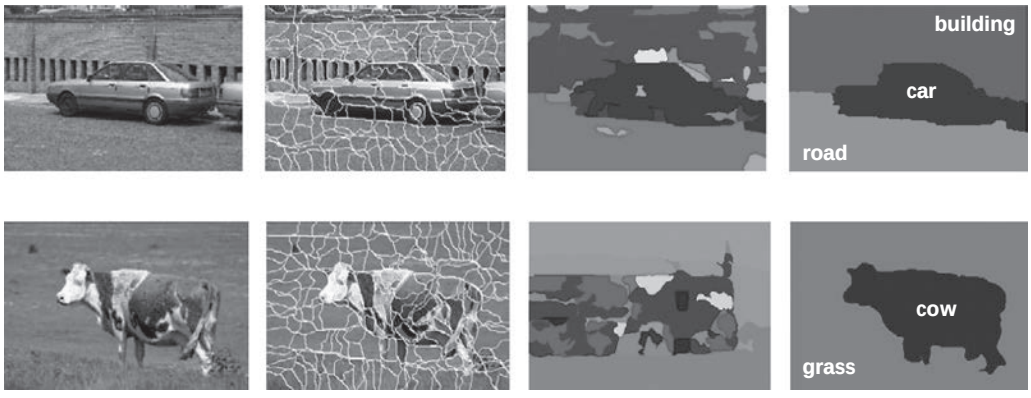


Figure 4.B.1 — Two examples of image segmentation results (a) The original image. (b) An oversegmentation known as superpixels; each superpixel is associated with a random variable that designates its segment assignment. The use of superpixels reduces the size of the problems. (c) Result of segmentation using node potentials alone, so that each superpixel is classified independently. (d) Result of segmentation using a pairwise Markov network encoding interactions between adjacent superpixels.

pairwise potentials. The difference in the quality of the results clearly illustrates the importance of modeling the correlations between the superpixels.

4.3 Markov Network Independencies

In section 4.1, we gave an intuitive justification of why an undirected graph seemed to capture the types of interactions in the Misconception example. We now provide a formal presentation of the undirected graph as a representation of independence assertions.

4.3.1 Basic Independencies

As in the case of Bayesian networks, the graph structure in a Markov network can be viewed as encoding a set of independence assumptions. Intuitively, in Markov networks, probabilistic influence "flows" along the undirected paths in the graph, but it is blocked if we condition on the intervening nodes.

observed variable

active path

Definition 4.8

Let $\mathcal{H}$ be a Markov network structure, and let $X_1 — \ldots — X_k$ be a path in $\mathcal{H}$. Let $\boldsymbol{Z} \subseteq \mathcal{X}$ be a set of observed variables. *The path $X_1 — \ldots — X_k$ is* active *given $\boldsymbol{Z}$ if none of the X_i's, $i = 1, \ldots, k$, is in $\boldsymbol{Z}$.* ■

Using this notion, we can define a notion of *separation* in the graph.

Definition 4.9

separation

global independencies

We say that a set of nodes $\boldsymbol{Z}$ separates $\boldsymbol{X}$ *and* $\boldsymbol{Y}$ *in* $\mathcal{H}$*, denoted* $\text{sep}_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$*, if there is no active path between any node* $X \in \boldsymbol{X}$ *and* $Y \in \boldsymbol{Y}$ *given* $\boldsymbol{Z}$*. We define the* global independencies *associated with* $\mathcal{H}$ *to be:*

$$\mathcal{I}(\mathcal{H}) = \{(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \ : \ \text{sep}_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})\}. \quad \blacksquare$$

As we will discuss, the independencies in $\mathcal{I}(\mathcal{H})$ are precisely those that are guaranteed to hold for every distribution P over $\mathcal{H}$. In other words, the separation criterion is sound for detecting independence properties in distributions over $\mathcal{H}$.

Note that the definition of separation is monotonic in $\boldsymbol{Z}$, that is, if $sep_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$, then $sep_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z}')$ for any $\boldsymbol{Z}' \supset \boldsymbol{Z}$. Thus, if we take separation as our definition of the independencies induced by the network structure, we are effectively restricting our ability to encode nonmonotonic independence relations. Recall that in the context of intercausal reasoning in Bayesian networks, nonmonotonic reasoning patterns are quite useful in many situations — for example, when two diseases are independent, but dependent given some common symptom. The nature of the separation property implies that such independence patterns cannot be expressed in the structure of a Markov network. We return to this issue in section 4.5.

As for Bayesian networks, we can show a connection between the independence properties implied by the Markov network structure, and the possibility of factorizing a distribution over the graph. As before, we can now state the analogue to both of our representation theorems for Bayesian networks, which assert the equivalence between the Gibbs factorization of a distribution P over a graph $\mathcal{H}$ and the assertion that $\mathcal{H}$ is an I-map for P, that is, that P satisfies the Markov assumptions $\mathcal{I}(\mathcal{H})$.

4.3.1.1 Soundness

soundness

We first consider the analogue to theorem 3.2, which asserts that a Gibbs distribution satisfies the independencies associated with the graph. In other words, this result states the *soundness* of the separation criterion.

Theorem 4.1

Let P *be a distribution over* $\mathcal{X}$*, and* $\mathcal{H}$ *a Markov network structure over* $\mathcal{X}$*. If* P *is a Gibbs distribution that factorizes over* $\mathcal{H}$*, then* $\mathcal{H}$ *is an I-map for* P*.*

Proof Let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ be any three disjoint subsets in $\mathcal{X}$ such that $\boldsymbol{Z}$ separates $\boldsymbol{X}$ and $\boldsymbol{Y}$ in $\mathcal{H}$. We want to show that $P \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$.

We start by considering the case where $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z} = \mathcal{X}$. As $\boldsymbol{Z}$ separates $\boldsymbol{X}$ from $\boldsymbol{Y}$, there are no direct edges between $\boldsymbol{X}$ and $\boldsymbol{Y}$. Hence, any clique in $\mathcal{H}$ is fully contained either in $\boldsymbol{X} \cup \boldsymbol{Z}$ or in $\boldsymbol{Y} \cup \boldsymbol{Z}$. Let $\mathcal{I}_{\boldsymbol{X}}$ be the indexes of the set of cliques that are contained in $\boldsymbol{X} \cup \boldsymbol{Z}$, and let $\mathcal{I}_{\boldsymbol{Y}}$ be the indexes of the remaining cliques. We know that

$$P(X_1, \ldots, X_n) = \frac{1}{Z} \prod_{i \in \mathcal{I}_{\boldsymbol{X}}} \phi_i(\boldsymbol{D}_i) \cdot \prod_{i \in \mathcal{I}_{\boldsymbol{Y}}} \phi_i(\boldsymbol{D}_i).$$

As we discussed, none of the factors in the first product involve any variable in $\boldsymbol{Y}$, and none in the second product involve any variable in $\boldsymbol{X}$. Hence, we can rewrite this product in the form:

$$P(X_1, \ldots, X_n) = \frac{1}{Z} f(\boldsymbol{X}, \boldsymbol{Z}) g(\boldsymbol{Y}, \boldsymbol{Z}).$$

From this decomposition, the desired independence follows immediately (exercise 2.5).

Now consider the case where $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z} \subset \mathcal{X}$. Let $\boldsymbol{U} = \mathcal{X} - (\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z})$. We can partition $\boldsymbol{U}$ into two disjoint sets $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ such that $\boldsymbol{Z}$ separates $\boldsymbol{X} \cup \boldsymbol{U}_1$ from $\boldsymbol{Y} \cup \boldsymbol{U}_2$ in $\mathcal{H}$. Using the preceding argument, we conclude that $P \models (\boldsymbol{X}, \boldsymbol{U}_1 \perp \boldsymbol{Y}, \boldsymbol{U}_2 \mid \boldsymbol{Z})$. Using the decomposition property (equation (2.8)), we conclude that $P \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$. ■

Hammersley-Clifford theorem

The other direction (the analogue to theorem 3.1), which goes from the independence properties of a distribution to its factorization, is known as the *Hammersley-Clifford theorem*. Unlike for Bayesian networks, this direction does not hold in general. As we will show, it holds only under the additional assumption that P is a positive distribution (see definition 2.5).

Theorem 4.2 *Let P be a positive distribution over $\mathcal{X}$, and $\mathcal{H}$ a Markov network graph over $\mathcal{X}$. If $\mathcal{H}$ is an I-map for P, then P is a Gibbs distribution that factorizes over $\mathcal{H}$.*

To prove this result, we would need to use the independence assumptions to construct a set of factors over $\mathcal{H}$ that give rise to the distribution P. In the case of Bayesian networks, these factors were simply CPDs, which we could derive directly from P. As we have discussed, the correspondence between the factors in a Gibbs distribution and the distribution P is much more indirect. The construction required here is therefore significantly more subtle, and relies on concepts that we develop later in this chapter; hence, we defer the proof to section 4.4 (theorem 4.8).

This result shows that, **for positive distributions, the global independencies imply that the distribution factorizes according the network structure. Thus, for this class of distributions, we have that a distribution P factorizes over a Markov network $\mathcal{H}$ if and only if $\mathcal{H}$ is an I-map of P.** The positivity assumption is necessary for this result to hold:

Example 4.4 *Consider a distribution P over four binary random variables X_1, X_2, X_3, X_4, which gives probability $1/8$ to each of the following eight configurations, and probability zero to all others:*

(0,0,0,0)	*(1,0,0,0)*	*(1,1,0,0)*	*(1,1,1,0)*
(0,0,0,1)	*(0,0,1,1)*	*(0,1,1,1)*	*(1,1,1,1)*

Let $\mathcal{H}$ be the graph X_1—X_2—X_3—X_4—X_1. Then P satisfies the global independencies with respect to $\mathcal{H}$. For example, consider the independence $(X_1 \perp X_3 \mid X_2, X_4)$. For the assignment $X_2 = x_2^1, X_4 = x_4^0$, we have that only assignments where $X_1 = x_1^1$ receive positive probability. Thus, $P(x_1^1 \mid x_2^1, x_4^0) = 1$, and X_1 is trivially independent of X_3 in this conditional distribution. A similar analysis applies to all other cases, so that the global independencies hold. However, the distribution P does not factorize according to $\mathcal{H}$. The proof of this fact is left as an exercise (see exercise 4.1). ■

4.3.1.2 Completeness

completeness

The preceding discussion shows the soundness of the separation condition as a criterion for detecting independencies in Markov networks: any distribution that factorizes over $\mathcal{G}$ satisfies the independence assertions implied by separation. The next obvious issue is the *completeness* of this criterion.

As for Bayesian networks, the strong version of completeness does not hold in this setting. In other words, it is not the case that every pair of nodes X and Y that are not separated in $\mathcal{H}$ are dependent in every distribution P which factorizes over $\mathcal{H}$. However, as in theorem 3.3, we can use a weaker definition of completeness that does hold:

Theorem 4.3 *Let $\mathcal{H}$ be a Markov network structure. If X and Y are not separated given $\boldsymbol{Z}$ in $\mathcal{H}$, then X and Y are dependent given $\boldsymbol{Z}$ in some distribution P that factorizes over $\mathcal{H}$.*

PROOF The proof is a constructive one: we construct a distribution P that factorizes over $\mathcal{H}$ where X and Y are dependent. We assume, without loss of generality, that all variables are binary-valued. If this is not the case, we can treat them as binary-valued by restricting attention to two distinguished values for each variable.

By assumption, X and Y are not separated given $\boldsymbol{Z}$ in $\mathcal{H}$; hence, they must be connected by some unblocked trail. Let $X = U_1 — U_2 — \ldots — U_k = Y$ be some minimal trail in the graph such that, for all i, $U_i \notin \boldsymbol{Z}$, where we define a minimal trail in $\mathcal{H}$ to be a path with no shortcuts: thus, for any i and $j \neq i \pm 1$, there is no edge $U_i — U_j$. We can always find such a path: If we have a nonminimal path where we have $U_i — U_j$ for $j > i + 1$, we can always "shortcut" the original trail, converting it to one that goes directly from U_i to U_j.

For any $i = 1, \ldots, k-1$, as there is an edge $U_i — U_{i+1}$, it follows that U_i, U_{i+1} must both appear in some clique $\boldsymbol{C}_i$. We pick some very large weight W, and for each i we define the clique potential $\phi_i(\boldsymbol{C}_i)$ to assign weight W if $U_i = U_{i+1}$ and weight 1 otherwise, regardless of the values of the other variables in the clique. Note that the cliques $\boldsymbol{C}_i$ for U_i, U_{i+1} and $\boldsymbol{C}_j$ for U_j, U_{j+1} must be different cliques: If $\boldsymbol{C}_i = \boldsymbol{C}_j$, then U_j is in the same clique as U_i, and we have an edge $U_i — U_j$, contradicting the minimality of the trail. Hence, we can define the clique potential for each clique $\boldsymbol{C}_i$ separately. We define the clique potential for any other clique to be uniformly 1.

We now consider the distribution P resulting from multiplying all of these clique potentials. Intuitively, the distribution $P(U_1, \ldots, U_k)$ is simply the distribution defined by multiplying the pairwise factors for the pairs U_i, U_{i+1}, regardless of the other variables (including the ones in $\boldsymbol{Z}$). One can verify that, in $P(U_1, \ldots, U_k)$, we have that $X = U_1$ and $Y = U_k$ are dependent. We leave the conclusion of this argument as an exercise (exercise 4.5). ■

We can use the same argument as theorem 3.5 to conclude that, for almost all distributions P that factorize over $\mathcal{H}$ (that is, for all distributions except for a set of measure zero in the space of factor parameterizations) we have that $\mathcal{I}(P) = \mathcal{I}(\mathcal{H})$.

Once again, we can view this result as telling us that our definition of $\mathcal{I}(\mathcal{H})$ is the maximal one. For any independence assertion that is not a consequence of separation in $\mathcal{H}$, we can always find a counterexample distribution P that factorizes over $\mathcal{H}$.

4.3.2 Independencies Revisited

When characterizing the independencies in a Bayesian network, we provided two definitions: the local independencies (each node is independent of its nondescendants given its parents), and the global independencies induced by d-separation. As we showed, these two sets of independencies are equivalent, in that one implies the other.

So far, our discussion for Markov networks provides only a global criterion. While the global criterion characterizes the entire set of independencies induced by the network structure, a local criterion is also valuable, since it allows us to focus on a smaller set of properties when examining the distribution, significantly simplifying the process of finding an I-map for a distribution P.

Thus, it is natural to ask whether we can provide a local definition of the independencies induced by a Markov network, analogously to the local independencies of Bayesian networks. Surprisingly, as we now show, in the context of Markov networks, there are three different possible definitions of the independencies associated with the network structure — two local ones and the global one in definition 4.9. While these definitions are related, they are equivalent only for positive distributions. As we will see, nonpositive distributions allow for deterministic dependencies between the variables. Such deterministic interactions can "fool" local independence tests, allowing us to construct networks that are not I-maps of the distribution, yet the local independencies hold.

4.3.2.1 Local Markov Assumptions

The first, and weakest, definition is based on the following intuition: Whenever two variables are directly connected, they have the potential of being directly correlated in a way that is not mediated by other variables. Conversely, when two variables are not directly linked, there must be some way of rendering them conditionally independent. Specifically, we can require that X and Y be independent given all other nodes in the graph.

Definition 4.10 pairwise independencies

Let $\mathcal{H}$ be a Markov network. We define the pairwise independencies *associated with $\mathcal{H}$ to be:*

$$\mathcal{I}_p(\mathcal{H}) = \{(X \perp Y \mid \mathcal{X} - \{X, Y\}) \ : \ X — Y \notin \mathcal{H}\}.$$ ■

Using this definition, we can easily represent the independencies in our Misconception example using a Markov network: We simply connect the nodes up in exactly the same way as the interaction structure between the students.

The second local definition is an undirected analogue to the local independencies associated with a Bayesian network. It is based on the intuition that we can block all influences on a node by conditioning on its immediate neighbors.

Definition 4.11 Markov blanket, local independencies

For a given graph $\mathcal{H}$, we define the Markov blanket *of X in $\mathcal{H}$, denoted $\mathrm{MB}_{\mathcal{H}}(X)$, to be the neighbors of X in $\mathcal{H}$. We define the* local independencies *associated with $\mathcal{H}$ to be:*

$$\mathcal{I}_\ell(\mathcal{H}) = \{(X \perp \mathcal{X} - \{X\} - \mathrm{MB}_{\mathcal{H}}(X) \mid \mathrm{MB}_{\mathcal{H}}(X)) \ : \ X \in \mathcal{X}\}.$$ ■

In other words, the local independencies state that X is independent of the rest of the nodes in the graph given its immediate neighbors. We will show that these local independence assumptions hold for any distribution that factorizes over $\mathcal{H}$, so that X's Markov blanket in $\mathcal{H}$ truly does separate it from all other variables.

4.3.2.2 Relationships between Markov Properties

We have now presented three sets of independence assertions associated with a network structure $\mathcal{H}$. For general distributions, $\mathcal{I}_p(\mathcal{H})$ is strictly weaker than $\mathcal{I}_\ell(\mathcal{H})$, which in turn is strictly weaker than $\mathcal{I}(\mathcal{H})$. However, all three definitions are equivalent for positive distributions.

Proposition 4.3 *For any Markov network $\mathcal{H}$, and any distribution P, we have that if $P \models \mathcal{I}_\ell(\mathcal{H})$ then $P \models \mathcal{I}_p(\mathcal{H})$.*

The proof of this result is left as an exercise (exercise 4.8).

Proposition 4.4 *For any Markov network $\mathcal{H}$, and any distribution P, we have that if $P \models \mathcal{I}(\mathcal{H})$ then $P \models \mathcal{I}_\ell(\mathcal{H})$.*

The proof of this result follows directly from the fact that if X and Y are not connected by an edge, then they are necessarily separated by all of the remaining nodes in the graph.

The converse of these inclusion results holds only for positive distributions (see definition 2.5). More specifically, if we assume the intersection property (equation (2.11)), all three of the Markov conditions are equivalent.

Theorem 4.4 *Let P be a positive distribution. If P satisfies $\mathcal{I}_p(\mathcal{H})$, then P satisfies $\mathcal{I}(\mathcal{H})$.*

PROOF We want to prove that for all disjoint sets $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$:

$$sep_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z}) \Longrightarrow P \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}). \tag{4.1}$$

The proof proceeds by descending induction on the size of $\boldsymbol{Z}$.

The base case is $|\boldsymbol{Z}| = n-2$; equation (4.1) follows immediately from the definition of $\mathcal{I}_p(\mathcal{H})$.

For the inductive step, assume that equation (4.1) holds for every $\boldsymbol{Z}'$ with size $|\boldsymbol{Z}'| = k$, and let $\boldsymbol{Z}$ be any set such that $|\boldsymbol{Z}| = k-1$. We distinguish between two cases.

In the first case, $\boldsymbol{X} \cup \boldsymbol{Z} \cup \boldsymbol{Y} = \mathcal{X}$. As $|\boldsymbol{Z}| < n-2$, we have that either $|\boldsymbol{X}| \geq 2$ or $|\boldsymbol{Y}| \geq 2$. Without loss of generality, assume that the latter holds; let $A \in \boldsymbol{Y}$ and $\boldsymbol{Y}' = \boldsymbol{Y} - \{A\}$. From the fact that $sep_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$, we also have that $sep_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y}' \mid \boldsymbol{Z})$ on one hand and $sep_{\mathcal{H}}(\boldsymbol{X}; A \mid \boldsymbol{Z})$ on the other hand. As separation is monotonic, we also have that $sep_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y}' \mid \boldsymbol{Z} \cup \{A\})$ and $sep_{\mathcal{H}}(\boldsymbol{X}; A \mid \boldsymbol{Z} \cup \boldsymbol{Y}')$. The separating sets $\boldsymbol{Z} \cup \{A\}$ and $\boldsymbol{Z} \cup \boldsymbol{Y}'$ are each at least size $|\boldsymbol{Z}| + 1 = k$ in size, so that equation (4.1) applies, and we can conclude that P satisfies:

$$(\boldsymbol{X} \perp \boldsymbol{Y}' \mid \boldsymbol{Z} \cup \{A\}) \quad \& \quad (\boldsymbol{X} \perp A \mid \boldsymbol{Z} \cup \boldsymbol{Y}').$$

Because P is positive, we can apply the intersection property (equation (2.11)) and conclude that $P \models (\boldsymbol{X} \perp \boldsymbol{Y}' \cup \{A\} \mid \boldsymbol{Z})$, that is, $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$.

The second case is where $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z} \neq \mathcal{X}$. Here, we might have that both $\boldsymbol{X}$ and $\boldsymbol{Y}$ are singletons. This case requires a similar argument that uses the induction hypothesis and properties of independence. We leave it as an exercise (exercise 4.9). ■

Our previous results entail that, for positive distributions, the three conditions are equivalent.

Corollary 4.1 *The following three statements are equivalent for a positive distribution P:*

1. $P \models \mathcal{I}_\ell(\mathcal{H})$.
2. $P \models \mathcal{I}_p(\mathcal{H})$.
3. $P \models \mathcal{I}(\mathcal{H})$.

This equivalence relies on the positivity assumption. In particular, for nonpositive distributions, we can provide examples of a distribution P that satisfies one of these properties, but not the stronger one.

Example 4.5

Let P be any distribution over $\mathcal{X} = \{X_1, \ldots, X_n\}$; let $\mathcal{X}' = \{X'_1, \ldots, X'_n\}$. We now construct a distribution $P'(\mathcal{X}, \mathcal{X}')$ whose marginal over $X_1, \ldots, X_n$ is the same as P, and where X'_i is deterministically equal to X_i. Let $\mathcal{H}$ be a Markov network over $\mathcal{X}, \mathcal{X}'$ that contains no edges other than X_i—X'_i. Then, in P', X_i is independent of the rest of the variables in the network given its neighbor X'_i, and similarly for X'_i; thus, $\mathcal{H}$ satisfies the local independencies for every node in the network. Yet clearly $\mathcal{H}$ is not an I-map for P', since $\mathcal{H}$ makes many independence assertions regarding the X_i's that do not hold in P (or in P'). ■

Thus, for nonpositive distributions, the local independencies do not imply the global ones.

A similar construction can be used to show that, for nonpositive distributions, the pairwise independencies do necessarily imply the local independencies.

Example 4.6

Let P be any distribution over $\mathcal{X} = \{X_1, \ldots, X_n\}$, and now consider two auxiliary sets of variables $\mathcal{X}'$ and $\mathcal{X}''$, and define $\mathcal{X}^* = \mathcal{X} \cup \mathcal{X}' \cup \mathcal{X}''$. We now construct a distribution $P'(\mathcal{X}^*)$ whose marginal over $X_1, \ldots, X_n$ is the same as P, and where X'_i and X''_i are both deterministically equal to X_i. Let $\mathcal{H}$ be the empty Markov network over $\mathcal{X}^*$. We argue that this empty network satisfies the pairwise assumptions for every pair of nodes in the network. For example, X_i and X'_i are rendered independent because $\mathcal{X}^* - \{X_i, X'_i\}$ contains X''_i. Similarly, X_i and X_j are independent given X'_i. Thus, $\mathcal{H}$ satisfies the pairwise independencies, but not the local or global independencies. ■

4.3.3 From Distributions to Graphs

Based on our deeper understanding of the independence properties associated with a Markov network, we can now turn to the question of encoding the independencies in a given distribution P using a graph structure. As for Bayesian networks, the notion of an I-map is not sufficient by itself: The complete graph implies no independence assumptions and is hence an I-map for any distribution. We therefore return to the notion of a minimal I-map, defined in definition 3.13, which was defined broadly enough to apply to Markov networks as well.

How can we construct a minimal I-map for a distribution P? Our discussion in section 4.3.2 immediately suggests two approaches for constructing a minimal I-map: one based on the pairwise Markov independencies, and the other based on the local independencies.

In the first approach, we consider the pairwise independencies. They assert that, if the edge $\{X, Y\}$ is not in $\mathcal{H}$, then X and Y must be independent given all other nodes in the graph, regardless of which other edges the graph contains. Thus, at the very least, to guarantee that $\mathcal{H}$ is an I-map, we must add direct edges between all pairs of nodes X and Y such that

$$P \not\models (X \perp Y \mid \mathcal{X} - \{X, Y\}). \tag{4.2}$$

We can now define $\mathcal{H}$ to include an edge X—Y for all X, Y for which equation (4.2) holds.

In the second approach, we use the local independencies and the notion of minimality. For each variable X, we define the neighbors of X to be a minimal set of nodes $\boldsymbol{Y}$ that render X independent of the rest of the nodes. More precisely, define:

Definition 4.12 Markov blanket

A set $\boldsymbol{U}$ is a Markov blanket *of X in a distribution P if $X \notin \boldsymbol{U}$ and if $\boldsymbol{U}$ is a minimal set of nodes such that*

$$(X \perp \mathcal{X} - \{X\} - \boldsymbol{U} \mid \boldsymbol{U}) \in \mathcal{I}(P). \tag{4.3}$$

■

We then define a graph $\mathcal{H}$ by introducing an edge $\{X, Y\}$ for all X and all $Y \in \mathrm{MB}_P(X)$. As defined, this construction is not unique, since there may be several sets $\boldsymbol{U}$ satisfying equation (4.3). However, theorem 4.6 will show that there is only one such minimal set. In fact, we now show that any positive distribution P has a unique minimal I-map, and that both of these constructions produce this I-map.

We begin with the proof for the pairwise definition:

Theorem 4.5

Let P be a positive distribution, and let $\mathcal{H}$ be defined by introducing an edge $\{X, Y\}$ for all X, Y for which equation (4.2) holds. Then the Markov network $\mathcal{H}$ is the unique minimal I-map for P.

Proof The fact that $\mathcal{H}$ is an I-map for P follows immediately from fact that P, by construction, satisfies $\mathcal{I}_p(\mathcal{H})$, and, therefore, by corollary 4.1, also satisfies $\mathcal{I}(\mathcal{H})$. The fact that it is minimal follows from the fact that if we eliminate some edge $\{X, Y\}$ from $\mathcal{H}$, the graph would imply the pairwise independence $(X \perp Y \mid \mathcal{X} - \{X, Y\})$, which we know to be false for P (otherwise, the edge would have been omitted in the construction of $\mathcal{H}$). The uniqueness of the minimal I-map also follows trivially: By the same argument, any other I-map $\mathcal{H}'$ for P must contain at least the edges in $\mathcal{H}$ and is therefore either equal to $\mathcal{H}$ or contains additional edges and is therefore not minimal. ■

It remains to show that the second definition results in the same minimal I-map.

Theorem 4.6

Let P be a positive distribution. For each node X, let $\mathrm{MB}_P(X)$ be a minimal set of nodes $\boldsymbol{U}$ satisfying equation (4.3). We define a graph $\mathcal{H}$ by introducing an edge $\{X, Y\}$ for all X and all $Y \in \mathrm{MB}_P(X)$. Then the Markov network $\mathcal{H}$ is the unique minimal I-map for P.

The proof is left as an exercise (exercise 4.11).

Both of the techniques for constructing a minimal I-map make the assumption that the distribution P is positive. As we have shown, for nonpositive distributions, neither the pairwise independencies nor the local independencies imply the global one. Hence, for a nonpositive distribution P, constructing a graph $\mathcal{H}$ such that P satisfies the pairwise assumptions for $\mathcal{H}$ does not guarantee that $\mathcal{H}$ is an I-map for P. Indeed, we can easily demonstrate that both of these constructions break down for nonpositive distributions.

Example 4.7

Consider a nonpositive distribution P over four binary variables A, B, C, D that assigns nonzero probability only to cases where all four variables take on exactly the same value; for example, we might have $P(a^1, b^1, c^1, d^1) = 0.5$ and $P(a^0, b^0, c^0, d^0) = 0.5$. The graph $\mathcal{H}$ shown in figure 4.7 is one possible output of applying the local independence I-map construction algorithm to P: For example, $P \models (A \perp C, D \mid B)$, and hence $\{B\}$ is a legal choice for $\mathrm{MB}_P(A)$. A similar analysis shows that this network satisfies the Markov blanket condition for all nodes. However, it is not an I-map for the distribution.

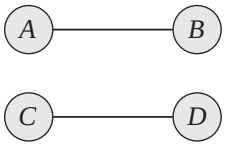


Figure 4.7 An attempt at an I-map for a nonpositive distribution P

If we use the pairwise independence I-map construction algorithm for this distribution, the network constructed is the empty network. For example, the algorithm would not place an edge between A and B, because $P \models (A \perp B \mid C, D)$. Exactly the same analysis shows that no edges will be placed into the graph. However, the resulting network is not an I-map for P. ■

Both these examples show that deterministic relations between variables can lead to failure in the construction based on local and pairwise independence. Suppose that A and B are two variables that are identical to each other and that both C and D are variables that correlated to both A and B so that $(C \perp D \mid A, B)$ holds. Since A is identical to B, we have that both $(A, D \perp C \mid B)$ and $(B, D \perp C \mid A)$ hold. In other words, it suffices to observe one of these two variables to capture the relevant information both have about C and separate C from D. In this case the Markov blanket of C is not uniquely defined. This ambiguity leads to the failure of both local and pairwise constructions. Clearly, identical variables are only one way of getting such ambiguities in local independencies. Once we allow nonpositive distribution, other distributions can have similar problems.

Having defined the notion of a minimal I-map for a distribution P, we can now ask to what extent it represents the independencies in P. More formally, we can ask whether every distribution has a perfect map. Clearly, the answer is no, even for positive distributions:

Example 4.8

Consider a distribution arising from a three-node Bayesian network with a v-structure, for example, the distribution induced in the Student *example over the nodes Intelligence, Difficulty, and Grade (figure 3.3). In the Markov network for this distribution, we must clearly have an edge between I and G and between D and G. Can we omit the edge between I and D? No, because we do not have that $(I \perp D \mid G)$ holds for the distribution; rather, we have the opposite: I and D are* dependent *given G. Therefore, the only minimal I-map for this P is the fully connected graph, which does not capture the marginal independence $(I \perp D)$ that holds in P.* ■

This example provides another counterexample to the strong version of completeness mentioned earlier. The only distributions for which separation is a sound and complete criterion for determining conditional independence are those for which $\mathcal{H}$ is a perfect map.

4.4 Parameterization Revisited

Now that we understand the semantics and independence properties of Markov networks, we revisit some alternative representations for the parameterization of a Markov network.

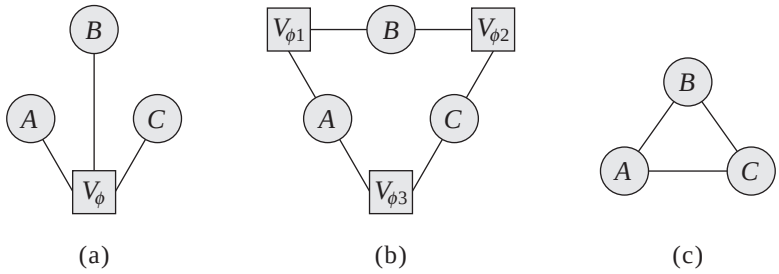


Figure 4.8 Different factor graphs for the same Markov network: (a) One factor graph over A, B, C, with a single factor over all three variables. (b) An alternative factor graph, with three pairwise factors. (c) The induced Markov network for both is a clique over A, B, C.

4.4.1 Finer-Grained Parameterization

4.4.1.1 Factor Graphs

A Markov network structure does not generally reveal all of the structure in a Gibbs parameterization. In particular, one cannot tell from the graph structure whether the factors in the parameterization involve maximal cliques or subsets thereof. Consider, for example, a Gibbs distribution P over a fully connected pairwise Markov network; that is, P is parameterized by a factor for each pair of variables $X, Y \in \mathcal{X}$. The clique potential parameterization would utilize a factor whose scope is the entire graph, and which therefore uses an exponential number of parameters. On the other hand, as we discussed in section 4.2.1, the number of parameters in the pairwise parameterization is quadratic in the number of variables. Note that the complete Markov network is not redundant in terms of conditional independencies — P does not factorize over any smaller network. Thus, although the finer-grained structure does not imply additional independencies in the distribution (see exercise 4.6), it is still very significant.

An alternative representation that makes this structure explicit is a *factor graph*. A factor graph is a graph containing two types of nodes: one type corresponds, as usual, to random variables; the other corresponds to factors over the variables. Formally:

Definition 4.13 factor graph, factorization

A factor graph $\mathcal{F}$ *is an undirected graph containing two types of nodes:* variable nodes *(denoted as ovals) and* factor nodes *(denoted as squares). The graph only contains edges between variable nodes and factor nodes. A factor graph $\mathcal{F}$ is parameterized by a set of factors, where each factor node V_ϕ is associated with precisely one factor ϕ, whose scope is the set of variables that are neighbors of V_ϕ in the graph. A distribution P* factorizes *over $\mathcal{F}$ if it can be represented as a set of factors of this form.* ■

Factor graphs make explicit the structure of the factors in the network. For example, in a fully connected pairwise Markov network, the factor graph would contain a factor node for each of the $\binom{n}{2}$ pairs of nodes; the factor node for a pair X_i, X_j would be connected to X_i and X_j; by contrast, a factor graph for a distribution with a single factor over $X_1, \ldots, X_n$ would have a single factor node connected to all of $X_1, \ldots, X_n$ (see figure 4.8). Thus, although the Markov networks for these two distributions are identical, their factor graphs make explicit the

$\epsilon_1(A,B)$			$\epsilon_2(B,C)$			$\epsilon_3(C,D)$			$\epsilon_4(D,A)$		
a^0	b^0	-3.4	b^0	c^0	-4.61	c^0	d^0	0	d^0	a^0	-4.61
a^0	b^1	-1.61	b^0	c^1	0	c^0	d^1	-4.61	d^0	a^1	0
a^1	b^0	0	b^1	c^0	0	c^1	d^0	-4.61	d^1	a^0	0
a^1	b^1	-2.3	b^1	c^1	-4.61	c^1	d^1	0	d^1	a^1	-4.61
(a)			(b)			(c)			(d)		

Figure 4.9 Energy functions for the Misconception **example**

difference in their factorization.

4.4.1.2 Log-Linear Models

Although factor graphs make certain types of structure more explicit, they still encode factors as complete tables over the scope of the factor. As in Bayesian networks, factors can also exhibit a type of context-specific structure — patterns that involve particular values of the variables. These patterns are often more easily seen in terms of an alternative parameterization of the factors that converts them into log-space.

More precisely, we can rewrite a factor $\phi(\boldsymbol{D})$ as

$$\phi(\boldsymbol{D}) = \exp(-\epsilon(\boldsymbol{D})),$$

energy function

where $\epsilon(\boldsymbol{D}) = -\ln \phi(\boldsymbol{D})$ is often called an *energy function*. The use of the word "energy" derives from statistical physics, where the probability of a physical state (for example, a configuration of a set of electrons), depends inversely on its energy. In this logarithmic representation, we have

$$P(X_1, \ldots, X_n) \propto \exp\left[-\sum_{i=1}^{m} \epsilon_i(\boldsymbol{D}_i)\right].$$

The logarithmic representation ensures that the probability distribution is positive. Moreover, the logarithmic parameters can take any value along the real line.

Any Markov network parameterized using positive factors can be converted to a logarithmic representation.

Example 4.9

Figure 4.9 shows the logarithmic representation of the clique potential parameters in figure 4.1. We can see that the "1" entries in the clique potentials translate into "0" entries in the energy function. ■

This representation makes certain types of structure in the potentials more apparent. For example, we can see that both $\epsilon_2(B,C)$ and $\epsilon_4(D,A)$ are constant multiples of an energy function that ascribes 1 to instantiations where the values of the two variables agree, and 0 to the instantiations where they do not.

We can provide a general framework for capturing such structure using the following notion:

Definition 4.14 feature

Let $\boldsymbol{D}$ be a subset of variables. We define a feature *$f(\boldsymbol{D})$ to be a function from $Val(\boldsymbol{D})$ to $\mathbb{R}$.* ■

A feature is simply a factor without the nonnegativity requirement. One type of feature of particular interest is the *indicator feature* that takes on value 1 for some values $\boldsymbol{y} \in Val(\boldsymbol{D})$ and 0 otherwise.

indicator feature

Features provide us with an easy mechanism for specifying certain types of interactions more compactly.

Example 4.10

Consider a situation where A_1 and A_2 each have ℓ values $a^1, \ldots, a^\ell$. Assume that our distribution is such that we prefer situations where A_1 and A_2 take on the same value, but otherwise have no preference. Thus, our energy function might have the following form:

$$\epsilon(A_1, A_2) = \begin{cases} -3 & A_1 = A_2 \\ 0 & \text{otherwise} \end{cases}$$

Represented as a full factor, this clique potential requires ℓ^2 values. However, it can also be represented as a log-linear function in terms of a feature $f(A_1, A_2)$ that is an indicator function for the event $A_1 = A_2$. The energy function is then simply a constant multiple -3 of this feature. ■

Thus, we can provide a more general definition for our notion of log-linear models:

Definition 4.15 log-linear model

A distribution P is a log-linear model *over a Markov network $\mathcal{H}$ if it is associated with:*

- *a set of features $\mathcal{F} = \{f_1(\boldsymbol{D}_1), \ldots, f_k(\boldsymbol{D}_k)\}$, where each $\boldsymbol{D}_i$ is a complete subgraph in $\mathcal{H}$,*
- *a set of weights $w_1, \ldots, w_k$,*

such that

$$P(X_1, \ldots, X_n) = \frac{1}{Z} \exp\left[-\sum_{i=1}^{k} w_i f_i(\boldsymbol{D}_i)\right].$$

■

Note that we can have several features over the same scope, so that we can, in fact, represent a standard set of table potentials. (See exercise 4.13.)

The log-linear model provides a much more compact representation for many distributions, especially in situations where variables have large domains such as text (such as box 4.E).

4.4.1.3 Discussion

We now have three representations of the parameterization of a Markov network. The Markov network denotes a product over potentials on cliques. A factor graph denotes a product of factors. And a set of features denotes a product over feature weights. Clearly, each representation is finer-grained than the previous one and as rich. A factor graph can describe the Gibbs distribution, and a set of features can describe all the entries in each of the factors of a factor graph.

☞ **Depending on the question of interest, different representations may be more appropriate. For example, a Markov network provides the right level of abstraction for discussing independence queries: The finer-grained representations of factor graphs or log-linear**

models do not change the independence assertions made by the model. On the other hand, as we will see in later chapters, factor graphs are useful when we discuss inference, and features are useful when we discuss parameterizations, both for hand-coded models and for learning.

Box 4.C — Concept: Ising Models and Boltzmann Machines. *One of the earliest types of Markov network models is the* Ising model*, which first arose in statistical physics as a model for the energy of a physical system involving a system of interacting atoms. In these systems, each atom is associated with a binary-valued random variable $X_i \in \{+1, -1\}$, whose value defines the direction of the atom's spin. The energy function associated with the edges is defined by a particularly simple parametric form:*

Ising model

$$\epsilon_{i,j}(x_i, x_j) = w_{i,j} x_i x_j \tag{4.4}$$

This energy is symmetric in X_i, X_j; it makes a contribution of $w_{i,j}$ to the energy function when $X_i = X_j$ (so both atoms have the same spin) and a contribution of $-w_{i,j}$ otherwise. Our model also contains a set of parameters u_i that encode individual node potentials; these bias individual variables to have one spin or another.

As usual, the energy function defines the following distribution:

$$P(\xi) = \frac{1}{Z} \exp\left(-\sum_{i<j} w_{i,j} x_i x_j - \sum_i u_i x_i \right).$$

As we can see, when $w_{i,j} > 0$ the model prefers to align the spins of the two atoms; in this case, the interaction is called ferromagnetic*. When $w_{i,j} < 0$ the interaction is called* antiferromagnetic*. When $w_{i,j} = 0$ the atoms are* non-interacting*.*

Much work has gone into studying particular types of Ising models, attempting to answer a variety of questions, usually as the number of atoms goes to infinity. For example, we might ask the probability of a configuration in which a majority of the spins are $+1$ or -1, versus the probability of more mixed configurations. The answer to this question depends heavily on the strength of the interaction between the variables; so, we can consider adapting this strength (by multiplying all weights by a temperature parameter*) and asking whether this change causes a phase transition in the probability of skewed versus mixed configurations. These questions, and many others, have been investigated extensively by physicists, and the answers are known (in some cases even analytically) for several cases.*

temperature parameter

Related to the Ising model is the Boltzmann distribution*; here, the variables are usually taken to have values $\{0, 1\}$, but still with the energy form of equation (4.4). Here, we get a nonzero contribution to the model from an edge (X_i, X_j) only when $X_i = X_j = 1$; however, the resulting energy can still be reformulated in terms of an Ising model (exercise 4.12).*

Boltzmann distribution

The popularity of the Boltzmann machine was primarily driven by its similarity to an activation model for neurons. To understand the relationship, we note that the probability distribution over each variable X_i given an assignment to is neighbors is $\mathrm{sigmoid}(z)$ where

$$z = -(\sum_j w_{i,j} x_j) - w_i.$$

This function is a sigmoid of a weighted combination of X_i's neighbors, weighted by the strength and direction of the connections between them. This is the simplest but also most popular mathematical approximation of the function employed by a neuron in the brain. Thus, if we imagine a process by which the network continuously adapts its assignment by resampling the value of each variable as a stochastic function of its neighbors, then the "activation" probability of each variable resembles a neuron's activity. This model is a very simple variant of a stochastic, recurrent neural network.

---

labeling MRF

Box 4.D — Concept: Metric MRFs. *One important class of MRFs comprises those used for* labeling. *Here, we have a graph of nodes $X_1, \ldots, X_n$ related by a set of edges $\mathcal{E}$, and we wish to assign to each X_i a label in the space $\mathcal{V} = \{v_1, \ldots, v_K\}$. Each node, taken in isolation, has its preferences among the possible labels. However, we also want to impose a soft "smoothness" constraint over the graph, in that neighboring nodes should take "similar" values.*

We encode the individual node preferences as node potentials in a pairwise MRF and the smoothness preferences as edge potentials. For reasons that will become clear, it is traditional to encode these models in negative log-space, using energy functions. As our objective in these models is inevitably the MAP objective, we can also ignore the partition function, and simply consider the energy function:

$$E(x_1, \ldots, x_n) = \sum_i \epsilon_i(x_i) + \sum_{(i,j)\in\mathcal{E}} \epsilon_{i,j}(x_i, x_j). \tag{4.5}$$

Our goal is then to minimize the energy:

$$\arg \min_{x_1,\ldots,x_n} E(x_1, \ldots, x_n).$$

We now need to provide a formal definition for the intuition of "smoothness" described earlier. There are many different types of conditions that we can impose; different conditions allow different methods to be applied.

Ising model

One of the simplest in this class of models is a slight variant of the Ising model, *where we have that, for any i, j:*

$$\epsilon_{i,j}(x_i, x_j) = \begin{cases} 0 & x_i = x_j \\ \lambda_{i,j} & x_i \neq x_j, \end{cases} \tag{4.6}$$

for $\lambda_{i,j} \geq 0$. In this model, we obtain the lowest possible pairwise energy (0) when two neighboring nodes X_i, X_j take the same value, and a higher energy $\lambda_{i,j}$ when they do not.

Potts model

This simple model has been generalized in many ways. The Potts model *extends it to the setting of more than two labels. An even broader class contains models where we have a distance function on the labels, and where we prefer neighboring nodes to have labels that are a smaller distance*

metric function

apart. More precisely, a function $\mu : \mathcal{V} \times \mathcal{V} \mapsto [0, \infty)$ is a metric *if it satisfies:*

- **Reflexivity:** $\mu(v_k, v_l) = 0$ *if and only if* $k = l$;
- **Symmetry:** $\mu(v_k, v_l) = \mu(v_l, v_k)$;

$\epsilon_1'(A,B)$			$\epsilon_2'(B,C)$		
a^0	b^0	-4.4	b^0	c^0	-3.61
a^0	b^1	-1.61	b^0	c^1	$+1$
a^1	b^0	-1	b^1	c^0	0
a^1	b^1	-2.3	b^1	c^1	-4.61

(a) (b)

Figure 4.10 Alternative but equivalent energy functions

- **Triangle Inequality:** $\mu(v_k, v_l) + \mu(v_l, v_m) \geq \mu(v_k, v_m)$.

semimetric

We say that μ is a semimetric *if it satisfies reflexivity and symmetry. We can now define a* metric MRF *(or a* semimetric MRF*) by defining $\epsilon_{i,j}(v_k, v_l) = \mu(v_k, v_l)$ for all i, j, where μ is a metric (semimetric). We note that, as defined, this model assumes that the distance metric used is the same for all pairs of variables. This assumption is made because it simplifies notation, it often holds in practice, and it reduces the number of parameters that must be acquired. It is not required for the inference algorithms that we present in later chapters. Metric interactions arise in many applications, and play a particularly important role in computer vision (see box 4.B and box 13.B).*

truncated norm

For example, one common metric used is some form of truncated p-norm *(usually $p = 1$ or $p = 2$):*

$$\epsilon(x_i, x_j) = \min(c\|x_i - x_j\|_p, \text{dist}_{\max}). \tag{4.7}$$

4.4.2 Overparameterization

Even if we use finer-grained factors, and in some cases, even features, the Markov network parameterization is generally overparameterized. That is, for any given distribution, there are multiple choices of parameters to describe it in the model. Most obviously, if our graph is a single clique over n binary variables $X_1, \ldots, X_n$, then the network is associated with a clique potential that has 2^n parameters, whereas the joint distribution only has $2^n - 1$ independent parameters.

A more subtle point arises in the context of a nontrivial clique structure. Consider a pair of cliques $\{A, B\}$ and $\{B, C\}$. The energy function $\epsilon_1(A, B)$ (or its corresponding clique potential) contains information not only about the interaction between A and B, but also about the distribution of the individual variables A and B. Similarly, $\epsilon_2(B, C)$ gives us information about the individual variables B and C. The information about B can be placed in either of the two cliques, or its contribution can be split between them in arbitrary ways, resulting in many different ways of specifying the same distribution.

Example 4.11

Consider the energy functions $\epsilon_1(A, B)$ and $\epsilon_2(B, C)$ in figure 4.9. The pair of energy functions shown in figure 4.10 result in an equivalent distribution: Here, we have simply subtracted 1 from $\epsilon_1(A, B)$ and added 1 to $\epsilon_2(B, C)$ for all instantiations where $B = b^0$. It is straightforward to

check that this results in an identical distribution to that of figure 4.9. In instances where $B \neq b^0$ the energy function returns exactly the same value as before. In cases where $B = b^0$, the actual values of the energy functions have changed. However, because the sum of the energy functions on each instance is identical to the original sum, the probability of the instance will not change. ■

Intuitively, the standard Markov network representation gives us too many places to account for the influence of variables in shared cliques. Thus, the same distribution can be represented as a Markov network (of a given structure) in infinitely many ways. It is often useful to pick one of this infinite set as our chosen parameterization for the distribution.

4.4.2.1 Canonical Parameterization

canonical parameterization

The *canonical parameterization* provides one very natural approach to avoiding this ambiguity in the parameterization of a Gibbs distribution P. This canonical parameterization requires that the distribution P be positive. It is most convenient to describe this parameterization using energy functions rather then clique potentials. For this reason, it is also useful to consider a log-transform of P: For any assignment ξ to $\mathcal{X}$, we use $\ell(\xi)$ to denote $\ln P(\xi)$. This transformation is well defined because of our positivity assumption.

The canonical parameterization of a Gibbs distribution over $\mathcal{H}$ is defined via a set of energy functions over all cliques. Thus, for example, the Markov network in figure 4.4b would have energy functions for the two cliques $\{A, B, D\}$ and $\{B, C, D\}$, energy functions for all possible pairs of variables except the pair $\{A, C\}$ (a total of five pairs), energy functions for all four singleton sets, and a constant energy function for the empty clique.

At first glance, it appears that we have only increased the number of parameters in the specification. However, as we will see, this approach uniquely associates the interaction parameters for a subset of variables with that subset, avoiding the ambiguity described earlier. As a consequence, many of the parameters in this canonical parameterization are often zero.

The canonical parameterization is defined relative to a particular fixed assignment $\xi^* = (x_1^*, \ldots, x_n^*)$ to the network variables $\mathcal{X}$. This assignment can be chosen arbitrarily. For any subset of variables $\boldsymbol{Z}$, and any assignment $\boldsymbol{x}$ to some subset of $\mathcal{X}$ that contains $\boldsymbol{Z}$, we define the assignment $\boldsymbol{x}_{\boldsymbol{Z}}$ to be $\boldsymbol{x}\langle \boldsymbol{Z} \rangle$, that is, the assignment in $\boldsymbol{x}$ to the variables in $\boldsymbol{Z}$. Conversely, we define $\xi^*_{-\boldsymbol{Z}}$ to be $\xi^*\langle \mathcal{X} - \boldsymbol{Z} \rangle$, that is, the assignment in ξ^* to the variables outside $\boldsymbol{Z}$. We can now construct an assignment $(\boldsymbol{x}_{\boldsymbol{Z}}, \xi^*_{-\boldsymbol{Z}})$ that keeps the assignments to the variables in $\boldsymbol{Z}$ as specified in $\boldsymbol{x}$, and augments it using the default values in ξ^*.

canonical energy function

The *canonical energy function* for a clique $\boldsymbol{D}$ is now defined as follows:

$$\epsilon^*_{\boldsymbol{D}}(\boldsymbol{d}) = \sum_{\boldsymbol{Z} \subseteq \boldsymbol{D}} (-1)^{|\boldsymbol{D}-\boldsymbol{Z}|} \ell(\boldsymbol{d}_{\boldsymbol{Z}}, \xi^*_{-\boldsymbol{Z}}), \tag{4.8}$$

where the sum is over all subsets of $\boldsymbol{D}$, including $\boldsymbol{D}$ itself and the empty set $\emptyset$. Note that all of the terms in the summation have a scope that is contained in $\boldsymbol{D}$, which in turn is part of a clique, so that these energy functions are legal relative to our Markov network structure.

This formula performs an inclusion-exclusion computation. For a set $\{A, B, C\}$, it first subtracts out the influence of all of the pairs: $\{A, B\}$, $\{B, C\}$, and $\{C, A\}$. However, this process oversubtracts the influence of the individual variables. Thus, their influence is added back in, to compensate. More generally, consider any subset of variables $\boldsymbol{Z} \subseteq \boldsymbol{D}$. Intuitively, it

$\epsilon_1^*(A,B)$			$\epsilon_2^*(B,C)$			$\epsilon_3^*(C,D)$			$\epsilon_4^*(D,A)$		
a^0	b^0	0	b^0	c^0	0	c^0	d^0	0	d^0	a^0	0
a^0	b^1	0	b^0	c^1	0	c^0	d^1	0	d^0	a^1	0
a^1	b^0	0	b^1	c^0	0	c^1	d^0	0	d^1	a^0	0
a^1	b^1	4.09	b^1	c^1	9.21	c^1	d^1	−9.21	d^1	a^1	9.21

$\epsilon_5^*(A)$		$\epsilon_6^*(B)$		$\epsilon_7^*(C)$		$\epsilon_8^*(D)$		$\epsilon_9^*(\emptyset)$
a^0	0	b^0	0	c^0	0	d^0	0	−3.18
a^1	−8.01	b^1	−6.4	c^1	0	d^1	0	

Figure 4.11 Canonical energy function for the Misconception **example**

makes a "contribution" once for every subset $\boldsymbol{U} \supseteq \boldsymbol{Z}$. Except for $\boldsymbol{U} = \boldsymbol{D}$, the number of times that $\boldsymbol{Z}$ appears is even — there is an even number of subsets $\boldsymbol{U} \supseteq \boldsymbol{Z}$ — and the number of times it appears with a positive sign is equal to the number of times it appears with a negative sign. Thus, we have effectively eliminated the net contribution of the subsets from the canonical energy function.

Let us consider the effect of the canonical transformation on our Misconception network.

Example 4.12

Let us choose (a^0, b^0, c^0, d^0) as our arbitrary assignment on which to base the canonical parameterization. The resulting energy functions are shown in figure 4.11. For example, the energy value $\epsilon_1^*(a^1, b^1)$ was computed as follows:

$$\ell(a^1, b^1, c^0, d^0) - \ell(a^1, b^0, c^0, d^0) - \ell(a^0, b^1, c^0, d^0) + \ell(a^0, b^0, c^0, d^0) = \\ -13.49 - -11.18 - -9.58 + -3.18 = 4.09$$

Note that many of the entries in the energy functions are zero. As discussed earlier, this phenomenon is fairly general, and occurs because we have accounted for the influence of small subsets of variables separately, leaving the larger factors to deal only with higher-order influences. We also note that these canonical parameters are not very intuitive, highlighting yet again the difficulties of constructing a reasonable parameterization of a Markov network by hand. ■

This canonical parameterization defines the same distribution as our original distribution P:

Theorem 4.7

Let P be a positive Gibbs distribution over $\mathcal{H}$, and let $\epsilon^*(\boldsymbol{D}_i)$ for each clique $\boldsymbol{D}_i$ be defined as specified in equation (4.8). Then

$$P(\xi) = \exp\left[\sum_i \epsilon^*_{\boldsymbol{D}_i}(\xi\langle \boldsymbol{D}_i \rangle)\right].$$

The proof for the case where $\mathcal{H}$ consists of a single clique is fairly simple, and it is left as an exercise (exercise 4.4). The general case follows from results in the next section.

The canonical parameterization gives us the tools to prove the Hammersley-Clifford theorem, which we restate for convenience.

Theorem 4.8 *Let P be a positive distribution over $\mathcal{X}$, and $\mathcal{H}$ a Markov network graph over $\mathcal{X}$. If $\mathcal{H}$ is an I-map for P, then P is a Gibbs distribution over $\mathcal{H}$.*

PROOF To prove this result, we need to show the existence of a Gibbs parameterization for any distribution P that satisfies the Markov assumptions associated with $\mathcal{H}$. The proof is constructive, and simply uses the canonical parameterization shown earlier in this section. Given P, we define an energy function for all subsets $\boldsymbol{D}$ of nodes in the graph, regardless of whether they are cliques in the graph. This energy function is defined exactly as in equation (4.8), relative to some specific fixed assignment ξ^* used to define the canonical parameterization. The distribution defined using this set of energy functions is P: the argument is identical to the proof of theorem 4.7, for the case where the graph consists of a single clique (see exercise 4.4).

It remains only to show that the resulting distribution is a Gibbs distribution over $\mathcal{H}$. To show that, we need to show that the factors $\epsilon^*(\boldsymbol{D})$ are identically 0 whenever $\boldsymbol{D}$ is not a clique in the graph, that is, whenever the nodes in $\boldsymbol{D}$ do not form a fully connected subgraph. Assume that we have $X, Y \in \boldsymbol{D}$ such that there is no edge between X and Y. For this proof, it helps to introduce the notation

$$\sigma_{\boldsymbol{Z}}[\boldsymbol{x}] = (\boldsymbol{x}_{\boldsymbol{Z}}, \xi^*_{-\boldsymbol{Z}}).$$

Plugging this notation into equation (4.8), we have that:

$$\epsilon^*_{\boldsymbol{D}}(\boldsymbol{d}) = \sum_{\boldsymbol{Z} \subseteq \boldsymbol{D}} (-1)^{|\boldsymbol{D}-\boldsymbol{Z}|} \ell(\sigma_{\boldsymbol{Z}}[\boldsymbol{d}]).$$

We now rearrange the sum over subsets $\boldsymbol{Z}$ into a sum over groups of subsets. Let $\boldsymbol{W} \subseteq \boldsymbol{D} - \{X, Y\}$; then $\boldsymbol{W}$, $\boldsymbol{W} \cup \{X\}$, $\boldsymbol{W} \cup \{Y\}$, and $\boldsymbol{W} \cup \{X, Y\}$ are all subsets of $\boldsymbol{Z}$. Hence, we can rewrite the summation over subsets of $\boldsymbol{D}$ as a summation over subsets of $\boldsymbol{D} - \{X, Y\}$:

$$\begin{aligned} \epsilon^*_{\boldsymbol{D}}(\boldsymbol{d}) \quad = \quad & \sum_{\boldsymbol{W} \subseteq \boldsymbol{D} - \{X,Y\}} (-1)^{|\boldsymbol{D}-\{X,Y\}-\boldsymbol{W}|} \\ & (\ell(\sigma_{\boldsymbol{W}}[\boldsymbol{d}]) - \ell(\sigma_{\boldsymbol{W} \cup \{X\}}[\boldsymbol{d}]) - \ell(\sigma_{\boldsymbol{W} \cup \{Y\}}[\boldsymbol{d}]) + \ell(\sigma_{\boldsymbol{W} \cup \{X,Y\}}[\boldsymbol{d}])). \end{aligned} \tag{4.9}$$

Now consider a specific subset $\boldsymbol{W}$ in this sum, and let $\boldsymbol{u}^*$ be $\xi^*\langle \mathcal{X} - \boldsymbol{D} \rangle$ — the assignment to $\mathcal{X} - \boldsymbol{D}$ in ξ. We now have that:

$$\begin{aligned} \ell(\sigma_{\boldsymbol{W} \cup \{X,Y\}}[\boldsymbol{d}])) - \ell(\sigma_{\boldsymbol{W} \cup \{X\}}[\boldsymbol{d}]) &= \ln \frac{P(x, y, \boldsymbol{w}, \boldsymbol{u}^*)}{P(x, y^*, \boldsymbol{w}, \boldsymbol{u}^*)} \\ &= \ln \frac{P(y \mid x, \boldsymbol{w}, \boldsymbol{u}^*) P(x, \boldsymbol{w}, \boldsymbol{u}^*)}{P(y^* \mid x, \boldsymbol{w}, \boldsymbol{u}^*) P(x, \boldsymbol{w}, \boldsymbol{u}^*)} \\ &= \ln \frac{P(y \mid x^*, \boldsymbol{w}, \boldsymbol{u}^*) P(x, \boldsymbol{w}, \boldsymbol{u}^*)}{P(y^* \mid x^*, \boldsymbol{w}, \boldsymbol{u}^*) P(x, \boldsymbol{w}, \boldsymbol{u}^*)} \\ &= \ln \frac{P(y \mid x^*, \boldsymbol{w}, \boldsymbol{u}^*) P(x^*, \boldsymbol{w}, \boldsymbol{u}^*)}{P(y^* \mid x^*, \boldsymbol{w}, \boldsymbol{u}^*) P(x^*, \boldsymbol{w}, \boldsymbol{u}^*)} \\ &= \ln \frac{P(x^*, y, \boldsymbol{w}, \boldsymbol{u}^*)}{P(x^*, y^*, \boldsymbol{w}, \boldsymbol{u}^*)} \\ &= \ell(\sigma_{\boldsymbol{W} \cup \{Y\}}[\boldsymbol{d}])) - \ell(\sigma_{\boldsymbol{W}}[\boldsymbol{d}]), \end{aligned}$$

where the third equality is a consequence of the fact that X and Y are not connected directly by an edge, and hence we have that $P \models (X \perp Y \mid \mathcal{X} - \{X, Y\})$. Thus, we have that each term in the outside summation in equation (4.9) adds to zero, and hence the summation as a whole is also zero, as required. ■

For positive distributions, we have already shown that all three sets of Markov assumptions are equivalent; putting these results together with theorem 4.1 and theorem 4.2, we obtain that, **for positive distributions, all four conditions — factorization and the three types of Markov assumptions — are all equivalent.**

4.4.2.2 Eliminating Redundancy

An alternative approach to the issue of overparameterization is to try to eliminate it entirely. We can do so in the context of a feature-based representation, which is sufficiently fine-grained to allow us to eliminate redundancies without losing expressive power. The tools for detecting and eliminating redundancies come from linear algebra.

linear dependence

We say that a set of features $f_1, \ldots, f_k$ is *linearly dependent* if there are constants $\alpha_0, \alpha_1, \ldots, \alpha_k$, not all of which are 0, so that for all ξ

$$\alpha_0 + \sum_i \alpha_i f_i(\xi) = 0.$$

This is the usual definition of linear dependencies in linear algebra, where we view each feature as a vector whose entries are the value of the feature in each of the possible instantiations.

Example 4.13

Consider again the Misconception *example. We can encode the log-factors in example 4.9 as a set of features by introducing indicator features of the form:*

$$f_{a,b}(A, B) = \begin{cases} 1 & A = a, B = b \\ 0 & \text{otherwise.} \end{cases}$$

Thus, to represent $\epsilon_1(A, B)$, we introduce four features that correspond to the four entries in the energy function. Since A, B take on exactly one of these possible four values, we have that

$$f_{a^0,b^0}(A, B) + f_{a^0,b^1}(A, B) + f_{a^1,b^0}(A, B) + f_{a^1,b^1}(A, B) = 1.$$

Thus, this set of features is linearly dependent. ■

Example 4.14

Now consider also the features that capture $\epsilon_2(B, C)$ and their interplay with the features that capture $\epsilon_1(A, B)$. We start by noting that the sum $f_{a^0,b^0}(A, B) + f_{a^1,b^0}(A, B)$ is equal to 1 when $B = b^0$ and 0 otherwise. Similarly, $f_{b^0,c^0}(B, C) + f_{b^0,c^1}(B, C)$ is also an indicator for $B = b^0$. Thus we get that

$$f_{a^0,b^0}(A, B) + f_{a^1,b^0}(A, B) - f_{b^0,c^0}(B, C) - f_{b^0,c^1}(B, C) = 0.$$

And so these four features are linearly dependent. ■

As we now show, linear dependencies imply non-unique parameterization.

Proposition 4.5 *Let $f_1, \ldots, f_k$ be a set of features with weights $\boldsymbol{w} = \{w_1, \ldots, w_k\}$ that form a log-linear representation of a distribution P. If there are coefficients $\alpha_0, \alpha_1, \ldots, \alpha_k$ such that for all ξ*

$$\alpha_0 + \sum_i \alpha_i f_i(\xi) = 0 \tag{4.10}$$

then the log-linear model with weights $\boldsymbol{w}' = \{w_1 + \alpha_1, \ldots, w_k + \alpha_k\}$ also represents P.

PROOF Consider the distribution

$$P_{\boldsymbol{w}'}(\xi) \propto \exp\left\{-\sum_i (w_i + \alpha_i) f_i(\xi)\right\}.$$

Using equation (4.10) we see that

$$-\sum_i (w_i + \alpha_i) f_i(\xi) = \alpha_0 - \sum_i w_i f_i(\xi).$$

Thus,

$$P_{\boldsymbol{w}'}(\xi) \propto e^{\alpha_0} \exp\left\{-\sum_i w_i f_i(\xi)\right\} \propto P(\xi).$$

We conclude that $P_{\boldsymbol{w}'}(\xi) = P(\xi)$. ∎

redundant

Motivated by this result, we say that a set of linearly dependent features is *redundant*. A *nonredundant* set of features is one where the features are not linearly dependent on each other. In fact, if the set of features is nonredundant, then each set of weights describes a unique distribution.

Proposition 4.6 *Let $f_1, \ldots, f_k$ be a set of nonredundant features, and let $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^k$. If $\boldsymbol{w} \neq \boldsymbol{w}'$ then $P_{\boldsymbol{w}} \neq P_{\boldsymbol{w}'}$.*

Example 4.15 *Can we construct a nonredundant set of features for the* Misconception *example? We can determine the number of nonredundant features by building the 16×16 matrix of the values of the 16 features (four factors with four features each) in the 16 instances of the joint distribution. This matrix has rank of 9, which implies that a subset of 8 features will be a nonredundant subset. In fact, there are several such subsets. In particular, the canonical parameterization shown in figure 4.11 has nine features of nonzero weight, which form a nonredundant parameterization. The equivalence of the canonical parameterization (theorem 4.7) implies that this set of features has the same expressive power as the original set of features. To verify this, we can show that adding any other feature will lead to a linear dependency. Consider, for example, the feature f_{a^1,b^0}. We can verify that*

$$f_{a^1,b^0} + f_{a^1,b^1} - f_{a^1} = 0.$$

Similarly, consider the feature f_{a^0,b^0}. Again we can find a linear dependency on other features:

$$f_{a^0,b^0} + f_{a^1} + f_{b^1} - f_{a^1,b^1} = 1.$$

Using similar arguments, we can show that adding any of the original features will lead to redundancy. Thus, this set of features can represent any parameterization in the original model. ∎

4.5 Bayesian Networks and Markov Networks

We have now described two graphical representation languages: Bayesian networks and Markov networks. Example 3.8 and example 4.8 show that these two representations are incomparable as a language for representing independencies: each can represent independence constraints that the other cannot. In this section, we strive to provide more insight about the relationship between these two representations.

4.5.1 From Bayesian Networks to Markov Networks

Let us begin by examining how we might take a distribution represented using one of these frameworks, and represent it in the other. One can view this endeavor from two different perspectives: Given a Bayesian network $\mathcal{B}$, we can ask how to represent the distribution $P_{\mathcal{B}}$ as a parameterized Markov network; or, given a graph $\mathcal{G}$, we can ask how to represent the independencies in $\mathcal{G}$ using an undirected graph $\mathcal{H}$. In other words, we might be interested in finding a minimal I-map for a distribution $P_{\mathcal{B}}$, or a minimal I-map for the independencies $\mathcal{I}(\mathcal{G})$. We can see that these two questions are related, but each perspective offers its own insights.

Let us begin by considering a distribution $P_{\mathcal{B}}$, where $\mathcal{B}$ is a parameterized Bayesian network over a graph $\mathcal{G}$. Importantly, the parameterization of $\mathcal{B}$ can also be viewed as a parameterization for a Gibbs distribution: We simply take each CPD $P(X_i \mid \mathrm{Pa}_{X_i})$ and view it as a factor of scope X_i, Pa_{X_i}. This factor satisfies additional normalization properties that are not generally true of all factors, but it is still a legal factor. This set of factors defines a Gibbs distribution, one whose partition function happens to be 1.

What is more important, **a Bayesian network conditioned on evidence $\boldsymbol{E} = \boldsymbol{e}$ also induces a Gibbs distribution: the one defined by the original factors *reduced* to the context $\boldsymbol{E} = \boldsymbol{e}$.**

Proposition 4.7 *Let $\mathcal{B}$ be a Bayesian network over $\mathcal{X}$ and $\boldsymbol{E} = \boldsymbol{e}$ an observation. Let $\boldsymbol{W} = \mathcal{X} - \boldsymbol{E}$. Then $P_{\mathcal{B}}(\boldsymbol{W} \mid \boldsymbol{e})$ is a Gibbs distribution defined by the factors $\Phi = \{\phi_{X_i}\}_{X_i \in \mathcal{X}}$, where*

$$\phi_{X_i} = P_{\mathcal{B}}(X_i \mid \mathrm{Pa}_{X_i})[\boldsymbol{E} = \boldsymbol{e}].$$

The partition function for this Gibbs distribution is $P(\boldsymbol{e})$.

The proof follows directly from the definitions. This result allows us to view any Bayesian network conditioned as evidence as a Gibbs distribution, and to bring to bear techniques developed for analysis of Markov networks.

What is the structure of the undirected graph that can serve as an I-map for a set of factors in a Bayesian network? In other words, what is the I-map for the Bayesian network structure $\mathcal{G}$? Going back to our construction, we see that we have created a factor for each family of X_i, containing all the variables in the family. Thus, in the undirected I-map, we need to have an edge between X_i and each of its parents, as well as between all of the parents of X_i. This observation motivates the following definition:

Definition 4.16 moralized graph

The moral graph $\mathcal{M}[\mathcal{G}]$ *of a Bayesian network structure $\mathcal{G}$ over $\mathcal{X}$ is the undirected graph over $\mathcal{X}$ that contains an undirected edge between X and Y if: (a) there is a directed edge between them (in either direction), or (b) X and Y are both parents of the same node.*[1] ■

1. The name *moralized graph* originated because of the supposed "morality" of marrying the parents of a node.

For example, figure 4.6a shows the moralized graph for the extended $\mathcal{B}^{student}$ network of figure 9.8.

The preceding discussion shows the following result:

Corollary 4.2 *Let $\mathcal{G}$ be a Bayesian network structure. Then for any distribution $P_{\mathcal{B}}$ such that $\mathcal{B}$ is a parameterization of $\mathcal{G}$, we have that $\mathcal{M}[\mathcal{G}]$ is an I-map for $P_{\mathcal{B}}$.*

One can also view the moralized graph construction purely from the perspective of the independencies encoded by a graph, avoiding completely the discussion of parameterizations of the network.

Proposition 4.8 *Let $\mathcal{G}$ be any Bayesian network graph. The moralized graph $\mathcal{M}[\mathcal{G}]$ is a minimal I-map for $\mathcal{G}$.*

Proof We want to build a Markov network $\mathcal{H}$ such that $\mathcal{I}(\mathcal{H}) \subseteq \mathcal{I}(\mathcal{G})$, that is, that $\mathcal{H}$ is an I-map for $\mathcal{G}$ (see definition 3.3). We use the algorithm for constructing minimal I-maps based on the Markov independencies. Consider a node X in $\mathcal{X}$: our task is to select as X's neighbors the smallest set of nodes $\boldsymbol{U}$ that are needed to render X independent of all other nodes in the network. We define the *Markov blanket* of X in a Bayesian network $\mathcal{G}$, denoted $\mathrm{MB}_{\mathcal{G}}(X)$, to be the nodes consisting of X's parents, X's children, and other parents of X's children. We now need to show that $\mathrm{MB}_{\mathcal{G}}(X)$ d-separates X from all other variables in $\mathcal{G}$; and that no subset of $\mathrm{MB}_{\mathcal{G}}(X)$ has that property. The proof uses straightforward graph-theoretic properties of trails, and it is left as an exercise (exercise 4.14). ■

Markov blanket

Now, let us consider how "close" the moralized graph is to the original graph $\mathcal{G}$. Intuitively, **the addition of the moralizing edges to the Markov network $\mathcal{H}$ leads to the loss of independence information implied by the graph structure.** For example, if our Bayesian network $\mathcal{G}$ has the form $X \rightarrow Z \leftarrow Y$, with no edge between X and Y, the Markov network $\mathcal{M}[\mathcal{G}]$ loses the information that X and Y are marginally independent (not given Z). However, information is not always lost. Intuitively, moralization causes loss of information about independencies only when it introduces new edges into the graph. We say that a Bayesian network $\mathcal{G}$ is *moral* if it contains no immoralities (as in definition 3.11); that is, for any pair of variables X, Y that share a child, there is a covering edge between X and Y. It is not difficult to show that:

moral graph

Proposition 4.9 *If the directed graph $\mathcal{G}$ is moral, then its moralized graph $\mathcal{M}[\mathcal{G}]$ is a perfect map of $\mathcal{G}$.*

Proof Let $\mathcal{H} = \mathcal{M}[\mathcal{G}]$. We have already shown that $\mathcal{I}(\mathcal{H}) \subseteq \mathcal{I}(\mathcal{G})$, so it remains to show the opposite inclusion. Assume by contradiction that there is an independence $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \in \mathcal{I}(\mathcal{G})$ which is not in $\mathcal{I}(\mathcal{H})$. Thus, there must exist some trail from $\boldsymbol{X}$ to $\boldsymbol{Y}$ in $\mathcal{H}$ which is active given $\boldsymbol{Z}$. Consider some such trail that is minimal, in the sense that it has no shortcuts. As $\mathcal{H}$ and $\mathcal{G}$ have precisely the same edges, the same trail must exist in $\mathcal{G}$. As, by assumption, it cannot be active in $\mathcal{G}$ given $\boldsymbol{Z}$, we conclude that it must contain a v-structure $X_1 \rightarrow X_2 \leftarrow X_3$. However, because $\mathcal{G}$ is moralized, we also have some edge between X_1 and X_3, contradicting the assumption that the trail is minimal. ■

Thus, a moral graph $\mathcal{G}$ can be converted to a Markov network without losing independence assumptions. This conclusion is fairly intuitive, inasmuch as the only independencies in $\mathcal{G}$ that are not present in an undirected graph containing the same edges are those corresponding to

v-structures. But if any v-structure can be short-cut, it induces no independencies that are not represented in the undirected graph.

We note, however, that very few directed graphs are moral. For example, assume that we have a v-structure $X \rightarrow Y \leftarrow Z$, which is moral due to the existence of an arc $X \rightarrow Z$. If Z has another parent W, it also has a v-structure $X \rightarrow Z \leftarrow W$, which, to be moral, requires some edge between X and W. We return to this issue in section 4.5.3.

4.5.1.1 Soundness of d-Separation

The connection between Bayesian networks and Markov networks provides us with the tools for proving the soundness of the d-separation criterion in Bayesian networks.

The idea behind the proof is to leverage the soundness of separation in undirected graphs, a result which (as we showed) is much easier to prove. Thus, we want to construct an undirected graph $\mathcal{H}$ such that active paths in $\mathcal{H}$ correspond to active paths in $\mathcal{G}$. A moment of thought shows that the moralized graph is not the right construct, because there are paths in the undirected graph that correspond to v-structures in $\mathcal{G}$ that may or may not be active. For example, if our graph $\mathcal{G}$ is $X \rightarrow Z \leftarrow Y$ and Z is not observed, d-separation tells us that X and Y are independent; but the moralized graph for $\mathcal{G}$ is the complete undirected graph, which does not have the same independence.

barren node

Therefore, to show the result, we first want to eliminate v-structures that are not active, so as to remove such cases. To do so, we first construct a subgraph where remove all *barren nodes* from the graph, thereby also removing all v-structures that do not have an observed descendant. The elimination of the barren nodes does not change the independence properties of the distribution over the remaining variables, but does eliminate paths in the graph involving v-structures that are not active. If we now consider only the subgraph, we can reduce d-separation to separation and utilize the soundness of separation to show the desired result.

upward closure

We first use these intuitions to provide an alternative formulation for d-separation. Recall that in definition 2.14 we defined the *upward closure* of a set of nodes $\boldsymbol{U}$ in a graph to be $\boldsymbol{U} \cup \mathit{Ancestors}_{\boldsymbol{U}}$. Letting $\boldsymbol{U}^*$ be the closure of a set $\boldsymbol{U}$, we can define the network induced over $\boldsymbol{U}^*$; importantly, as all parents of every node in $\boldsymbol{U}^*$ are also in $\boldsymbol{U}^*$, we have all the variables mentioned in every CPD, so that the induced graph defines a coherent probability distribution. We let $\mathcal{G}^+[\boldsymbol{U}]$ be the induced Bayesian network over $\boldsymbol{U}$ and its ancestors.

Proposition 4.10 *Let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ be three disjoint sets of nodes in a Bayesian network $\mathcal{G}$. Let $\boldsymbol{U} = \boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$, and let $\mathcal{G}' = \mathcal{G}^+[\boldsymbol{U}]$ be the induced Bayesian network over $\boldsymbol{U} \cup \text{Ancestors}_{\boldsymbol{U}}$. Let $\mathcal{H}$ be the moralized graph $\mathcal{M}[\mathcal{G}']$. Then* $\text{d-sep}_{\mathcal{G}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$ *if and only if* $\text{sep}_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$.

Example 4.16 *To gain some intuition for this result, consider the Bayesian network $\mathcal{G}$ of figure 4.12a (which extends our* Student *network). Consider the d-separation query* $\text{d-sep}_{\mathcal{G}}(D; I \mid L)$. *In this case, $\boldsymbol{U} = \{D, I, L\}$, and hence the moralized graph $\mathcal{M}[\mathcal{G}^+[\boldsymbol{U}]]$ is the graph shown in figure 4.12b, where we have introduced an undirected moralizing edge between D and I. In the resulting graph, D and I are not separated given L, exactly as we would have concluded using the d-separation procedure on the original graph.*

On the other hand, consider the d-separation query $\text{d-sep}_{\mathcal{G}}(D; I \mid S, A)$. *In this case, $\boldsymbol{U} = \{D, I, S, A\}$. Because D and I are not spouses in $\mathcal{G}^+[\boldsymbol{U}]$, the moralization process does not add*

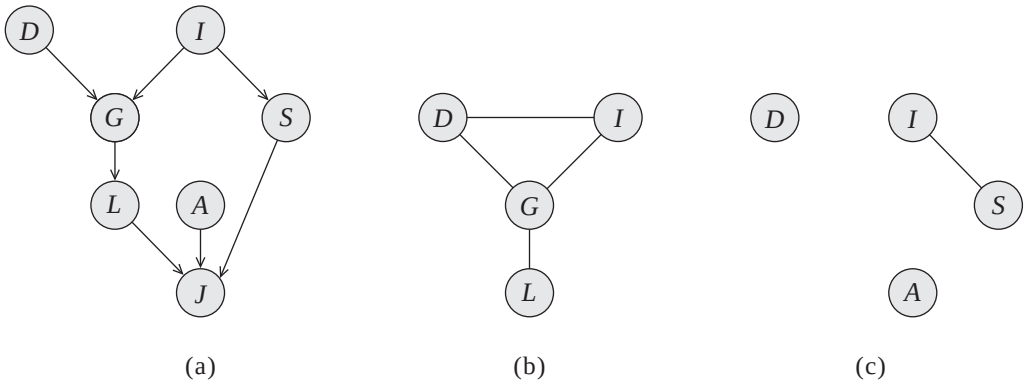


Figure 4.12 Example of alternative definition of d-separation based on Markov networks. (a) A Bayesian network $\mathcal{G}$. (b) The Markov network $\mathcal{M}[\mathcal{G}^+[D, I, L]]$. (c) The Markov network $\mathcal{M}[\mathcal{G}^+[D, I, A, S]]$.

an edge between them. The resulting moralized graph is shown in figure 4.12c. As we can see, we have that $\text{sep}_{\mathcal{M}[\mathcal{G}^+[\boldsymbol{U}]]}(D; I \mid S, A)$*, as desired.* ■

The proof for the general case is similar and is left as an exercise (exercise 4.15).

With this result, the soundness of d-separation follows easily. We repeat the statement of theorem 3.3:

Theorem 4.9

If a distribution $P_{\mathcal{B}}$ factorizes according to $\mathcal{G}$, then $\mathcal{G}$ is an I-map for P.

PROOF As in proposition 4.10, let $\boldsymbol{U} = \boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$, let $\boldsymbol{U}^* = \boldsymbol{U} \cup \mathit{Ancestors}_{\boldsymbol{U}}$, let $\mathcal{G}_{\boldsymbol{U}^*} = \mathcal{G}^+[\boldsymbol{U}]$ be the induced graph over $\boldsymbol{U}^*$, and let $\mathcal{H}$ be the moralized graph $\mathcal{M}[\mathcal{G}_{\boldsymbol{U}^*}]$. Let $P_{\boldsymbol{U}^*}$ be the Bayesian network distribution defined over $\mathcal{G}_{\boldsymbol{U}^*}$ in the obvious way: the CPD for any variable in $\boldsymbol{U}^*$ is the same as in $\mathcal{B}$. Because $\boldsymbol{U}^*$ is upwardly closed, all variables used in these CPDs are in $\boldsymbol{U}^*$.

Now, consider an independence assertion $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \in \mathcal{I}(\mathcal{G})$; we want to prove that $P_{\mathcal{B}} \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$. By definition 3.7, if $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \in \mathcal{I}(\mathcal{G})$, we have that $\mathit{d\text{-}sep}_{\mathcal{G}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$. It follows that $\mathit{sep}_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$, and hence that $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \in \mathcal{I}(\mathcal{H})$. $P_{\boldsymbol{U}^*}$ is a Gibbs distribution over $\mathcal{H}$, and hence, from theorem 4.1, $P_{\boldsymbol{U}^*} \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$. Using exercise 3.8, the distribution $P_{\boldsymbol{U}^*}(\boldsymbol{U}^*)$ is the same as $P_{\mathcal{B}}(\boldsymbol{U}^*)$. Hence, it follows also that $P_{\mathcal{B}} \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$, proving the desired result. ■

4.5.2 From Markov Networks to Bayesian Networks

The previous section dealt with the conversion from a Bayesian network to a Markov network. We now consider the converse transformation: finding a Bayesian network that is a minimal I-map for a Markov network. It turns out that the transformation in this direction is significantly more difficult, both conceptually and computationally. Indeed, the Bayesian network that is a minimal I-map for a Markov network might be considerably larger than the Markov network.

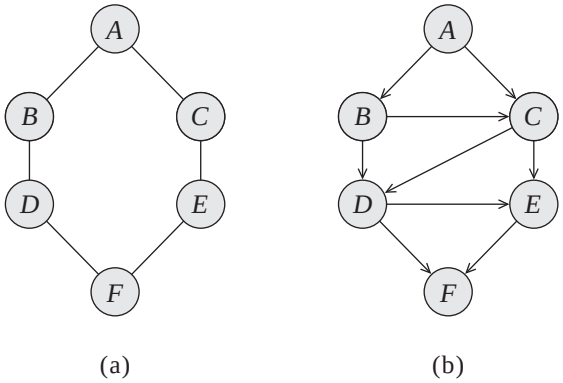


Figure 4.13 Minimal I-map Bayesian networks for a nonchordal Markov network. (a) A Markov network $\mathcal{H}_\ell$ with a loop. (b) A minimal I-map $\mathcal{G}_\ell$ Bayesian network for $\mathcal{H}$.

Example 4.17

Consider the Markov network structure $\mathcal{H}_\ell$ of figure 4.13a, and assume that we want to find a Bayesian network I-map for $\mathcal{H}_\ell$. As we discussed in section 3.4.1, we can find such an I-map by enumerating the nodes in $\mathcal{X}$ in some ordering, and define the parent set for each one in turn according to the independencies in the distribution. Assume we enumerate the nodes in the order A, B, C, D, E, F. The process for A and B is obvious. Consider what happens when we add C. We must, of course, introduce A as a parent for C. More interestingly, however, C is not independent of B given A; hence, we must also add B as a parent for C. Now, consider the node D. One of its parents must be B. As D is not independent of C given B, we must add C as a parent for B. We do not need to add A, as D is independent of A given B and C. Similarly, E's parents must be C and D. Overall, the minimal Bayesian network I-map according to this ordering has the structure $\mathcal{G}_\ell$ shown in figure 4.13b. ■

A quick examination of the structure $\mathcal{G}_\ell$ shows that we have added several edges to the graph, resulting in a set of triangles crisscrossing the loop. In fact, the graph $\mathcal{G}_\ell$ in figure 4.13b is chordal: all loops have been partitioned into triangles.

One might hope that a different ordering might lead to fewer edges being introduced. Unfortunately, this phenomenon is a general one: any Bayesian network I-map for this Markov network must add triangulating edges into the graph, so that the resulting graph is chordal (see definition 2.24). In fact, we can show the following property, which is even stronger:

Theorem 4.10

Let $\mathcal{H}$ be a Markov network structure, and let $\mathcal{G}$ be any Bayesian network minimal I-map for $\mathcal{H}$. Then $\mathcal{G}$ can have no immoralities (see definition 3.11).

PROOF Let $X_1, \ldots, X_n$ be a topological ordering for $\mathcal{G}$. Assume, by contradiction, that there is some immorality $X_i \rightarrow X_j \leftarrow X_k$ in $\mathcal{G}$ such that there is no edge between X_i and X_k; assume (without loss of generality) that $i < k < j$.

Owing to minimality of the I-map $\mathcal{G}$, if X_i is a parent of X_j, then X_i and X_j are not separated by X_j's other parents. Thus, $\mathcal{H}$ necessarily contains one or more paths between X_i

and X_j that are not cut by X_k (or by X_j's other parents). Similarly, $\mathcal{H}$ necessarily contains one or more paths between X_k and X_j that are not cut by X_i (or by X_j's other parents).

Consider the parent set $\boldsymbol{U}$ that was chosen for X_k. By our previous argument, there are one or more paths in $\mathcal{H}$ between X_i and X_k via X_j. As $i < k$, and X_i is not a parent of X_k (by our assumption), we have that $\boldsymbol{U}$ must cut all of those paths. To do so, $\boldsymbol{U}$ must cut either all of the paths between X_i and X_j, or all of the paths between X_j and X_k: As long as there is at least one active path from X_i to X_j and one from X_j to X_k, there is an active path between X_i and X_k that is not cut by $\boldsymbol{U}$. Assume, without loss of generality, that $\boldsymbol{U}$ cuts all paths between X_j and X_k (the other case is symmetrical). Now, consider the choice of parent set for X_j, and recall that it is the (unique) minimal subset among $X_1, \ldots, X_{j-1}$ that separates X_j from the others. In a Markov network, this set consists of all nodes in $X_1, \ldots, X_{j-1}$ that are the first on some uncut path from X_j. As $\boldsymbol{U}$ separates X_k from X_j, it follows that X_k cannot be the first on any uncut path from X_j, and therefore X_k cannot be a parent of X_j. This result provides the desired contradiction. ■

Because any nontriangulated loop of length at least 4 in a Bayesian network graph necessarily contains an immorality, we conclude:

Corollary 4.3 *Let $\mathcal{H}$ be a Markov network structure, and let $\mathcal{G}$ be any minimal I-map for $\mathcal{H}$. Then $\mathcal{G}$ is necessarily chordal.*

triangulation

Thus, the process of turning a Markov network into a Bayesian network requires that we add enough edges to a graph to make it chordal. This process is called *triangulation*. As in the transformation from Bayesian networks to Markov networks, the addition of edges leads to the loss of independence information. For instance, in example 4.17, the Bayesian network $\mathcal{G}_\ell$ in figure 4.13b loses the information that C and D are independent given A and F. In the transformation from directed to undirected models, however, the edges added are only the ones that are, in some sense, implicitly there — the edges required by the fact that each factor in a Bayesian network involves an entire family (a node and its parents). By contrast, the transformation from Markov networks to Bayesian networks can lead to the introduction of a large number of edges, and, in many cases, to the creation of very large families (exercise 4.16).

4.5.3 Chordal Graphs

We have seen that the conversion in either direction between Bayesian networks to Markov networks can lead to the addition of edges to the graph and to the loss of independence information implied by the graph structure. It is interesting to ask when a set of independence assumptions can be represented perfectly by both a Bayesian network and a Markov network. It turns out that this class is precisely the class of undirected chordal graphs.

The proof of one direction is fairly straightforward, based on our earlier results.

Theorem 4.11 *Let $\mathcal{H}$ be a nonchordal Markov network. Then there is no Bayesian network $\mathcal{G}$ which is a perfect map for $\mathcal{H}$ (that is, such that $\mathcal{I}(\mathcal{H}) = \mathcal{I}(\mathcal{G})$).*

PROOF The proof follows from the fact that the minimal I-map for $\mathcal{G}$ must be chordal. Hence, any I-map $\mathcal{G}$ for $\mathcal{I}(\mathcal{H})$ must include edges that are not present in $\mathcal{H}$. Because any additional edge eliminates independence assumptions, it is not possible for any Bayesian network $\mathcal{G}$ to precisely encode $\mathcal{I}(\mathcal{H})$. ■

To prove the other direction of this equivalence, we first prove some important properties of chordal graphs. As we will see, chordal graphs and the properties we now show play a central role in the derivation of exact inference algorithms for graphical models. For the remainder of this discussion, we restrict attention to connected graphs; the extension to the general case is straightforward. The basic result we show is that we can decompose any connected chordal graph $\mathcal{H}$ into a *tree of cliques* — a tree whose nodes are the maximal cliques in $\mathcal{H}$ — so that the structure of the tree precisely encodes the independencies in $\mathcal{H}$. (In the case of disconnected graphs, we obtain a forest of cliques, rather than a tree.)

We begin by introducing some notation. Let $\mathcal{H}$ be a connected undirected graph, and let $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_k$ be the set of maximal cliques in $\mathcal{H}$. Let $\mathcal{T}$ be any tree-structured graph whose nodes correspond to the maximal cliques $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_k$. Let $\boldsymbol{C}_i, \boldsymbol{C}_j$ be two cliques in the tree that are directly connected by an edge; we define $\boldsymbol{S}_{i,j} = \boldsymbol{C}_i \cap \boldsymbol{C}_j$ to be a *sepset* between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$. Let $\boldsymbol{W}_{<(i,j)}$ ($\boldsymbol{W}_{<(j,i)}$) be all of the variables that appear in any clique on the $\boldsymbol{C}_i$ ($\boldsymbol{C}_j$) side of the edge. Thus, each edge decomposes $\mathcal{X}$ into three disjoint sets: $\boldsymbol{W}_{<(i,j)} - \boldsymbol{S}_{i,j}$, $\boldsymbol{W}_{<(j,i)} - \boldsymbol{S}_{i,j}$, and $\boldsymbol{S}_{i,j}$.

sepset

Definition 4.17 *We say that a tree $\mathcal{T}$ is a* clique tree *for $\mathcal{H}$ if:*

clique tree

- *each node corresponds to a clique in $\mathcal{H}$, and each maximal clique in $\mathcal{H}$ is a node in $\mathcal{T}$;*
- *each sepset $\boldsymbol{S}_{i,j}$ separates $\boldsymbol{W}_{<(i,j)}$ and $\boldsymbol{W}_{<(j,i)}$ in $\mathcal{H}$.* ■

Note that this definition implies that each separator $\boldsymbol{S}_{i,j}$ renders its two sides conditionally independent in $\mathcal{H}$.

Example 4.18 *Consider the Bayesian network graph $\mathcal{G}_\ell$ in figure 4.13b. Since it contains no immoralities, its moralized graph $\mathcal{H}'_\ell$ is simply the same graph, but where all edges have been made undirected. As $\mathcal{G}_\ell$ is chordal, so is $\mathcal{H}'_\ell$. The clique tree for $\mathcal{H}'_\ell$ is simply a chain $\{A, B, C\} \rightarrow \{B, C, D\} \rightarrow \{C, D, E\} \rightarrow \{D, E, F\}$, which clearly satisfies the separation requirements of the clique tree definition.* ■

Theorem 4.12 *Every undirected chordal graph $\mathcal{H}$ has a clique tree $\mathcal{T}$.*

PROOF We prove the theorem by induction on the number of nodes in the graph. The base case of a single node is trivial. Now, consider a chordal graph $\mathcal{H}$ of size > 1. If $\mathcal{H}$ consists of a single clique, then the theorem holds trivially. Therefore, consider the case where we have at least two nodes X_1, X_2 that are not connected directly by an edge. Assume that X_1 and X_2

are connected, otherwise the inductive step holds trivially. Let $\boldsymbol{S}$ be a minimal subset of nodes that separates X_1 and X_2.

The removal of the set $\boldsymbol{S}$ breaks up the graph into at least two disconnected components — one containing X_1, another containing X_2, and perhaps additional ones. Let $\boldsymbol{W}_1, \boldsymbol{W}_2$ be some partition of the variables in $\mathcal{X} - \boldsymbol{S}$ into two disjoint components, such that $\boldsymbol{W}_i$ encompasses the connected component containing X_i. (The other connected components can be assigned to $\boldsymbol{W}_1$ or $\boldsymbol{W}_2$ arbitrarily.) We first show that $\boldsymbol{S}$ must be a complete subgraph. Let Z_1, Z_2 be any two variables in $\boldsymbol{S}$. Due to the minimality of $\boldsymbol{S}$, each Z_i must lie on a path between X_1 and X_2 that does not go through any other node in $\boldsymbol{S}$. (Otherwise, we could eliminate Z_i from $\boldsymbol{S}$ while still maintaining separation.) We can therefore construct a minimal path from Z_1 to Z_2 that goes only through nodes in $\boldsymbol{W}_1$ by constructing a path from Z_1 to X_1 to Z_2 that goes only through $\boldsymbol{W}_1$, and by eliminating any shortcuts. We can similarly construct a minimal path from Z_1 to Z_2 that goes only through nodes in $\boldsymbol{W}_2$. The two paths together form a cycle of length ≥ 4. Because of chordality, the cycle must have a chord, which, by construction, must be the edge Z_1—Z_2.

Now consider the induced graph $\mathcal{H}_1 = \mathcal{H}[\boldsymbol{W}_1 \cup \boldsymbol{S}]$. As $X_2 \notin \mathcal{H}_1$, this induced graph is smaller than $\mathcal{H}$. Moreover, $\mathcal{H}_1$ is chordal, so we can apply the inductive hypothesis. Let $\mathcal{T}_1$ be the clique tree for $\mathcal{H}_1$. Because $\boldsymbol{S}$ is a complete connected subgraph, it is either a maximal clique or a subset of some maximal clique in $\mathcal{H}_1$. Let $\boldsymbol{C}_1$ be some clique in $\mathcal{T}_1$ containing $\boldsymbol{S}$ (there may be more than one such clique). We can similarly define $\mathcal{H}_2$ and $\boldsymbol{C}_2$ for X_2. If neither $\boldsymbol{C}_1$ nor $\boldsymbol{C}_2$ is equal to $\boldsymbol{S}$, we construct a tree $\mathcal{T}$ that contains the union of the cliques in $\mathcal{T}_1$ and $\mathcal{T}_2$, and connects $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ by an edge. Otherwise, without loss of generality, let $\boldsymbol{C}_1 = \boldsymbol{S}$; we create $\mathcal{T}$ by merging $\mathcal{T}_1$ minus $\boldsymbol{C}_1$ into $\mathcal{T}_2$, making all of $\boldsymbol{C}_1$'s neighbors adjacent to $\boldsymbol{C}_2$ instead.

It remains to show that the resulting structure is a clique tree for $\mathcal{H}$. First, we note that there is no clique in $\mathcal{H}$ that intersects both $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$; hence, any maximal clique in $\mathcal{H}$ is a maximal clique in either $\mathcal{H}_1$ or $\mathcal{H}_2$ (or both in the possible case of $\boldsymbol{S}$), so that all maximal cliques in $\mathcal{H}$ appear in $\mathcal{T}$. Thus, the nodes in $\mathcal{T}$ are precisely the maximal cliques in $\mathcal{H}$. Second, we need to show that any $\boldsymbol{S}_{i,j}$ separates $\boldsymbol{W}_{<(i,j)}$ and $\boldsymbol{W}_{<(j,i)}$. Consider two variables $X \in \boldsymbol{W}_{<(i,j)}$ and $Y \in \boldsymbol{W}_{<(j,i)}$. First, assume that $X, Y \in \mathcal{H}_1$; as all the nodes in $\mathcal{H}_1$ are on the $\mathcal{T}_1$ side of the tree, we also have that $\boldsymbol{S}_{i,j} \subset \mathcal{H}_1$. Any path between two nodes in $\mathcal{H}_1$ that goes through $\boldsymbol{W}_2$ can be shortcut to go only through $\mathcal{H}_1$. Thus, if $\boldsymbol{S}_{i,j}$ separates X, Y in $\mathcal{H}_1$, then it also separates them in $\mathcal{H}$. The same argument applies for $X, Y \in \mathcal{H}_2$. Now, consider $X \in \boldsymbol{W}_1$ and $Y \in \boldsymbol{W}_2$. If $\boldsymbol{S}_{i,j} = \boldsymbol{S}$, the result follows from the fact that $\boldsymbol{S}$ separates $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. Otherwise, assume that $\boldsymbol{S}_{i,j}$ is in $\mathcal{T}_1$, on the path from X to $\boldsymbol{C}_1$. In this case, we have that $\boldsymbol{S}_{i,j}$ separates X from $\boldsymbol{S}$, and $\boldsymbol{S}$ separates $\boldsymbol{S}_{i,j}$ from Y. The conclusion now follows from the transitivity of graph separation.

We have therefore constructed a clique tree for $\mathcal{H}$, proving the inductive claim. ∎

Using this result, we can show that the independencies in an undirected graph $\mathcal{H}$ can be captured perfectly in a Bayesian network if and only if $\mathcal{H}$ is chordal.

Theorem 4.13 *Let $\mathcal{H}$ be a chordal Markov network. Then there is a Bayesian network $\mathcal{G}$ such that $\mathcal{I}(\mathcal{H}) = \mathcal{I}(\mathcal{G})$.*

Proof Let $\mathcal{T}$ be the clique tree for $\mathcal{H}$, whose existence is guaranteed by theorem 4.12. We can select an ordering over the nodes in the Bayesian network as follows. We select an arbitrary

clique $\boldsymbol{C}_1$ to be the root of the clique tree, and then order the cliques $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_k$ using any topological ordering, that is, where cliques closer to the root are ordered first. We now order the nodes in the network in any ordering consistent with the clique ordering: if X_l first appears in $\boldsymbol{C}_i$ and X_m first appears in $\boldsymbol{C}_j$, for $i < j$, then X_l must precede X_m in the ordering. We now construct a Bayesian network using the procedure Build-Minimal-I-Map of algorithm 3.2 applied to the resulting node ordering $X_1, \ldots, X_n$ and to $\mathcal{I}(\mathcal{H})$.

Let $\mathcal{G}$ be the resulting network. We first show that, when X_i is added to the graph, then X_i's parents are precisely $\boldsymbol{U}_i = \mathrm{Nb}_{X_i} \cap \{X_1, \ldots, X_{i-1}\}$, where Nb_{X_i} is the set of neighbors of X_i in $\mathcal{H}$. In other words, we want to show that X_i is independent of $\{X_1, \ldots, X_{i-1}\} - \boldsymbol{U}_i$ given $\boldsymbol{U}_i$. Let $\boldsymbol{C}_k$ be the first clique in the clique ordering to which X_i belongs. Then $\boldsymbol{U}_i \subset \boldsymbol{C}_k$. Let $\boldsymbol{C}_l$ be the parent of $\boldsymbol{C}_k$ in the rooted clique tree. According to our selected ordering, all of the variables in $\boldsymbol{C}_l$ are ordered before any variable in $\boldsymbol{C}_k - \boldsymbol{C}_l$. Thus, $\boldsymbol{S}_{l,k} \subset \{X_1, \ldots, X_{i-1}\}$. Moreover, from our choice of ordering, none of $\{X_1, \ldots, X_{i-1}\} - \boldsymbol{U}_i$ are in any descendants of $\boldsymbol{C}_k$ in the clique tree. Thus, they are all in $\boldsymbol{W}_{<(l,k)}$. From theorem 4.12, it follows that $\boldsymbol{S}_{l,k}$ separates X_i from all of $\{X_1, \ldots, X_{i-1}\} - \boldsymbol{U}_i$, and hence that X_i is independent of all of $\{X_1, \ldots, X_{i-1}\} - \boldsymbol{U}_i$ given $\boldsymbol{U}_i$. It follows that $\mathcal{G}$ and $\mathcal{H}$ have the same set of edges. Moreover, we note that all of $\boldsymbol{U}_i$ are in $\boldsymbol{C}_k$, and hence are connected in $\mathcal{G}$. Therefore, $\mathcal{G}$ is moralized. As $\mathcal{H}$ is the moralized undirected graph of $\mathcal{G}$, the result now follows from proposition 4.9. ■

For example, the graph $\mathcal{G}_\ell$ of figure 4.13b, and its moralized network $\mathcal{H}'_\ell$ encode precisely the same independencies. By contrast, as we discussed, there exists no Bayesian network that encodes precisely the independencies in the nonchordal network $\mathcal{H}_\ell$ of figure 4.13a.

Thus, we have shown that chordal graphs are precisely the intersection between Markov networks and Bayesian networks, in that the independencies in a graph can be represented exactly in both types of models if and only if the graph is chordal.

4.6 Partially Directed Models

So far, we have presented two distinct types of graphical models, based on directed and undirected graphs. We can unify both representations by allowing models that incorporate both directed and undirected dependencies. We begin by describing the notion of *conditional random field*, a Markov network with a directed dependency on some subset of variables. We then present a generalization of this framework to the class of *chain graphs*, an entire network in which undirected components depend on each other in a directed fashion.

4.6.1 Conditional Random Fields

So far, we have described the Markov network representation as encoding a joint distribution over $\mathcal{X}$. The same undirected graph representation and parameterization can also be used to encode a *conditional distribution* $P(\boldsymbol{Y} \mid \boldsymbol{X})$, where $\boldsymbol{Y}$ is a set of *target variables* and $\boldsymbol{X}$ is a (disjoint) set of *observed variables*. We will also see a directed analogue of this concept in section 5.6. In the case of Markov networks, this representation is generally called a *conditional random field* (CRF).

target variable

observed variable

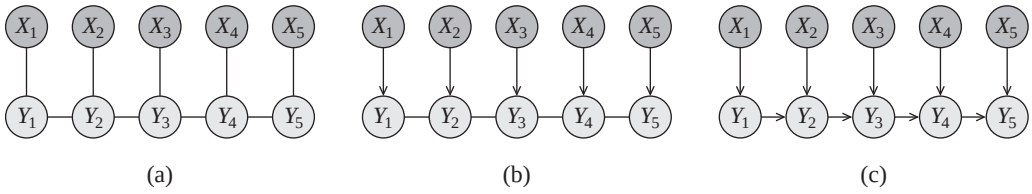


Figure 4.14 Different linear-chain graphical models: (a) a linear-chain-structured conditional random field, where the feature variables are denoted using grayed-out ovals; (b) a partially directed variant; (c) a fully directed, non-equivalent model. The X_i's are assumed to be always observed when the network is used, and hence they are shown as darker gray.

4.6.1.1 CRF Representation and Semantics

More formally, a CRF is an undirected graph whose nodes correspond to $\boldsymbol{Y} \cup \boldsymbol{X}$. At a high level, this graph is parameterized in the same way as an ordinary Markov network, as a set of factors $\phi_1(\boldsymbol{D}_1), \ldots, \phi_m(\boldsymbol{D}_m)$. (As before, these factors can also be encoded more compactly as a log-linear model; for uniformity of presentation, we view the log-linear model as encoding a set of factors.) However, rather than encoding the distribution $P(\boldsymbol{Y}, \boldsymbol{X})$, we view it as representing the conditional distribution $P(\boldsymbol{Y} \mid \boldsymbol{X})$. To have the network structure and parameterization correspond naturally to a conditional distribution, we want to avoid representing a probabilistic model over $\boldsymbol{X}$. We therefore disallow potentials that involve only variables in $\boldsymbol{X}$.

Definition 4.18 conditional random field

A conditional random field *is an undirected graph $\mathcal{H}$ whose nodes correspond to $\boldsymbol{X} \cup \boldsymbol{Y}$; the network is annotated with a set of factors $\phi_1(\boldsymbol{D}_1), \ldots, \phi_m(\boldsymbol{D}_m)$ such that each $\boldsymbol{D}_i \not\subseteq \boldsymbol{X}$. The network encodes a conditional distribution as follows:*

$$\begin{aligned} P(\boldsymbol{Y} \mid \boldsymbol{X}) &= \frac{1}{Z(\boldsymbol{X})} \tilde{P}(\boldsymbol{Y}, \boldsymbol{X}) \\ \tilde{P}(\boldsymbol{Y}, \boldsymbol{X}) &= \prod_{i=1}^{m} \phi_i(\boldsymbol{D}_i) \\ Z(\boldsymbol{X}) &= \sum_{\boldsymbol{Y}} \tilde{P}(\boldsymbol{Y}, \boldsymbol{X}). \end{aligned} \tag{4.11}$$

Two variables in $\mathcal{H}$ are connected by an (undirected) edge whenever they appear together in the scope of some factor. ■

The only difference between equation (4.11) and the (unconditional) Gibbs distribution of definition 4.3 is the different normalization used in the partition function $Z(\boldsymbol{X})$. The definition of a CRF induces a different value for the partition function for every assignment $\boldsymbol{x}$ to $\boldsymbol{X}$. This difference is denoted graphically by having the feature variables grayed out.

Example 4.19

Consider a CRF over $\boldsymbol{Y} = \{Y_1, \ldots, Y_k\}$ and $\boldsymbol{X} = \{X_1, \ldots, X_k\}$, with an edge Y_i—Y_{i+1} ($i = 1, \ldots, k-1$) and an edge Y_i—X_i ($i = 1, \ldots, k$), as shown in figure 4.14a. The distribution

represented by this network has the form:

$$\begin{aligned} P(\boldsymbol{Y} \mid \boldsymbol{X}) &= \frac{1}{Z(\boldsymbol{X})}\tilde{P}(\boldsymbol{Y}, \boldsymbol{X}) \\ \tilde{P}(\boldsymbol{Y}, \boldsymbol{X}) &= \prod_{i=1}^{k-1} \phi(Y_i, Y_{i+1}) \prod_{i=1}^{k} \phi(Y_i, X_i) \\ Z(\boldsymbol{X}) &= \sum_{\boldsymbol{Y}} \tilde{P}(\boldsymbol{Y}, \boldsymbol{X}). \end{aligned}$$

■

Note that, unlike the definition of a conditional Bayesian network, the structure of a CRF may still contain edges between variables in $\boldsymbol{X}$, which arise when two such variables appear together in a factor that also contains a target variable. However, these edges do not encode the structure of any distribution over $\boldsymbol{X}$, since the network explicitly does not encode any such distribution.

☞ **The fact that we avoid encoding the distribution over the variables in $\boldsymbol{X}$ is one of the main strengths of the CRF representation. This flexibility allows us to incorporate into the model a rich set of observed variables whose dependencies may be quite complex or even poorly understood. It also allows us to include continuous variables whose distribution may not have a simple parametric form. This flexibility allows us to use domain knowledge in order to define a rich set of features characterizing our domain, without worrying about modeling their joint distribution.** For example, returning to the vision MRFs of box 4.B, rather than defining a joint distribution over pixel values and their region assignment, we can define a conditional distribution over segment assignments *given* the pixel values. The use of a conditional distribution here allows us to avoid making a parametric assumption over the (continuous) pixel values. Even more important, we can use image-processing routines to define rich features, such as the presence or direction of an image gradient at a pixel. Such features can be highly informative in determining the region assignment of a pixel. However, the definition of such features usually relies on multiple pixels, and defining a correct joint distribution or a set of independence assumptions over these features is far from trivial. The fact that we can condition on these features and avoid this whole issue allows us the flexibility to include them in the model. See box 4.E for another example.

4.6.1.2 Directed and Undirected Dependencies

A CRF defines a conditional distribution of $\boldsymbol{Y}$ on $\boldsymbol{X}$; thus, it can be viewed as a partially directed graph, where we have an undirected component over Y, which has the variables in $\boldsymbol{X}$ as parents.

Example 4.20

naive Markov

Consider a CRF over the binary-valued variables $\boldsymbol{X} = \{X_1, \ldots, X_k\}$ and $\boldsymbol{Y} = \{Y\}$, and a pairwise potential between Y and each X_i; this model is sometimes known as a naive Markov *model, due to its similarity to the naive Bayes model. Assume that the pairwise potentials defined via the following log-linear model*

$$\phi_i(X_i, Y) = \exp\{w_i \mathbf{I}\{X_i = 1, Y = 1\}\}.$$

We also introduce a single-node potential $\phi_0(Y) = \exp\{w_0 \mathbf{I}\{Y = 1\}\}$. Following equation (4.11),

we now have:

$$\begin{aligned}\tilde{P}(Y = 1 \mid x_1, \ldots, x_k) &= \exp\left\{w_0 + \sum_{i=1}^{k} w_i x_i\right\} \\ \tilde{P}(Y = 0 \mid x_1, \ldots, x_k) &= \exp\{0\} = 1.\end{aligned}$$

In this case, we can show (exercise 5.16) that

$$P(Y = 1 \mid x_1, \ldots, x_k) = \text{sigmoid}\left(w_0 + \sum_{i=1}^{k} w_i x_i\right),$$

where

$$\text{sigmoid}(z) = \frac{e^z}{1 + e^z}$$

sigmoid

logistic CPD

is the sigmoid *function. This conditional distribution $P(Y \mid \boldsymbol{X})$ is of great practical interest: It defines a CPD that is not structured as a table, but that is induced by a small set of parameters $w_0, \ldots, w_k$ — parameters whose number is linear, rather than exponential, in the number of parents. This type of CPD, often called a* logistic CPD*, is a natural model for many real-world applications, inasmuch as it naturally aggregates the influence of different parents. We discuss this CPD in greater detail in section 5.4.2 as part of our general presentation of structured CPDs.* ■

The partially directed model for the CRF of example 4.19 is shown in figure 4.14b. We may be tempted to believe that we can construct an equivalent model that is fully directed, such as the one in figure 4.14c. In particular, conditioned on any assignment $\boldsymbol{x}$, the posterior distributions over $\boldsymbol{Y}$ in the two models satisfy the same independence assignments (the ones defined by the chain structure). However, the two models are not equivalent: In the Bayesian network, we have that Y_1 is independent of X_2 if we are not given Y_2. By contrast, in the original CRF, the unnormalized marginal measure of $\boldsymbol{Y}$ depends on the entire parameterization of the chain, and specifically the values of all of the variables in $\boldsymbol{X}$. A sound conditional Bayesian network for this distribution would require edges from all of the variables in $\boldsymbol{X}$ to each of the variables Y_i, thereby losing much of the structure in the distribution. See also box 20.A for further discussion.

Box 4.E — Case Study: CRFs for Text Analysis. *One important use for the CRF framework is in the domain of text analysis. Various models have been proposed for different tasks, including part-of-speech labeling, identifying named entities (people, places, organizations, and so forth), and extracting structured information from the text (for example, extracting from a reference list the publication titles, authors, journals, years, and the like). Most of these models share a similar structure: We have a target variable for each word (or perhaps short phrase) in the document, which encodes the possible labels for that word. Each target variable is connected to a set of feature variables that capture properties relevant to the target distinction. These methods are very popular in text analysis, both because the structure of the networks is a good fit for this domain, and because they produce state-of-the-art results for a broad range of natural-language processing problems.*

As a concrete example, consider the named entity recognition task, as described by Sutton and McCallum (2004, 2007). Entities often span multiple words, and the type of an entity may not be

apparent from individual words; for example, "New York" is a location, but "New York Times" is an organization. The problem of extracting entities from a word sequence of length T can be cast as a graphical model by introducing for each word, $X_t, 1 \leq t \leq T$, a target variable, Y_t, which indicates the entity type of the word. The outcomes of Y_t include B-Person, I-Person, B-Location, I-Location, B-Organization, I-Organization, *and* Other. *In this so-called "BIO notation,"* Other *indicates that the word is not part of an entity, the* B- *outcomes indicate the beginning of a named entity phrase, and the* I- *outcomes indicate the inside or end of the named entity phrase. Having a distinguishing label for the beginning versus inside of an entity phrase allows the model to segment adjacent entities of the same type.*

linear-chain CRF

A common structure for this problem is a linear-chain CRF *often having two factors for each word: one factor $\phi_t^1(Y_t, Y_{t+1})$ to represent the dependency between neighboring target variables, and another factor $\phi_t^2(Y_t, X_1, \ldots, X_T)$ that represents the dependency between a target and its context in the word sequence. Note that the second factor can depend on arbitrary features of the entire input word sequence. We generally do not encode this model using table factors, but using a log-linear model. Thus, the factors are derived from a number of feature functions, such as $f_t(Y_t, X_t) = \mathbf{1}\{Y_t =$ B-Organization$, X_t =$ "Times"$\}$. We note that, just as logistic CPDs are the conditional analog of the naive Bayes classifier (example 4.20), the linear-chain CRF is the conditional analog of the* hidden Markov model *(HMM) that we present in section 6.2.3.1.*

hidden Markov model

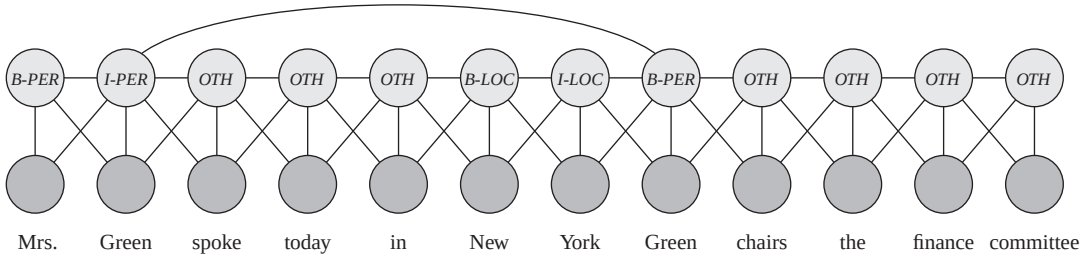
A large number of features of the word X_t and neighboring words are relevant to the named entity decision. These include features of the word itself: is it capitalized; does it appear in a list of common person names; does it appear in an atlas of location names; does it end with the character string "ton"; is it exactly the string "York"; is the following word "Times." Also relevant are aggregate features of the entire word sequence, such as whether it contains more than two sports-related words, which might be an indicator that "New York" is an organization (sports team) rather than a location. In addition, including features that are conjunctions of all these features often increases accuracy. The total number of features can be quite large, often in the hundreds of thousands or more if conjunctions of word pairs are used as features. However, the features are sparse, meaning that most features are zero for most words.

Note that the same feature variable can be connected to multiple target variables, so that Y_t would typically be dependent on the identity of several words in a window around position t. These contextual features are often highly indicative: for example, "Mrs." before a word and "spoke" after a word are both strong indicators that the word is a person. These context words would generally be used as a feature for multiple target variables. Thus, if we were using a simple naive-Bayes-style generative model, where each target variable is a parent of its associated feature, we either would have to deal with the fact that a context word has multiple parents or we would have to duplicate its occurrences (with one copy for each target variable for which it is in the context), and thereby overcount its contribution.

Linear-chain CRFs frequently provide per-token accuracies in the high 90 percent range on many natural data sets. Per-field precision and recall (where the entire phrase category and boundaries must be correct) are more often around 80–95 percent, depending on the data set.

Although the linear-chain model is often effective, additional information can be incorporated into the model by augmenting the graphical structure. For example, often when a word occurs multiple times in the same document, it often has the same label. This knowledge can be incorporated by including factors that connect identical words, resulting in a skip-chain CRF, *as shown in figure 4.E.1a. The first occurrence of the word "Green" has neighboring words that provide strong*

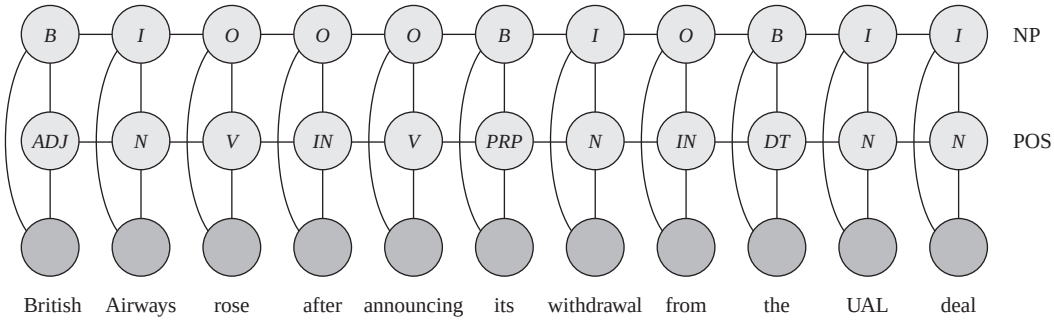
skip-chain CRF



KEY

B-PER	Begin person name	*I-LOC*	Within location name
I-PER	Within person name	*OTH*	Not an entitiy
B-LOC	Begin location name		

(a)



KEY

B	Begin noun phrase	*V*	Verb
I	Within noun phrase	*IN*	Preposition
O	Not a noun phrase	*PRP*	Possesive pronoun
N	Noun	*DT*	Determiner (e.g., a, an, the)
ADJ	Adjective		

(b)

Figure 4.E.1 — Two models for text analysis based on a linear chain CRF Gray nodes indicate $\boldsymbol{X}$ and clear nodes $\boldsymbol{Y}$. The annotations inside the $\boldsymbol{Y}$ are the true labels. (a) A skip chain CRF for named entity recognition, with connections between adjacent words and long-range connections between multiple occurrences of the same word. (b) A pair of coupled linear-chain CRFs that performs joint part-of-speech labeling and noun-phrase segmentation. Here, B indicates the beginning of a noun phrase, I other words in the noun phrase, and O words not in a noun phrase. The labels for the second chain are parts of speech.

evidence that it is a PERSON*'s name; however, the second occurrence is much more ambiguous. By augmenting the original linear-chain CRF with an additional long-range factor that prefers its connected target variables to have the same value, the model is more likely to predict correctly that the second occurrence is also a* PERSON*. This example demonstrates another flexibility of conditional models, which is that the graphical structure over $\boldsymbol{Y}$ can easily depend on the value of the $\boldsymbol{X}$'s.*

CRFs having a wide variety of model structures have been successfully applied to many different tasks. Joint inference of both part-of-speech labels and noun-phrase segmentation has been performed with two connected linear chains (somewhat analogous to a coupled hidden Markov mode*, shown in figure 6.3). This structure is illustrated in figure 4.E.1b.*

coupled HMM

4.6.2 Chain Graph Models ⋆

We now present a more general framework that builds on the CRF representation and can be used to provide a general treatment of the independence assumptions made in these partially directed models. Recall from definition 2.21 that, in a *partially directed acyclic graph* (PDAG), the nodes can be disjointly partitioned into several *chain components*. An edge between two nodes in the same chain component must be undirected, while an edge between two nodes in different chain components must be directed. Thus, PDAGs are also called *chain graphs*.

partially directed acyclic graph

chain component

chain graph

4.6.2.1 Factorization

As in our other graphical representations, the structure of a PDAG $\mathcal{K}$ can be used to define a factorization for a probability distribution over $\mathcal{K}$. Intuitively, the factorization for PDAGs represents the distribution as a product of each of the chain components given its parents. Thus, we call such a representation a *chain graph model*.

chain graph model

Intuitively, each chain component $\boldsymbol{K}_i$ in the chain graph model is associated with a CRF that defines $\mathcal{P}(\boldsymbol{K}_i \mid \mathrm{Pa}_{\boldsymbol{K}_i})$ — the conditional distribution of $\boldsymbol{K}_i$ given its parents in the graph. More precisely, each is defined via a set of factors that involve the variables in $\boldsymbol{K}_i$ and their parents; the distribution $\mathcal{P}(\boldsymbol{K}_i \mid \mathrm{Pa}_{\boldsymbol{K}_i})$ is defined by using the factors associated with $\boldsymbol{K}_i$ to define a CRF whose target variables are $\boldsymbol{K}_i$ and whose observable variables are $\mathrm{Pa}_{\boldsymbol{K}_i}$.

To provide a formal definition, it helps to introduce the concept of a moralized PDAG.

Definition 4.19 moralized graph

Let $\mathcal{K}$ be a PDAG and $\boldsymbol{K}_1, \ldots, \boldsymbol{K}_\ell$ be its chain components. We define $\mathrm{Pa}_{\boldsymbol{K}_i}$ to be the parents of nodes in $\boldsymbol{K}_i$. The moralized graph *of $\mathcal{K}$ is an undirected graph $\mathcal{M}[\mathcal{K}]$ produced by first connecting, using undirected edges, any pair of nodes $X, Y \in \mathrm{Pa}_{\boldsymbol{K}_i}$ for all $i = 1, \ldots, \ell$, and then converting all directed edges into undirected edges.* ■

This definition generalizes our earlier notion of a moralized directed graph. In the case of directed graphs, each node is its own chain component, and hence we are simply adding undirected edges between the parents of each node.

Example 4.21

Figure 4.15 shows a chain graph and its moral graph. We have added the edge between A and B, since they are both parents of the chain component $\{C, D, E\}$, and edges between C, E, and H, because they are parents of the chain component $\{I\}$. Note that we did not add an edge between

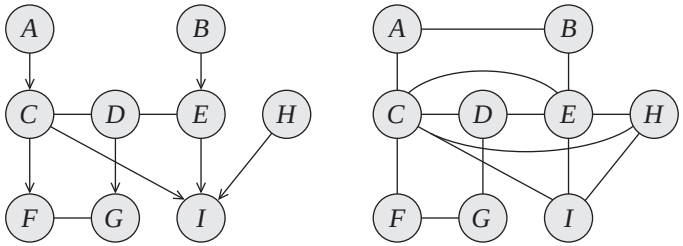


Figure 4.15 A chain graph $\mathcal{K}$ and its moralized version

D and H (even though D and C, E are in the same chain component), since D is not a parent of I. ■

We can now define the factorization of a chain graph:

Definition 4.20 chain graph distribution

Let $\mathcal{K}$ be a PDAG, and $\boldsymbol{K}_1, \ldots, \boldsymbol{K}_\ell$ be its chain components. A chain graph distribution *is defined via a set of factors $\phi_i(\boldsymbol{D}_i)$ $(i = 1, \ldots, m)$, such that each $\boldsymbol{D}_i$ is a complete subgraph in the moralized graph $\mathcal{M}[\mathcal{K}^+[\boldsymbol{D}_i]]$. We associate each factor $\phi_i(\boldsymbol{D}_i)$ with a single chain component $\boldsymbol{K}_j$, such that $\boldsymbol{D}_i \subseteq \boldsymbol{K}_i \cup \mathrm{Pa}_{\boldsymbol{K}_i}$ and define $P(\boldsymbol{K}_i \mid \mathrm{Pa}_{\boldsymbol{K}_i})$ as a CRF with these factors, and with $\boldsymbol{Y}_i = \boldsymbol{K}_i$ and $\boldsymbol{X}_i = \mathrm{Pa}_{\boldsymbol{K}_i}$. We now define*

$$P(\mathcal{X}) = \prod_{i=1}^{\ell} P(\boldsymbol{K}_i \mid \mathrm{Pa}_{\boldsymbol{K}_i}).$$

■

We say that a distribution P factorizes over $\mathcal{K}$ if it can be represented as a chain graph distribution over $\mathcal{K}$.

Example 4.22

In the chain graph model defined by the graph of figure 4.15, we require that the conditional distribution $P(C, D, E \mid A, B)$ factorize according to the graph of figure 4.16a. Specifically, we would have to define the conditional probability as a normalized product of factors:

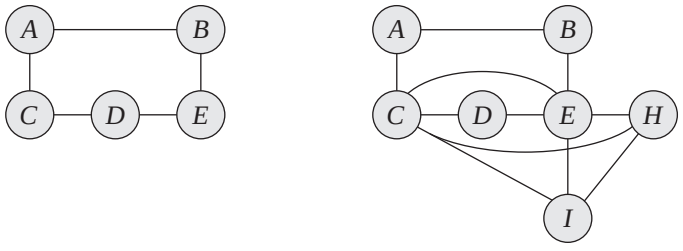
$$\frac{1}{Z(A, B)} \phi_1(A, C)\phi_2(B, E)\phi_3(C, D)\phi_4(D, E).$$

A similar factorization applies to $P(F, G \mid C, D)$. ■

4.6.2.2 Independencies in Chain Graphs

As for undirected graphs, there are three distinct interpretations for the independence properties induced by a PDAG. Recall that in a PDAG, we have both the notion of parents of X (variables Y such that $Y \to X$ is in the graph) and neighbors of X (variables Y such that Y—X is in the graph). Recall that the union of these two sets is the *boundary* of X, denoted $\mathrm{Boundary}_X$. Also recall, from definition 2.15, that the descendants of X are those nodes Y that can be reached using any directed path, where a directed path can involve both directed and undirected edges but must contain at least one edge directed from X to Y, and no edges directed from Y to

boundary



(a) (b)

Figure 4.16 Example for definition of c-separation in a chain graph. (a) The Markov network $\mathcal{M}[\mathcal{K}^+[C, D, E]]$. (b) The Markov network $\mathcal{M}[\mathcal{K}^+[C, D, E, I]]$.

X. Thus, in the case of PDAGs, it follows that if Y is a descendant of X, then Y must be in a "lower" chain component.

Definition 4.21 pairwise independencies

For a PDAG $\mathcal{K}$, we define the pairwise independencies *associated with $\mathcal{K}$ to be:*

$$\mathcal{I}_p(\mathcal{K}) = \{(X \perp Y \mid (\text{NonDescendants}_X - \{X, Y\})) : X, Y \text{ non-adjacent}, Y \in \text{NonDescendants}_X\}.$$ ■

This definition generalizes the pairwise independencies for undirected graphs: in an undirected graph, nodes have no descendants, so $\text{NonDescendants}_X = \mathcal{X}$. Similarly, it is not too hard to show that these independencies also hold in a directed graph.

Definition 4.22 local independencies

For a PDAG $\mathcal{K}$, we define the local independencies *associated with $\mathcal{K}$ to be:*

$$\mathcal{I}_\ell(\mathcal{K}) = \{(X \perp \text{NonDescendants}_X - \text{Boundary}_X \mid \text{Boundary}_X) : X \in \mathcal{X}\}.$$ ■

This definition generalizes the definition of local independencies for both directed and undirected graphs. For directed graphs, NonDescendants_X is precisely the set of nondescendants, whereas Boundary_X is the set of parents. For undirected graphs, NonDescendants_X is $\mathcal{X}$, whereas $\text{Boundary}_X = \text{Nb}_X$.

We define the global independencies in a PDAG using the definition of moral graph. Our definition follows the lines of proposition 4.10.

Definition 4.23 c-separation

Let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z} \subset \mathcal{X}$ be three disjoint sets, and let $\boldsymbol{U} = \boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$. We say that $\boldsymbol{X}$ is c-separated *from $\boldsymbol{Y}$ given $\boldsymbol{Z}$ if $\boldsymbol{X}$ is separated from $\boldsymbol{Y}$ given $\boldsymbol{Z}$ in the undirected graph $\mathcal{M}[\mathcal{K}^+[\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}]]$.* ■

Example 4.23

Consider again the PDAG of figure 4.15. Then C is c-separated from E given D, A, because C and E are separated given D, A in the undirected graph $\mathcal{M}[\mathcal{K}^+[\{C, D, E\}]]$, shown in figure 4.16a. However, C is not c-separated from E given only D, since there is a path between C and E via A, B. On the other hand, C is not separated from E given D, A, I. The graph $\mathcal{M}[\mathcal{K}^+[\{C, D, E, I\}]]$ is shown in figure 4.16b. As we can see, the introduction of I into the set $\boldsymbol{U}$ causes us to introduce a direct edge between C and E in order to moralize the graph. Thus, we cannot block the path between C and E using D, A, I. ■

This notion of c-separation clearly generalizes the notion of separation in undirected graphs, since the ancestors of a set $\boldsymbol{U}$ in an undirected graph are simply the entire set of nodes $\mathcal{X}$. It also generalizes the notion of d-separation in directed graphs, using the equivalent definition provided in proposition 4.10. Using the definition of c-separation, we can finally define the notion of global Markov independencies:

Definition 4.24 global independencies

Let $\mathcal{K}$ be a PDAG. We define the global independencies *associated with $\mathcal{K}$ to be:*

$$\mathcal{I}(\mathcal{K}) = \{(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \ : \ \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z} \subset \mathcal{X}, \boldsymbol{X} \text{ is c-separated from } \boldsymbol{Y} \text{ given } \boldsymbol{Z}\}.$$ ■

As in the case of undirected models, these three criteria for independence are not equivalent for nonpositive distributions. The inclusions are the same: the global independencies imply the local independencies, which in turn imply the pairwise independencies. Because undirected models are a subclass of PDAGs, the same counterexamples used in section 4.3.3 show that the inclusions are strict for nonpositive distributions. For positive distributions, we again have that the three definitions are equivalent.

We note that, as in the case of Bayesian networks, the parents of a chain component are always fully connected in $\mathcal{M}[\mathcal{K}[\boldsymbol{K}_i \cup \mathrm{Pa}_{\boldsymbol{K}_i}]]$. Thus, while the structure over the parents helps factorize the distribution over the chain components containing the parents, it does not give rise to independence assertions in the conditional distribution over the child chain component. Importantly, however, it does give rise to structure in the form of the parameterization of $P(\boldsymbol{K}_i \mid \mathrm{Pa}_{\boldsymbol{K}_i})$, as we saw in example 4.20.

As in the case of directed and undirected models, we have an equivalence between the requirement of factorization of a distribution and the requirement that it satisfy the independencies associated with the graph. Not surprisingly, since PDAGs generalize undirected graphs, this equivalence only holds for positive distributions:

Theorem 4.14

A positive distribution P factorizes over a PDAG $\mathcal{K}$ if and only if $P \models \mathcal{I}(\mathcal{K})$.

We omit the proof.

4.7 Summary and Discussion

In this chapter, we introduced *Markov networks*, an alternative graphical modeling language for probability distributions, based on undirected graphs.

We showed that Markov networks, like Bayesian networks, can be viewed as defining a set of independence assumptions determined by the graph structure. In the case of undirected models, there are several possible definitions for the independence assumptions induced by the graph, which are equivalent for positive distributions. As in the case of Bayesian network, we also showed that the graph can be viewed as a data structure for specifying a probability distribution in a factored form. The factorization is defined as a product of factors (general nonnegative functions) over cliques in the graph. We showed that, for positive distributions, the two characterizations of undirected graphs — as specifying a set of independence assumptions and as defining a factorization — are equivalent.

Markov networks also provide useful insight on Bayesian networks. In particular, we showed how a Bayesian network can be viewed as a Gibbs distribution. More importantly, the unnormalized measure we obtain by introducing evidence into a Bayesian network is also a Gibbs

distribution, whose partition function is the probability of the evidence. This observation will play a critical role in providing a unified view of inference in graphical models.

We investigated the relationship between Bayesian networks and Markov networks and showed that the two represent different families of independence assumptions. The difference in these independence assumptions is a key factor in deciding which of the two representations to use in encoding a particular domain. There are domains where interactions have a natural directionality, often derived from causal intuitions. **In this case, the independencies derived from the network structure directly reflect patterns such as intercausal reasoning. Markov networks represent only monotonic independence patterns: observing a variable can only serve to remove dependencies, not to activate them.** Of course, we can encode a distribution with "causal" connections as a Gibbs distribution, and it will exhibit the same nonmonotonic independencies. However, these independencies will not be manifest in the network structure.

In other domains, the interactions are more symmetrical, and attempts to force a directionality give rise to models that are unintuitive and that often are incapable of capturing the independencies in the domain (see, for example, section 6.6). As a consequence, the use of undirected models has increased steadily, most notably in fields such as computer vision and natural language processing, where the acyclicity requirements of directed graphical models are often at odds with the nature of the model. The flexibility of the undirected model also allows the distribution to be decomposed into factors over multiple overlapping "features" without having to worry about defining a single normalized generating distribution for each variable. Conversely, **this very flexibility and the associated lack of clear semantics for the model parameters often make it difficult to elicit models from experts. Therefore, many recent applications use learning techniques to estimate parameters from data, avoiding the need to provide a precise semantic meaning for each of them.**

Finally, the question of which class of models better encodes the properties of the distribution is only one factor in the selection of a representation. There are other important distinctions between these two classes of models, especially when it comes to learning from data. We return to these topics later in the book (see, for example, box 20.A).

4.8 Relevant Literature

contingency table

The representation of a probability distribution as an undirected graph has its roots in the *contingency table* representation that is a staple in statistical modeling. The idea of representing probabilistic interactions in this representation dates back at least as early as the work of Bartlett (1935). This line of work is reviewed in detail by Whittaker (1990) and Lauritzen (1996), and we refer the reader to those sources and the references therein.

A parallel line of work involved the development of the Markov network (or Markov random field) representation. Here, the starting point was a graph object rather than a distribution object (such as a contingency table). Isham (1981) surveys some of the early work along these lines.

The connection between the undirected graph representation and the Gibbs factorization of the distribution was first made in the unpublished work of Hammersley and Clifford (1971). As a consequence, they also showed the equivalence of the different types of (local, pairwise, and global) independence properties for undirected graphs in the case of positive distributions.

Lauritzen (1982) made the connection between MRFs and contingency tables, and proved

some of the key results regarding the independence properties arising form the undirected representation. The line of work analyzing independence properties was then significantly extended by Pearl and Paz (1987). The history of these developments and other key references are presented by Pearl (1988) and Lauritzen (1996). The independence properties of chain graphs were studied in detail by Frydenberg (1990); see also Lauritzen (1996). Studený and Bouckaert (1998) also provide an alternative definition of the independence properties in chain graphs, one that is equivalent to c-separation but more directly analogous to the definition of d-separation in directed graphs.

Factor graphs were presented by Kschischang et al. (2001a) and extended by Frey (2003) to encompass both Bayesian networks and Markov networks. The framework of conditional random fields (CRFs) was first proposed by Lafferty, McCallum, and Pereira (2001). They have subsequently been used in a broad range of applications in natural language processing, computer vision, and many more. Skip-chain CRFs were introduced by Sutton and McCallum (2004), and factorial CRFs by Sutton et al. (2007). Sutton and McCallum (2007) also provide an overview of this framework and some of its applications.

Ising models were first proposed by Ising (1925). The literature on this topic is too vast to mention; we refer the reader to any textbook in the area of statistical physics. The connection between Markov networks and these models in statistical physics is the origin of some of the terminology associated with these models, such as partition function or energy. In fact, many of the recent developments in inference for these models arise from approximations that were first proposed in the statistical physics community. Boltzmann machines were first proposed by Hinton and Sejnowski (1983).

Computer vision is another application domain that has motivated much of the work in undirected graphical models. The applications of MRFs to computer vision are too numerous to list; they span problems in low-level vision (such as image denoising, stereo reconstruction, or image segmentation) and in high-level vision (such as object recognition). Li (2001) provides a detailed description of some early applications; Szeliski et al. (2008) describe some applications that are viewed as standard benchmark problems for MRFs in the computer vision field.

4.9 Exercises

Exercise 4.1

Complete the analysis of example 4.4, showing that the distribution P defined in the example does not factorize over $\mathcal{H}$. (Hint: Use a proof by contradiction.)

Exercise 4.2

In this exercise, you will prove that the modified energy functions $\epsilon_1'(A, B)$ and $\epsilon_2'(B, C)$ of figure 4.10 result in precisely the same distribution as our original energy functions. More generally, for any constants λ^1 and λ^0, we can redefine

$$\begin{aligned} \epsilon_1'(a, b^i) &:= \epsilon_1(a, b^i) + \lambda^i \\ \epsilon_2'(b^i, c) &:= \epsilon_2(b^i, c) - \lambda^i \end{aligned}$$

Show that the resulting energy function is equivalent.

Exercise 4.3⋆

Provide an example a class of Markov networks $\mathcal{H}_n$ over n such that the size of the largest clique in $\mathcal{H}_n$ is constant, yet *any* Bayesian network I-map for $\mathcal{H}_n$ is exponentially large in n.

Exercise 4.4⋆

Prove theorem 4.7 for the case where $\mathcal{H}$ consists of a single clique.

Exercise 4.5

Complete the proof of theorem 4.3, by showing that U_1 and U_k are dependent given $\boldsymbol{Z}$ in the distribution P defined by the product of potentials described in the proof.

Exercise 4.6⋆

Consider a factor graph $\mathcal{F}$, as in definition 4.13. Define the minimal Markov network $\mathcal{H}$ that is guaranteed to be an I-map for any distribution defined over $\mathcal{F}$. Prove that $\mathcal{H}$ is a sound and complete representation of the independencies in $\mathcal{F}$:

a. If $sep_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$ holds, then $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ holds for all distributions over $\mathcal{F}$.
b. If $sep_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$ does not hold, then there is some distribution P that factorizes over $\mathcal{F}$ such that $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ does not hold in P.

Exercise 4.7⋆

The canonical parameterization in the Hammersley-Clifford theorem is stated in terms of the maximal cliques in a Markov network. In this exercise, you will show that it also captures the finer-grained representation of factor graphs. Specifically, let P be a distribution that factorizes over a factor graph $\mathcal{F}$, as in definition 4.13. Show that the canonical parameterization of P also factorizes over $\mathcal{F}$.

Exercise 4.8

Prove proposition 4.3. More precisely, let P satisfy $\mathcal{I}_\ell(\mathcal{H})$, and assume that X and Y are two nodes in $\mathcal{H}$ that are not connected directly by an edge. Prove that P satisfies $(X \perp Y \mid \mathcal{X} - \{X, Y\})$.

Exercise 4.9⋆

Complete the proof of theorem 4.4. Assume that equation (4.1) holds for all disjoint sets $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$, with $|\boldsymbol{Z}| \geq k$. Prove that equation (4.1) also holds for any disjoint $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ such that $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z} \neq \mathcal{X}$ and $|\boldsymbol{Z}| = k - 1$.

Exercise 4.10

We define the following properties for a set of independencies:

strong union
- **Strong Union**:

$$(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \Longrightarrow (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{W}). \tag{4.12}$$

  In other words, additional evidence $\boldsymbol{W}$ cannot induce dependence.

transitivity
- **Transitivity**: For all disjoint sets $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ and all variables A:

$$\neg(\boldsymbol{X} \perp A \mid \boldsymbol{Z}) \& \neg(A \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \Longrightarrow \neg(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}). \tag{4.13}$$

  Intuitively, this statement asserts that if $\boldsymbol{X}$ and $\boldsymbol{Y}$ are both correlated with some A (given $\boldsymbol{Z}$), then they are also correlated with each other (given $\boldsymbol{Z}$). We can also write the contrapositive of this statement, which is less obvious but easier to read. For all $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, A$:

$$(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) \longrightarrow (\boldsymbol{X} \perp A \mid \boldsymbol{Z}) \vee (A \perp \boldsymbol{Y} \mid \boldsymbol{Z}).$$

Prove that if $\mathcal{I} = \mathcal{I}(\mathcal{H})$ for some Markov network $\mathcal{H}$, then $\mathcal{I}$ satisfies strong union and transitivity.

Exercise 4.11⋆

In this exercise you will prove theorem 4.6. Consider some specific node X, and let $\mathcal{U}$ be the set of all subsets $\boldsymbol{U}$ satisfying definition 4.12. Define $\boldsymbol{U}^*$ to be the intersection of all $\boldsymbol{U} \in \mathcal{U}$.

a. Prove that $\boldsymbol{U}^* \in \mathcal{U}$. Conclude that $\mathrm{MB}_P(X) = \boldsymbol{U}^*$.

b. Prove that if $P \models (X \perp Y \mid \mathcal{X} - \{X, Y\})$, then $Y \notin \mathrm{MB}_P(X)$.
c. Prove that if $Y \notin \mathrm{MB}_P(X)$, then $P \models (X \perp Y \mid \mathcal{X} - \{X, Y\})$.
d. Conclude that $\mathrm{MB}_P(X)$ is precisely the set of neighbors of X in the graph defined in theorem 4.5, showing that the construction of theorem 4.6 also produces a minimal I-map.

Exercise 4.12

Show that a Boltzmann machine distribution (with variables taking values in $\{0, 1\}$) can be rewritten as an Ising model, where we use the value space $\{-1, +1\}$ (mapping 0 to -1).

Exercise 4.13

Show that we can represent any Gibbs distribution as a log-linear model, as defined in definition 4.15.

Exercise 4.14

Complete the proof of proposition 4.8. In particular, show the following:

a. For any variable X, let $\boldsymbol{W} = \mathcal{X} - \{X\} - \mathrm{MB}_{\mathcal{G}}(X)$. Then $d\text{-}sep_{\mathcal{G}}(X; \boldsymbol{W} \mid \mathrm{MB}_{\mathcal{G}}(X))$.
b. The set $\mathrm{MB}_{\mathcal{G}}(X)$ is the minimal set for which this property holds.

Exercise 4.15

Prove proposition 4.10.

Exercise 4.16⋆

Provide an example of a class of Markov networks $\mathcal{H}_n$ over n nodes for arbitrarily large n (not necessarily for every n), where the size of the largest clique is a constant independent of n, yet the size of the largest clique in any chordal graph $\mathcal{H}_n^C$ that contains $\mathcal{H}_n$ is exponential in n. Explain why the size of the largest clique is necessarily exponential in n for all $\mathcal{H}_n^C$.

Exercise 4.17⋆

In this exercise, you will prove that the chordality requirement for graphs is equivalent to two other conditions of independent interest.

Definition 4.25 Markov network decomposition

Let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ be disjoint sets such that $\mathcal{X} = \boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$ and $\boldsymbol{X}, \boldsymbol{Y} \neq \emptyset$. We say that $(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{Y})$ is a decomposition *of a Markov network $\mathcal{H}$ if $\boldsymbol{Z}$ separates $\boldsymbol{X}$ from $\boldsymbol{Y}$ and $\boldsymbol{Z}$ is a complete subgraph in $\mathcal{H}$.* ■

Definition 4.26

We say that a graph $\mathcal{H}$ is decomposable *if there is a decomposition $(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{Y})$ of $\mathcal{H}$, such that the graphs induced by $\boldsymbol{X} \cup \boldsymbol{Z}$ and $\boldsymbol{Y} \cup \boldsymbol{Z}$ are also decomposable.* ■

Show that, for any undirected graph $\mathcal{H}$, the following conditions are equivalent:

a. $\mathcal{H}$ is decomposable;
b. $\mathcal{H}$ is chordal;
c. for every X, Y, every minimal set $\boldsymbol{Z}$ that separates X and Y is complete.

The proof of equivalence proceeds by induction on the number of vertices in $\mathcal{X}$. Assume that the three conditions are equivalent for all graphs with $|\mathcal{X}| \leq n$, and consider a graph $\mathcal{H}$ with $|\mathcal{X}| = n + 1$.

a. Prove that if $\mathcal{H}$ is decomposable, it is chordal.
b. Prove that if $\mathcal{H}$ is chordal, then for any X, Y, and any minimal set $\boldsymbol{Z}$ that separates X and Y, $\boldsymbol{Z}$ is complete.
c. Prove that for any X, Y, any minimal set $\boldsymbol{Z}$ that separates X and Y is complete, then $\mathcal{H}$ is decomposable.

Exercise 4.18

Let $\mathcal{G}$ be a Bayesian network structure and $\mathcal{H}$ a Markov network structure over $\mathcal{X}$ such that the skeleton of $\mathcal{G}$ is precisely $\mathcal{H}$. Prove that if $\mathcal{G}$ has no immoralities, then $\mathcal{I}(\mathcal{G}) = \mathcal{I}(\mathcal{H})$.

Exercise 4.19

Consider the PDAG of figure 4.15. Write down all c-separation statements that are valid given $\{G\}$; write down all valid statements given $\{G, D\}$; write down all statements that are valid given $\{C, D, E\}$.

5 *Local Probabilistic Models*

In chapter 3 and chapter 4, we discussed the representation of global properties of independence by graphs. These properties of independence allowed us to factorize a high-dimensional joint distribution into a product of lower-dimensional CPDs or factors. So far, we have mostly ignored the representation of these factors. In this chapter, we examine CPDs in more detail. We describe a range of representations and consider their implications in terms of additional regularities we can exploit. We have chosen to phrase our discussion in terms of CPDs, since they are more constrained than factors (because of the local normalization constraints). However, many of the representations we discuss in the context of CPDs can also be applied to factors.

5.1 Tabular CPDs

When dealing with spaces composed solely of discrete-valued random variables, we can always resort to a *tabular* representation of CPDs, where we encode $P(X \mid \mathrm{Pa}_X)$ as a table that contains an entry for each joint assignment to X and Pa_X. For this table to be a proper CPD, we require that all the values are nonnegative, and that, for each value pa_X, we have

$$\sum_{x \in Val(X)} P(x \mid \mathrm{pa}_X) = 1. \tag{5.1}$$

It is clear that this representation is as general as possible. We can represent every possible discrete CPD using such a table. As we will also see, table-CPDs can be used in a natural way in inference algorithms that we discuss in chapter 9. These advantages often lead to the perception that *table-CPDs*, also known as *conditional probability tables* (CPTs), are an inherent part of the Bayesian network representation.

table-CPD

However, the tabular representation also has several significant disadvantages. First, it is clear that if we consider random variables with infinite domains (for example, random variables with continuous values), we cannot store each possible conditional probability in a table. But even in the discrete setting, we encounter difficulties. The number of parameters needed to describe a table-CPD is the number of joint assignments to X and Pa_X, that is, $|Val(\mathrm{Pa}_X)| \cdot |Val(X)|$.[1] This number grows exponentially in the number of parents. Thus, for example, if we have 5 binary parents of a binary variable X, we need specify $2^5 = 32$ values; if we have 10 parents, we need to specify $2^{10} = 1,024$ values.

1. We can save some space by storing only independent parameters, but this saving is not significant.

Clearly, the tabular representation rapidly becomes large and unwieldy as the number of parents grows. This problem is a serious one in many settings. Consider a medical domain where a symptom, *Fever*, depends on 10 diseases. It would be quite tiresome to ask our expert 1,024 questions of the format: "What is the probability of high fever when the patient has disease A, does not have disease B, ...?" Clearly, our expert will lose patience with us at some point!

This example illustrates another problem with the tabular representation: it ignores *structure* within the CPD. If the CPD is such that there are no similarity between the various cases, that is, each combination of disease has drastically different probability of high fever, then the expert might be more patient. However, in this example, like many others, there is some regularity in the parameters for different values of the parents of X. For example, it might be the case that, if the patient suffers from disease A, then she is certain to have high fever and thus $P(X \mid \mathrm{pa}_X)$ is the same for all values pa_X in which A is true. Indeed, many of the representations we consider in this chapter attempt to describe such regularities explicitly and to exploit them in order to reduce the number of parameters needed to specify a CPD.

The key insight that allows us to avoid these problems is the following observation: **A CPD needs to specify a conditional probability $P(x \mid \mathrm{pa}_X)$ for every assignment of values pa_X and x, but it does not have to do so by listing each such value explicitly. We should view CPDs not as tables listing all of the conditional probabilities, but rather as functions that given pa_x and x, return the conditional probability $P(x \mid \mathrm{pa}_X)$.** This implicit representation suffices in order to specify a well-defined joint distribution as a BN. In the remainder of the chapter, we will explore some of the possible representations of such functions.

5.2 Deterministic CPDs

5.2.1 Representation

deterministic CPD

Perhaps the simplest type of nontabular CPD arises when a variable X is a *deterministic* function of its parents Pa_X. That is, there is a function $f : Val(\mathrm{Pa}_X) \mapsto Val(X)$, such that

$$P(x \mid \mathrm{pa}_X) = \begin{cases} 1 & x = f(\mathrm{pa}_X) \\ 0 & \text{otherwise.} \end{cases}$$

For example, in the case of binary-valued variables, X might be the "or" of its parents. In a continuous domain, we might want to assert in $P(X \mid Y, Z)$ that X is equal to $Y + Z$.

Of course, the extent to which this representation is more compact than a table (that is, takes less space in the computer) depends on the expressive power that our BN modeling language offers us for specifying deterministic functions. For example, some languages might allow a vocabulary that includes only logical OR and AND of the parents, so that all other functions must be specified explicitly as a table.[2] In a domain with continuous variables, a language might choose to allow only linear dependencies of the form $X = 2Y + -3Z + 1$, and not arbitrary functions such as $X = \sin(y + e^z)$.

Deterministic relations are useful in modeling many domains. In some cases, they occur naturally. Most obviously, when modeling constructed artifacts such as machines or electronic circuits, deterministic dependencies are often part of the device specification. For example, the

2. Other logical functions, however, can be described by introducing intermediate nodes and composing ORs and ANDs.

behavior of an OR gate in an electronic circuit (in the case of no faults) is that the gate output is a deterministic OR of the gate inputs. However, we can also find deterministic dependencies in "natural" domains.

Example 5.1

Recall that the genotype of a person is determined by two copies of each gene, called alleles. *Each allele can take on one of several values corresponding to different genetic tendencies. The person's phenotype is often a deterministic function of these values. For example, the gene responsible for determining blood type has three values: a, b, and o. Letting G_1 and G_2 be variables representing the two alleles, and T the variable representing the phenotypical blood type, then we have that:*

$$T = \begin{cases} ab & \text{if } G_1 \text{ or } G_2 \text{ is } a \text{ and the other is } b \\ a & \text{if at least one of } G_1 \text{ or } G_2 \text{ is equal to } a \text{ and the other is either } a \text{ or } o \\ b & \text{if at least one of } G_1 \text{ or } G_2 \text{ is equal to } b \text{ and the other is either } b \text{ or } o \\ o & \text{if } G_1 = o \text{ and } G_2 = o \end{cases}$$

■

Deterministic variables can also help simplify the dependencies in a complex model.

Example 5.2

When modeling a car, we might have four variables $T_1, \ldots, T_4$, each corresponding to a flat in one of the four tires. When one or more of these tires is flat, there are several effects; for example, the steering may be affected, the ride can be rougher, and so forth. Naively, we can make all of the T_i's parents of all of the affected variables — Steering, Ride, and so on. However, it can significantly simplify the model to introduce a new variable Flat-Tire, which is the deterministic OR of $T_1, \ldots, T_4$. We can then replace a complex dependency of Steering and Ride on $T_1, \ldots, T_4$ with a dependency on a single parent Flat-Tire, significantly reducing their indegree. If these variables have other parents, the savings can be considerable. ■

5.2.2 Independencies

Aside from a more compact representation, we get an additional advantage from making the structure explicit. Recall that conditional independence is a numerical property — it is defined using equality of probabilities. However, the graphical structure in a BN makes certain properties of a distribution explicit, allowing us to deduce that some independencies hold without looking at the numbers. By making structure explicit in the CPD, we can do even more of the same.

Example 5.3

Consider the simple network structure in figure 5.1. If C is a deterministic function of A and B, what new conditional independencies do we have? Suppose that we are given the values of A and B. Then, since C is deterministic, we also know the value of C. As a consequence, we have that D and E are independent. Thus, we conclude that $(D \perp E \mid A, B)$ holds in the distribution. Note that, had C not been a deterministic function of A and B, this independence would not necessarily hold. Indeed, d-separation would not deduce that D and E are independent given A and B. ■

Can we augment the d-separation procedure to discover independencies in cases such as this? Consider an independence assertion $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$; in our example, we are interested in the case where $\boldsymbol{Z} = \{A, B\}$. The variable C is not in $\boldsymbol{Z}$ and is therefore not considered observed.

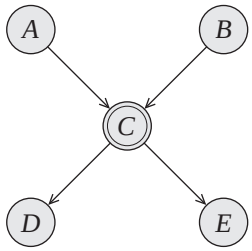


Figure 5.1 Example of a network with a deterministic CPD. The double-line notation represents the fact that C is a deterministic function of A and B.

Algorithm 5.1 Computing d-separation in the presence of deterministic CPDs

```
Procedure DET-SEP(
    Graph, // network structure
    D, // set of deterministic variables
    X, Y, Z // query
)
    Let Z+ ← Z
    While there is an X_i such that
          (1) X_i ∈ D // X_i has a deterministic CPD
          (2) Pa_{X_i} ⊆ Z+
       Z+ ← Z+ ∪ {X_i}
    return d-sep_G(X; Y | Z+)
```

But when A and B are observed, then the value of C is also known with certainty, so we can consider it as part of our observed set $\boldsymbol{Z}$. In our example, this simple modification would suffice for inferring that D and E are independent given A and B.

In other examples, however, we might need to continue this process. For example, if we had another variable F that was a deterministic function of C, then F is also de facto observed when C is observed, and hence when A and B are observed. Thus, F should also be introduced into $\boldsymbol{Z}$. Thus, we have to extend $\boldsymbol{Z}$ iteratively to contain all the variables that are determined by it. This discussion suggests the simple procedure shown in algorithm 5.1.

deterministic separation

This algorithm provides a procedural definition for *deterministic separation* of $\boldsymbol{X}$ from $\boldsymbol{Y}$ given $\boldsymbol{Z}$. This definition is sound, in the same sense that d-separation is sound.

Theorem 5.1

Let $\mathcal{G}$ be a network structure, and let $\boldsymbol{D}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ be sets of variables. If $\boldsymbol{X}$ is deterministically separated from $\boldsymbol{Y}$ given $\boldsymbol{Z}$ (as defined by DET-SEP*$(\mathcal{G}, \boldsymbol{D}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$), then for all distributions P such that $P \models \mathcal{I}_\ell(\mathcal{G})$ and where, for each $X \in \boldsymbol{D}$, $P(X \mid \mathrm{Pa}_X)$ is a deterministic CPD, we have that $P \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$.*

The proof is straightforward and is left as an exercise (exercise 5.1).

Does this procedure capture all of the independencies implied by the deterministic functions? As with d-separation, the answer must be qualified: Given only the graph structure and the set

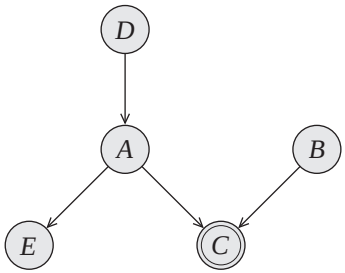


Figure 5.2 A slightly more complex example with deterministic CPDs

of deterministic CPDs, we cannot find additional independencies.

Theorem 5.2 *Let $\mathcal{G}$ be a network structure, and let $\boldsymbol{D}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ be sets of variables. If* DET-SEP$(\mathcal{G}, \boldsymbol{D}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ *returns false, then there is a distribution P such that $P \models \mathcal{I}_\ell(\mathcal{G})$ and where, for each $X \in \boldsymbol{D}$, $P(X \mid \mathrm{Pa}_X)$ is deterministic CPD, but we have that $P \not\models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$.*

Of course, the DET-SEP procedure detects independencies that are derived purely from the fact that a variable is a deterministic function of its parents. However, particular deterministic functions can imply additional independencies.

Example 5.4 *Consider the network of figure 5.2, where C is the exclusive or of A and B. What additional independencies do we have here? In the case of XOR (although not for all other deterministic functions), the values of C and B fully determine that of A. Therefore, we have that $(D \perp E \mid B, C)$ holds in the distribution.* ■

Specific deterministic functions can also induce other independencies, ones that are more refined than the variable-level independencies discussed in chapter 3.

Example 5.5 *Consider the Bayesian network of figure 5.1, but where we also know that the deterministic function at C is an OR. Assume we are given the evidence $A = a^1$. Because C is an OR of its parents, we immediately know that $C = c^1$, regardless of the value of B. Thus, we can conclude that B and D are now independent: In other words, we have that*

$$P(D \mid B, a^1) = P(D \mid a^1).$$

On the other hand, if we are given $A = a^0$, the value of C is not determined, and it does depend on the value of B. Hence, the corresponding statement conditioned on a^0 is false. ■

Thus, deterministic variables can induce a form of independence that is different from the standard notion on which we have focused so far. Up to now, we have restricted attention to independence properties of the form $(X \perp Y \mid Z)$, which represent the assumption that $P(\boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{Z}) = P(\boldsymbol{X} \mid \boldsymbol{Z})$ for all values of $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$. Deterministic functions can imply a type of independence that only holds for *particular* values of some variables.

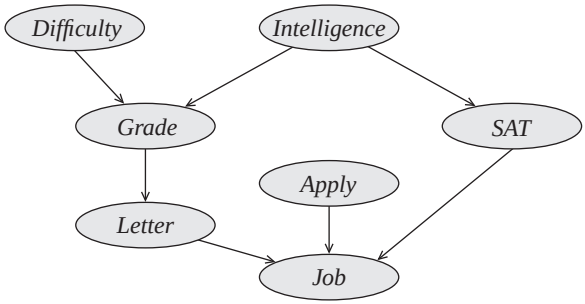


Figure 5.3 The Student **example augmented with a *Job* variable**

Definition 5.1 context-specific independence

Let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ be pairwise disjoint sets of variables, let $\boldsymbol{C}$ be a set of variables (that might overlap with $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$), and let $\boldsymbol{c} \in Val(\boldsymbol{C})$. We say that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are contextually independent *given $\boldsymbol{Z}$ and the* context $\boldsymbol{c}$ *denoted $(\boldsymbol{X} \perp_c \boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{c})$, if*

$$P(\boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{c}) = P(\boldsymbol{X} \mid \boldsymbol{Z}, \boldsymbol{c}) \text{ whenever } P(\boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{c}) > 0.$$ ■

Independence statements of this form are called *context-specific independencies* (CSI). They arise in many forms in the context of deterministic dependencies.

Example 5.6

As we saw in example 5.5, we can have that some value of one parent A can be enough to determine the value of the child C. Thus, we have that $(C \perp_c B \mid a^1)$, and hence also that $(D \perp_c B \mid a^1)$. We can make additional conclusions if we use properties of the OR function. For example, if we know that $C = c^0$, we can conclude that both $A = a^0$ and $B = b^0$. Thus, in particular, we can conclude both that $(A \perp_c B \mid c^0)$ and that $(D \perp_c E \mid c^0)$. Similarly, if we know that $C = c^1$ and $B = b^0$, we can conclude that $A = a^1$, and hence we have that $(D \perp_c E \mid b^0, c^1)$. ■

☞ It is important to note that **context-specific independencies can also arise when we have tabular CPDs. However, in the case of tabular CPDs, the independencies would only become apparent if we examine the network parameters. By making the structure of the CPD explicit, we can use qualitative arguments to deduce these independencies.**

5.3 Context-Specific CPDs

5.3.1 Representation

Structure in CPDs does not arise only in the case of deterministic dependencies. A very common type of regularity arises when we have precisely the same effect in several contexts.

Example 5.7

We augment our Student *example to model the event that the student will be offered a job at Acme Consulting. Thus, we have a binary-valued variable J, whose value is j^1 if the student is offered this job, and j^0 otherwise. The probability of this event depends on the student's SAT scores and the strength of his recommendation letter. We also have to represent the fact that our student might*

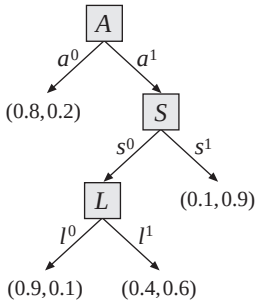


Figure 5.4 A tree-CPD for $P(J \mid A, S, L)$. Internal nodes in the tree denote tests on parent variables. Leaves are annotated with the distribution over J.

choose not to apply for a job at Acme Consulting. Thus, we have a binary variable Applied, whose value (a^1 or a^0) indicates whether the student applied or not. The structure of the augmented network is shown in figure 5.3.

Now, we need to describe the CPD $P(J \mid A, S, L)$. In our domain, even if the student does not apply, there is still a chance that Acme Consulting is sufficiently desperate for employees to offer him a job anyway. (This phenomenon was quite common during the days of the Internet Gold Rush.) In this case, however, the recruiter has no access to the student's SAT scores or recommendation letters, and therefore the decision to make an offer cannot depend on these variables. Thus, among the 8 values of the parents A, S, L, the four that have $A = a^0$ must induce an identical distribution over the variable J.

We can elaborate this model even further. Assume that our recruiter, knowing that SAT scores are a far more reliable indicator of the student's intelligence than a recommendation letter, first considers the SAT score. If it is high, he generates an offer immediately. (As we said, Acme Consulting is somewhat desperate for employees.) If, on the other hand, the SAT score is low, he goes to the effort of obtaining the professor's letter of recommendation, and makes his decision accordingly. In this case, we have yet more regularity in the CPD: $P(J \mid a^1, s^1, l^1) = P(J \mid a^1, s^1, l^0)$. ■

In this simple example, we have a CPD in which several values of Pa_J specify the same conditional probability over J. In general, we often have CPDs where, for certain partial assignments $\boldsymbol{u}$ to subsets $\boldsymbol{U} \subset \mathrm{Pa}_X$, the values of the remaining parents are not relevant. In such cases, several different distributions $P(X \mid \mathrm{pa}_X)$ are identical. In this section, we discuss how we might capture this regularity in our CPD representation and what implications this structure has on conditional independence. There are many possible approaches for capturing functions over a scope $\boldsymbol{X}$ that are constant over certain subsets of instantiations to $\boldsymbol{X}$. In this section, we present two common and useful choices: trees and rules.

5.3.1.1 Tree-CPDs

A very natural representation for capturing common elements in a CPD is via a *tree*, where the leaves of the tree represent different possible (conditional) distributions over J, and where the path to each leaf dictates the contexts in which this distribution is used.

Example 5.8

Figure 5.4 shows a tree for the CPD of the variable J in example 5.7. Given this tree, we find $P(J \mid A, S, L)$ by traversing the tree from the root downward. At each internal node, we see a test on one of the attributes. For example, in the root node of our tree we see a test on the value A. We then follow the arc that is labeled with the value a, which is given in the current setting of the parents. Assume, for example, that we are interested in $P(J \mid a^1, s^1, l^0)$. Thus, we have that $A = a^1$, and we would follow the right-hand arc labeled a^1. The next test is over S. We have $S = s^1$, and we would also follow the right-hand arc. We have now reached a leaf, which is annotated with a particular distribution over J: $P(j^1) = 0.9$, and $P(j^0) = 0.1$. This distribution is the one we use for $P(J \mid a^1, s^1, l^0)$. ■

Formally, we use the following recursive definition of trees.

Definition 5.2
tree-CPD

A tree-CPD *representing a CPD for variable X is a rooted tree; each* t-node *in the tree is either a* leaf t-node *or an* interior t-node. *Each leaf is labeled with a distribution $P(X)$. Each interior t-node is labeled with some variable $Z \in \mathrm{Pa}_X$. Each interior t-node has a set of outgoing* arcs *to its children, each one associated with a unique* variable assignment *$Z = z_i$ for $z_i \in Val(Z)$.*

A branch *β through a tree-CPD is a path beginning at the root and proceeding to a leaf node. We assume that no branch contains two interior nodes labeled by the same variable. The* parent context *induced by branch β is the set of variable assignments $Z = z_i$ encountered on the arcs along the branch.* ■

Note that, to avoid confusion, we use t-nodes and arcs for a tree-CPD, as opposed to our use of nodes and edges as the terminology in a BN.

Example 5.9

Consider again the tree in figure 5.4. There are four branches in this tree. One induces the parent context $\langle a^0 \rangle$, corresponding to the situation where the student did not apply for the job. A second induces the parent context $\langle a^1, s^1 \rangle$, corresponding to an application with a high SAT score. The remaining two branches induce complete assignments to all the parents of J: $\langle a^1, s^0, l^1 \rangle$ and $\langle a^1, s^0, l^0 \rangle$. Thus, this representation breaks down the conditional distribution of J given its parents into four parent contexts by grouping the possible assignments in $Val(\mathrm{Pa}_J)$ into subsets that have the same effect on J. Note that now we need only 4 parameters to describe the behavior of J, instead of 8 in the table representation. ■

Regularities of this type occur in many domains. Some events can occur only in certain situations. For example, we can have a *Wet* variable, denoting whether we get wet; that variable would depend on the *Raining* variable, but only in the context where we are outside. Another type of example arises in cases where we have a sensor, for example, a thermometer; in general, the thermometer depends on the temperature, but not if it is broken.

This type of regularity is very common in cases where a variable can depend on one of a large set of variables: it depends only on one, but we have uncertainty about the choice of variable on which it depends.

Example 5.10

Let us revisit example 3.7, where George had to decide whether to give the recruiter at Acme Consulting the letter from his professor in Computer Science 101 or his professor in Computer Science 102. George's chances of getting a job can depend on the quality of both letters L_1 and L_2,

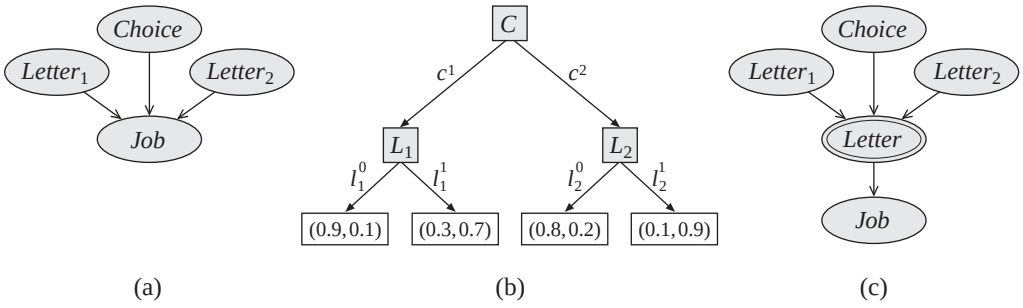


Figure 5.5 The OneLetter **example.** (a) The network fragment. (b) tree-CPD for $P(J \mid C, L_1, L_2)$. (c) Modified network with a new variable L that has a multiplexer CPD.

and hence both are parents. However, depending on which choice C George makes, the dependence will only be on one of the two. Figure 5.5a shows the network fragment, and b shows the tree-CPD for the variable J. (For simplicity, we have eliminated the dependence on S and A that we had in figure 5.4.) ■

More formally, we define the following:

Definition 5.3

multiplexer CPD

A CPD $P(Y \mid A, Z_1, \ldots, Z_k)$ is said to be a multiplexer CPD *if $Val(A) = \{1, \ldots, k\}$, and*

$$P(Y \mid a, Z_1, \ldots, Z_k) = \mathbf{1}\{Y = Z_a\},$$

selector variable

where a is the value of A. The variable A is called the selector variable *for the CPD.* ■

In other words, the value of the multiplexer variable is a copy of the value of one of its parents, $Z_1, \ldots, Z_k$. The role of A is to select the parent who is being copied. Thus, we can think of a multiplexer CPD as a switch.

We can apply this definition to example 5.10 by introducing a new variable L, which is a multiplexer of L_1 and L_2, using C as the selector. The variable J now depends directly only on L. The modified network is shown in figure 5.5c.

This type of model arises in many settings. For example, it can arise when we have different actions in our model; it is often the case that the set of parents for a variables varies considerably based on the action taken. Configuration variables also result in such situations: depending on the specific configuration of a physical system, the interactions between variables might differ (see box 5.A).

correspondence

data association

Another setting where this type of model is particularly useful is in dealing with uncertainty about *correspondence* between different objects. This problem arises, for example, in *data association*, where we obtain sensor measurements about real-world objects, but we are uncertain about which object gave rise to which sensor measurement. For example, we might get a blip on a radar screen without knowing which of several airplanes the source of the signal. Such cases also arise in robotics (see box 15.A and box 19.D). Similar situations also arise in other applications, such as *identity resolution*: associating names mentioned in text to the real-

identity resolution

correspondence variable

world objects to which they refer (see box 6.D). We can model this type of situation using a *correspondence variable* U that associates, with each sensor measurement, the identity u of the object that gave rise to the measurement. The actual sensor measurement is then defined using a multiplexer CPD that depends on the correspondence variable U (which plays the role of the selector variable), and on the value of $A(u)$ for all u from which the measurement could have been derived. The value of the measurement will be the value of $A(u)$ for $U = u$, usually with some added noise due to measurement error. Box 12.D describes this problem in more detail and presents algorithms for dealing with the difficult inference problem it entails.

Trees provide a very natural framework for representing context-specificity in the CPD. In particular, it turns out that people find it very convenient to represent this type of structure using trees. Furthermore, the tree representation lends itself very well to automated learning algorithms that construct a tree automatically from a data set.

---

troubleshooting

Box 5.A — Case Study: Context-Specificity in Diagnostic Networks. *A common setting where context-specific CPDs arise is in* troubleshooting *of physical systems, as described, for example, by Heckerman, Breese, and Rommelse (1995). In such networks, the context specificity is due to the presence of alternative configurations. For example, consider a network for diagnosis of faults in a printer, developed as part of a suite of troubleshooting networks for Microsoft's Windows 95™ operating system. This network, shown in figure 5.A.1a, models the fact that the printer can be hooked up either to the network via an Ethernet cable or to a local computer via a cable, and therefore depends on both the status of the local transport medium and the network transport medium. However, the status of the Ethernet cable only affects the printer's output if the printer is hooked up to the network. The tree-CPD for the variable Printer-Output is shown in figure 5.A.1b. Even in this very simple network, this use of local structure in the CPD reduced the number of parameters required from 145 to 55.*

We return to the topic of Bayesian networks for troubleshooting in box 21.C and box 23.C.

5.3.1.2 Rule CPDs

As we seen, trees are appealing for several reasons. However, trees are a global representation that captures the entire CPD in a single data structure. In many cases, it is easier to reason using a CPD if we break down the dependency structure into finer-grained elements. A finer-grained representation of context-specific dependencies is via *rules*. Roughly speaking, each rule corresponds to a single entry in the CPD of the variable. It specifies a context in which the CPD entry applies and its numerical value.

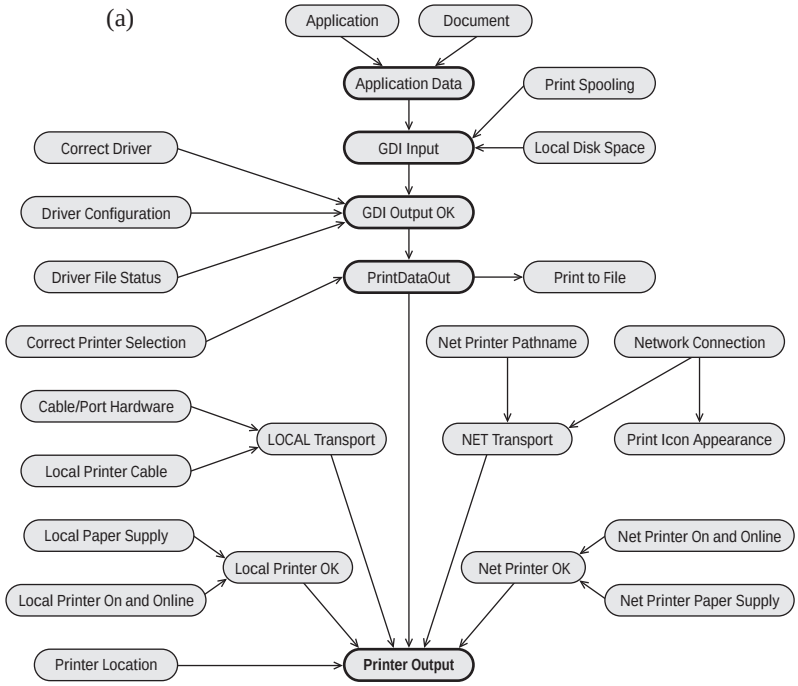
Definition 5.4

rule

scope

A rule ρ *is a pair* $\langle \mathbf{c}; p \rangle$ *where* $\mathbf{c}$ *is an assignment to some subset of variables* $\boldsymbol{C}$*, and* $p \in [0, 1]$*. We define* $\boldsymbol{C}$ *to be the* scope *of* ρ*, denoted* $Scope[\rho]$. ■

This representation decomposes a tree-CPD into its most basic elements.



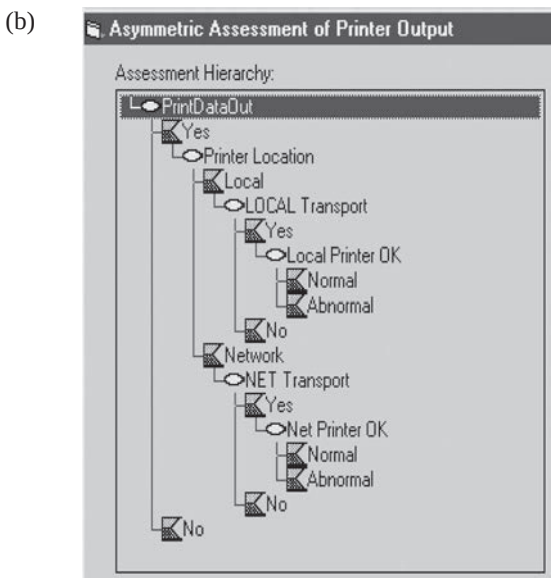


Figure 5.A.1 — Context-specific independencies for diagnostic networks. (a) A real-life Bayesian network that uses context-specific dependencies. The network is used for diagnosing printing problems in Microsoft's online troubleshooting system. (b) The structure of the tree-CPD for the *Printer Output* variable in that network.

Example 5.11

Consider the tree of figure 5.4. There are eight entries in the CPD tree, such that each one corresponds to a branch in the tree and an assignment to the variable J itself. Thus, the CPD defines eight rules:

$$\left\{\begin{array}{l}
\rho_1{:}\langle a^0, j^0; 0.8\rangle \\
\rho_2{:}\langle a^0, j^1; 0.2\rangle \\
\rho_3{:}\langle a^1, s^0, l^0, j^0; 0.9\rangle \\
\rho_4{:}\langle a^1, s^0, l^0, j^1; 0.1\rangle \\
\rho_5{:}\langle a^1, s^0, l^1, j^0; 0.4\rangle \\
\rho_6{:}\langle a^1, s^0, l^1, j^1; 0.6\rangle \\
\rho_7{:}\langle a^1, s^1, j^0; 0.1\rangle \\
\rho_8{:}\langle a^1, s^1, j^1; 0.9\rangle
\end{array}\right\}$$

For example, the rule ρ_4 is derived by following the branch a^1, s^0, l^0 and then selecting the probability associated with the assignment $J = j^1$. ■

Although we can decompose any tree-CPD into its constituent rules, we wish to define rule-based CPDs as an independent notion. To define a coherent CPD from a set of rules, we need to make sure that each conditional distribution of the form $P(X \mid \mathrm{Pa}_X)$ is specified by precisely one rule. Thus, the rules in a CPD must be mutually exclusive and exhaustive.

Definition 5.5

rule-based CPD

A rule-based CPD $P(X \mid \mathrm{Pa}_X)$ *is a set of rules $\mathcal{R}$ such that:*

- *For each rule $\rho \in \mathcal{R}$, we have that Scope[ρ] $\subseteq \{X\} \cup \mathrm{Pa}_X$.*
- *For each assignment $(x, \boldsymbol{u})$ to $\{X\} \cup \mathrm{Pa}_X$, we have precisely one rule $\langle \boldsymbol{c}; p\rangle \in \mathcal{R}$ such that $\boldsymbol{c}$ is compatible with $(x, \boldsymbol{u})$. In this case, we say that $P(X = x \mid \mathrm{Pa}_X = \boldsymbol{u}) = p$.*
- *The resulting CPD $P(X \mid \boldsymbol{U})$ is a legal CPD, in that*

$$\sum_x P(x \mid \boldsymbol{u}) = 1.$$

■

The rule set in example 5.11 satisfies these conditions.

Consider the following, more complex, example.

Example 5.12

Let X be a variable with $\mathrm{Pa}_X = \{A, B, C\}$, and assume that X's CPD is defined via the following set of rules:

$\rho_1{:}\langle a^1, b^1, x^0; 0.1\rangle$ $\quad$ $\rho_2{:}\langle a^1, b^1, x^1; 0.9\rangle$
$\rho_3{:}\langle a^0, c^1, x^0; 0.2\rangle$ $\quad$ $\rho_4{:}\langle a^0, c^1, x^1; 0.8\rangle$
$\rho_5{:}\langle b^0, c^0, x^0; 0.3\rangle$ $\quad$ $\rho_6{:}\langle b^0, c^0, x^1; 0.7\rangle$
$\rho_7{:}\langle a^1, b^0, c^1, x^0; 0.4\rangle$ $\quad$ $\rho_8{:}\langle a^1, b^0, c^1, x^1; 0.6\rangle$
$\rho_9{:}\langle a^0, b^1, c^0; 0.5\rangle$

This set of rules defines the following CPD:

X	$a^0b^0c^0$	$a^0b^0c^1$	$a^0b^1c^0$	$a^0b^1c^1$	$a^1b^0c^0$	$a^1b^0c^1$	$a^1b^1c^0$	$a^1b^1c^1$
x^0	0.3	0.2	0.5	0.2	0.3	0.4	0.1	0.1
x^1	0.7	0.8	0.5	0.8	0.7	0.6	0.9	0.9

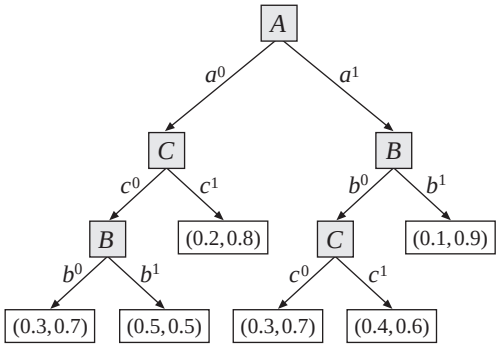


Figure 5.6 A tree-CPD for the rule-based CPD $P(X \mid A, B, C)$ of example 5.12.

For example, the CPD entry $P(x^0 \mid a^0, b^1, c^1)$ is determined by the rule ρ_3, resulting in the CPD entry 0.2; we can verify that no other rule is compatible with the context a^0, b^1, c^1, x^1. We can also verify that each of the CPD entries is also compatible with precisely one context, and hence that the different contexts are mutually exclusive and exhaustive.

Note that both CPD entries $P(x^1 \mid a^0, b^1, c^0)$ and $P(x^0 \mid a^0, b^1, c^0)$ are determined by a single rule ρ_9. As the probabilities for the different contexts in this case must sum up to 1, this phenomenon is only possible when the rule defines a uniform distribution, as it does in this case.■

This perspective views rules as a decomposition of a CPD. We can also view a rule as a finer-grained factorization of an entire distribution.

Proposition 5.1

Let $\mathcal{B}$ be a Bayesian network, and assume that each CPD $P(X \mid \mathrm{Pa}_X)$ in $\mathcal{B}$ is represented as a set of rules $\mathcal{R}_X$. Let $\mathcal{R}$ be the multiset defined as $\uplus_{X \in \mathcal{X}} \mathcal{R}_X$, where $\uplus$ denotes multiset join, which puts together all of the rule instances (including duplicates). Then, the probability of any instantiation ξ to the network variables $\mathcal{X}$ can be computed as

$$P(\xi) = \prod_{\langle \mathbf{c}; p \rangle \in \mathcal{R}, \xi \sim \mathbf{c}} p.$$

The proof is left as an exercise (exercise 5.3).

The rule representation is more than a simple transformation of tree-CPDs. In particular, although every tree-CPDs can be represented compactly as a set of rules, the converse does not necessarily hold: not every rule-based CPD can be represented compactly as a tree.

Example 5.13

Consider the rule-based CPD of example 5.12. In any rule set that is derived from a tree, one variable — the one at the root — appears in all rules. In the rule set $\mathcal{R}$, none of the parent variables A, B, C appears in all rules, and hence the rule set is not derived from a tree. If we try to represent it as a tree-CPD, we would have to select one of A, B, or C to be the root. Say, for example, that we select A to be the root. In this case, rules that do not contain A would necessarily correspond to more than one branch (one for a^1 and one for a^0). Thus, the transformation would result in more branches than rules. For example, figure 5.6 shows a minimal tree-CPD that represents the rule-based CPD of example 5.12. ■

5.3.1.3 Other Representations

The tree and rule representations provide two possibilities for representing context-specific structure. We have focused on these two approaches as they have been demonstrated to be useful for representation, for inference, or for learning. However, other representations are also possible, and can also be used for these tasks. In general, if we abstract away from the details of these representations, we see that they both simply induce *partitions* of $Val(\{X\} \cup \mathrm{Pa}_X)$, defined by the branches in the tree on one hand or the rule contexts on the other. Each partition is associated with a different entry in X's CPD.

This perspective allows us to understand the strengths and limitations of the different representations. In both trees and rules, all the partitions are described via an assignment to a subset of the variables. Thus, for example, we cannot represent the partition that contains only a^1, s^1, l^0 and a^1, s^0, l^1, a partition that we might obtain if the recruiter lumped together candidates that had a high SAT score or a strong recommendation letter, but not both. As defined, these representations also require that we either split on a variable (within a branch of the tree or within a rule) or ignore it entirely. In particular, this restriction does not allow us to capture dependencies that utilize a taxonomic hierarchy on some parent attribute, as described in box 5.B

Of course, we can still represent distributions with these properties by simply having multiple tree branches or multiple rules that are associated with the same parameterization. However, this solution both is less compact and fails to capture some aspects of the structure of the CPD. A very flexible representation, that allows these structures, might use general logical formulas to describe partitions. This representation is very flexible, and it can precisely capture any partition we might consider; however, the formulas might get fairly complex. Somewhat more restrictive is the use of a *decision diagram*, which allows different t-nodes in a tree to share children, avoiding duplication of subtrees where possible. This representation is more general than trees, in that any structure that can be represented compactly as a tree can be represented compactly as a decision diagram, but the converse does not hold. Decision diagrams are incomparable to rules, in that there are examples where each is more compact than the other. In general, different representations offer different trade-offs and might be appropriate for different applications.

decision diagram

---

multinet

Box 5.B — Concept: Multinets and Similarity Networks. *The* multinet *representation provides a more global approach to capturing context-specific independence. In its simple form, a multinet is a network centered on a single distinguished class variable C, which is a root of the network. The multinet defines a separate network $\mathcal{B}_c$ for each value of C, where the structure as well as the parameters can differ for these different networks. In most cases, a multinet defines a single network where every variable X has as its parents C, and all variables Y in* any *of the networks $\mathcal{B}_c$. However, the CPD of X is such that, in context $C = c$, it depends only on $\mathrm{Pa}_X^{\mathcal{B}_c}$. In some cases, however, a subtlety arises, where Y is a parent of X in $\mathcal{B}_{c^1}$, and X is a parent of Y in $\mathcal{B}_{c^2}$. In this case, the Bayesian network induced by the multinet is cyclic; nevertheless, because of the context-specific independence properties of this network, it specifies a coherent distribution. (See also exercise 5.2.) Although, in most cases, a multinet can be represented as a standard BN with context-specific CPDs, it is nevertheless useful, since it explicitly shows the independencies in a graphical form, making them easier to understand and elicit.*

similarity network

A related representation, the similarity network*, was developed as part of the Pathfinder system (see box 3.D). In a similarity network, we define a network $\mathcal{B}_S$ for certain subsets of values $S \subset Val(C)$, which contains only those attributes relevant for distinguishing between the values in S. The underlying assumption is that, if a variable X does not appear in the network $\mathcal{B}_S$, then $P(X \mid C = c)$ is the same for all $c \in S$. Moreover, if X does not have Y as a parent in this network, then X is contextually independent of Y given $C \in S$ and X's other parents in this network. A similarity network easily captures structure where the dependence of X on C is defined in terms of a taxonomic hierarchy on C. For example, we might have that our class variable is Disease. While Sore-throat depends on Disease, it does not have a different conditional distribution for every value d of Disease. For example, we might partition diseases into diseases that do not cause sore throat and those that do, and the latter might be further split into diffuse disease (causing soreness throughout the throat) and localized diseases (such as abscesses). Using this partition, we might have only three different conditional distributions for $P(\text{Sore-Throat} \mid \text{Disease} = d)$. Multinets facilitate elicitation both by focusing the expert's attention on attributes that matter, and by reducing the number of distinct probabilities that must be elicited.*

5.3.2 Independencies

In many of our preceding examples, we used phrases such as "In the case a^0, where the student does not apply, the recruiter's decision cannot depend on the variables S and L." These phrases suggest that context-specific CPDs induce context-specific independence. In this section, we analyze the independencies induced by context-specific dependency models.

Consider a CPD $P(X \mid \mathrm{Pa}_X)$, where certain distributions over X are shared across different instantiations of Pa_X. The structure of such a CPD allows us to infer certain independencies *locally* without having to consider any global aspects of the network.

Example 5.14

Returning to example 5.7, we can see that $(J \perp_c S, L \mid a^0)$: By the definition of the CPD, $P(J \mid a^0, s, l)$ is the same for all values of s and l. Note that this equality holds regardless of the structure or the parameters of the network in which this CPD is embedded. Similarly, we have that $(J \perp_c L \mid a^1, s^1)$. ■

In general, if we define $\boldsymbol{c}$ to be the context associated with a branch in the tree-CPD for X, then X is independent of the remaining parents ($\mathrm{Pa}_X - \mathit{Scope}[\boldsymbol{c}]$) given the context $\boldsymbol{c}$. However, there might be additional CSI statements that we can determine locally, conditioned on contexts that are not induced by complete branches.

Example 5.15

Consider, the tree-CPD of figure 5.5b. Here, once George chooses to request a letter from one professor, his job prospects still depend on the quality of that professor's letter, but not on that of the other. More precisely, we have that $(J \perp_c L_2 \mid c^1)$; note that c^1 is not the full assignment associated with a branch. ■

Example 5.16

More interestingly, consider again the tree of figure 5.4, and suppose we are given the context s^1. Clearly, we should only consider branches that are consistent with this value. There are two such

branches. One associated with the assignment a^0 and the other with the assignment a^1, s^1. We can immediately see that the choice between these two branches does not depend on the value of L. Thus, we conclude that $(J \perp_c L \mid s^1)$ holds in this case. ■

We can generalize this line of reasoning by considering the rules compatible with a particular context $\boldsymbol{c}$. Intuitively, if none of these rules mentions a particular parent Y of X, then X is conditionally independent of Y given $\boldsymbol{c}$. More generally, we can define the notion of conditioning a rule on a context:

Definition 5.6

reduced rule

Let $\rho = \langle \boldsymbol{c}'; p \rangle$ be a rule and $\boldsymbol{C} = \boldsymbol{c}$ be a context. If $\boldsymbol{c}'$ is compatible with $\boldsymbol{c}$, we say that $\rho \sim \boldsymbol{c}$. In this case, let $\boldsymbol{c}'' = \boldsymbol{c}'\langle Scope[\boldsymbol{c}'] - Scope[\boldsymbol{c}]\rangle$ be the assignment in $\boldsymbol{c}'$ to the variables in $Scope[\boldsymbol{c}'] - Scope[\boldsymbol{c}]$. We then define the reduced rule *$\rho[\boldsymbol{c}] = \langle \boldsymbol{c}''; p \rangle$. For $\mathcal{R}$ a set of rules, we define the* reduced rule set

$$\mathcal{R}[\boldsymbol{c}] = \{\rho[\boldsymbol{c}] \ : \ \rho \in \mathcal{R}, \rho \sim \boldsymbol{c}\}.$$

■

Example 5.17

In the rule set $\mathcal{R}$ of example 5.12, $\mathcal{R}[a^1]$ is the set

$$\begin{array}{ll} \rho_1' : \langle b^1, x^0; 0.1 \rangle & \rho_2 : \langle b^1, x^1; 0.9 \rangle \\ \rho_5 : \langle b^0, c^0, x^0; 0.3 \rangle & \rho_6 : \langle b^0, c^0, x^1; 0.7 \rangle \\ \rho_7' : \langle b^0, c^1, x^0; 0.4 \rangle & \rho_8' : \langle b^0, c^1, x^1; 0.6 \rangle. \end{array}$$

Thus, we have left only the rules compatible with a^1, and eliminated a^1 from the context in the rules where it appeared. ■

Proposition 5.2

Let $\mathcal{R}$ be the rules in the rule-based CPD for a variable X, and let $\mathcal{R}_{\boldsymbol{c}}$ be the rules in $\mathcal{R}$ that are compatible with $\boldsymbol{c}$. Let $\boldsymbol{Y} \subseteq \mathrm{Pa}_X$ be some subset of parents of X such that $\boldsymbol{Y} \cap Scope[\boldsymbol{c}] = \emptyset$. If for every $\rho \in \mathcal{R}[\boldsymbol{c}]$, we have that $\boldsymbol{Y} \cap Scope[\rho] = \emptyset$, then $(X \perp_c \boldsymbol{Y} \mid \mathrm{Pa}_X - \boldsymbol{Y}, \boldsymbol{c})$.

The proof is left as an exercise (exercise 5.4).

This proposition specifies a computational tool for deducing "local" CSI relations from the rule representation. We can check whether a variable Y is being tested in the reduced rule set given a context in linear time in the number of rules. (See also exercise 5.6 for a similar procedure for trees.)

This procedure, however, is incomplete in two ways. First, since the procedure does not examine the actual parameter values, it can miss additional independencies that are true for the specific parameter assignments. However, as in the case of completeness for d-separation in BNs, this violation only occurs in degenerate cases. (See exercise 5.7.)

The more severe limitation of this procedure is that it only tests for independencies between X and some of its parents given a context and the other parents. Are there are other, more global, implications of such CSI relations?

Example 5.18

Consider example 5.7 again. In general, finding out that Gump got the job at Acme will increase our belief that he is intelligent, via evidential reasoning. However, now assume that we know that Gump did not apply. Intuitively, we now learn nothing about his intelligence from the fact that he got the job. ■

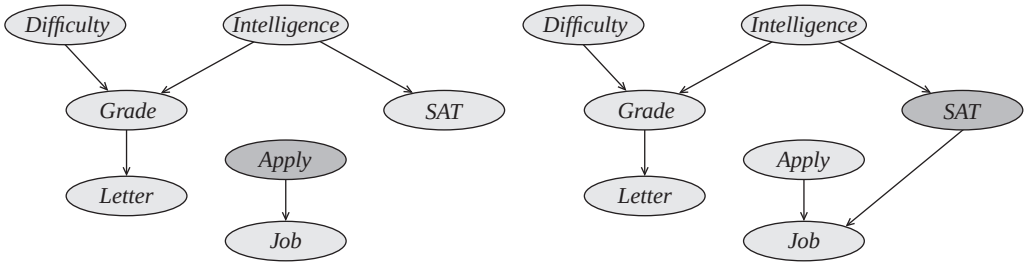


Figure 5.7 The graph of figure 5.3, after we remove spurious edges: (a) in the context $A = a^0$; (b) in the context $S = s^1$.

Can we capture this intuition formally? Consider the dependence structure in the context $A = a^0$. Intuitively, in this context, the edges $S \to J$ and $L \to J$ are both redundant, since we know that $(J \perp_c S, L \mid a^0)$. Thus, our intuition is that we should check for d-separation in the graph without this edge. Indeed, we can show that this is a sound check for CSI conditions.

Definition 5.7 spurious edge

Let $P(X \mid \mathrm{Pa}_X)$ be a CPD, let $Y \in \mathrm{Pa}_X$, and let $\boldsymbol{c}$ be a context. We say that the edge $Y \to X$ is spurious *in the context $\boldsymbol{c}$ if $P(X \mid \mathrm{Pa}_X)$ satisfies $(X \perp_c Y \mid \mathrm{Pa}_X - \{Y\}, \boldsymbol{c}')$, where $\boldsymbol{c}' = \boldsymbol{c}\langle \mathrm{Pa}_X \rangle$ is the restriction of $\boldsymbol{c}$ to variables in Pa_X.* ■

If we represent CPDs with rules, then we can determine whether an edge is spurious by examining the reduced rule set. Let $\mathcal{R}$ be the rule-based CPD for $P(X \mid \mathrm{Pa}_X)$, then the edge $Y \to X$ is spurious in context $\boldsymbol{c}$ if Y does not appear in the reduced rule set $\mathcal{R}[\boldsymbol{c}]$.

Algorithm 5.2 Computing d-separation in the presence of context-specific CPDs

```
    Procedure CSI-sep (
      G,    // Bayesian network structure
      c,    // Context
      X, Y, Z    // Is X CSI-separated from Y given Z, c
    )
1     G' ← G
2     for each edge Y → X in G'
3       if Y → X is spurious given c in G then
4         Remove Y → X in G'
5     return d-sep_G'(X; Y | Z, c)
6
```

CSI-separation

Now we can define *CSI-separation*, a variant of d-separation that takes CSI into account. This notion, defined procedurally in algorithm 5.2, is straightforward: we use local considerations to remove spurious edges and then apply standard d-separation to the resulting graph. We say that

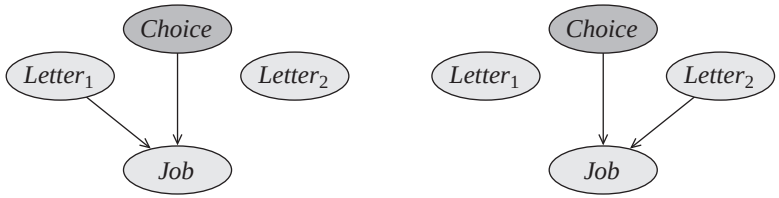


Figure 5.8 Two reductions of the CPD for the OneLetter **example:** (a) in the context $C = c^1$; (b) in the context $C = c^2$.

$\boldsymbol{X}$ is *CSI-separated* from $\boldsymbol{Y}$ given $\boldsymbol{Z}$ in the context $\boldsymbol{c}$ if CSI-SEP$(\mathcal{G}, \boldsymbol{c}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ returns *true*.

As an example, consider the network of example 5.7, in the context $A = a^0$. In this case, we get that the arcs $S \to J$ and $L \to J$ are spurious, leading to the reduced graph in figure 5.7a. As we can see, J and I are d-separated in the reduced graph, as are J and D. Thus, using CSI-SEP, we get that I and J are d-separated given the context a^0. Figure 5.7b shows the reduced graph in the context s^1.

It is not hard to show that CSI-separation provides a sound test for determining context-specific independence.

Theorem 5.3

Let $\mathcal{G}$ be a network structure, let P be a distribution such that $P \models \mathcal{I}_\ell(\mathcal{G})$, let $\boldsymbol{c}$ be a context, and let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ be sets of variables. If $\boldsymbol{X}$ is CSI-separated from $\boldsymbol{Y}$ given $\boldsymbol{Z}$ in the context $\boldsymbol{c}$, then $P \models (\boldsymbol{X} \perp_c \boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{c})$.

The proof is left as an exercise (exercise 5.8).

Of course, we also want to know if CSI-separation is complete — that is, whether it discovers all the context-specific independencies in the distribution. At best, we can hope for the same type of qualified completeness that we had before: discovering all CSI assertions that are a direct consequence of the structural properties of the model, regardless of the particular choice of parameters. In this case, the structural properties consist of the graph structure (as usual) and the structure of the rule sets or trees. Unfortunately, even this weak notion of completeness does not hold in this case.

Example 5.19

Consider the example of figure 5.5b and the context $C = c^1$. In this context, the arc $L_2 \to J$ is spurious. Thus, there is no path between L_1 and L_2, even given J. Hence, CSI-SEP *will report that L_1 and L_2 are d-separated given J and the context $C = c^1$. This case is shown in figure 5.8a. Therefore, we conclude that $(L_1 \perp_c L_2 \mid J, c^1)$. Similarly, in the context $C = c^2$, the arc $L_1 \to J$ is spurious, and we have that L_1 and L_2 are d-separated given J and c^2, and hence that $(L_1 \perp_c L_2 \mid J, c^2)$. Thus, reasoning by cases, we conclude that once we know the value of C, we have that L_1 and L_2 are always d-separated given J, and hence that $(L_1 \perp L_2 \mid J, C)$.*

Can we get this conclusion using CSI-separation? Unfortunately, the answer is no. If we invoke CSI-separation with the empty context, then no edges are spurious and CSI-separation reduces to d-separation. Since both L_1 and L_2 are parents of J, we conclude that they are not *separated given J and C.* ■

The problem here is that CSI-separation does not perform reasoning by cases. Of course, if we want to determine whether X and Y are independent given $\boldsymbol{Z}$ and a context $\boldsymbol{c}$, we can invoke CSI-separation on the context $\boldsymbol{c}, \boldsymbol{z}$ for each possible value of $\boldsymbol{Z}$, and see if X and Y are separated in all of these contexts. This procedure, however, is exponential in the number of variables of $\boldsymbol{Z}$. Thus, it is practical only for small evidence sets. Can we do better than reasoning by cases? The answer is that sometimes we cannot. See exercise 5.10 for a more detailed examination of this issue.

5.4 Independence of Causal Influence

In this section, we describe a very different type of structure in the local probability model. Consider a variable Y whose distribution depends on some set of causes $X_1, \ldots, X_k$. In general, Y can depend on its parents in arbitrary ways — the X_i can interact with each other in complex ways, making the effect of each combination of values unrelated to any other combination. However, in many cases, the combined influence of the X_i's on Y is a simple combination of the influence of each of the X_i's on Y in isolation. In other words, each of the X_i's influences Y independently, and the influence of several of them is simply combined in some way.

We begin by describing two very useful models of this type — the *noisy-or* model, and the class of *generalized linear models*. We then provide a general definition for this type of interaction.

5.4.1 The Noisy-Or Model

Let us begin by considering an example in which a different professor writes a recommendation letter for a student. Unlike our earlier example, this professor teaches a small seminar class, where she gets to know every student. The quality of her letter depends on two things: whether the student participated in class, for example, by asking good questions (Q); and whether he wrote a good final paper (F). Roughly speaking, each of these events is enough to cause the professor to write a good letter. However, the professor might fail to remember the student's participation. On the other hand, she might not have been able to read the student's handwriting, and hence may not appreciate the quality of his final paper. Thus, there is some noise in the process.

Let us consider each of the two causes in isolation. Assume that $P(l^1 \mid q^1, f^0) = 0.8$, that is, the professor is 80 percent likely to remember class participation. On the other hand, $P(l^1 \mid q^0, f^1) = 0.9$, that is, the student's handwriting is readable in 90 percent of the cases. What happens if both occur: the student participates in class and writes a good final paper? The key assumption is that these are two independent *causal mechanisms* for causing a strong letter, and that the letter is weak only if neither of them succeeded. The first causal mechanism — class participation q^1 — fails with probability 0.2. The second mechanism — a good final paper f^1 — fails with probability 0.1. If both q^1 and f^1 occurred, the probability that both mechanisms fail (independently) is $0.2 \cdot 0.1 = 0.02$. Thus, we have that $P(l^0 \mid q^1, f^1) = 0.02$

causal mechanism

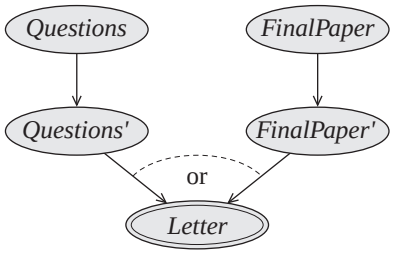


Figure 5.9 Decomposition of the noisy-or model for *Letter*

and $P(l^1 \mid q^1, f^1) = 0.98$. In other words, our CPD for $P(L \mid Q, F)$ is:

Q, F	l^0	l^1
$q^0 f^0$	1	0
$q^0 f^1$	0.1	0.9
$q^1 f^0$	0.2	0.8
$q^1 f^1$	0.02	0.98

noisy-or CPD

This type of interaction between causes is called the *noisy-or* model. Note that we assumed that a student cannot end up with a strong letter if he neither participated in class nor wrote a good final paper. We relax this assumption later on.

An alternative way of understanding this interaction is by assuming that the letter-writing process can be represented by a more elaborate probabilistic model, as shown in figure 5.9. This figure represents the conditional distribution for the *Letter* variable given *Questions* and *FinalPaper*. It also uses two intermediate variables that reveal the associated causal mechanisms. The variable Q' is true if the professor remembers the student's participation; the variable F' is true if the professor could read and appreciate the student's high-quality final paper. The letter is strong if and only if one of these events holds. We can verify that the conditional distribution $P(L \mid Q, F)$ induced by this network is precisely the one shown before.

noise parameter

The probability that Q causes L (0.8 in this example) is called the *noise parameter*, and denoted λ_Q. In the context of our decomposition, $\lambda_Q = P(q'^1 \mid q^1)$. Similarly, we have a noise parameter λ_F, which in this context is $\lambda_F = P(f'^1 \mid f^1)$.

leak probability

We can also incorporate a *leak probability* that represents the probability — say 0.0001 — that the professor would write a good recommendation letter for no good reason, simply because she is having a good day. We simply introduce another variable into the network to represent this event. This variable has no parents, and is true with probability $\lambda_0 = 0.0001$. It is also a parent of the *Letter* variable, which remains a deterministic or.

The decomposition of this CPD clearly shows why this local probability model is called a noisy-or. The basic interaction of the effect with its causes is that of an OR, but there is some noise in the "effective value" of each cause.

We can define this model in the more general setting:

Definition 5.8 noisy-or CPD

Let Y be a binary-valued random variable with k binary-valued parents $X_1, \ldots, X_k$. The CPD $P(Y \mid X_1, \ldots, X_k)$ is a noisy-or *if there are $k + 1$* noise parameters *$\lambda_0, \lambda_1, \ldots, \lambda_k$ such that*

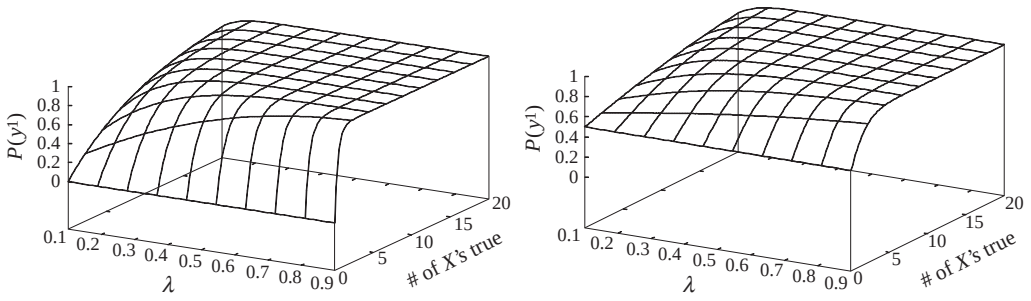


Figure 5.10 The behavior of the noisy-or model as a function of λ and the number of parents that have value *true*: (a) with a leak probability of 0; (b) with a leak probability of 0.5.

$$P(y^0 \mid X_1, \ldots, X_k) = (1 - \lambda_0) \prod_{i \,:\, X_i = x_i^1} (1 - \lambda_i) \tag{5.2}$$

$$P(y^1 \mid X_1, \ldots, X_k) = 1 - [(1 - \lambda_0) \prod_{i \,:\, X_i = x_i^1} (1 - \lambda_i)]$$

We note that, if we interpret x_i^1 as 1 and x_i^0 as 0, we can rewrite equation (5.2) somewhat more compactly as:

$$P(y^0 \mid x_1, \ldots, x_k) = (1 - \lambda_0) \prod_{i=1}^{k} (1 - \lambda_i)^{x_i}. \tag{5.3}$$

Although this transformation might seem cumbersome, it will turn out to be very useful in a variety of settings.

Figure 5.10 shows a graph of the behavior of a special-case noisy or model, where all the variables have the same noise parameter λ. The graph shows the probability of the child Y in terms of the parameter λ and the number of X_i's that have the value *true*.

The noisy-or model is applicable in a wide variety of settings, but perhaps the most obvious is in the medical domain. For example, as we discussed earlier, a symptom variable such as *Fever* usually has a very large number of parents, corresponding to different diseases that can cause the symptom. However, it is often a reasonable approximation to assume that the different diseases use different causal mechanisms, and that if any disease succeeds in activating its mechanism, the symptom is present. Hence, the noisy-or model is a reasonable approximation.

Box 5.C — Concept: BN2O Networks. *A class of networks that has received some attention in the domain of medical diagnosis is the class of* BN2O networks.

BN2O network

A BN2O network, illustrated in figure 5.C.1, is a two-layer Bayesian network, where the top layer corresponds to a set of causes, such as diseases, and the second to findings that might indicate these

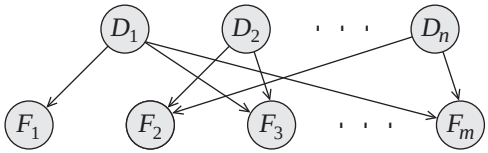


Figure 5.C.1 — A two-layer noisy-or network

causes, such as symptoms or test results. All variables are binary-valued, and the variables in the second layer all have noisy-or models. Specifically, the CPD of F_i is given by:

$$P(f_i^0 \mid \mathrm{Pa}_{F_i}) \quad = \quad (1-\lambda_{i,0}) \prod_{D_j \in \mathrm{Pa}_{F_i}} (1-\lambda_{i,j})^{d_j}.$$

These networks are conceptually very simple and require a small number of easy-to-understand parameters: Each edge denotes a causal association between a cause d_i and a finding f_j; each is associated with a parameter $\lambda_{i,j}$ that encodes the probability that d_i, in isolation, causes f_j to manifest. Thus, these networks resemble a simple set of noisy rules, a similarity that greatly facilitates the knowledge-elicitation task. Although simple, BN2O networks are a reasonable first approximation for a medical diagnosis network.

BN2O networks also have another useful property. In Bayesian networks, observing a variable generally induces a correlation between all of its parents. In medical diagnosis networks, where findings can be caused by a large number of diseases, this phenomenon might lead to significant complexity, both cognitively and in terms of inference. However, in medical diagnosis, most of the findings in any specific case are false — a patient generally only has a small handful of symptoms. As discussed in section 5.4.4, the parents of a noisy-or variable F are conditionally independent given that we observe that F is false. As a consequence, a BN2O network where we observe $F = f^0$ is equivalent to a network where F disappears from the network entirely (see exercise 5.13). This observation can greatly reduce the cost of inference.

5.4.2 Generalized Linear Models

generalized linear model

An apparently very different class of models that also satisfy independence of causal influence are the *generalized linear models*. Although there are many models of this type, in this section we focus on models that define probability distributions $P(Y \mid X_1, \ldots, X_k)$ where Y takes on values in some discrete finite space. We first discuss the case where Y and all of the X_i's are binary-valued. We then extend the model to deal with the multinomial case.

5.4.2.1 Binary-Valued Variables

Roughly speaking, our models in this case are a soft version of a linear threshold function. As a motivating example, we can think of applying this model in a medical setting: In practice, our body's immune system is constantly fighting off multiple invaders. Each of them adds to the

burden, with some adding more than others. We can imagine that when the total burden passes some threshold, we begin to exhibit a fever and other symptoms of infection. That is, as the total burden increases, the probability of fever increases. This requires us to clarify two terms in this discussion. The first is the "total burden" value and how it depends on the particular possible disease causes. The second is a specification of how the probability of fever depends on the total burden.

More generally, we examine a CPD of Y given $X_1, \ldots, X_k$. We assume that the effect of the X_i's on Y can be summarized via a linear function $f(X_1, \ldots, X_k) = \sum_{i=1}^{k} w_i X_i$, where we again interpret x_i^1 as 1 and x_i^0 as 0. In our example, this function will be the total burden on the immune system, and the w_i coefficient describes how much burden is contributed by each disease cause.

The next question is how the probability of $Y = y^1$ depends on $f(X_1, \ldots, X_k)$. In general, this probability undergoes a phase transition around some threshold value τ: when $f(X_1, \ldots, X_k) \geq \tau$, then Y is very likely to be 1; when $f(X_1, \ldots, X_k) < \tau$, then Y is very likely to be 0. It is easier to eliminate τ by simply defining $f(X_1, \ldots, X_k) = w_0 + \sum_{i=1}^{k} w_i X_i$, so that w_0 takes the role of $-\tau$.

To provide a realistic model for immune system example and others, we do not use a hard threshold function to define the probability of Y, but rather a smoother transition function. One common choice (although not the only one) is the *sigmoid* or *logit* function:

sigmoid

$$\text{sigmoid}(z) = \frac{e^z}{1 + e^z}.$$

Figure 5.11a shows the sigmoid function. This function implies that the probability *saturates* to 1 when $f(X_1, \ldots, X_k)$ is large, and saturates to 0 when $f(X_1, \ldots, X_k)$ is small. And so, activation of another disease cause for a sick patient will not change the probability of fever by much, since it is already close to 1. Similarly, if the patient is healthy, a minor burden on the immune system will not increase the probability of fever, since $f(X_1, \ldots, X_k)$ is far from the threshold. In the area of the phase transition, the behavior is close to linear.

We can now define:

Definition 5.9 logistic CPD

Let Y be a binary-valued random variable with k parents $X_1, \ldots, X_k$ that take on numerical values. The CPD $P(Y \mid X_1, \ldots, X_k)$ is a logistic CPD *if there are $k + 1$* weights *$w_0, w_1, \ldots, w_k$ such that:*

$$P(y^1 \mid X_1, \ldots, X_k) \quad = \quad \text{sigmoid}(w_0 + \sum_{i=1}^{k} w_i X_i). \qquad \blacksquare$$

We have already encountered this CPD in example 4.20, where we saw that it can be derived by taking a naive Markov network and reformulating it as a conditional distribution.

log-odds

We can interpret the parameter w_i in terms of its effect on the *log-odds* of Y. In general, the odds ratio for a binary variable is the ratio of the probability of y^1 and the probability of y^0. It is the same concept used when we say that the odds of some event (for example, a sports team winning the Super Bowl) are "2 to 1." Consider the odds ratio for the variable Y, where we use Z to represent $w_0 + \sum_i w_i X_i$:

$$O(\boldsymbol{X}) = \frac{P(y^1 \mid X_1, \ldots, X_k)}{P(y^0 \mid X_1, \ldots, X_k)} = \frac{e^Z/(1 + e^Z)}{1/(1 + e^Z)} = e^Z.$$

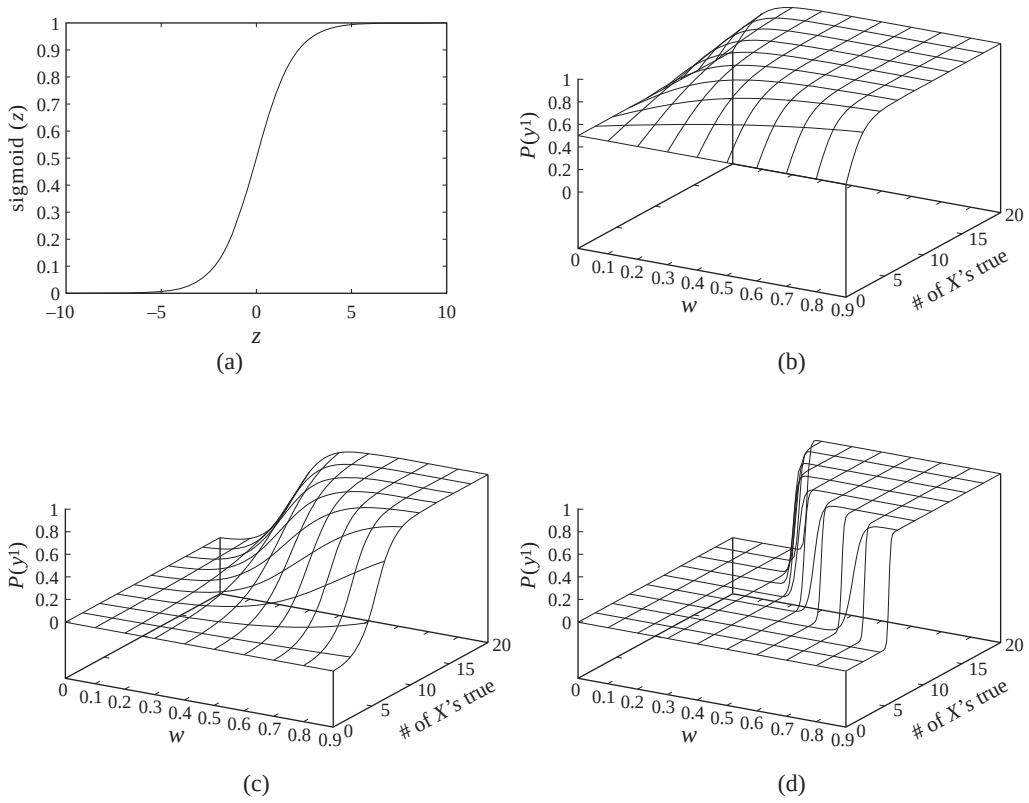


Figure 5.11 The behavior of the sigmoid CPD: (a) The sigmoid function. (b),(c), & (d) The behavior of the linear sigmoid model as a function of w and the number of parents that have value *true*: (b) when the threshold $w_0 = 0$; (c) when $w_0 = -5$; (d) when w and w_0 are multiplied by 10.

Now, consider the effect on this odds ratio as some variable X_j changes its value from *false* to *true*. Let $\boldsymbol{X}_{-j}$ be the variables in $X_1, \ldots, X_k$ except for X_j. Then:

$$\frac{O(\boldsymbol{X}_{-j}, x_j^1)}{O(\boldsymbol{X}_{-j}, x_j^0)} = \frac{\exp(w_0 + \sum_{i \neq j} w_i X_i + w_j)}{\exp(w_0 + \sum_{i \neq j} w_i X_i)} = e^{w_j}.$$

Thus, $X_j =$ *true* changes the odds ratio by a multiplicative factor of e^{w_j}. A positive coefficient $w_j > 0$ implies that $e^{w_j} > 1$ so that the odds ratio increases, hence making y^1 more likely. Conversely, a negative coefficient $w_j < 0$ implies that $e^{w_j} < 1$ and hence the odds ratio decreases, making y^1 less likely.

Figure 5.11b shows a graph of the behavior of a special case of the logistic CPD model, where all the variables have the same weight w. The graph shows $P(Y \mid X_1, \ldots, X_k)$ as a function of w and the number of X_i's that take the value *true*. The graph shows two cases: one where $w_0 = 0$ and the other where $w_0 = -5$. In the first case, the probability starts out at 0.5

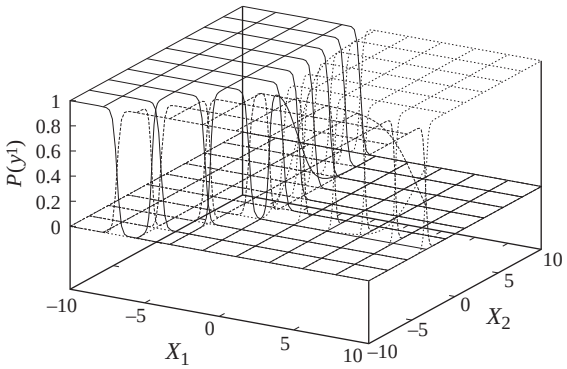


Figure 5.12 A multinomial logistic CPD: The distribution of $P(Y \mid X_1, X_2)$ using the multinomial logistic model for $\ell_1(X_1, X_2) = -3X_1 - 2X_2 + 1$, $\ell_2(X_1, X_2) = 5X_1 - 8X_2 - 4$, and $\ell_3 = x - y + 10$.

when none of the causes are in effect, and rapidly goes up to 1; the rate of increase is, as expected, much higher for high values of w. It is interesting to compare this graph to the graph of figure 5.10b that shows the behavior of the noisy-or model with $\lambda_0 = 0.5$. The graphs exhibit very similar behavior for $\lambda = w$, showing that the incremental effect of a new cause is similar in both. However, the logistic CPD also allows for a negative influence of some X_i on Y by making w_i negative. Furthermore, the parameterization of the logistic model also provides substantially more flexibility in generating qualitatively different distributions. For example, as shown in figure 5.11c, setting w_0 to a different value allows us to obtain the threshold effect discussed earlier. Furthermore, as shown in figure 5.11d, we can adapt the scale of the parameters to obtain a sharper transition. However, the noisy-or model is cognitively very plausible in many settings. Furthermore, as we discuss, it has certain benefits both in reasoning with the models and in learning the models from data.

5.4.2.2 Multivalued Variables

We can extend the logistic CPD to the case where Y takes on multiple values $y^1, \ldots, y^m$. In this case, we can imagine that the different values of Y are each supported in a different way by the X_i's, where the support is again defined via a linear function. The choice of Y can be viewed as a soft version of "winner takes all," where the y^i that has the most support gets probability 1 and the others get probability 0.

More precisely, we have:

Definition 5.10 multinomial logistic CPD

Let Y be an m-valued random variable with k parents $X_1, \ldots, X_k$ that take on numerical values. The CPD $P(Y \mid X_1, \ldots, X_k)$ is a multinomial logistic *if for each $j = 1, \ldots, m$, there are $k + 1$*

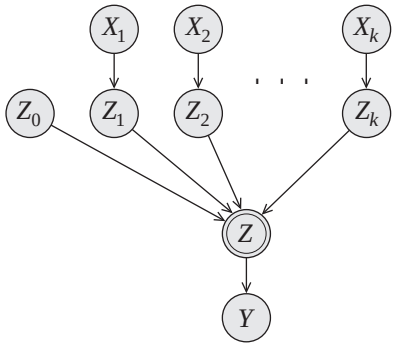


Figure 5.13 Independence of causal influence

weights $w_{j,0}, w_{j,1}, \ldots, w_{j,k}$ *such that:*

$$\begin{aligned} \ell_j(X_1, \ldots, X_k) &= w_{j,0} + \sum_{i=1}^{k} w_{j,i} X_i \\ P(y^j \mid X_1, \ldots, X_k) &= \frac{\exp(\ell_j(X_1, \ldots, X_k))}{\sum_{j'=1}^{m} \exp(\ell_{j'}(X_1, \ldots, X_k))}. \end{aligned}$$

■

Figure 5.12 shows one example of this model for the case of two parents and a three-valued child Y. We note that one of the weights $w_{j,1}, \ldots, w_{j,k}$ is redundant, as it can be folded into the bias term $w_{j,0}$.

We can also deal with the case where the parent variables X_i take on more than two values. The approach taken is usually straightforward. If $X_i = x_i^1, \ldots, x_i^m$, we define a new set of binary-valued variables $X_{i,1}, \ldots, X_{i,m}$, where $X_{i,j} = x_{i,j}^1$ precisely when $X_i = j$. Each of these new variables gets its own coefficient (or set of coefficients) in the logistic function. For example, if we have a binary-valued child Y with an m-valued parent X, our logistic function would be parameterized using $m + 1$ weights, $w_0, w_1, \ldots, w_m$, such that

$$P(y^1 \mid X) = \text{sigmoid}(w_0 + \sum_{j=1}^{m} w_j \mathbf{I}\{X = x^j\}). \tag{5.4}$$

We note that, for any assignment to X_i, precisely one of the weights $w_1, \ldots, w_m$ will make a contribution to the linear function. As a consequence, one of the weights is redundant, since it can be folded into the bias weight w_0.

We noted before that we can view a binary-valued logistic CPD as a conditional version of a naive Markov model. We can generalize this observation to the nonbinary case, and show that the multinomial logit CPD is also a particular type of pairwise CRF (see exercise 5.16).

5.4.3 The General Formulation

causal independence

Both of these models are specials case of a general class of local probability models, which satisfy a property called *causal independence* or *independence of causal influence* (*ICI*). These

models all share the property that the influence of multiple causes can be decomposed into separate influences. We can define the resulting class of models more precisely as follows:

Definition 5.11 *Let Y be a random variable with parents $X_1, \ldots, X_k$. The CPD $P(Y \mid X_1, \ldots, X_k)$ exhibits* independence of causal influence *if it is described via a network fragment of the structure shown in figure 5.13, where the CPD of Z is a deterministic function f.* ■

Intuitively, each variable X_i can be transformed separately using its own individual noise model. The resulting variables Z_i are combined using some deterministic combination function. Finally, an additional stochastic choice can be applied to the result Z, so that the final value of Y is not necessarily a deterministic function of the variables Z_i's. The key here is that any stochastic parts of the model are applied independently to each of the X_i's, so that there can be no interactions between them. The only interaction between the X_i's occurs in the context of the function f.

As stated, this definition is not particularly meaningful. Given an arbitrarily complex function f, we can represent any CPD using the representation of figure 5.13. (See exercise 5.15.) It is possible to place various restrictions on the form of the function f that would make the definition more meaningful. For our purposes, we provide a fairly stringent definition that fortunately turns out to capture the standard uses of ICI models.

Definition 5.12 *We say that a deterministic binary function $x \diamond y$ is* commutative *if $x \diamond y = y \diamond x$, and* associative *if $(x \diamond y) \diamond z = x \diamond (y \diamond z)$. We say that a function $f(x_1, \ldots, x_k)$ is a* symmetric decomposable function *if there is a commutative associative function $x \diamond y$ such that $f(x_1, \ldots, x_k) = x_1 \diamond x_2 \diamond \cdots \diamond x_k$.*[3] ■

Definition 5.13 symmetric ICI

We say that the CPD $P(Y \mid X_1, \ldots, X_k)$ exhibits symmetric ICI *if it is described via a network fragment of the structure shown in figure 5.13, where the CPD of Z is a deterministic symmetric decomposable function f. The CPD exhibits* fully symmetric ICI *if the CPDs of the different Z_i variables are identical.* ■

There are many instantiations of the symmetric ICI model, with different noise models — $P(Z_i \mid X_i)$ — and different combination functions. Our noisy-or model uses the combination function OR and a simple noise model with binary variables. The generalized linear models use the Z_i to produce $w_i X_i$, and then summation as the combination function f. The final soft thresholding effect is accomplished in the distribution of Y given Z.

These types of models turn out to be very useful in practice, both because of their cognitive plausibility and because they provide a significant reduction in the number of parameters required to represent the distribution. The number of parameters in the CPD is linear in the number of parents, as opposed to the usual exponential.

Box 5.D — Case Study: Noisy Rule Models for Medical Diagnosis. *As discussed in box 5.C, noisy rule interactions such as noisy-or are a simple yet plausible first approximation of models for medical diagnosis. A generalization that is also useful in this setting is the* noisy-max model. *Like* noisy-max

3. Because $\diamond$ is associative, the order of application of the operations does not matter.

the application of the noisy-or model for diagnosis, the parents X_i correspond to different diseases that the patient might have. In this case, however, the value space of the symptom variable Y can be more refined than simply $\{present, absent\}$; it can encode the severity of the symptom. Each Z_i corresponds (intuitively) to the effect of the disease X_i on the symptom Y in isolation, that is, the severity of the symptom in case only the disease X_i is present. The value of Z is the maximum of the different Z_i's.

Both noisy-or and noisy-max models have been used in several medical diagnosis networks. Two of the largest are the QMR-DT (Shwe et al. 1991) and CPCS (Pradhan et al. 1994) networks, both based on various versions of a knowledge-based system called QMR (Quick Medical Reference), compiled for diagnosis of internal medicine. QMR-DT is a BN2O network (see box 5.C) that contains more than five hundred significant diseases, about four thousand associated findings, and more than forty thousand disease-finding associations.

CPCS is a somewhat smaller network, containing close to five hundred variables and more than nine hundred edges. Unlike QMR-DT, the network contains not only diseases and findings but also variables for predisposing factors and intermediate physiological states. Thus, CPCS has at least four distinct layers. All variables representing diseases and intermediate states take on one of four values. A specification of the network using full conditional probability tables would require close to 134 million parameters. However, the network is constructed using only noisy-or and noisy-max interactions, so that the number of actual parameters is only 8,254. Furthermore, most of the parameters were generated automatically from "frequency weights" in the original knowledge base. Thus, the number of parameters that were, in fact, elicited during the construction of the network is around 560.

Finally, the symmetric ICI models allow certain decompositions of the CPD that can be exploited by probabilistic inference algorithms for computational gain, when the domain of the variables Z_i and the variable Z are reasonably small.

5.4.4 Independencies

As we have seen, structured CPDs often induce independence properties that go beyond those represented explicitly in the Bayesian network structure. Understanding these independencies can be useful for gaining insight into the properties of our distribution. Also, as we will see, the additional structure can be exploited for improving the performance of various probabilistic inference algorithms.

The additional independence properties that arise in general ICI models $P(Y \mid X_1, \ldots, X_k)$ are more indirect than those we have seen in the context of deterministic CPDs or tree-CPDs. In particular, they do not manifest directly in terms of the original variables, but only if we decompose it by adding auxiliary variables. In particular, as we can easily see from figure 5.13, each X_i is conditionally independent of Y, and of the other X_j's, given Z_i.

We can obtain even more independencies by decomposing the CPD of Z in various ways. For example, assume that $k = 4$, so that our CPD has the form $P(Y \mid X_1, X_2, X_3, X_4)$. We can

introduce two new variables W_1 and W_2, such that:

$$\begin{aligned} W_1 &= Z_0 \diamond Z_1 \diamond Z_2 \\ W_2 &= Z_3 \diamond Z_4 \\ Z &= W_1 \diamond W_2 \end{aligned}$$

By the associativity of $\diamond$, the decomposed CPD is precisely equivalent to the original one. In this CPD, we can use the results of section 5.2 to conclude, for example, that X_4 is independent of Y given W_2.

Although these independencies might appear somewhat artificial, it turns out that the associated decomposition of the network can be exploited by inference algorithms (see section 9.6.1). However, as we will see, they are only useful when the domain of the intermediate variables (W_1 and W_2 in our example) are small. This restriction should not be surprising given our earlier observation that any CPD can be decomposed in this way if we allow the Z_i's and Z to be arbitrarily complex.

The independencies that we just saw are derived simply from the fact that the CPD of Z is deterministic and symmetric. As in section 5.2, there are often additional independencies that are associated with the particular choice of deterministic function. The best-known independence of this type is the one arising for noisy-or models:

Proposition 5.3 *Let $P(Y \mid X_1, \ldots, X_k)$ be a noisy-or CPD. Then for each $i \neq j$, X_i is independent of X_j given $Y = y^0$.*

The proof is left as an exercise (exercise 5.11). Note that this independence is not derived from the network structure via d-separation: Instantiating Y enables the v-structure between X_i and X_j, and hence potentially renders them correlated. Furthermore, this independence is context-specific: it holds only for the specific value $Y = y^0$. Other deterministic functions are associated with other context-specific independencies.

5.5 Continuous Variables

So far, we have restricted attention to discrete variables with finitely many values. In many situations, some variables are best modeled as taking values in some continuous space. Examples include variables such as position, velocity, temperature, and pressure. Clearly, we cannot use a table representation in this case. One common solution is to circumvent the entire issue by discretizing all continuous variables. Unfortunately, this solution can be problematic in many cases. In order to get a reasonably accurate model, we often have to use a fairly fine discretization, with tens or even hundreds of values. For example, when applying probabilistic models to a robot navigation task, a typical discretization granularity might be 15 centimeters for the x and y coordinates of the robot location. For a reasonably sized environment, each of these variables might have more than a thousand values, leading to more than a million discretized values for the robot's position. CPDs of this magnitude are outside the range of most systems.

Furthermore, **when we discretize a continuous variable we often lose much of the structure that characterizes it. It is not generally the case that each of the million values that defines a robot position can be associated with an arbitrary probability. Basic**

continuity assumptions that hold in almost all domains imply certain relationships that hold between probabilities associated with "nearby" discretized values of a continuous variable. However, such constraints are very hard to capture in a discrete distribution, where there is no notion that two values of the variable are "close" to each other.

Fortunately, nothing in our formulation of a Bayesian network requires that we restrict attention to discrete variables. Our only requirement is that the CPD $P(X \mid \mathrm{Pa}_X)$ represent, for every assignment of values pa_X to Pa_X, a distribution over X. In this case, X might be continuous, in which case the CPD would need to represent distributions over a continuum of values; we might also have some of X's parents be continuous, so that the CPD would also need to represent a continuum of different probability distributions. However, as we now show, we can provide implicit representations for CPDs of this type, allowing us to apply all of the machinery we developed for the continuous case as well as for *hybrid networks* involving both discrete and continuous variables.

hybrid network

In this section, we describe how continuous variables can be integrated into the BN framework. We first describe the purely continuous case, where the CPDs involve only continuous variables, both as parents and as children. We then examine the case of hybrid networks, which involve both discrete and continuous variables.

There are many possible models one could use for any of these cases; we briefly describe only one prototypical example for each of them, focusing on the models that are most commonly used. Of course, there is an unlimited range of representations that we can use: any parametric representation for a CPD is eligible in principle. The only difficulty, as far as representation is concerned, is in creating a language that allows for it. Other tasks, such as inference and learning, are a different issue. As we will see, these tasks can be difficult even for very simple hybrid models.

The most commonly used parametric form for continuous density functions is the Gaussian distribution. We have already described the univariate Gaussian distribution in chapter 2. We now describe how it can be used within the context of a Bayesian network representation.

First, let us consider the problem of representing a dependency of a continuous variable Y on a continuous parent X. One simple solution is to decide to model the distribution of Y as a Gaussian, whose parameters depend on the value of X. In this case, we need to have a set of parameters for every one of the infinitely many values $x \in Val(X)$. A common solution is to decide that the mean of Y is a linear function of X, and that the variance of Y does not depend on X. For example, we might have that

$$p(Y \mid x) = \mathcal{N}(-2x + 0.9; 1).$$

Example 5.20

Consider a vehicle (for example, a car) moving over time. For simplicity, assume that the vehicle is moving along a straight line, so that its position (measured in meters) at the t'th second is described using a single variable $X^{(t)}$. Let $V^{(t)}$ represent the velocity of the car at the kth second, measured in meters per second. Then, under ideal motion, we would have that $X^{(t+1)} = X^{(t)} + V^{(t)}$ — if the car is at meter #510 along the road, and its current velocity is 15 meters/second, then we expect its position at the next second to be meter #525. However, there is invariably some stochasticity in the motion. Hence, it is much more realistic to assert that the car's position $X^{(t+1)}$ is described using a Gaussian distribution whose mean is 525 and whose variance is 5 meters. ■

This type of dependence is called a *linear Gaussian* model. It extends to multiple continuous

parents in a straightforward way:

Definition 5.14

linear Gaussian CPD

Let Y be a continuous variable with continuous parents $X_1, \ldots, X_k$. We say that Y has a linear Gaussian *model if there are parameters $\beta_0, \ldots, \beta_k$ and σ^2 such that*

$$p(Y \mid x_1, \ldots, x_k) = \mathcal{N}\left(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k; \sigma^2\right).$$

In vector notation,

$$p(Y \mid \boldsymbol{x}) = \mathcal{N}\left(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}; \sigma^2\right).$$

■

Viewed slightly differently, this formulation says that Y is a linear function of the variables $X_1, \ldots, X_k$, with the addition of Gaussian noise with mean 0 and variance σ^2:

$$Y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \epsilon,$$

where ϵ is a Gaussian random variable with mean 0 and variance σ^2, representing the noise in the system.

This simple model captures many interesting dependencies. However, there are certain facets of the situation that it might not capture. For example, the variance of the child variable Y cannot depend on the actual values of the parents. In example 5.20, we might wish to construct a model in which there is more variance about a car's future position if it is currently moving very quickly. The linear Gaussian model cannot capture this type of interaction.

Of course, we can easily extend this model to have the mean and variance of Y depend on the values of its parents in arbitrary way. For example, we can easily construct a richer representation where we allow the mean of Y to be $\sin(x_1)^{x_2}$ and its variance to be $(x_3/x_4)^2$. However, the linear Gaussian model is a very natural one, which is a useful approximation in many practical applications. Furthermore, as we will see in section 7.2, networks based on the linear Gaussian model provide us with an alternative representation for multivariate Gaussian distributions, one that directly reveals more of the underlying structure.

Box 5.E — Case Study: Robot Motion and Sensors. *One interesting application of hybrid models is in the domain of* robot localization. *In this application, the robot must keep track of its location as it moves in an environment, and obtains sensor readings that depend on its location. This application is an example of a temporal model, a topic that will be discussed in detail in section 6.2; we also return to the robot example specifically in box 15.A. There are two main local probability models associated with this application. The first specifies the robot dynamics — the distribution over its position at the next time step L' given its current position L and the action taken A; the second specifies the robot sensor model — the distribution over its observed sensor reading S at the current time given its current location L.*

robot localization

We describe one model for this application, as proposed by Fox et al. (1999) and Thrun et al. (2000). Here, the robot location L is a three-dimensional vector containing its X, Y coordinates and an angular orientation θ. The action A specifies a distance to travel and a rotation (offset from the current θ). The model uses the assumption that the errors in both translation and rotation are normally distributed with zero mean. Specifically, $P(L' \mid L, A)$ is defined as a product of two independent Gaussians with cut off tails, $P(\theta' \mid \theta, A)$ and $P(X', Y' \mid X, Y, A)$, whose variances

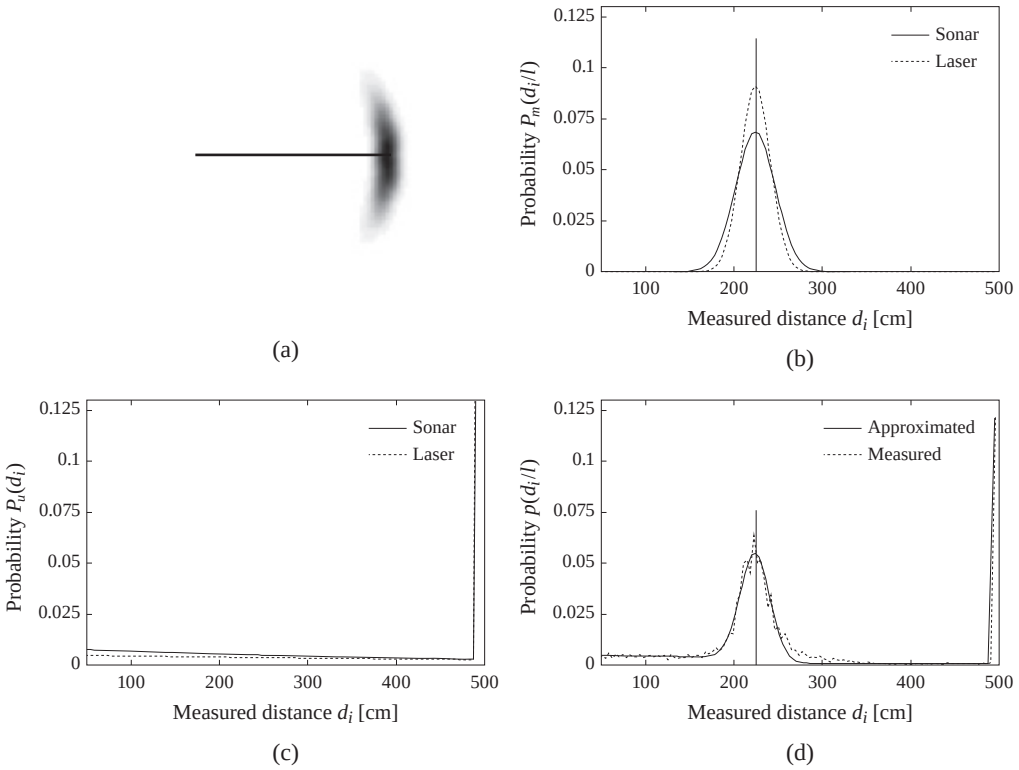


Figure 5.E.1 — Probabilistic model for robot localization track. (a) A typical "banana-shaped" distribution for the robot motion model. The figure shows the projection of the conditional distribution over $L' = \langle X', Y', \theta' \rangle$ onto the X', Y' space, given the robot's starting position and action shown. (b) Two distributions $P_m(D \mid L)$ for the distance returned by a range sensor given that the distance to the closest obstacle is $o_L = 230cm$. The figure shows a distribution for both an ultrasound sensor and a laser range finder; the laser sensor has a higher accuracy than the ultrasound sensor, as indicated by the smaller variance. (c) Two distributions $P_u(D)$ for the distance returned by a range sensor, for an ultrasound sensor and a laser range finder. The relatively large probability of measuring 500 centimeters owes to the fact that the maximum range of the proximity sensors is set to 500 centimeters. Thus, this distance represents the probability of measuring *at least* 500 centimeters. (d) Overall model distribution (solid line) and empirical distribution (dashed line) of $P(D \mid o_L)$, for $o_L = 230cm$ for a laser sensor.

are proportional to the length of the motion. The robot's conditional distribution over (X', Y') is a banana-shaped cloud (see figure 5.E.1a, where the banana shape is due to the noise in the rotation.

The sensor is generally some type of range sensor, either a sonar or a laser, which provides a reading D of the distance between the robot and the nearest obstacle along the direction of the sensor. There are two distinct cases to consider. If the sensor signal results from an obstacle in the map, then the resulting distribution is modeled by a Gaussian distribution with mean at the distance to this obstacle. Letting o_L be the distance to the closest obstacle to the position L (along the sensor beam), we can define $P_m(D \mid L) = \mathcal{N}\left(o_L; \sigma^2\right)$, where the variance σ^2 represents the uncertainty of the measured distance, based on the accuracy of the world model and the accuracy of the sensor. Figure 5.E.1b shows an example of such a distribution for an ultrasound sensor and a laser range finder. The laser sensor has a higher accuracy than the ultrasound sensor, as indicated by the smaller variance.

The second case arises when the sensor beam is reflected by an obstacle not represented in the world model (for example, a dynamic obstacle, such as a person or a chair, which is not in the robot's map). Assuming that these objects are equally distributed in the environment, the probability $P_u(D)$ of detecting an unknown obstacle at distance D is independent of the location of the robot and can be modeled by an exponential distribution. This distribution results from the observation that a distance d is measured if the sensor is not reflected by an obstacle at a shorter distance and is reflected at distance d. An example exponential distribution is shown in figure 5.E.1c.

Only one of these two cases can hold for a given measurement. Thus, $P(D \mid L)$ is a combination of the two distributions P_m and P_u. The combined probability $P(D \mid L)$ is based on the observation that d is measured in one of two cases:

- *The sensor beam in not reflected by an unknown obstacle before reaching distance d, and is reflected by the known obstacle at distance d (an event that happens only with some probability).*
- *The beam is reflected neither by an unknown obstacle nor by the known obstacle before reaching distance d, and it is reflected by an unknown obstacle at distance d.*

Overall, the probability of sensor measurements is computed incrementally for the different distances starting at 0cm*; for each distance, we consider the probability that the sensor beam reaches the corresponding distance and is reflected either by the closest obstacle in the map (along the sensor beam) or by an unknown obstacle. Putting these different cases together, we obtain a single distribution for $P(D \mid L)$. This distribution is shown in figure 5.E.1d, along with an empirical distribution obtained from data pairs consisting of the distance o_L to the closest obstacle on the map and the measured distance d during the typical operation of the robot.*

5.5.1 Hybrid Models

We now turn our attention to models incorporating both discrete and continuous variables. We have to address two types of dependencies: a continuous variable with continuous and discrete parents, and a discrete variable with continuous and discrete parents.

Let us first consider the case of a continuous child X. If we ignore the discrete parents of X, we can simply represent the CPD of X as a linear Gaussian of X's continuous parents. The

simplest way of making the continuous variable X depend on a discrete variable U is to define a different set of parameters for every value of the discrete parent. More precisely:

Definition 5.15 conditional linear Gaussian CPD

Let X be a continuous variable, and let $\boldsymbol{U} = \{U_1, \ldots, U_m\}$ be its discrete parents and $\boldsymbol{Y} = \{Y_1, \ldots, Y_k\}$ be its continuous parents. We say that X has a conditional linear Gaussian *(CLG) CPD if, for every value $\boldsymbol{u} \in Val(\boldsymbol{U})$, we have a set of $k+1$ coefficients $a_{\boldsymbol{u},0}, \ldots, a_{\boldsymbol{u},k}$ and a variance $\sigma^2_{\boldsymbol{u}}$ such that*

$$p(X \mid \boldsymbol{u}, \boldsymbol{y}) = \mathcal{N}\left(a_{\boldsymbol{u},0} + \sum_{i=1}^{k} a_{\boldsymbol{u},i} y_i; \sigma^2_{\boldsymbol{u}}\right)$$

■

If we restrict attention to this type of CPD, we get an interesting class of models. More precisely, we have:

Definition 5.16 CLG network

A Bayesian network is called a CLG network *if every discrete variable has only discrete parents and every continuous variable has a CLG CPD.* ■

Gaussian mixture distribution

Note that the conditional linear Gaussian model does not allow for continuous variables to have discrete children. A CLG model induces a joint distribution that has the form of a *mixture* — a weighted average — of Gaussians. The mixture contains one Gaussian component for each instantiation of the discrete network variables; the weight of the component is the probability of that instantiation. Thus, the number of mixture components is (in the worst case) exponential in the number of discrete network variables.

Finally, we address the case of a discrete child with a continuous parent. The simplest model is a threshold model. Assume we have a binary discrete variable U with a continuous parent Y. We may want to define:

$$P(u^1) = \begin{cases} 0.9 & y \leq 65 \\ 0.05 & \text{otherwise.} \end{cases}$$

Such a model may be appropriate, for example, if Y is the temperature (in Fahrenheit) and U is the thermostat turning the heater on.

The problem with the threshold model is that the change in probability is discontinuous as a function of Y, which is both inconvenient from a mathematical perspective and implausible in many settings. However, we can address this problem by simply using the logistic model or its multinomial extension, as defined in definition 5.9 or definition 5.10.

Figure 5.14 shows how a multinomial CPD can be used to model a simple sensor that has three values: *low*, *medium* and *high*. The probability of each of these values depends on the value of the continuous parent Y. As discussed in section 5.4.2, we can easily accommodate a variety of noise models for the sensor: we can make it less reliable in borderline situations by making the transitions between regions more moderate. It is also fairly straightforward to generalize the model to allow the probabilities of the different values in each of the regions to be values other than 0 or 1.

As for the conditional linear Gaussian CPD, we address the existence of discrete parents for Y by simply introducing a separate set of parameters for each instantiation of the discrete parents.

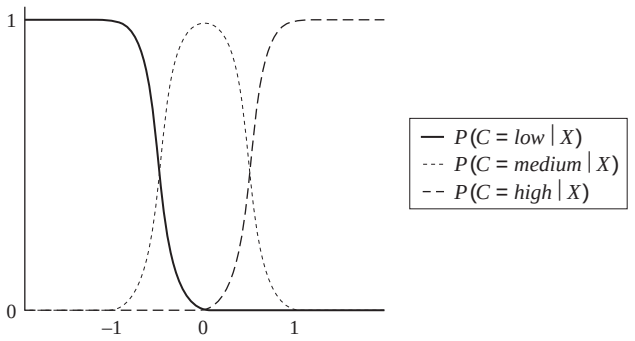


Figure 5.14 Generalized linear model for a thermostat

5.6 Conditional Bayesian Networks

The previous sections all describe various compact representations of a CPD. Another very useful way of compactly representing a conditional probability distribution is via a Bayesian network fragment. We have already seen one very simple example of this idea: Our decomposition of the noisy-or CPD for the *Letter* variable, shown in figure 5.9. There, our decomposition used a Bayesian network to represent the internal model of the *Letter* variable. The network included explicit variables for the parents of the variable, as well as auxiliary variables that are not in the original network. This entire network represented the CPD for *Letter*. In this section, we generalize this idea to a much wider setting.

Note that the network fragment in this example is not a full Bayesian network. In particular, it does not specify a probabilistic model — parents and a CPD — for the parent variables *Questions* and *FinalPaper*. This network fragment specifies not a joint distribution over the variables in the fragment, but a *conditional* distribution of *Letter given Questions* and *FinalPaper*. More generally, we can define the following:

Definition 5.17 conditional Bayesian network

A conditional Bayesian network $\mathcal{B}$ *over* $\boldsymbol{Y}$ *given* $\boldsymbol{X}$ *is defined as a directed acyclic graph* $\mathcal{G}$ *whose nodes are* $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$*, where* $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ *are disjoint. The variables in* $\boldsymbol{X}$ *are called* inputs*, the variables in* $\boldsymbol{Y}$ outputs*, and the variables in* $\boldsymbol{Z}$ encapsulated*. The variables in* $\boldsymbol{X}$ *have no parents in* $\mathcal{G}$*. The variables in* $\boldsymbol{Z} \cup \boldsymbol{Y}$ *are associated with a conditional probability distribution. The network defines a conditional distribution using a chain rule:*

$$P_{\mathcal{B}}(\boldsymbol{Y}, \boldsymbol{Z} \mid \boldsymbol{X}) = \prod_{X \in \boldsymbol{Y} \cup \boldsymbol{Z}} P(X \mid \mathrm{Pa}_X^{\mathcal{G}}).$$

The distribution $P_{\mathcal{B}}(\boldsymbol{Y} \mid \boldsymbol{X})$ *is defined as the marginal of* $P_{\mathcal{B}}(\boldsymbol{Y}, \boldsymbol{Z} \mid \boldsymbol{X})$*:*

$$P_{\mathcal{B}}(\boldsymbol{Y} \mid \boldsymbol{X}) = \sum_{\boldsymbol{Z}} P_{\mathcal{B}}(\boldsymbol{Y}, \boldsymbol{Z} \mid \boldsymbol{X}).$$

■

conditional random field

The *conditional random field* of section 4.6.1 is the undirected analogue of this definition.

The notion of a conditional BN turns out to be useful in many settings. In particular, we can use it to define an encapsulated CPD.

Definition 5.18 encapsulated CPD

Let Y be a random variable with k parents $X_1, \ldots, X_k$. The CPD $P(Y \mid X_1, \ldots, X_k)$ is an encapsulated CPD *if it is represented using a conditional Bayesian network over Y given $X_1, \ldots, X_k$.* ■

At some level, it is clear that the representation of an individual CPD for a variable Y as a conditional Bayesian network $\mathcal{B}_Y$ does not add expressive power to the model. After all, we could simply take the network $\mathcal{B}_Y$ and "substitute it in" for the atomic CPD $P(Y \mid \mathrm{Pa}_Y)$. One key advantage of the encapsulated representation over a more explicit model is that the encapsulation can simplify the model significantly from a cognitive perspective. Consider again our noisy-or model. Externally, to the rest of the network, we can still view *Letter* as a single variable with its two parents: *Questions* and *FinalPaper*. All of the internal structure is encapsulated, so that, to the rest of the network, the variable can be viewed as any other variable. In particular, a knowledge engineer specifying the network does not have to ascribe meaning to the encapsulated variables.

The encapsulation advantage can be even more significant when we want to describe a complex system where components are composed of other, lower-level, subsystems. When specifying a model for such a system, we would like to model each subsystem separately, without having to consider the internal model of its lower level components.

In particular, consider a model for a physical device such as a computer; we might construct such a model for fault diagnosis purposes. When modeling the computer, we would like to avoid thinking about the detailed structure and fault models of its individual components, such as the hard drive, and within the hard drive the disk surfaces, the controller, and more, each of which has yet other components. By using an encapsulated CPD, we can decouple the model of the computer from the detailed model of the hard drive. We need only specify which global aspects of the computer state the hard drive behavior depends on, and which it influences. Furthermore, we can hierarchically compose encapsulated CPDs, modeling, in turn, the hard drive's behavior in terms of its yet-lower-level components.

In figure 5.15 we show a simple hierarchical model for a computer system. This high-level model for a computer, figure 5.15a, uses encapsulated CPDs for *Power-Source*, *Motherboard*, *Hard-Drive*, *Printer*, and more. The *Hard-Drive* CPD has inputs *Temperature*, *Age* and *OS-Status*, and the outputs *Status* and *Full*. Although the hard drive has a rich internal state, the only aspects of its state that influence objects outside the hard drive are whether it is working properly and whether it is full. The *Temperature* input of the hard drive in a computer is outside the probabilistic model and will be mapped to the *Temperature* parent of the *Hard-Drive* variable in the computer model. A similar mapping happens for other inputs.

The *Hard-Drive* encapsulated network, figure 5.15b, in turn uses encapsulated CPDs for *Controller*, *Surface1*, *Drive-Mechanism*, and more. The hierarchy can continue as necessary. In this case, the model for the variable *Motor* (in the *Drive-Mechanism*) is "simple," in that none of its CPDs are encapsulated.

One obvious observation that can be derived from looking at this example is that an encapsulated CPD is often appropriate for more than one variable in the model. For example, the encapsulated CPD for the variable *Surface1* in the hard drive is almost certainly the same as the CPDs for the variables *Surface2*, *Surface2*, and *Surface4*. Thus, we can imagine creating a *template* of an encapsulated CPD, and reusing it multiple times, for several variables in the model. This idea forms the basis for a framework known as *object-oriented Bayesian networks*.

object-oriented Bayesian network

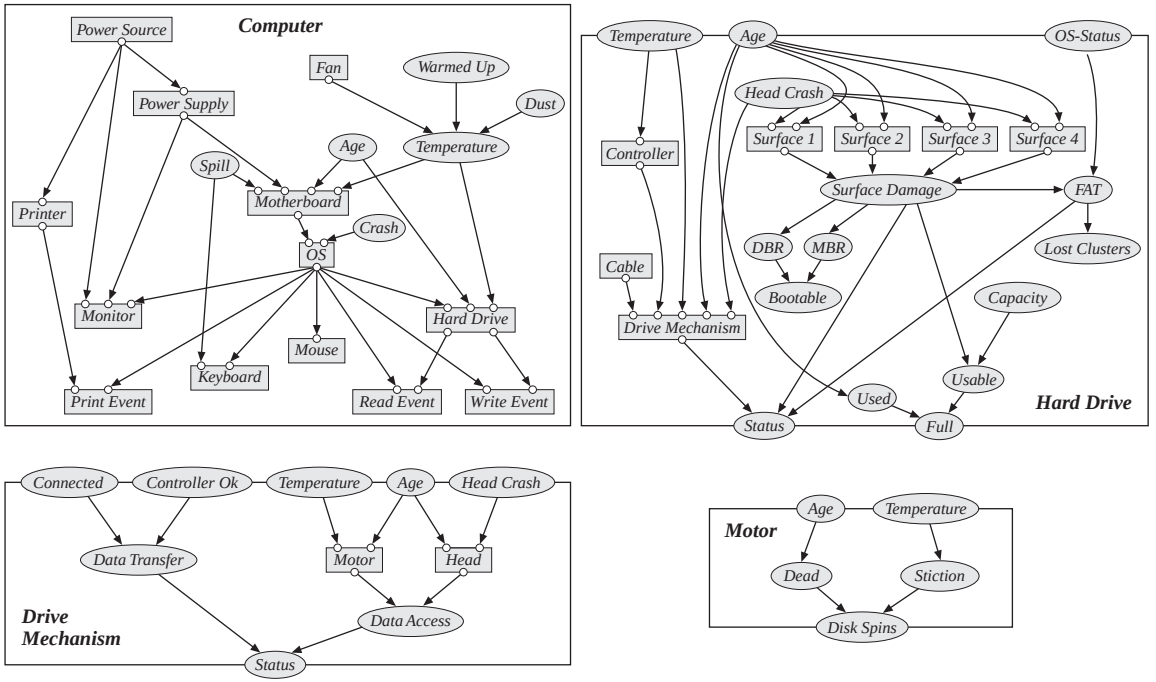


Figure 5.15 Example of encapsulated CPDs for a computer system model: Four levels of a hierarchy of encapsulated CPDs in a model of a computer system. Variables with encapsulated CPDs are shown as rectangles, while nonhierarchical variables are ovals (as usual). Each encapsulated CPD is contained in a box. Input variables intersect the top edge of the box, indicating the fact that their values are received from outside the class, while output variables intersect the bottom. The rectangles representing the complex components also have little bubbles on their borders, showing that variables are passed into and out of those components.

5.7 Summary

In this chapter, we have shown that our ability to represent structure in the distribution does not end at the level of the graph. In many cases, here is important structure within the CPDs that we wish to make explicit. In particular, we discussed several important types of discrete structured CPDs.

- deterministic functions;
- asymmetric, or context specific, dependencies;
- cases where different influences combine independently within the CPD, including noisy-or, logistic functions, and more.

In many cases, we showed that the additional structure provides not only a more compact parameterization, but also additional independencies that are not visible at the level of the original graph.

As we discussed, the idea of structured CPDs is critical in the case of continuous variables, where a table-based representation is clearly irrelevant. We discussed various representations for CPDs in hybrid (discrete/continuous) networks, of which the most common is the linear Gaussian representation. For this case, we showed some important connections between the linear Gaussian representation and multivariate Gaussian distributions.

Finally, we discussed the notion of a conditional Bayesian network, which allows us to decompose a conditional probability distribution recursively, into another Bayesian network.

5.8 Relevant Literature

This chapter addresses the issue of modeling the probabilistic dependence of one variable on some set of others. This issue plays a central role in both statistics and (supervised) machine learning, where much of the work is devoted precisely to the prediction of one variable from others. Indeed, many of the representations described in this chapter are derived from statistical models such as decision trees or regression models. More information on these models can be found in a variety of statistics textbooks (for example, Breiman et al. 1984; Duda et al. 2000; Hastie et al. 2001; McCullagh and Nelder 1989).

The extension of d-separation to the case of Bayesian networks with deterministic CPDs was done by Geiger et al. (1990). Nielsen et al. (2000) proposed a specific representation for deterministic CPDs with particular benefits for inference.

The notion of context-specific independence, which recurs throughout much of this chapter, was first formalized by Shimony (1991). Representations for asymmetric dependencies, which most directly capture CSI, were a key component in Heckerman's similarity networks and the related multinets (Heckerman and Nathwani 1992a; Geiger and Heckerman 1996). Smith, Holtzman, and Matheson (1993) proposed a "conditional" version of influence diagrams, which used many of the same concepts. There have been various proposals for local probabilistic models that encode a mapping between context and parameters. Poole (1993a); Poole and Zhang (2003) used a rule-based representation; Boutilier, Friedman, Goldszmidt, and Koller (1996) proposed both the use of a a fully general partition, and the more specific tree-CPDs; and Chickering, Heckerman, and Meek (1997) suggested the use of general DAG-structured CPDs, which allow "paths" corresponding to different context to "merge." Boutilier *et al.* also define the notion of CSI-separation, which extends the notion of d-separation to networks involving asymmetric dependencies; they provide an efficient algorithm for CSI-separation based on cutting spurious arcs, and they discuss how to check for spurious arcs for various types of local probability models.

The notion of causal independence, and specifically the noisy-or model, was proposed independently by Pearl (1986b) and by Peng and Reggia (1986). This model was subsequently generalized to allow a variety of interactions, such as AND or MAX (Pearl 1988; Heckerman 1993; Srinivas 1993; Pradhan et al. 1994).

Lauritzen and Wermuth (1989) introduced the notion of conditional linear Gaussian Bayesian networks. Shachter and Kenley (1989) used linear Gaussian dependencies in the context of Gaussian influence diagrams, a framework that was later extended to the hybrid case Poland (1994). Lerner et al. (2001) suggest the use of a softmax CPD as a local probability model for discrete variables with continuous parents. Murphy (1998) and Lerner (2002) provide a good

introduction to these topics.

Several papers have proposed the use of richer (semiparametric or nonparametric) models for representing dependencies in networks involving continuous variables, including kernel estimators (Hofmann and Tresp 1995), neural networks (Monti and Cooper 1997), and Gaussian processes (Friedman and Nachman 2000). These representations are generally opaque to humans, and therefore are useful only in the context of learning networks from data.

The notion of encapsulated Bayesian networks as a representation of local probability models was introduced by Srinivas (1994) in the context of modeling hierarchically structured physical systems. This idea was later generalized within the framework of object-oriented Bayesian networks by Koller and Pfeffer (1997), which allows the definition of a general object model as a network fragment, in a way that encapsulates it from the rest of the model. A further generalization was proposed by Heckerman and Meek (1997), which allows the dependencies in the encapsulated network fragment to be oriented in the opposite direction than is implied by the parent-child relations of the CPD.

5.9 Exercises

Exercise 5.1⋆

Prove theorem 5.1.

Exercise 5.2⋆⋆

a. Show that a multinet where each Bayesian network $\mathcal{B}_c$ is acyclic always defines a coherent probability distribution — one where all of the probabilities sum to 1. Your proof should apply even when the induced Bayesian network that contains the union of all of the edges in the networks $\mathcal{B}_c$ contains a cycle.

b. Now, consider a more general case, where each variable X is associated with a rule-based CPD $P(X \mid \mathrm{Pa}_X)$ (as in definition 5.5). Provide a general sufficient condition on the set of rule-based CPDs that guarantee that the distribution defined by the chain rule:

$$P(X_1, \ldots, X_n) = \prod_i P(X_i \mid \mathrm{Pa}_{X_i})$$

is coherent. Your condition should also encompass cases (including the case of multinets) where the induced global network — one containing a directed edge $Y \rightarrow X$ whenever $Y \in \mathrm{Pa}_X$ — is not necessarily acyclic.

Exercise 5.3

Prove proposition 5.1.

Exercise 5.4⋆

Prove proposition 5.2.

Exercise 5.5

In this exercise, we consider the use of tree-structured local models in the undirected setting.

a. Show how we can use a structure similar to tree-CPDs to represent a factor in a Markov network. What do the values at the leaves of such a tree represent?

b. Given a context $\boldsymbol{U} = \boldsymbol{u}$, define a simple algorithm that takes a tree factor $\phi(\boldsymbol{Y})$ and returns the reduced factor $\phi[\boldsymbol{U} = \boldsymbol{u}](\boldsymbol{Y} - \boldsymbol{U})$ (see definition 4.5).

c. The preceding expression takes $\boldsymbol{Y} - \boldsymbol{U}$ to be the scope of the reduced factor. In some cases it turns out that we can further reduce the scope. Give an example and specify a general rule for when a variable in $\boldsymbol{Y} - \boldsymbol{U}$ can be eliminated from the scope of the reduced tree-factor.

Exercise 5.6

Provide an algorithm for constructing a tree-CPD that is the reduced tree reduced for a given context $\boldsymbol{c}$. In other words, assume you are given a tree-CPD for $P(X \mid \mathrm{Pa}_X)$ and a context $\boldsymbol{c}$ such that $\mathit{Scope}[\boldsymbol{c}] \subset \mathrm{Pa}_X$. Provide a linear time algorithm for constructing a tree that represents $P(X \mid \mathrm{Pa}_X, \boldsymbol{c})$.

Exercise 5.7

Consider a BN $\mathcal{B}$ with a variable X that has a tree-CPD. Assume that all of the distributions at the leaves of X's tree-CPD are different. Let $\boldsymbol{c} \in \mathit{Val}(\boldsymbol{C})$ for $\boldsymbol{C} \subseteq \mathrm{Pa}_X$ be a context, and let $\boldsymbol{Z} \subseteq \mathrm{Pa}_X$. Show that if $P_{\mathcal{B}} \models (X \perp_c \boldsymbol{Z} \mid \mathrm{Pa}_X - \boldsymbol{Z}, \boldsymbol{c})$, then $T^{\boldsymbol{c}}$ does not test any variable in $\boldsymbol{Z}$.

Exercise 5.8⋆

Prove theorem 5.3. (Hint: Use exercise 5.5 and follow the lines of the proof of theorem 4.9.)

Exercise 5.9⋆

Prove the following statement, or disprove it by finding a counterexample: CSI-separation statements are monotonic in the context; that is, if $\boldsymbol{c}$ is an assignment to a set of variables $\boldsymbol{C}$, and $\boldsymbol{C} \subset \boldsymbol{C}'$, and $\boldsymbol{c}'$ is an assignment to $\boldsymbol{C}'$ that is consistent with $\boldsymbol{c}$, then if $\boldsymbol{X}$ and $\boldsymbol{Y}$ are CSI-separated given $\boldsymbol{c}$, they are also CSI-separated given $\boldsymbol{c}'$.

Exercise 5.10⋆⋆

Prove that the problem of determining CSI-separation given a set of variables is $\mathcal{NP}$-complete. More precisely, we define the following decision problem:

> Each instance is a graph $\mathcal{G}$, where some of the variables have tree-structured CPDs of known structure, and a query $X, Y, \boldsymbol{Z}$. An instance is in the language if there is a $\boldsymbol{z} \in \mathit{Val}(\boldsymbol{Z})$ such that X and Y are not CSI-separated given $\boldsymbol{z}$.

Show that this problem is $\mathcal{NP}$-complete.

a. Show that this problem is in $\mathcal{NP}$.
b. Provide a reduction to this problem from 3-SAT. Hint: The set $\boldsymbol{Z}$ corresponds to the propositional variables, and an assignment $\boldsymbol{z}$ to some particular truth assignment. Define a variable Y_i for the ith clause in the 3-SAT formula. Provide a construction that contains an active path from Y_i to Y_{i+1} in a context $\boldsymbol{z}$ iff the assignment $\boldsymbol{z}$ satisfies the ith clause.

Exercise 5.11

In this exercise, we consider the context-specific independencies arising in ICI models.

a. Prove proposition 5.3.
b. What context-specific independencies arise if $P(Y \mid X_1, \ldots, X_k)$ is a noisy-and (analogous to figure 5.9, but where the aggregation function is a deterministic AND)?
c. What context-specific independencies arise if $P(Y \mid X_1, \ldots, X_k)$ is a noisy-max (where the aggregation function is a deterministic max), where we assume that Y and the X_i's take ordinal values $v_1, \ldots, v_l$?

Exercise 5.12⋆

In this exercise, we study intercausal reasoning in noisy-or models. Consider the v-structure Bayesian network: $X \rightarrow Z \leftarrow Y$, where X, Y, and Z are all binary random variables. Assume that the CPD for Z is a noisy-or, as in equation (5.2). Show that this network must satisfy the *explaining away* property:

explaining away

$$P(x^1 \mid z^1) \geq P(x^1 \mid y^1, z^1).$$

Exercise 5.13

Consider a BN2O network $\mathcal{B}$, as described in box 5.C, and assume we are given a negative observation $F_1 = F_1^0$. Show that the the posterior distribution $P_{\mathcal{B}}(\cdot \mid F_1 = F_1^0$ can be encoded using another BN2O network $\mathcal{B}'$ that has the same structure as $\mathcal{B}$, except that F_1 is omitted from the network. Specify the parameters of $\mathcal{B}'$ in terms of the parameters of $\mathcal{B}$.

Exercise 5.14⋆

Consider a BN2O network $\mathcal{B}$, as described in box 5.C, and the task of medical diagnosis: Computing the posterior probability in some set of diseases given evidence concerning some of the findings. However, we are only interested in computing the probability of a particular subset of the diseases, so that we wish (for reasons of computational efficiency) to remove from the network those disease variables that are not of interest at the moment.

a. Begin by considering a particular variable F_i, and assume (without loss of generality) that the parents of F_i are $D_1, \ldots, D_k$ and that we wish to maintain only the parents $D_1, \ldots, D_\ell$ for $\ell < k$. Show how we can construct a new noisy-or CPD for F_i that preserves the correct joint distribution over $D_1, \ldots, D_\ell, F_i$.

b. We now remove some fixed set of disease variables $\mathcal{D}$ from the network, executing this pruning procedure for all the finding variables F_i, removing all parents $D_j \in \mathcal{D}$. Is this transformation exact? In other words, if we compute the posterior probability over some variable $D_i \notin \mathcal{D}$, will we get the correct posterior probability (relative to our original model)? Justify your answer.

Exercise 5.15⋆

Consider the symmetric ICI model, as defined in definition 5.13. Show that, if we allow the domain of the intermediate variables Z_i to be arbitrarily large, we can represent any CPD $P(Y \mid X_1, \ldots, X_k)$ using this type of model.

Exercise 5.16

a. Consider a naive Markov model over the multivalued variables $\boldsymbol{X} = \{X_1, \ldots, X_k\}$ and $\boldsymbol{Y} = \{Y\}$, with the pairwise potentials defined via the following log-linear model

$$\phi_i(x_i^l, y^m) = \exp\left\{w_i^{lm} \mathbf{1}\{X_i = x_i^l, Y = y^m\}\right\}.$$

Again, we have a single-node potential $\phi_0(y^m) = \exp\{w_0^m \mathbf{1}\{Y = y^m\}\}$. Show that the distribution $P(Y \mid X_1, \ldots, X_k)$ defined by this model when viewed as a CRF is equivalent to the multinomial logistic CPD of equation (5.4).

b. Determine the appropriate form of CPD for which this result holds for a CRF defined in terms of a general log-linear model, where features are not necessarily pairwise.

6 *Template-Based Representations*

6.1 Introduction

A probabilistic graphical model (whether a Bayesian network or a Markov network) specifies a joint distribution over a fixed set $\mathcal{X}$ of random variables. This fixed distribution is then used in a variety of different situations. For example, a network for medical diagnosis can be applied to multiple patients, each with different symptoms and diseases. However, in this example, the different situations to which the network is applied all share the same general structure — all patients can be described by the same set of attributes, only the attributes' values differ across patients. We call this type of model *variable-based*, since the focus of the representation is a set of random variables.

In many domains, however, the probabilistic model relates to a much more complex space than can be encoded as a fixed set of variables. In a temporal setting, we wish to represent distributions over systems whose state changes over time. For example, we may be monitoring a patient in an intensive care unit. In this setting, we obtain sensor readings at regular intervals — heart rate, blood pressure, EKG — and are interested in tracking the patient's state over time. As another example, we may be interested in tracking a robot's location as it moves in the world and gathers observations. Here, we want a single model to apply to trajectories of different lengths, or perhaps even infinite trajectories.

An even more complex setting arises in our Genetics example; here, each pedigree (family tree) consists of an entire set of individuals, all with their own properties. Our probabilistic model should encode a joint distribution over the properties of all of the family members. Clearly, we cannot define a single variable-based model that applies universally to this application: each family has a different family tree; the networks that represent the genetic inheritance process within the tree have different random variables, and different connectivities. Yet the mechanism by which genes are transmitted from parent to child is identical both for different individuals within a pedigree and across different pedigrees.

In both of these examples, and in many others, we might hope to construct a single, compact model that provides a *template* for an entire class of distributions from the same type: trajectories of different lengths, or different pedigrees. In this chapter, we define representations that allow us to define distributions over richly structured spaces, consisting of multiple objects, interrelated in a variety of ways. These template-based representations have been used in two main settings. The first is temporal modeling, where the language of *dynamic Bayesian networks* allows us to construct a single compact model that captures the properties of the system dy-

namics, and to produce distributions over different trajectories. The second involves domains such as the Genetics example, where we have multiple objects that are somehow related to each other. Here, various languages have been proposed that allow us to produce distributions over different worlds, each with its own set of individuals and set of relations between them.

Once we consider higher-level representations that allow us to model objects, relations, and probabilistic statements about those entities, we open the door to very rich and expressive languages and to queries about concepts that are not even within the scope of a variable-based framework. For example, in the Genetics example, our space consists of multiple people with different types of relationships such as *Mother*, *Father-of*, and perhaps *Married*. In this type probability space, we can also express uncertainty about the identity of Michael's father, or how many children Great-aunt Ethel had. Thus, we may wish to construct a probability distribution over a space consisting of distinct pedigree structures, which may even contain a varying set of objects. As we will see, this richer modeling language will allow us both to answer new types of queries, and to provide more informed answers to "traditional" queries.

6.2 Temporal Models

Our focus in this section is on modeling dynamic settings, where we are interested in reasoning about the state of the world as it evolves over time. We can model such settings in terms of a

system state

system state, whose value at time t is a snapshot of the relevant attributes (hidden or observed) of the system at time t. We assume that the system state is represented, as usual, as an assignment of values to some set of random variables $\mathcal{X}$. We use $X_i^{(t)}$ to represent the instantiation of the variable X_i at time t. Note that X_i itself is no longer a variable that takes a value; rather, it is a

template variable

template variable. This template is instantiated at different points in time t, and each $X_i^{(t)}$ is a variable that takes a value in $Val(X_i)$. For a set of variables $\boldsymbol{X} \subseteq \mathcal{X}$, we use $\boldsymbol{X}^{(t_1:t_2)}$ $(t_1 < t_2)$ to denote the set of variables $\{\boldsymbol{X}^{(t)} : t \in [t_1, t_2]\}$. As usual, we use the notation $\boldsymbol{x}^{(t:t')}$ for an assignment of values to this set of variables.

trajectory

Each "possible world" in our probability space is now a *trajectory*: an assignment of values to each variable $X_i^{(t)}$ for each relevant time t. Our goal therefore is to represent a joint distribution over such trajectories. Clearly, the space of possible trajectories is a very complex probability space, so representing such a distribution can be very difficult. We therefore make a series of simplifying assumptions that help make this representational problem more tractable.

Example 6.1 *Consider a vehicle localization task, where a moving car tries to track its current location using the data obtained from a, possibly faulty, sensor. The system state can be encoded (very simply) using the: Location — the car's current location, Velocity — the car's current velocity, Weather — the current weather, Failure — the failure status of the sensor, and Obs — the current observation. We have one such set of variables for every point t. A joint probability distribution over all of these sets defines a probability distribution over trajectories of the car. Using this distribution, we can answer a variety queries, such as: Given a sequence of observations about the car, where is it now? Where is it likely to be in ten minutes? Did it stop at the red light?* ■

6.2.1 Basic Assumptions

time slice

Our first simplification is to discretize the timeline into a set of *time slices*: measurements of the system state taken at intervals that are regularly spaced with a predetermined time granularity Δ. Thus, we can now restrict our set of random variables to $\mathcal{X}^{(0)}, \mathcal{X}^{(1)}, \ldots$, where $\mathcal{X}^{(t)}$ are the ground random variables that represent the system state at time $t \cdot \Delta$. For example, in the patient monitoring example, we might be interested in monitoring the patient's state every second, so that $\Delta = 1$*sec.* This assumption simplifies our problem from representing distributions over a continuum of random variables to representing distributions over countably many random variables, sampled at discrete intervals.

Consider a distribution over trajectories sampled over a prefix of time $t = 0, \ldots, T$ — $P(\mathcal{X}^{(0)}, \mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(T)})$, often abbreviated $P(\mathcal{X}^{(0:T)})$. We can reparameterize the distribution using the chain rule for probabilities, in a direction consistent with time:

$$P(\mathcal{X}^{(0:T)}) = P(\mathcal{X}^{(0)}) \prod_{t=0}^{T-1} P(\mathcal{X}^{(t+1)} \mid \mathcal{X}^{(0:t)}).$$

Thus, the distribution over trajectories is the product of conditional distributions, for the variables in each time slice given the preceding ones. We can considerably simplify this formulation by using our usual tool — conditional independence assumptions. One very natural approach is to assume that the future is conditionally independent of the past given the present:

Definition 6.1

Markov assumption

We say that a dynamic system over the template variables $\mathcal{X}$ satisfies the Markov assumption *if, for all $t \geq 0$,*

$$(\mathcal{X}^{(t+1)} \perp \mathcal{X}^{(0:(t-1))} \mid \mathcal{X}^{(t)}).$$

Markovian system

Such systems are called Markovian. ■

The Markov assumptions states that the variables in $\mathcal{X}^{(t+1)}$ cannot depend directly on variables in $\mathcal{X}^{(t')}$ for $t' < t$. If we were to draw our dependency model as an (infinite) Bayesian network, the Markov assumption would correspond to the constraint on the graph that there are no edges into $\mathcal{X}^{(t+1)}$ from variables in time slices $t - 1$ or earlier. Like many other conditional independence assumptions, the Markov assumption allows us to define a more compact representation of the distribution:

$$P(\mathcal{X}^{(0)}, \mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(T)}) = P(\mathcal{X}^{(0)}) \prod_{t=0}^{T-1} P(\mathcal{X}^{(t+1)} \mid \mathcal{X}^{(t)}). \tag{6.1}$$

Like any conditional independence assumption, the Markov assumption may or may not be reasonable in a particular setting.

Example 6.2

Let us return to the setting of example 6.1, but assume we had, instead, selected $\mathcal{X} = \{L, O\}$, where L is the location of the object and O its observed location. At first glance, we might be tempted to make the Markov assumption in this setting: after all, the location at time $t + 1$ does not appear to depend directly on the location at time $t - 1$. However, assuming the object's motion is coherent, the location at time $t + 1$ is not independent of the previous locations given only the location at

time t, because the previous locations give us information about the object's direction of motion and speed. By adding Velocity, we make the Markov assumption closer to being satisfied. If, however, the driver is more likely to accelerate and decelerate sharply in certain types of weather (say heavy winds), then our V, L model does not satisfy the Markov assumption relative to V; we can, again, make the model more Markovian by adding the Weather variable. Finally, in many cases, a sensor failure at one point is usually accompanied with a sensor failure at nearby time points, rendering nearby Obs variables correlated. By adding all of these variables into our state model, we define a state space whereby the Markov assumption is arguably a reasonable approximation. ■

Philosophically, one might argue whether, given a sufficiently rich description of the world state, the past is independent of the future given the present. However, that question is not central to the use of the Markov assumption in practice. Rather, **we need only consider whether the Markov assumption is a sufficiently reasonable approximation to the dependencies in our distribution. In most cases, if we use a reasonably rich state description, the approximation is quite reasonable.** In other cases, we can also define models that are *semi-Markov*, where the independence assumption is relaxed (see exercise 6.1).

semi-Markov

Because the process can continue indefinitely, equation (6.1) still leaves us with the task of acquiring an infinite set of conditional distributions, or a very large one, in the case of finite-horizon processes. Therefore, we usually make one last simplifying assumption:

Definition 6.2 stationary dynamical system; transitional model

We say that a Markovian dynamic system is stationary *(also called* time invariant *or* homogeneous*) if $P(\mathcal{X}^{(t+1)} \mid \mathcal{X}^{(t)})$ is the same for all t. In this case, we can represent the process using a* transition model *$P(\mathcal{X}' \mid \mathcal{X})$, so that, for any $t \geq 0$,*

$$P(\mathcal{X}^{(t+1)} = \xi' \mid \mathcal{X}^{(t)} = \xi) = P(\mathcal{X}' = \xi' \mid \mathcal{X} = \xi).$$ ■

6.2.2 Dynamic Bayesian Networks

The Markov and stationarity assumptions described in the previous section allow us to represent the probability distribution over infinite trajectories very compactly: We need only represent the initial state distribution and the transition model $P(\mathcal{X}' \mid \mathcal{X})$. This transition model is a conditional probability distribution, which we can represent using a conditional Bayesian network, as described in section 5.6.

Example 6.3

Let us return to the setting of example 6.1. Here, we might want to represent the system dynamics using the model shown in figure 6.1a, the current observation depends on the car's location (and the map, which is not explicitly modeled) and on the error status of the sensor. Bad weather makes the sensor more likely to fail. And the car's location depends on the previous position and the velocity. All of the variables are interface variables except for Obs, since we assume that the sensor observation is generated at each time point independently given the other variables. ■

This type of conditional Bayesian network is called a *2-time-slice Bayesian network (2-TBN).*

Definition 6.3 2-TBN; interface variable

A 2-time-slice Bayesian network (2-TBN) *for a process over $\mathcal{X}$ is a conditional Bayesian network over $\mathcal{X}'$ given $\mathcal{X}_I$, where $\mathcal{X}_I \subseteq \mathcal{X}$ is a set of* interface variables. ■

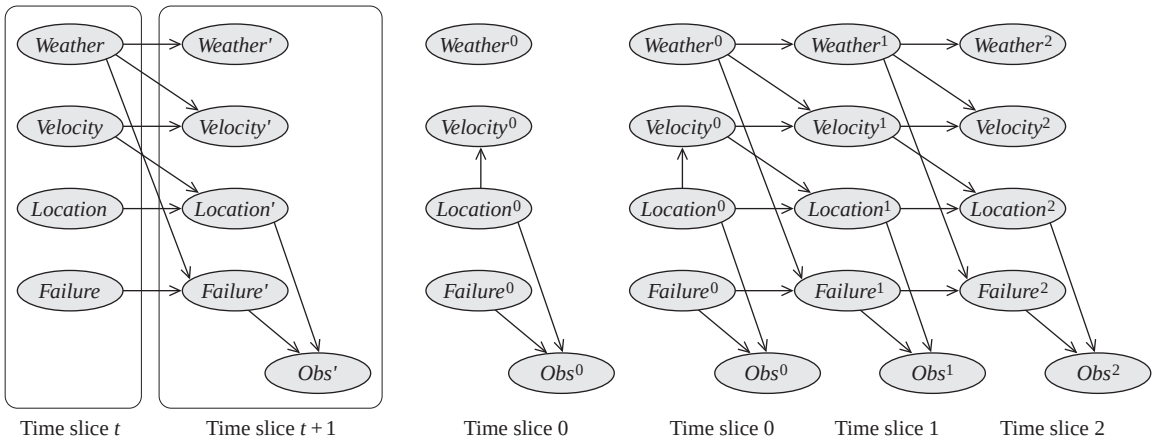


Figure 6.1 A highly simplified DBN for monitoring a vehicle: (a) the 2-TBN; (b) the time 0 network; (c) resulting unrolled DBN over three time slices.

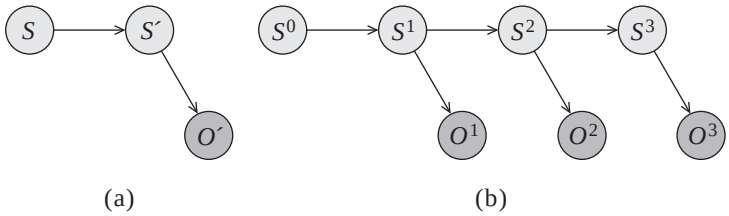


Figure 6.2 HMM as a DBN: (a) The 2-TBN for a generic HMM. (b) The unrolled DBN for four time slices.

As a reminder, in a conditional Bayesian network, only the variables $\mathcal{X}'$ have parents or CPDs. The interface variables $\mathcal{X}_I$ are those variables whose values at time t have a direct effect on the variables at time $t+1$. Thus, only the variables in $\mathcal{X}_I$ can be parents of variables in $\mathcal{X}'$. In our example, all variables except O are in the interface.

Overall, the 2-TBN represents the conditional distribution:

$$P(\mathcal{X}' \mid \mathcal{X}) = P(\mathcal{X}' \mid \mathcal{X}_I) = \prod_{i=1}^{n} P(X_i' \mid \mathrm{Pa}_{X_i'}). \tag{6.2}$$

template factor

For each template variable X_i, the CPD $P(X_i' \mid \mathrm{Pa}_{X_i'})$ is a *template factor*: it will be instantiated multiple times within the model, for multiple variables $X_i^{(t)}$ (and their parents).

hidden Markov model

Perhaps the simplest nontrivial example of a temporal model of this kind is the *hidden Markov model* (see section 6.2.3.1). It has a single state variable S and a single observation variable O. Viewed as a DBN, an HMM has the structure shown in figure 6.2.

Example 6.4

Consider a robot moving around in a grid. Most simply, the robot is the only aspect of the world that is changing, so that the state of the system S is simply the robot's position. Our transition model $P(S' \mid S)$ then represents the probability that, if the robot is in some state (position) s, it will move to another state s'. Our task is to keep track of the robot's location, using a noisy sensor (for example, a sonar) whose value depends on the robot's location. The observation model $P(O \mid S)$ tells us the probability of making a particular sensor reading o given that the robot's current position is s. (See box 5.E for more details on the state transition and observation models in a real robot localization task.) ■

inter-time-slice edge

intra-time-slice edge

In a 2-TBN, some of the edges are *inter-time-slice edges*, going between time slices, whereas others are *intra-time-slice edges*, connecting variables in the same time slice. Intuitively, our decision of how to relate two variables depends on how tight the coupling is between them. If the effect of one variable on the other is immediate — much shorter than the time granularity in the model — the influence would manifest (roughly) within a time slice. If the effect is slightly longer-term, the influence manifests from one time slice to the next. In our simple examples, the effect on the observations is almost immediate, and hence is modeled as an intra-time-slice edge, whereas other dependencies are inter-time-slice. In other examples, when time slices have a coarser granularity, more effects might be short relative to the length of the time slice, and so we might have other dependencies that are intra-time-slice.

persistence edge

persistent variable

Many of the inter-time-slice edges are of the form $X \rightarrow X'$. Such edges are called *persistence edges*, and they represent the tendency of the variable X (for example, sensor failure) to persist over time with high probability. A variable X for which we have an edge $X \rightarrow X'$ in the 2-TBN is called a *persistent variable*.

Based on the stationarity property, a 2-TBN defines the probability distribution $P(\mathcal{X}^{(t+1)} \mid \mathcal{X}^{(t)})$ for any t. Given a distribution over the initial states, we can *unroll* the network over sequences of any length, to define a Bayesian network that induces a distribution over trajectories of that length. In these networks, all the copies of the variable $X_i^{(t)}$ for $t > 0$ have the same dependency structure and the same CPD. Figure 6.1 demonstrates a transition model, initial state network, and a resulting unrolled DBN, for our car example.

Definition 6.4

dynamic Bayesian network

unrolled Bayesian network

A dynamic Bayesian network (DBN) *is a pair $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, where $\mathcal{B}_0$ is a Bayesian network over $\mathcal{X}^{(0)}$, representing the initial distribution over states, and $\mathcal{B}_{\rightarrow}$ is a 2-TBN for the process. For any desired time span $T \geq 0$, the distribution over $\mathcal{X}^{(0:T)}$ is defined as a* unrolled Bayesian network, *where, for any $i = 1, \ldots, n$:*

- *the structure and CPDs of $X_i^{(0)}$ are the same as those for X_i in $\mathcal{B}_0$,*
- *the structure and CPD of $X_i^{(t)}$ for $t > 0$ are the same as those for X_i' in $\mathcal{B}_{\rightarrow}$.* ■

Thus, we can view a DBN as a compact representation from which we can generate an infinite set of Bayesian networks (one for every $T > 0$).

factorial HMM

coupled HMM

Figure 6.3 shows two useful classes of DBNs that are constructed from HMMs. A *factorial HMM*, on the left, is a DBN whose 2-TBN has the structure of a set of chains $X_i \rightarrow X_i'$ ($i = 1, \ldots, n$), with a single (always) observed variable Y', which is a child of all the variables X_i'. This type of model is very useful in a variety of applications, for example, when several sources of sound are being heard simultaneously through a single microphone. A *coupled HMM*,

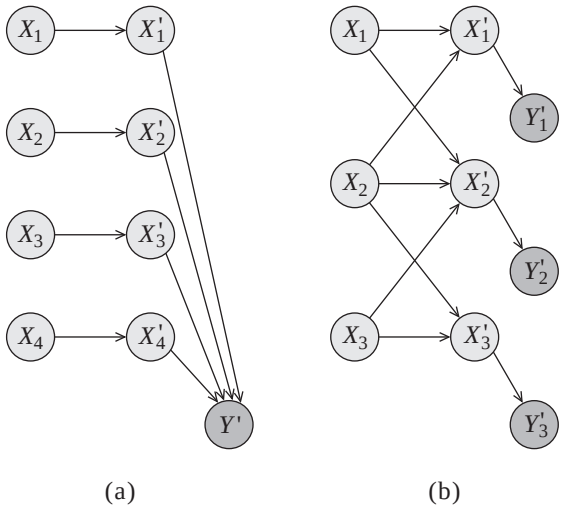


Figure 6.3 Two classes of DBNs constructed from HMMs: (a) A factorial HMM. (b) A coupled HMM.

on the right, is also constructed from a set of chains X_i, but now, each chain is an HMM with its private own observation variable Y_i. The chains now interact directly via their state variables, with each chain affecting its adjacent chains. These models are also useful in a variety of applications. For example, consider monitoring the temperature in a building over time (for example, for fire alarms). Here, X_i might be the true (hidden) state of the ith room, and Y_i the value returned by the room's own temperature sensor. In this case, we would expect to have interactions between the hidden states of adjacent rooms.

In DBNs, it is often the case that our observation pattern is constant over time. That is, we can partition the variables $\mathcal{X}$ into disjoint subsets $\boldsymbol{X}$ and $\boldsymbol{O}$, such that the variables in $\boldsymbol{X}^{(t)}$ are always hidden and $\boldsymbol{O}^{(t)}$ are always observed. For uniformity of presentation, we generally make this assumption; however, the algorithms we present also apply to the more general case.

A DBN can enable fairly sophisticated reasoning patterns.

Example 6.5

By explicitly encoding sensor failure, we allow the agent to reach the conclusion that the sensor has failed. Thus, for example, if we suddenly get a reading that tells us something unexpected, for example, the car is suddenly 15 feet to the left of where we thought it was 0.1 seconds ago, then in addition to considering the option that the car has suddenly teleported, we will also consider the option that the sensor has simply failed. Note that the model only considers options that are built into it. If we had no "sensor failure" variable, and had the sensor reading depend only on the current location, then the different sensor readings would be independent given the car's trajectory, so that there would be no way to explain correlations of unexpected sensor readings except via the trajectory. Similarly, if the system knows (perhaps from a weather report or from prior observations) that it is raining, it will expect the sensor to be less accurate, and therefore be less likely to believe that the car is out of position. ■

Box 6.A — Case Study: HMMs and Phylo-HMMs for Gene Finding. *HMMs are a primary tool in algorithms that extract information from biological sequences. Key applications (among many) include: modeling families of related proteins within and between organisms, finding genes in DNA sequences, and modeling the correlation structure of the genetic variation between individuals in a population. We describe the second of these applications, as an illustration of the methods used.*

The DNA of an organism is composed of two paired helical strands consisting of a long sequence of nucleotides, each of which can take on one of four values — A,C,G,T; in the double helix structure, A is paired with T and C is paired with G, to form a base pair. The DNA sequence consists of multiple regions that can play different roles. Some regions are genes, whose DNA is transcribed into mRNA, some of which is subsequently translated into protein. In the translation process, triplets of base pairs, known as codons, are converted into amino acids. There are $4^3 = 64$ *different codons, but only 20 different amino acids, so that the code is redundant. Not all transcribed regions are necessarily translated. Genes can contain exons, which are translated, and introns, which are spliced out during the translation process. The DNA thus consists of multiple genes that are separated by intergenic regions; and genes are themselves structured, consisting of multiple exons separated by introns. The sequences in each of these regions is characterized by certain statistical properties; for example, a region that produces protein has a very regular codon structure, where the codon triplets exhibit the usage statistics of the amino acids they produce. Moreover, boundaries between these regions are also often demarcated with sequence elements that help the cell determine where transcription should begin and end, and where translation ought to begin and end. Nevertheless, the signals in the sequence are not always clear, and therefore identifying the relevant sequence units (genes, exons, and more) is a difficult task.*

HMMs are a critical tool in this analysis. Here, we have a hidden state variable for each base pair, which denotes the type of region to which this base pair belongs. To satisfy the Markov assumption, one generally needs to refine the state space. For example, to capture the codon structure, we generally include different hidden states for the first, second, and third base pairs within a codon. This larger state space allows us to encode the fact that coding regions are sequences of triplets of base pairs, as well as encode the different statistical properties of these three positions. We can further refine the state space to include different statistics for codons in the first exon and in the last exon in the gene, which can exhibit different characteristics than exons in the middle of the gene. The observed state of the HMM naturally includes the base pair itself, with the observation model reflecting the different statistics of the nucleotide composition of the different regions. It can also include other forms of evidence, such as the extent to which measurements of mRNA taken from the cell have suggested that a particular region is transcribed. And, very importantly, it can contain evidence regarding the conservation of a base pair across other species. This last key piece of evidence derives from the fact that base pairs that play a functional role in the cell, such as those that code for protein, are much more likely to be conserved across related species; base pairs that are nonfunctional, such as most of those in the intergenic regions, evolve much more rapidly, since they are not subject to selective pressure. Thus, we can use conservation as evidence regarding the role of a particular base pair.

phylogenetic HMM

One way of incorporating the evolutionary model more explicitly into the model is via a phylogenetic HMM *(of which we now present a simplified version). Here, we encode not a single DNA sequence, but the sequences of an entire phylogeny (or evolutionary tree) of related species. We*

let $X_{k,i}$ be the ith nucleotide for species s_k. We also introduce a species-independent variable Y_i denoting the functional role of the ith base pair (intergenic, intron, and so on). The base pair $X_{k,i}$ will depend on the corresponding base pair $X_{\ell,i}$ where s_ℓ is the ancestral species from which s_k evolved. The parameters of this dependency will depend on the evolutionary distance between s_k and s_l (the extent to which s_k has diverged) and on the rate at which a base pair playing a particular role evolves. For example, as we mentioned, a base pair in an intergenic region generally evolves much faster than one in a coding region. Moreover, the base pair in the third position in a codon also often evolves more rapidly, since this position encodes most of the redundancy between codons and amino acids, and so allows evolution without changing the amino acid composition. Thus, overall, we define $X_{k,i}$'s parents in the model to be Y_i (the type of region in which $X_{k,i}$ resides), $X_{k,i-1}$ (the previous nucleotide in species s) and $X_{\ell,i}$ (the ith nucleotide in the parent species s_ℓ). This model captures both the correlations in the functional roles (as in the simple gene finding model) and the fact that evolution of a particular base pair can depend on the adjacent base pairs. This model allows us to combine information from multiple species in order to infer which are the regions that are functional, and to suggest a segmentation of the sequence into its constituent units.

Overall, the structure of this model is roughly a set of trees connected by chains: For each i we have a tree over the variables $\{X_{k,i}\}_{s_k}$, where the structure of the tree is that of the evolutionary tree; in addition, all of the $X_{k,i}$ are connected by chains to $X_{k,i+1}$; finally, we also have the variables Y_i, which also form a chain and are parents of all of the $X_{k,i}$. Unfortunately, the structure of this model is highly intractable for inference, and requires the use of approximate inference methods; see exercise 11.29.

6.2.3 State-Observation Models

state-observation model

An alternative way of thinking about a temporal process is as a *state-observation model*. In a state-observation model, we view the system as evolving naturally on its own, with our observations of it occurring in a separate process. This view separates out the system dynamics from our observation model, allowing us to consider each of them separately. It is particularly useful when our observations are obtained from a (usually noisy) sensor, so that it makes sense to model separately the dynamics of the system and our ability to sense it.

A state-observation model utilizes two independence assumptions: that the state variables evolve in a Markovian way, so that

$$(\boldsymbol{X}^{(t+1)} \perp \boldsymbol{X}^{(0:(t-1))} \mid \boldsymbol{X}^{(t)});$$

and that the observation variables at time t are conditionally independent of the entire state sequence given the state variables at time t:

$$(\boldsymbol{O}^{(t)} \perp \boldsymbol{X}^{(0:(t-1))}, \boldsymbol{X}^{(t+1:\infty)} \mid \boldsymbol{X}^{(t)}).$$

transition model

observation model

We now view our probabilistic model as consisting of two components: the *transition model*, $P(\boldsymbol{X}' \mid \boldsymbol{X})$, and the *observation model*, $P(\boldsymbol{O} \mid \boldsymbol{X})$.

From the perspective of DBNs, this type of model corresponds to a 2-TBN structure where the observation variables $\boldsymbol{O}'$ are all leaves, and have parents only in $\boldsymbol{X}'$. This type of situation arises

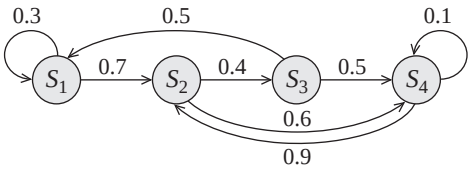


Figure 6.4 A simple 4-state HMM

quite naturally in a variety of real-world systems, where we do not have direct observations of the system state, but only access to a set of (generally noisy) sensors that depend on the state. The sensor observations do not directly effect the system dynamics, and therefore are naturally viewed as leaves.

We note that we can convert any 2-TBN to a state-observation representation as follows: For any observed variable Y that does not already satisfy the structural restrictions, we introduce a new variable $\tilde{Y}$ whose only parent is Y, and that is deterministically equal to Y. Then, we view Y as being hidden, and we interpret our observations of Y as observations on $\tilde{Y}$. In effect, we construct $\tilde{Y}$ to be a perfectly reliable sensor of Y. Note, however, that, while the resulting transformed network is probabilistically equivalent to the original, it does obscure structural independence properties of the network (for example, various independencies given that Y is observed), which are now apparent only if we account for the deterministic dependency between Y and $\tilde{Y}$.

It is often convenient to view a temporal system as a state-observation model, both because it lends a certain uniformity of notation to a range of different systems, and because the state transition and observation models often induce different computational operations, and it is convenient to consider them separately.

State-observation models encompass two important architectures that have been used in a wide variety of applications: hidden Markov models and linear dynamical systems. We now briefly describe each of them.

6.2.3.1 Hidden Markov Models

hidden Markov model

A *hidden Markov model*, which we illustrated in figure 6.2, is the simplest example of a state-observation model. While an HMM is a special case of a simple DBN, it is often used to encode structure that is left implicit in the DBN representation. Specifically, the transition model $P(S' \mid S)$ in an HMM is often assumed to be sparse, with many of the possible transitions having zero probability. In such cases, HMMs are often represented using a different graphical notation, which visualizes this sparse transition model. In this representation, the HMM transition model is encoded using a directed (generally cyclic) graph, whose nodes represent the *different states* of the system, that is, the values in $Val(S)$. We have a directed arc from s to s' if it is possible to transition from s to s' — that is, $P(s' \mid s) > 0$. The edge from s to s' can also be annotated with its associated transition probability $P(s' \mid s)$.

Example 6.6

Consider an HMM with a state variable S that takes 4 values s_1, s_2, s_3, s_4, and with a transition

model:

	s_1	s_2	s_3	s_4
s_1	0.3	0.7	0	0
s_2	0	0	0.4	0.6
s_3	0.5	0	0	0.5
s_4	0	0.9	0	0.1

where the rows correspond to states s and the columns to successor states s' (so that each row must sum to 1). The transition graph for this model is shown in figure 6.4. ■

Importantly, the transition graph for an HMM is a very different entity from the graph encoding a graphical model. Here, the nodes in the graph are *state*, or possible values of the state variable; the directed edges represent possible transitions between the states, or entries in the CPD that have nonzero probability. Thus, the weights of the edges leaving a node must sum to 1. This graph representation can also be viewed as *probabilistic finite-state automaton*. Note that this graph-based representation does not encode the observation model of the HMM. In some cases, the observation model is deterministic, in that, for each s, there is a single observation o for which $P(o \mid s) = 1$ (although the same observation can arise in multiple states). In this case, the observation is often annotated on the node associated with the state.

probabilistic finite-state automaton

It turns out that HMMs, despite their simplicity, are an extremely useful architecture. For example, they are the primary architecture for speech recognition systems (see box 6.B) and for many problems related to analysis of biological sequences (see, for example, box 6.A). Moreover, these applications and others have inspired a variety of valuable generalizations of the basic HMM framework (see, for example, Exercises 6.2–6.5).

Box 6.B — Case Study: HMMs for Speech Recognition. *Hidden Markov models are currently the key technology in all* speech- recognition *systems. The HMM for speech is composed of three distinct layers: the* language model*, which generates sentences as sequences of words; the* word model*, where words are described as a sequence of phonemes; and the* acoustic model*, which shows the progression of the acoustic signal through a phoneme.*

speech recognition

language model

At the highest level, the language model represents a probability distribution over sequences of words in the language. Most simply, one can use a bigram model*, which is a Markov model over words, defined via a probability distribution $P(W_i \mid W_{i-1})$ for each position i in the sentence. We can view this model as a Markov model where the state is the current word in the sequence. (Note that this model does not take into account the actual position in the sentence, so that $P(W_i \mid W_{i-1})$ is the same for all $i > 1$.) A somewhat richer model is the* trigram model*, where the states correspond to pairs of successive words in the sentence, so that our model defines a probability distribution $P(W_i \mid W_{i-1}, W_{i-2})$. Both of these distributions define a ridiculously naive model of language, since they only capture local correlations between neighboring words, with no attempt at modeling global coherence. Nevertheless, these models prove surprisingly hard to beat, probably because they are quite easy to train robustly from the (virtually unlimited amounts of) available training data, without the need for any manual labeling.*

The middle layer describes the composition of individual words in terms of phonemes *— basic phonetic units corresponding to distinct sounds. These units vary not just on the basic sound uttered*

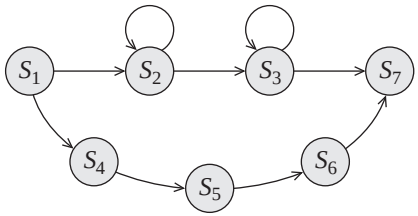


Figure 6.B.1 — A phoneme-level HMM for a fairly complex phoneme.

("p" versus "b"), but also on whether the sound is breathy, aspirated, nasalized, and more. There is an international agreement on an International Phonetic Alphabet, *which contains about 100 phonemes. Each word is modeled as a sequence of phonemes. Of course, a word can have multiple different pronunciations, in which case it corresponds to several such sequences.*

At the acoustic level, the acoustic signal is segmented into short time frames (around 10–25ms). A given phoneme lasts over a sequence of these partitions. The phoneme is also not homogenous. Different acoustics are associated with its beginning, its middle, and its end. We thus create an HMM for each phoneme, with its hidden variable corresponding to stages in the expression of the phoneme. HMMs for phonemes are usually quite simple, with three states, but can get more complicated, as in figure 6.B.1. The observation represents some set of features extracted from the acoustic signal; the feature vector is generally either discretized into a set of bins or treated as a continuous observation with a Gaussian or mixture of Gaussian distribution.

hierarchical HMM

Given these three models, we can put them all together to form a single huge hierarchical HMM *that defines a joint probability distribution over a state space encompassing words, phonemes, and basic acoustic units. In a bigram model, the states in the space have the form (w, i, j), where w is the current word, i is a phoneme within that word, and j is an acoustic position within that phoneme. The sequence of states corresponding to a word w is governed by a word-HMM representing the distribution over pronunciations of w. This word-HMM has a start-state and an end-state. When we exit from the end-state of the HMM for one word w, we branch to the start-state of another word w' with probability $P(w' \mid w)$. Each sequence is thus a trajectory through acoustic HMMs of individual phonemes, transitioning from the end-state of one phoneme's HMM to the start-state of the next phoneme's HMM.*

A hierarchical HMM can be converted into a DBN, whose variables represent the states of the different levels of the hierarchy (word, phoneme, and intraphone state), along with some auxiliary variables to capture the "control architecture" of the hierarchical HMM; see exercise 6.5. The DBN formulation has the benefit of being a much more flexible framework in which to introduce extensions to the model. One extension addresses the coarticulation problem, *where the proximity of one phoneme changes the pronunciation of another. Thus, for example, the last phoneme in the word "don't" sounds very different if the word after it is "go" or if it is "you." Similarly, we often pronounce "going to" as "gonna." The reason for coarticulation is the fact that a person's speech articulators (such as the tongue or the lips) have some inertia and therefore do not always move all the way to where they are supposed to be. Within the DBN framework, we can easily solve this problem by introducing a dependency of the pronunciation model for one phoneme on the value of the preceding phoneme and the next one. Note that "previous" and "next" need to be interpreted*

with care: These are not the values of the phoneme variable at the previous or next states in the HMM, which are generally exactly the same as the current phoneme; rather, these are the values of the variables prior to the previous phoneme change, and following the next phoneme change. This extension gives rise to a non-Markovian model, which is more easily represented as a structured graphical model. Another extension that is facilitated by a DBN structure is the introduction of variables that denote states at which a transition between phonemes occurs. These variables can then be connected to observations that are indicative of such a transition, such as a significant change in the spectrum. Such features can also be incorporated into the standard HMM model, but it is difficult to restrict the model so that these features affect only our beliefs in phoneme transitions.

Finally, graphical model structure has also been used to model the structure in the Gaussian distribution over the acoustic signal features given the state. Here, two "traditional" models are: a diagonal Gaussian over the features, a model that generally loses many important correlations between the features; and a full covariance Gaussian, a model that requires many parameters and is hard to estimate from data (especially since the Gaussian is different for every state in the HMM). As we discuss in chapter 7, graphical models provide an intermediate point along the spectrum: we can use a Gaussian graphical model that captures the most important of the correlations between the features. The structure of this Gaussian can be learned from data, allowing a flexible trade-off to be determined based on the available data.

6.2.3.2 Linear Dynamical Systems

linear dynamical system

Kaman filter

Another very useful temporal model is a *linear dynamical system*, which represents a system of one or more real-valued variables that evolve linearly over time, with some Gaussian noise. Such systems are also often called *Kalman filters*, after the algorithm used to perform tracking. **A linear dynamical system can be viewed as a dynamic Bayesian network where the variables are all continuous and all of the dependencies are linear Gaussian.**

Linear dynamical systems are often used to model the dynamics of moving objects and to track their current positions given noisy measurements. (See also box 15.A.)

Example 6.7

Recall example 5.20, where we have a (vector) variable X denoting a vehicle's current position (in each relevant dimension) and a variable V denoting its velocity (also in each dimension). As we discussed earlier, a first level approximation may be a model where $P(X' \mid X, V) = \mathcal{N}\left(X + V\Delta; \sigma_X^2\right)$ and $P(V' \mid V) = \mathcal{N}\left(V; \sigma_V^2\right)$ (where Δ, as before, is the length of our time slice). The observation — for example, a GPS signal measured from the car — is a noisy Gaussian measurement of X. ■

These systems and their extensions are at the heart of most target tracking systems, for example, tracking airplanes in an air traffic control system using radar data.

Traditionally, linear dynamical systems have not been viewed from the perspective of factorized representations of the distribution. They are traditionally represented as a state-observation model, where the state and observation are both vector-valued random variables, and the transition and observation models are encoded using matrices. More precisely, the model is generally

defined via the following set of equations:

$$P(\boldsymbol{X}^{(t)} \mid \boldsymbol{X}^{(t-1)}) = \mathcal{N}\left(A\boldsymbol{X}^{(t-1)}; Q\right), \tag{6.3}$$

$$P(O^{(t)} \mid \boldsymbol{X}^{(t)}) = \mathcal{N}\left(H\boldsymbol{X}^{(t)}; R\right), \tag{6.4}$$

where: $\boldsymbol{X}$ is an n-vector of state variables, O is an m-vector of observation variables, A is an $n \times n$ matrix defining the linear transition model, Q is an $n \times n$ matrix defining the Gaussian noise associated with the system dynamics, H is an $n \times m$ matrix defining the linear observation model, and R is an $m \times m$ matrix defining the Gaussian noise associated with the observations. This type of model encodes independence structure implicitly, in the parameters of the matrices (see exercise 7.5).

There are many interesting generalizations of the basic linear dynamical system, which can also be placed within the DBN framework. For example, a nonlinear variant, often called an *extended Kalman filter*, is a system where the state and observation variables are still vectors of real numbers, but where the state transition and observation models can be nonlinear functions rather than linear matrix multiplications as in equation (6.3) and equation (6.4). Specifically, we usually write:

extended Kalman filter

$$P(\boldsymbol{X}^{(t)} \mid \boldsymbol{X}^{(t-1)}) = f(\boldsymbol{X}^{(t-1)}, \boldsymbol{U}^{(t-1)})$$
$$P(O^{(t)} \mid \boldsymbol{X}^{(t)}) = g(\boldsymbol{X}^{(t)}, \boldsymbol{W}^{(t)}),$$

where f and g are deterministic nonlinear functions, and $\boldsymbol{U}^{(t)}, \boldsymbol{W}^{(t)}$ are Gaussian random variables that explicitly encode the noise in the transition and observation models, respectively. In other words, rather than model the system in terms of stochastic CPDs, we use an equivalent representation that partitions the model into a deterministic function and a noise component.

Another interesting class of models are systems where the continuous dynamics are linear, but that also include discrete variables. For example, in our tracking example, we might introduce a discrete variable that denotes the driver's target lane in the freeway: the driver can stay in her current lane, or she can switch to a lane on the right or a lane on the left. Each of these discrete settings leads to different dynamics for the vehicle velocity, in both the lateral (across the road) and frontal velocity.

switching linear dynamical system

Systems that model such phenomena are called *switching linear dynamical system (SLDS)*. In such models, we system can switch between a set of discrete *modes*. While within a fixed mode, the system evolves using standard linear (or nonlinear) Gaussian dynamics, but the equations governing the dynamics are different in different modes. We can view this type of system as a DBN by including a discrete variable D that encodes the mode, where $\mathrm{Pa}_{D'} = \{D\}$, and allowing D to be the parent of (some of) the continuous variables in the model, so that they use a conditional linear Gaussian CPDs.

6.3 Template Variables and Template Factors

Having seen one concrete example of a template-based model, we now describe a more general framework that provides the fundamental building blocks for a template-based model. This framework provides a more formal perspective on the temporal models of the previous section, and a sound foundation for the richer models in the remaining sections of this chapter.

Template Attributes The key concept in the definition of the models we describe in this chapter is that of a *template* that is instantiated many times within the model. A template for a random variable allows us to encode models that contain multiple random variables with the same value space and the same semantics. For example, we can have a *Blood-Type* template, which has a particular value space (say, *A*, *B*, *AB*, or *O*) and is reused for multiple individuals within a pedigree. That is, when reasoning about a pedigree, as in box 3.B, we would want to have multiple instances of blood-type variables, such as *Blood-Type*(*Bart*) or *Blood-Type*(*Homer*). We use the word *attribute* or *template variable* to distinguish a template, such as *Blood-Type*, from a specific random variable, such as *Blood-Type*(*Bart*).

attribute

template variable

In a very different type of example, we can have a template attribute *Location*, which can be instantiated to produce random variables *Location*(t) for a set of different time points t. This type of model allows us to represent a joint distribution over a vehicle's position at different points in time, as in the previous section.

In these example, the template was a property of a single object — a person. More broadly, attributes may be properties of entire tuples of objects. For example, a student's grade in a course is associated with a student-course pair; a person's opinion of a book is associated with the person-book pair; the affinity between a regulatory protein in the cell and one of its gene targets is also associated with the pair. More specifically, in our Student example, we want to have a *Grade* template, which we can instantiate for different (student,course) pairs s, c to produce multiple random variables *Grade*(s, c), such as *Grade*(*George*, *CS101*).

Because many domains involve heterogeneous objects, such as courses and students, it is convenient to view the world as being composed of a set of *objects*. Most simply, objects can be divided into a set of mutually exclusive and exhaustive *classes* $\mathcal{Q} = \mathsf{Q}_1, \ldots, \mathsf{Q}_k$. In the Student scenario, we might have a Student class and a Course class.

object

class

Attributes have a tuple of *arguments*, each of which is associated with a particular class of objects. This class defines the set of objects that can be used to instantiate the argument in a given domain. For example, in our *Grade* template, we have one argument S that can be instantiated with a "student" object, and another argument C that can be instantiated with a "course" object. Template attributes thus provide us with a "generator" for random variables in a given probability space.

argument

Definition 6.5 *An* attribute A *is a function* $A(U_1, \ldots, U_k)$*, whose* range *is some set* $Val(A)$ *and where each argument* U_i *is a typed* logical variable *associated with a particular class* $\mathsf{Q}[U_i]$*. The tuple* $U_1, \ldots, U_k$ *is called the* argument signature *of the attribute* A*, and denoted* $\alpha(A)$. ■

attribute

logical variable

argument signature

From here on, we assume without loss of generality that each logical variable U_i is uniquely associated with a particular class $\mathsf{Q}[U_i]$; thus, any mention of a logical variable U_i uniquely specifies the class over which it ranges.

For example, the argument signature of the *Grade* attribute would have two logical variables S, C, where S is of class Student and C is of class Course. We note that the classes associated with an attribute's argument signature are not necessarily distinct. For example, we might have a binary-valued *Cited* attribute with argument signature A_1, A_2, where both are of type Article. We assume, for simplicity of presentation, that attribute names are uniquely defined; thus, for example, the attribute denoting the age for a person will be named differently from the attribute denoting the age for a car.

This last example demonstrates a basic concept in this framework: that of a *relation*. A

relation

relation is a property of a tuple of objects, which tells us whether the objects in this tuple satisfy a certain relationship with each other. For example, *Took-Course* is a relation over student-course object pairs s, c, which is true is student s took the course c. As another example *Mother* is a relation between person-person pairs p_1, p_2, which is true if p_1 is the mother of p_2. Relations are not restricted to involving only pairs of objects. For example, we can have a *Go* relation, which takes triples of objects — a person, a source location, and a destination location.

At some level, a relation is simply a binary-valued attribute, as in our *Cited* example. However, this perspective obscures the fundamental property of a relation — that it *relates* a tuple of objects to each other. Thus, when we introduce the notion of probabilistic dependencies that can occur between related objects, the presence or absence of a relation between a pair of objects will play a central role in defining the probabilistic dependency model.

Instantiations Given a set of template attributes, we can instantiate them in different ways, to produce probability spaces with multiple random variables of the same type. For example, we can consider a particular university, with a set of students and a set of courses, and use the notion of a template attribute to define a probability space that contains a random variable *Grade*(s, c) for different (student,course) pairs s, c. The resulting model encodes a joint distribution over the grades of multiple students in multiple courses. Similarly, in a temporal model, we can have the template attribute *Location*(T); we can then select a set of relevant time points and generate a trajectory with specific random variables *Location*(t)

To instantiate a set of template attributes to a particular setting, we need to define a set of objects for each class in our domain. For example, we may want to take a particular set of students and set of courses and define a model that contains a ground random variable *Intelligence*(s) and *SAT*(s) for every student object s, a ground random variable *Difficulty*(c) for every course object c, and ground random variables *Grade*(s, c) and *Satisfaction*(s, c) for every valid pair of (student,course) objects.

More formally, we now show how a set of template attributes can be used to generate an infinite set of probability spaces, each involving instantiations of the template attributes induced by some set of objects. We begin with a simple definition, deferring discussion of some of the more complicated extensions to section 6.6.

Definition 6.6 object skeleton

Let $\mathcal{Q}$ be a set of classes, and $\aleph$ a set of template attributes over $\mathcal{Q}$. An object skeleton κ *specifies a fixed, finite set of objects $\mathcal{O}^\kappa[\mathsf{Q}]$ for every $\mathsf{Q} \in \mathcal{Q}$. We also define*

$$\mathcal{O}^\kappa[U_1, \ldots, U_k] = \mathcal{O}^\kappa[\mathsf{Q}[U_1]] \times \ldots \times \mathcal{O}^\kappa[\mathsf{Q}[U_k]].$$

By default, we define $\Gamma_\kappa[A] = \mathcal{O}^\kappa[\alpha(A)]$ to be the set of possible assignments to the logical variables in the argument signature of A. However, an object skeleton may also specify a subset of legal assignments. $\Gamma_\kappa[A] \subset \mathcal{O}^\kappa[\alpha(A)]$. ■

We can now define the set of instantiations of the attributes:

Student	*Intelligence*	*SAT*
George	*low*	*High*
Alice	*high*	*High*

Course	*Difficulty*
CS101	*high*
Econ101	*low*

Student	Course	*Grade*	*Satisfaction*
George	**CS101**	*C*	*low*
George	**Econ101**	*A*	*high*
Alice	**CS101**	*A*	*low*

Figure 6.5 One possible world for the University **example.** Here, we have two student objects and two course objects. The attributes *Grade* and *Satisfaction* are restricted to three of their possible four legal assignments.

Definition 6.7

ground random variable

Let κ be an object skeleton over $\mathcal{Q}, \aleph$. We define sets of ground random variables:

$$\begin{aligned} \mathcal{X}_\kappa[A] &= \{A(\gamma) \ : \ \gamma \in \Gamma_\kappa[A]\} \\ \mathcal{X}_\kappa[\aleph] &= \cup_{A \in \aleph} \mathcal{X}_\kappa[A]. \end{aligned} \tag{6.5}$$

Note that we are abusing notation here, identifying an assignment $\gamma = \langle U_1 \mapsto u_1, \ldots, U_k \mapsto u_k \rangle$ with the tuple $\langle u_1 \ldots, u_k \rangle$; this abuse of notation is unambiguous in this context due to the ordering of the tuples. ■

The ability to specify a subset of $\mathcal{O}^\kappa[\alpha(A)]$ is useful in eliminating the need to consider random variables that do not really appear in the model. For example, in most cases, not every student takes every course, and so we would not want to include a *Grade* variable for every possible (student, course) pair at our university. See figure 6.5 as an example.

Clearly, the set of random variables is different for different skeletons; hence the model is a template for an infinite set of probability distributions, each spanning a different set of objects that induces a different set of random variables. In a sense, this is similar to the situation we had in DBNs, where the same 2TBN could induce a distribution over different numbers of time slices. Here, however, the variation between the different instantiations of the template is significantly greater.

Our discussion so far makes several important simplifying assumptions. First, we portrayed the skeleton as defining a set of objects for each of the classes. As we discuss in later sections, it can be important to allow the skeleton to provide additional background information about the set of possible worlds, such as some relationships that hold between objects (such as the structure of a family tree). Conversely, we may also want the skeleton to provide less information: In particular, the premise underlying equation (6.5) is that the set of objects is predefined by the skeleton. As we briefly discuss in section 6.6.2, we may also want to deal with settings in which we have uncertainty over the number of objects in the domain. In this case, different possible worlds may have different sets of objects, so that a random variable such as $A(u)$ may be defined in some worlds (those that contain the object u) but not in others. Settings like this pose significant challenges, a discussion of which is outside the scope of this book.

Template Factors The final component in a template-based probabilistic model is one that defines the actual probability distribution over a set of a ground random variables generated from a set of template attributes. Clearly, we want the specification of the model to be defined in a template-based way. Specifically, we would like to take a factor — whether an undirected factor or a CPD — and instantiate it to apply to multiple scopes in the domain. We have already seen one simple example of this notion: in a 2-TBN, we had a template CPD $P(X'_i \mid \mathrm{Pa}_{X'_i})$, which we instantiated to apply to different scopes $X_i^{(t)}, \mathrm{Pa}_{X_i^{(t)}}$, by instantiating any occurrence X'_j to $X_j^{(t)}$, and any occurrence X_j to $X_j^{(t-1)}$. In effect, there we had template variables of the form X_j and X'_j as arguments to the CPD, and we instantiated them in different ways for different time points. We can now generalize this notion by defining a factor with arguments. Recall that a factor ϕ is a function from a tuple of random variables $\boldsymbol{X} = \mathit{Scope}[\phi]$ to the reals; this function returns a number for each assignment $\boldsymbol{x}$ to the variables $\boldsymbol{X}$. We can now define

Definition 6.8 *A* template factor *is a function ξ defined over a tuple of template attributes $A_1, \ldots, A_l$, where each A_j has a range $Val(A)$. It defines a mapping from $Val(A_1) \times \ldots \times Val(A_l)$ to $\mathbb{R}$. Given a tuple of random variables $X_1, \ldots, X_l$, such that $Val(X_j) = Val(A_j)$ for all $j = 1, \ldots, l$, we define $\xi(X_1, \ldots, X_l)$ to be the* instantiated factor *from $\boldsymbol{X}$ to $\mathbb{R}$.* ■

template factor

instantiated factor

In the subsequent sections of this chapter, we use these notions to define various languages for encoding template-based probabilistic models. As we will see, some of these representational frameworks subsume and generalize on the DBN framework defined earlier.

6.4 Directed Probabilistic Models for Object-Relational Domains

Based on the framework described in the previous section, we now describe template-based representation languages that can encode directed probabilistic models.

6.4.1 Plate Models

plate model

We begin our discussion by presenting the *plate model*, the simplest and best-established of the object-relational frameworks. Although restricted in several important ways, the plate modeling framework is perhaps the approach that has been most commonly used in practice, notably for encoding the assumptions made in various learning tasks. This framework also provides an excellent starting point for describing the key ideas of template-based languages and for motivating some of the extensions that have been pursued in richer languages.

In the plate formalism, object types are called *plates*. The fact that multiple objects in the class share the same set of attributes and same probabilistic model is the basis for the use of the term "plate," which suggests a stack of identical objects. We begin with some motivating examples and then describe the formal framework.

6.4.1.1 Examples

Example 6.8 *The simplest example of a plate model, shown in figure 6.6, describes multiple random variables generated from the same distribution. In this case, we have a set of random variables $X(d)$ ($d \in \mathcal{D}$)*

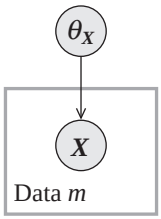


Figure 6.6 Plate model for a set of coin tosses sampled from a single coin

that all have the same domain $Val(X)$ and are sampled from the same distribution. In a plate representation, we encode the fact that these variables are all generated from the same template by drawing only a single node $X(d)$ and enclosing it in a box denoting that d ranges over $\mathcal{D}$, so that we know that the box represents an entire "stack" of these identically distributed variables. This box is called a plate*, with the analogy that it represents a stack of identical plates.*

plate

In many cases, we want to explicitly encode the fact that these variables have an identical distribution. We therefore often explicitly add to the model the variable θ_X, which denotes the parameterization of the CPD from which the variables $X(d)$ are sampled. Most simply, if the X's are coin tosses of a single (possibly biased) coin, θ_X would take on values in the range $[0, 1]$, and its value would denote the bias of the coin (the probability with which it comes up "Heads"). ■

The idea of including the CPD parameters directly in the probabilistic model plays a central role in our discussion of learning later in this book (see section 17.3). For the moment, we note only that including the parameters directly in the model allows us to make explicit the fact that all of the variables $X(d)$ are sampled from the same CPD. By contrast, we could also have used a model where a variable $\theta_X(d)$ is included *inside* the plate, allowing us to encode the setting where each of the coin tosses was sampled from a different coin. We note that this transformation is equivalent to adding the coin ID d as a parent to X; however, the explicit placement of θ_X within the plate makes the nature of the dependence more explicit. In this chapter, to reduce clutter, we use the convention that parameters not explicitly included in the model (or in the figure) are outside of all plates.

Example 6.9

Let us return to our Student *example. We can have a* Student *plate that includes the* attributes *$I(S), G(S)$. As shown in figure 6.7a, we can have $G(S)$ depend on $I(s)$. In this model we have a set of (Intelligence,Grade) pairs, one for each student. The figure also shows the* ground Bayesian network *that would result from instantiating this model for two students. As we discussed, this model implicitly makes the assumption that the CPDs for $P(I(s))$ and for $P(G(s) \mid I(s))$ is the same for all students s. Clearly, we can further enrich the model by introducing additional variables, such as an SAT-score variable for each student.* ■

ground Bayesian network

Our examples thus far have included only a single type of object, and do not significantly expand the expressive power of our language beyond that of plain graphical models. The key benefit of the plate framework is that it allows for multiple plates that can overlap with each other in various ways.

Example 6.10

nested plate

Assume we want to capture the fact that a course has multiple students in it, each with his or her own grade, and that this grade depends on the difficulty of the course. Thus, we can introduce a second type of plate, labeled Course*, where the Grade attribute is now associated with a (student, course) pair. There are several ways in which we can modify the model to include courses. In figure 6.7b, the* Student *plate is* nested *within the* Course *plate. The Difficulty variable is enclosed within the* Course *plate, whereas Intelligence and Grade are enclosed within both plates. We thus have that Grade*(s, c) *for a particular (student, course) pair* (s, c) *depends on Difficulty*(c) *and on Intelligence*(s, c)*.* ■

This formulation ignores an important facet of this problem. As illustrated in figure 6.7b, it induces networks where the *Intelligence* variable is associated not with a student, but rather with a (student, course) pair. Thus, if we have the same student in two different courses, we would have two different variables corresponding to his intelligence, and these could take on different values. This formulation may make sense in some settings, where different notions of "intelligence" may be appropriate to different topics (for example, math versus art); however, it is clearly not a suitable model for all settings. Fortunately, the plate framework allows us to come up with a different formulation.

Example 6.11

plate intersection

Figure 6.7c shows a construction that avoids this limitation. Here, the Student *plate and the* Course *plates* intersect*, so that the Intelligence attribute is now associated only with the* Student *plate, and Difficulty with the* Course *plate; the Grade attribute is associated with the pair (comprising the intersection between the plates). The interpretation of this dependence is that, for any pair of (student, course) objects* s, c*, the attribute Grade*(s, c) *depends on Intelligence*(s) *and on Difficulty*(c)*. The figure also shows the network that results for two students both taking two courses.* ■

In these examples, we see that even simple plate representations can induce fairly complex ground Bayesian networks. Such networks model a rich network of interdependencies between different variables, allowing for paths of influence that one may not anticipate.

Example 6.12

Consider the plate model of figure 6.7c, where we know that a student Jane took CS101 and got an A. This fact changes our belief about Jane's intelligence and increases our belief that CS101 is an easy class. If we now observe that Jane got a C in Math 101, it decreases our beliefs that she is intelligent, and therefore should increase our beliefs that CS101 is an easy class. If we now observe that George got a C in CS101, our probability that George has high intelligence is significantly lower. Thus, our beliefs about George's intelligence can be affected by the grades of other students in other classes.

Figure 6.8 shows a ground Bayesian network induced by a more complex skeleton involving fifteen students and four courses. Somewhat surprisingly, the additional pieces of "weak" evidence regarding other students in other courses can accumulate to change our conclusions fairly radically: Considering only the evidence that relates directly to George's grades in the two classes that he took, our posterior probability that George has high intelligence is 0.8*. If we consider our entire body of evidence about all students in all classes, this probability decreases from 0.8 to 0.25. When we examine the evidence more closely, this conclusion is quite intuitive. We note, for example, that of the students who took CS101, only George got a C. In fact, even Alice, who got a C in both of her other classes, got an A in CS101. This evidence suggests strongly that CS101 is not a difficult class, so that George's grade of a C in CS101 is a very strong indicator that he does not have high intelligence.*

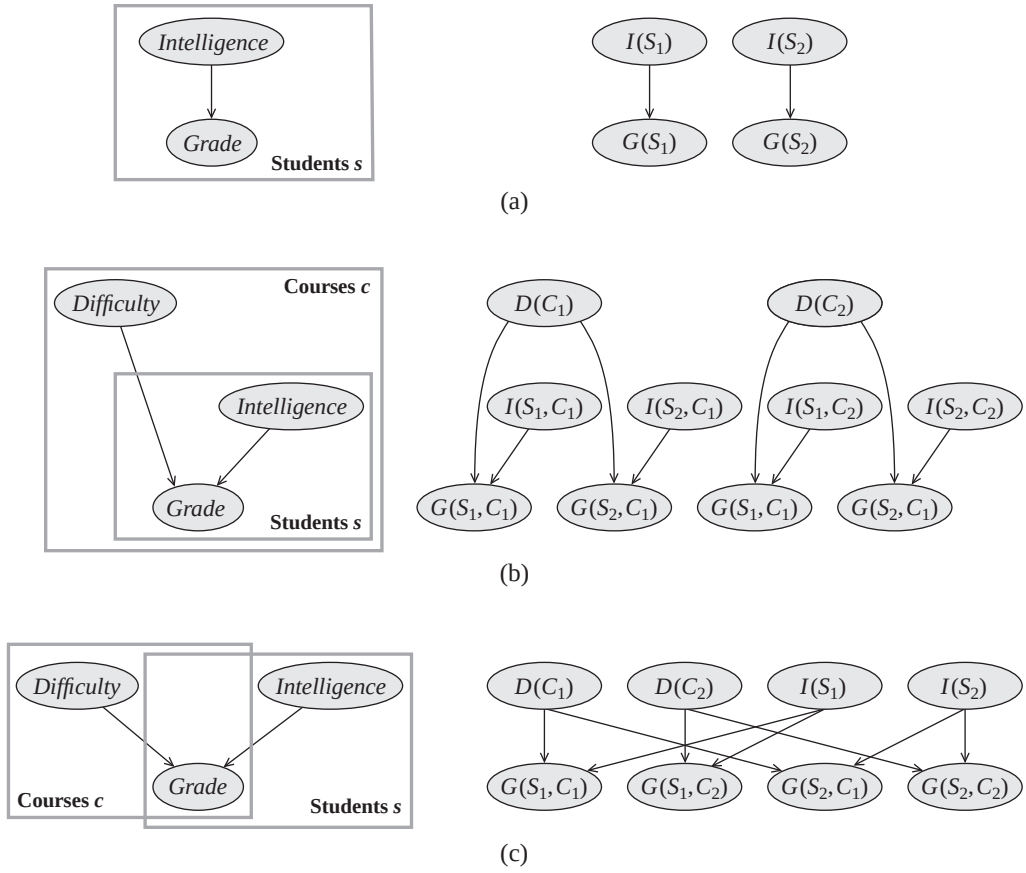


Figure 6.7 Plate models and induced ground Bayesian networks for a simplified Student **example.** (a) Single plate: Multiple independent samples from the same distribution. (b) Nested plates: Multiple courses, each with a separate set of students. (c) Intersecting plates: Multiple courses with overlapping sets of students.

Thus, we obtain much more informed conclusions by defining probabilistic models that encompass all of the relevant evidence. ■

As we can see, **a plate model provides a language for encoding models with repeated structure and shared parameters. As in the case of DBNs, the models are represented at the *template level*; given a particular set of objects, they can then be instantiated to induce a *ground Bayesian network* over the random variables induced by these objects. Because there are infinitely many sets of objects, this template can induce an infinite set of ground networks.**

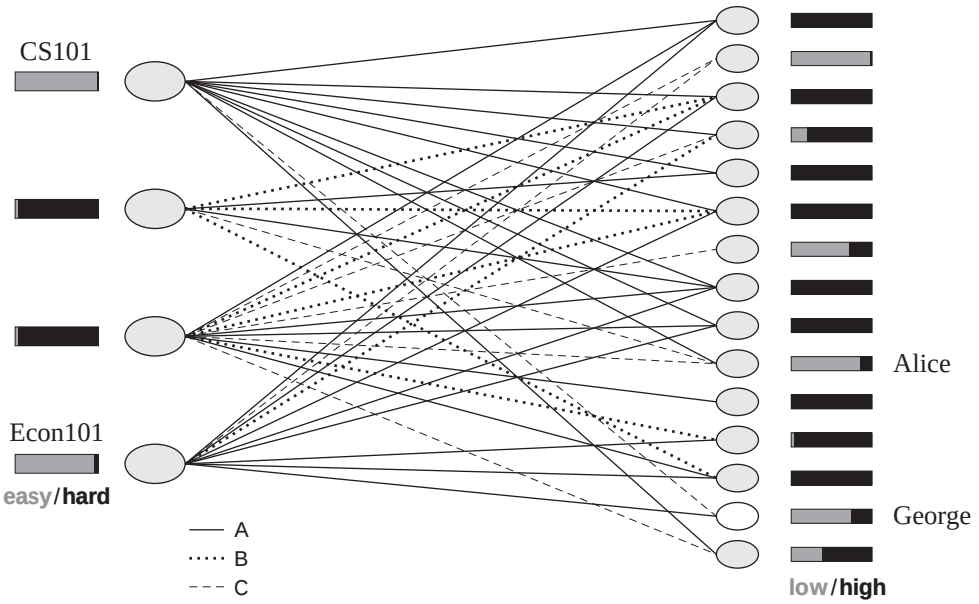


Figure 6.8 Illustration of probabilistic interactions in the University **domain.** The ground network contains random variables for the *Intelligence* of fifteen students (right ovals), including George (denoted by the white oval), and the *Difficulty* for four courses (left ovals), including CS101 and Econ101. There are also observed random variables for some subset of the (student, course) pairs. For clarity, these observed grades are not denoted as variables in the network, but rather as edges relating the relevant (student, course) pairs. Thus, for example George received an A in Econ101 but a C in CS101. Also shown are the final probabilities obtained by running inference over the resulting network.

6.4.1.2 Plate Models: Formal Framework

We now provide a more formal description of the plate modeling language: its representation and its semantics. The plate formalism uses the basic object-relational framework described in section 6.4. As we mentioned earlier, plates correspond to object types.

Each template attribute in the model is embedded in zero, one, or more plates (when plates intersect). If an attribute A is embedded in a set of plates $\mathsf{Q}_1, \ldots, \mathsf{Q}_k$, we can view it as being associated with the argument signature $U_1, \ldots, U_k$, where each logical variable U_i ranges over the objects in the plate (class) Q_i. Recall that a plate model can also have attributes that are external to any plate; these are attributes for which there is always a single copy in the model. We can view this attribute as being associated with an argument signature of arity zero.

In a plate model, the set of random variables induced by a template attribute A is defined by the complete set of assignments: $\Gamma_\kappa[A] = \mathcal{O}^\kappa[\alpha(A)]$. Thus, for example, we would have a *Grade* random variable for every (student,course) pair, whereas, intuitively, these variables are only defined in cases where the student has taken the course. We can take the values of such variables to be unobserved, and, if the model is well designed, its descendants in the probabilistic dependency graph will also be unobserved. In this case, the resulting random variables in the

network will be barren, and can be dropped from the network without affecting the marginal distribution over the others. This solution, however, while providing the right semantics, is not particularly elegant.

We now define the probabilistic dependency structures allowed in the plate framework. To provide a simple, well-specified dependency structure, plate models place strong restrictions on the types of dependencies allowed. For example, in example 6.11, if we define *Intelligence* to be a parent of *Difficulty* (reflecting, say, our intuition that intelligent students may choose to take harder classes), the semantics of the ground model is not clear: for a ground random variable $D(c)$, the model does not specify which specific $I(s)$ is the parent. To avoid this problem, plate models require that an attribute can only depend on attributes in the same plate. This requirement is precisely the intuition behind the notion of plate intersection: Attributes in the intersection of plates can depend on other attributes in any of the plates to which they belong.

Formally, we have the following definition:

Definition 6.9 plate model

A plate model $\mathcal{M}_{Plate}$ *defines, for each template attribute* $A \in \aleph$ *with argument signature* $U_1, \ldots, U_k$*:*

template parent

- *a set of* template parents

$$\mathrm{Pa}_A = \{B_1(\boldsymbol{U}_1), \ldots, B_l(\boldsymbol{U}_l)\}$$

such that for each $B_i(\boldsymbol{U}_i)$*, we have that* $\boldsymbol{U}_i \subseteq \{U_1, \ldots, U_k\}$*. The variables* $\boldsymbol{U}_i$ *are the* argument signature *of the parent* B_i*.*

parent argument signature

- *a template CPD* $P(A \mid \mathrm{Pa}_A)$*.* ■

This definition allows the *Grade* attribute *Grade*(S, C) to depend on *Intelligence*(S), but not vice versa. Note that, as in Bayesian networks, this definition allows any form of CPD, with or without local structure.

Note that the straightforward graphical representation of plates fails to make certain distinctions that are clear in the symbolic representation.

Example 6.13

Assume that our model contains an attribute Cited(U_1, U_2)*, where* U_1, U_2 *are both in the* Paper *class. We might want the dependency model of this attribute to depend on properties of both papers, for example, Topic*(U_1)*, Topic*(U_2)*, or Review-Paper*(U_1)*. To encode this dependency graphically, we first need to have two sets of attributes from the* Paper *class, one for* U_1 *and the other for* U_2*. Moreover, we need to denote somehow which attributes of which of the two arguments are the parents. The symbolic representation makes these distinctions unambiguously.* ■

To instantiate the template parents and template CPDs, it helps to introduce some shorthand notation. Let $\gamma = \langle U_1 \mapsto u_1, \ldots, U_k \mapsto u_k \rangle$ be some assignment to some set of logical variables, and $B(U_{i_1}, \ldots, U_{i_l})$ be an attribute whose argument signature involves only a subset of these variables. We define $B(\gamma)$ to be the ground random variable $B(u_{i_1}, \ldots, u_{i_l})$.

The template-level plate model, when applied to a particular skeleton, defines a ground probabilistic model, in the form of a Bayesian network:

Definition 6.10 ground Bayesian network

A plate model $\mathcal{M}_{Plate}$ *and object skeleton* κ *define a* ground Bayesian network $\mathcal{B}_\kappa^{\mathcal{M}_{Plate}}$ *as fol-*

lows. Let $A(U_1, \ldots, U_k)$ *be any template attribute in* $\aleph$. *Then, for any assignment* $\gamma = \langle U_1 \mapsto u_1, \ldots, U_k \mapsto u_k \rangle \in \Gamma_\kappa[A]$, *we have a variable* $A(\gamma)$ *in the ground network, with parents* $B(\gamma)$ *for all* $B \in \mathrm{Pa}_A$, *and the instantiated CPD* $P(A(\gamma) \mid \mathrm{Pa}_A(\gamma))$. ■

Thus, in our example, we have that the network contains a set of ground random variables *Grade*(s, c), one for every student s and every course c. Each such variable depends on *Intelligence*(s) and on *Difficulty*(c).

The ground network $\mathcal{B}_\kappa^{\mathcal{M}_{Plate}}$ specifies a well-defined joint distribution over $\mathcal{X}_\kappa[\aleph]$, as required. The BN in figure 6.7b is precisely the network structure we would obtain from this definition, using the plate model of figure 6.7 and the object skeleton $\mathcal{O}^\kappa[\mathsf{Student}] = \{s_1, s_2\}$ and $\mathcal{O}^\kappa[\mathsf{Course}] = \{c_1, c_2\}$. In general, despite the compact parameterization (only one local probabilistic model for every attribute in the model), the resulting ground Bayesian network can be quite complex, and models a rich set of interactions. As we saw in example 6.12, the ability to incorporate all of the relevant evidence into the single network shown in the figure can significantly improve our ability to obtain meaningful conclusions even from weak indicators.

☞ **The plate model is simple and easy to understand, but it is also highly limited in several ways. Most important is the first condition of definition 6.9, whereby** $A(U_1, \ldots, U_k)$ **can only depend on attributes of the form** $B(U_{i_1}, \ldots, U_{i_l})$**, where** $U_{i_1}, \ldots, U_{i_l}$ **is a subtuple of** $U_1, \ldots, U_k$**. This restriction significantly constrains our ability to encode a rich network of probabilistic dependencies between the objects in the domain. For example, in the** Genetics **domain, we cannot encode a dependence of** *Genotype*(U_1) **on** *Genotype*(U_2)**, where** U_2 **is (say) the mother of** U_1**. Similarly, we cannot encode temporal models such as those described in section 6.2, where the car's position at a point in time depends on its position at the previous time point.** In the next section, we describe a more expressive representation that addresses these limitations.

6.4.2 Probabilistic Relational Models

As we discussed, the greatest limitation of the plate formalism is its restriction on the argument signature of an attribute's parents. In particular, in our genetic inheritance example, we would like to have a model where *Genotype*(u) depends on *Genotype*(u'), where u' is the mother of u. This type of dependency is not encodable within plate models, because it uses a logical variable in the attribute's parent that is not used within the attribute itself. To allow such models, we must relax this restriction on plate models. However, relaxing this assumption without care can lead to nonsensical models. In particular, if we simply allow *Genotype*(U) to depend on *Genotype*(U'), we end up with a dependency model where every ground variable *Genotype*(u) depends on every other such variable. Such models are intractably dense, and (more importantly) cyclic. What we really want is to allow a dependence of *Genotype*(U) on *Genotype*(U'), but only for those assignments to U' that correspond to U's mother. We now describe one representation that allows such dependencies, and then discuss some of the subtleties that arise when we introduce this significant extension to our expressive power.

6.4.2.1 Contingent Dependencies

To capture such situations, we introduce the notion of a *contingent dependency*, which specifies the context in which a particular dependency holds. A contingent dependency is defined in

terms of a *guard* — a formula that must hold for the dependency to be applicable.

Example 6.14

Consider again our University *example. As usual, we can define Grade(S, C) for a student S and a course C to have the parents Difficulty(C) and Intelligence(S). Now, however, we can make this dependency contingent on the guard Registered(S, C). Here, the parent's argument signature is the same as the child's. More interestingly, contingent dependencies allow us to model the dependency of the student's satisfaction in a course, Satisfaction(S, C), on the teaching ability of the professor who teaches the course. In this setting, we can make Teaching-Ability(P) the parent of Satisfaction(S, C), where the dependency is contingent on the guard Registered$(S, C) \wedge$ Teaches(P, C). Note that here, we have more logical variables in the parents of Satisfaction(S, C) than in the attribute itself: the attribute's argument signature is S, C, whereas its* parent argument signature *is the tuple S, C, P.* ■

We can also represent chains of dependencies within objects in the same class.

Example 6.15

For example, to encode temporal models, we could have Location(U) depend on Location(V), contingent on the guard Precedes(V, U). In our Genetics *example, for the attribute Genotype(U), we would define the template parents Genotype(V) and Genotype(W), the guard Mother$(V, U) \wedge$ Father(W, U), and the parent signature U, V, W.* ■

We now provide the formal definition underlying these examples:

Definition 6.11

For a template attribute A, we define a contingent dependency model *as a tuple consisting of:*

contingent dependency model

- *A* parent argument signature $\alpha(\mathrm{Pa}_A)$*, which is a tuple of typed logical variables U_i such that $\alpha(\mathrm{Pa}_A) \supseteq \alpha(A)$.*

parent argument signature

- *A* guard Γ*, which is a binary-valued formula defined in terms of a set of template attributes Pa_A^Γ over the argument signature $\alpha(\mathrm{Pa}_A)$.*

guard

- *a set of* template parents

template parent

$$\mathrm{Pa}_A = \{B_1(\boldsymbol{U}_1), \ldots, B_l(\boldsymbol{U}_l)\}$$

such that for each $B_i(\boldsymbol{U}_i)$, we have that $\boldsymbol{U}_i \subseteq \alpha(\mathrm{Pa}_A)$. ■

probabilistic relational model

A *probabilistic relational model* (PRM) $\mathcal{M}_{PRM}$ defines, for each $A \in \aleph$ a contingent dependency model, as in definition 6.11, and a template CPD. The structure of the template CPD in this case is more complex, and we discuss it in detail in section 6.4.2.2.

Intuitively, the template parents in a PRM, as in the plate model, define a template for the parent assignments in the ground network, which will correspond to specific assignments of the logical variables to objects of the appropriate type. In this setting, however, the set of logical variables in the parents is not necessarily a subset of the logical variables in the child.

The ability to introduce new logical variables into the specification of an attribute's parents gives us significant expressive power, but introduces some significant challenges. These challenges clearly manifest in the construction of the ground network.

Definition 6.12 ground Bayesian network

A PRM $\mathcal{M}_{PRM}$ and object skeleton κ define a ground Bayesian network $\mathcal{B}_\kappa^{\mathcal{M}_{PRM}}$ *as follows. Let $A(U_1, \ldots, U_k)$ be any template attribute in $\aleph$. Then, for any assignment $\gamma \in \Gamma_\kappa[A]$, we have a variable $A(\gamma)$ in the ground network. This variable has, for any $B \in \mathrm{Pa}_A^\Gamma \cup \mathrm{Pa}_A$ and any assignment γ' to $\alpha(\mathrm{Pa}_A) - \alpha(A)$, the parent that is the instantiated variable $B(\gamma, \gamma')$.* ■

An important subtlety in this definition is that the attributes that appear in the guard are also parents of the ground variable. This requirement is necessary, because the values of the guard attributes determine whether there is a dependency on the parents or not, and hence they affect the probabilistic model.

Using this definition for the model of example 6.14, we have that *Satisfaction*(s, c) has the parents: *Teaching-Ability*(p), *Registered*(s, c), and *Teaches*(p, c) for every professor p. The guard in the contingent dependency is intended to encode the fact that the dependency on *Teaching-Ability*(p) is only present for a subset of individuals p, but it is not obvious how that fact affects the construction of our model. The situation is even more complex in example 6.15, where we have as parents of *Genotype*(u) all of the variables of the form *Father*(v, u) and *Genotype*(v), for all person objects v, and similarly for *Mother*(v, u) and *Genotype*(v). In both cases, the resulting ground network is very densely connected. In the Genetics network, it is also obviously cyclic. We will describe how to encode such dependencies correctly within the CPD of the ground network, and how to deal with the issues of potential cyclicity.

6.4.2.2 CPDs in the Ground Network

As we just discussed, the ground network induced by a PRM can introduce a dependency of a variable on a set of parents that is not fixed in advance, and which may be arbitrarily large. How do we encode a probabilistic dependency model for such dependencies?

Exploiting the Guard Structure The first key observation is that the notion of a contingent dependency is intended to specifically capture context-specific independence: In definition 6.12, if the guard for a parent B of A is false for a particular assignment (γ, γ'), then there is no dependency of $A(\gamma)$ on $B(\gamma, \gamma')$. For example, unless *Mother*(v, u) is true for a particular pair (u, v), we have no dependence of *Genotype*(u) on *Genotype*(v). Similarly, unless *Registered*$(s, c) \wedge$ *Teaches*(p, c) is true, there is no dependence of *Satisfaction*(s, c) on *Teaching-Ability*(p). We can easily capture this type of context-specific independence in the CPD using a variant of the multiplexer CPD of definition 5.3.

While this approach helps us specify the CPD in these networks of potentially unbounded indegree, it does not address the fundamental problem: the dense, and often cyclic, connectivity structure. A common solution to this problem is to assume that the guard predicates are properties of the basic relational structure in the domain, and are often fixed in advance. For example, in the temporal setting, the *Precedes* relation is always fixed: time point $t - 1$ always precedes time point t. Somewhat less obviously, in our Genetics example, it may be reasonable to assume that the pedigree is known in advance.

relational skeleton

We can encode this assumption by defining a *relational skeleton* κ_r, which defines a certain set of facts (usually relationships between objects) that are given in advance, and are not part of the probabilistic model. In cases where the values of the attributes in the guards are specified as part of the relational skeleton, we can simply use that information to determine the set of parents that

are active in the model, usually a very limited set. Thus, for example, if *Registered* and *Teaches* are part of the relational skeleton, then *Satisfaction*(s, c) has the parent *Teaching-Ability*(p) only when *Registered*(s, c) and *Teaches*(p, c) both hold. Similarly, in the Genetics example, if the pedigree structure is given in the skeleton, we would have that *Genotype*(v) is a parent of *Genotype*(u) only if *Mother*(v, u) or *Father*(v, u) are present in the skeleton. Moreover, we see that, assuming a legal pedigree, the resulting ground network in the Genetics domain is guaranteed to be acyclic. Indeed, the resulting model produces ground networks that are precisely of the same type demonstrated in box 3.B. **The use of contingent dependencies allows us to exploit relations that are determined by our skeleton to produce greatly simplified models, and to make explicit the fact that the model is acyclic.**

The situation becomes more complex, however, if the guard predicates are associated with a probabilistic model, and therefore are random variables in the domain. Because the guards are typically associated with relational structure, we refer to this type of uncertainty as *relational uncertainty*. Relational uncertainty introduces significant complexity into our model, as we now cannot use background knowledge (from our skeleton) to simplify the dependency structure in contingent dependencies. In this case, when the family tree is uncertain, we may indeed have that *Genotype*(u) can depend on every other variable *Genotype*(v), a model that is cyclic and ill defined. However, if we restrict the distribution over the *Mother* and *Father* relations so as to ensure that only "reasonable" pedigrees get positive probability, we can still guarantee that our probabilistic model defines a coherent probability distribution. However, defining a probability distribution over the *Mother* and *Father* relations that is guaranteed to have this property is far from trivial; we return to this issue in section 6.6.

relational uncertainty

Aggregating Dependencies By itself, the use of guards may not fully address the problem of defining a parameterization for the CPDs in a PRM. Consider again a dependency of A on an attribute B that involves some set of logical variables $\boldsymbol{U}'$ that are not in $\alpha(A)$. Even if we assume that we have a relational skeleton that fully determines the values of all the guards, there may be multiple assignments γ' to $\boldsymbol{U}'$ for which the guard holds, and hence multiple different ground parents $B(\gamma, \gamma')$ — one for each distinct assignment γ' to $\boldsymbol{U}'$. Even in our simple University example, there may be multiple instructors for a course c, and therefore multiple ground variables *Teaching-Ability*(p) that are parents of a ground variable *Satisfaction*(s, c). In general, the number of possible instantiations of a given parent B is not known in advance, and may not even be bounded. Thus, we need to define a mechanism for specifying a template-level local dependency model that allows a variable number of parents. Moreover, because the parents corresponding to different instantiations are interchangeable, the local dependency model must be symmetric.

There are many possible ways of specifying such a model. One approach is to use one of the symmetric local probability models that we saw in chapter 5. For example, we can use a noisy-or (section 5.4.1) or logistic model (see section 5.4.2), where all parents have the same parameter. An alternative approach is to define an *aggregator CPD* that uses certain aggregate statistics or summaries of the set of parents of a variable. (See exercise 6.7 for some analysis of the expressive power of such CPDs.)

aggregator CPD

Example 6.16

Let us consider again the dependence of Satisfaction(S, C) *on Teaching-Ability*(P)*, with the guard Taking*$(S, C) \wedge$ *Teaching*(C, P)*. Assuming, for the moment, that both Satisfaction and Teaching-*

Ability are binary-valued, we might use a noisy-or model: Given a parameterization for Satisfaction given a single Teaching-Ability, we can use the noisy-or model to define a general CPD for Satisfaction(s, c) *given any set of parents* Teaching-Ability(p_1), ..., Teaching-Ability(p_m). *Alternatively, we can assume that the student's satisfaction depends on the worst instructor and best instructor in the course. In this case, we might aggregate the teaching abilities using the* $\min$ *and* $\max$ *functions, and then use a CPD of our choosing to denote the student's satisfaction as a function of the resulting pair of values. As another example, a student's job prospects can depend on the average grade in all the courses she has taken.* ■

When designing such a combination rule, it is important to consider any possible boundary cases. On one side, in many settings the set of parents can be empty: a course may have no instructors (if it is a seminar composed entirely of invited lecturers); or a person may not have a parent in the pedigree. On the other side, we may also need to consider cases where the number of parents is large, in which case noisy-or and logistic models often become degenerate (see figure 5.10 and figure 5.11).

The situation becomes even more complex when there are multiple distinct parents at the template level, each of which may result in a set of ground parents. For example, a student's satisfaction in a course may depend both on the teaching ability of multiple instructors and on the quality of the design of the different problem sets in the course. We therefore need to address both the aggregation of each type of parent set (instructors or problem sets) as well as combining them into a single CPD. Thus, we need to define some way of combining a set of CPDs $\{P(X \mid Y_{i,1}, \ldots, Y_{i,j_i}) \ : \ i = 1, \ldots, l\}$, to a single joint CPD $\{P(X \mid Y_{1,1}, \ldots, Y_{1,j_1}, \ldots, Y_{l,1}, \ldots, Y_{l,j_l})$. Here, as before, there is no single right answer, and the particular choice is likely to depend heavily on the properties of the application.

We note that the issue of multiple parents is distinct from the multiple parents that arise when we have relational uncertainty. In the case of relational uncertainty, we also have multiple parents for a variable in the ground network; yet, it may well be the case that, in any situation, at most one assignment to the logical variables in the parent signature will satisfy the guard condition. For example, even if we are uncertain about John's paternity, we would like it to be the case that, in any world, there is a unique object v for which *Father*$(v, John)$ holds.

(As we discuss in section 6.6, however, defining a probabilistic model that ensures this type of constraint can be far from trivial.) In this type of situation, the concept of a guard is again useful, since it allows us to avoid defining local dependency models with a variable number of parents in domains (such as genetic inheritance) where such situations do not actually arise.

6.4.2.3 Checking Acyclicity

One important issue in relational dependency models is that the dependency structure of the ground network is, in general, not determined in advance, but rather induced from the model structure and the skeleton. How, then, can we guarantee that we obtain a coherent probability distribution? Most obviously, we can simply check, *post hoc*, that any particular ground network resulting from this process is acyclic. However, this approach is unsatisfying from a model design perspective. When constructing a model, whether by hand or using learning methods, we would like to have some guarantees that it will lead to coherent probability distributions.

Thus, we would like to provide a test that we can execute on a model $\mathcal{M}_{PRM}$ *at the template level*, and which will guarantee that ground distributions induced from $\mathcal{M}_{PRM}$ will be coherent.

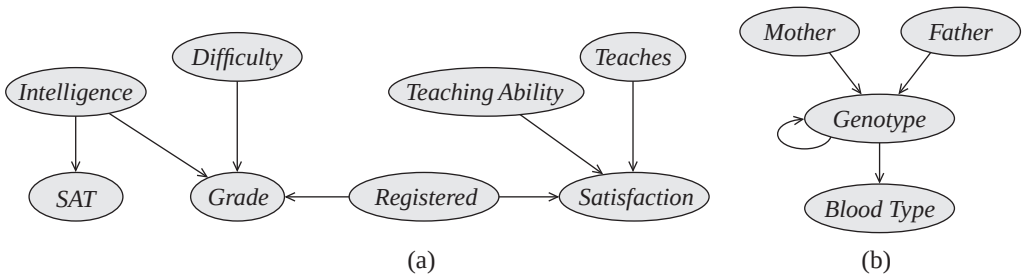


Figure 6.9 Examples of dependency graphs: (a) Dependency graph for the University example. (b) Dependency graph for the Genetics example.

One approach for doing so is to construct a template-level graph that encodes a set of potential dependencies that may happen at the ground level. The nodes in the graph are the template-level attributes; there is an edge from B to A if there is any possibility that a ground variable of type B will influence one of type A.

Definition 6.13 template dependency graph

A template dependency graph *for a template dependency model $\mathcal{M}_{PRM}$ contains a node for each template-level attribute A, and a directed edge from B to A whenever there is an attribute of type B in $\mathrm{Pa}_A^\Gamma \cup \mathrm{Pa}_A$.* ■

This graph can easily be constructed from the definition of the dependency model. For example, the template dependency graph of our University model (example 6.14) is shown in figure 6.9a.

It is not difficult to show that if template dependency graph for a model $\mathcal{M}_{PRM}$ is acyclic (as in this case), it is clear that any ground network generated from $\mathcal{M}_{PRM}$ must also be acyclic (see exercise 6.8). However, a cycle in the template dependency graph for $\mathcal{M}_{PRM}$ does not imply that every ground network induced by $\mathcal{M}_{PRM}$ is cyclic. (This is the case, however, for any nondegenerate instantiation of a plate model; see exercise 6.9.) Indeed, there are template models that, although cyclic at the template levels, reflect a natural dependency structure whose ground networks are guaranteed to be acyclic in practice.

Example 6.17

Consider the template dependency graph of our Genetics *domain, shown in figure 6.9b. The template-level self-loop involving Genotype*(Person) *reflects a ground-level dependency of a person's genotype on that of his or her parents. This type of dependency can only lead to cycles in the ground network if the pedigree is cyclic, that is, a person is his/her own ancestor. Because such cases (time travel aside) are impossible, this template model cannot result in cyclic ground networks* for the skeletons that arise in practice. *Intuitively, in this case, we have an (acyclic) ordering $\prec$ over the objects (people) in the domain, which implies that u' can be a parent of u only when $u' \prec u$; therefore, Genotype(u) can depend on Genotype(u') only when $u' \prec u$. This ordering on objects is acyclic, and therefore so is the resulting dependency structure.* ■

The template dependency graph does not account for these constraints on the skeleton, and therefore we cannot conclude by examining the graph whether cycles can occur in ground

networks for such ordered skeletons. However, exercise 6.10 discusses a richer form of the template dependency network that explicitly incorporates such constraints, and it is therefore able to determine that our Genetics model results in acyclic ground networks for any skeleton representing an acyclic pedigree.

So far in our discussion of acyclicity, we have largely sidestepped the issue of relational uncertainty. As we discussed, in the case of relational uncertainty, the ground network contains many "potential" edges, only a few of which will ever be "active" simultaneously. In such cases, the resulting ground network may not even be acyclic, even though it may well define a coherent (and sparse) probability distribution for every relational structure. Indeed, as we discussed in example 6.17, there are models that are potentially cyclic but are guaranteed to be acyclic by virtue of specific constraints on the dependency structure. It is thus possible to guarantee the coherence of a cyclic model of this type by ensuring that there is no positive-probability assignment to the guards that actually induces a cyclic dependence between the attributes.

6.5 Undirected Representation

The previous sections describe template-based formalisms that use a directed graphical model as their foundation. One can define similar extensions for undirected graphical models. Many of the ideas underlying this extension are fairly similar to the directed case. However, the greater flexibility of the undirected representation in avoiding local normalization requirements and acyclicity constraints can be particularly helpful in the context of these richer representations. Eliminating these requirements allows us to easily encode a much richer set of patterns about the relationships between objects in the domain; see, for example, box 6.C. In particular, as we discuss in section 6.6, these benefits can be very significant when we wish to define distributions over complex relational structures.

The basic component in a template-based undirected model is some expression, written in terms of template-level attributes with logical variables as arguments, and associated with a template factor. For a given object skeleton, each possible assignment γ to the logical variables in the expression induces a factor in the ground undirected network, all sharing the same parameters. As for variable-based undirected representations, one can parameterize a template-based undirected probabilistic model using full factors, or using features, as in a log-linear model. This decision is largely orthogonal to other issues. We choose to use log-linear features, which are the finest-grained representation and subsume table factors.

Example 6.18

Let us begin by revisiting our Misconception *example in section 4.1. Now, assume that we are interested in defining a probabilistic model over an entire set of students, where some number of pairs study together. We define a binary predicate (relation) Study-Pair*(S_1, S_2)*, which is true when two students* S_1, S_2 *study together, and a predicate (attribute) Misconception*(S) *that encodes the level of understanding of a student* S*. We can now define a* template feature f_M*, over pairs Misconception*(S_1)*, Misconception*(S_2)*, which takes value 1 whenever*

template feature

$$[\textit{Study-Pair}(S_1, S_2) = \textit{true} \wedge \textit{Misconception}(S_1) = \textit{Misconception}(S_2)] = \textit{true} \tag{6.6}$$

and has value 0 otherwise. ■

Definition 6.14

relational Markov network

feature argument

A relational Markov network $\mathcal{M}_{RMN}$ *is defined in terms of a set* Λ *of* template features, *where each* $\lambda \in \Lambda$ *comprises:*

- *a real-valued* template feature f_λ *whose* arguments *are* $\aleph(\lambda) = \{A_1(\boldsymbol{U}_1), \ldots, A_l(U_l)\}$;
- *a weight* $w_\lambda \in \mathbb{R}$.

We define $\alpha(\lambda)$ *so that for all* i, $\boldsymbol{U}_i \subseteq \alpha(\lambda)$. ■

In example 6.18, we have that $\alpha(\lambda_M) = \{S_1, S_2\}$, both of type Student; and

$$\aleph(\lambda_M) = \{\textit{Study-Pair}(S_1, S_2), \textit{Misconception}(S_1), \textit{Misconception}(S_2)\}.$$

object skeleton

To specify a ground network using an RMN, we must provide an *object skeleton* κ that defines a finite set of objects $\mathcal{O}^\kappa[\mathsf{Q}]$ for each class Q. As before, we can also define a restricted set $\Gamma_\kappa[A] \subset \mathcal{O}^\kappa[\alpha(A)]$. Given a skeleton, we can now define a *ground Gibbs distribution* in the natural way:

Example 6.19

Continuing example 6.18, assume we are given a skeleton containing a particular set of students and the set of study pairs within this set. This model induces a Markov network where the ground random variables have the form Misconception(s) *for every student* s *in the domain. In this model, we have a feature* f_M *for every triple of variables* Misconception(s_1), Misconception(s_2), Study-Pair(s_1, s_2). *As usual in log-linear models, features can be associated with a weight; in this example, we might choose* $w_M = 10$. *In this case, the unnormalized measure for a given assignment to the ground variables would be* $\exp(10K)$, *where* K *is the number of pairs* s_1, s_2 *for which equation (6.6) holds.* ■

More formally:

Definition 6.15

ground Gibbs distribution

Given an RMN $\mathcal{M}_{RMN}$ *and an object skeleton* κ, *we can define a* ground Gibbs distribution $P_\kappa^{\mathcal{M}_{RMN}}$ *as follows:*

- *The variables in the network are* $\mathcal{X}_\kappa[\aleph]$ *(as in definition 6.7);*
- $P_\kappa^{\mathcal{M}_{RMN}}$ *contains a term*

$$\exp(w_\lambda \cdot f_\lambda(\gamma))$$

for each feature template $\lambda \in \Lambda$ *and each assignment* $\gamma \in \Gamma_\kappa[\alpha(\lambda)]$. ■

As always, a (ground) Gibbs distribution defines a Markov network, where we connect every pair of variables that appear together in some factor.

In the directed setting, the dense connectivity arising in ground networks raised several concerns: acyclicity, aggregation of dependencies, and computational cost of the resulting model. The first of these is obviously not a concern in undirected models. The other two, however, deserve some discussion.

Although better hidden, the issue of aggregating the contribution of multiple assignments of a feature also arises in undirected model. Here, the definition of the Gibbs distribution dictates

the form of the aggregation we use. In this case, each grounding of the feature defines a factor in the unnormalized measure, and they are combined by a product operation, or an addition in log-space. In other words, each occurrence of the feature has a log-linear contribution to the unnormalized density. Importantly, however, this type of aggregation may not be appropriate for every application.

Example 6.20 *Consider a model for "viral marketing" — a social network of individuals related by the Friends(P, P') relation, where the attribute of interest Gadget(P) is the purchase of some cool new gadget G. We may want to construct a model where it is more likely that two friends either both own or both do not own G. That is, we have a feature similar to λ_M in example 6.18. In the log-linear model, the unnormalized probability that a person p purchases G grows log-linearly with the number k_p of his friends who own G. However, a more realistic model may involve a saturation effect, where the impact diminishes as k_p grows; that is, the increase in probability of Gadget(P) between $k_p = 0$ and $k_p = 1$ is greater than the increase in probability between $k_p = 20$ and $k_p = 21$.* ■

Thus, in concrete applications, we may wish to extend the framework to allow for other forms of combination, or even simply to define auxiliary variables corresponding to relevant aggregates (for example, the value of k_p in our example).

The issue of dense connectivity is as much an issue in the undirected case as in the directed case. The typical solution is similar: If we have background knowledge in the form of a relational skeleton, we can significantly simplify the resulting model. Here, the operation is very simple: we simply reduce every one of the factors in our model using the evidence contained in the relational skeleton, producing a reduced Markov network, as in definition 4.7. In this network, we would eliminate any ground variables whose values are observed in the skeleton and instantiate their (fixed) values in any ground factors containing them. In many cases, this process can greatly simplify the resulting features, often making them degenerate.

Example 6.21 *Returning to example 6.18, assume now that our skeleton specifies the instantiation of the relation Study-Pair, so that we know exactly which pairs of students study together and which do not. Now, consider the reduced Markov network obtained by conditioning on the skeleton. As all the variables Study-Pair(s_1, s_2) are observed, they are all eliminated from the network. Moreover, for any pair of students s_1, s_2 for which Study-Pair$(s_1, s_2) =$ false, the feature $\lambda_M(s_1, s_2)$ necessarily takes the value 0, regardless of the values of the other variables in the feature. Because this ground feature is vacuous and has no impact on the distribution, it can be eliminated from the model. The resulting Markov network is much simpler, containing edges only between pairs of students who study together (according to the information in the relational skeleton).* ■

We note that we could have introduced the notion of a guarded dependency, as we did for PRMs. However, this component is far less useful here than it was in the directed case, where it also served a role in eliminating the need to aggregate parents that are not actually present in the network and in helping clarify the acyclicity in the model. Neither of these issues arises in the undirected framework, obviating the need for the additional notational complexity.

Finally, we mention one subtlety that is specific to the undirected setting. An undirected model uses nonlocal factors, which can have a dramatic influence on the global probability measure of the model. Thus, the probability distribution defined by an undirected relational model is not modular: Introducing a new object into the domain can drastically change the

distribution over the properties of existing objects, even when the newly introduced object seems to have no meaningful interactions with the previous objects.

Example 6.22

Let us return to our example 6.19, and assume that any pair of students study together with some probability p; that is, we have an additional template feature over Study-Pair(S_1, S_2) that takes the value $\log p$ when this binary attribute is true and $\log(1-p)$ otherwise.

Assume that we have a probability distribution over the properties of some set of students $\mathcal{O}^\kappa[\mathsf{Student}] = \{s_1, \ldots, s_n\}$, and let us study how this distribution changes if we add a new student s_{n+1}. Consider an assignment to the properties of $s_1, \ldots, s_n$ in which m of the n students s_i have Misconception$(s_i) = 1$, whereas the remaining $n - m$ have Misconception$(s_i) = 0$. We can now consider the following situations with respect to s_{n+1}: he studies with k of the m students for whom Misconception$(s_i) = 1$, with ℓ of the $n - m$ students for whom Misconception$(s_i) = 0$, and himself has Misconception$(s_{n+1}) = c$ (for $c \in \{0, 1\}$). The probability of each such event is

$$\binom{m}{k}\binom{(n-m)}{\ell} p^\ell (1-p)^{(n-m-\ell)})(10^{kc} \cdot 10^{\ell(1-c)}),$$

where the first two terms come from the factors over the Study-Pair(S_1, S_2) structure, and the final term comes from the template feature λ_M. We want to compute the marginal distribution over our original variables (not involving s_{n+1}), to see whether introducing s_{n+1} changes this distribution. Thus, we sum out over all of the preceding events, which (using simple algebra) is $(10p + (1-p))^m + (10p + (1-p))^{n-m}$.

This analysis shows that the assignments to our original variables are multiplied by very different terms, depending on the value m. In particular, the probability of joint assignments where $m = 0$, so that all students agree, are multiplied by a factor of $(10p + (1-p))^n$, whereas the probability of joint assignments where the students are equally divided in their opinion are multiplied by $(10p + (1-p))^{n/2}$, an exponentially smaller factor. Thus, adding a new student, even one about whom we appear to know nothing, can drastically change the properties of our probability distribution. ■

☞ **Thus, for undirected models, it can be problematic to construct (by learning or by hand) a template-based model for domains of a certain size, and apply it to models of a very different size. The impact of the domain size on the probability distribution varies, and therefore the implications regarding our ability to apply learning in this setting need to be evaluated per application.**

Box 6.C — Case Study: Collective Classification of Web Pages. *One application that calls for interesting models of interobject relationships is the classification of a network of interlinked webpages. One example is that of a university website, where webpages can be associated with students, faculty members, courses, projects, and more. We can associate each webpage w with a hidden variable $T(w)$ whose value denotes the type of the entity to which the webpage belongs. In a standard classification setting, we would use some learned classifier to label each webpage based on its features, such as the words on the page. However, we can obtain more information by also considering the interactions between the entities and the correlations they induce over their labels. For example, an examination of the data reveals that student pages are more likely to link to faculty webpages than to other student pages.*

One can capture this type of interaction both in a directed and in an undirected model. In a directed model, we might have a binary attribute Links(W_1, W_2) that takes the value true if W_1 links to W_2 and false otherwise. We can then have Links(W_1, W_2) depend on $T(W_1)$ and $T(W_2)$, capturing the dependence of the link probability on the classes of the two linked pages. An alternative approach is to use an undirected model, where we directly introduce a pairwise template feature over $T(W_1), T(W_2)$ for pairs W_1, W_2 such that Links(W_1, W_2). Here, we can give higher potentials to pairs of types that tend to link, for example, student-faculty, faculty-course, faculty-project, project-student, and more.

A priori, both models appear to capture the basic structure of the domain. However, the directed model has some significant disadvantages in this setting. First, since the link structure of the webpages is known, the Links(W_1, W_2) is always observed. Thus, we have an active v-structure connecting every pair of webpages, whether they are linked or not. The computational disadvantages of this requirement are obvious. Less obvious but equally important is the fact that there are many more non-links than links, and so the signal from the absent links tends to overwhelm the signal that could derived from the links that are present. In an undirected model, the absent links are simply omitted from the model; we simply introduce a potential that correlates the topics of two webpages only if they are linked. Therefore, an undirected model generally achieves much better performance on this task.

Another important advantage of the undirected model for this task is its flexibility in incorporating a much richer set of interactions. For example, it is often the case that a faculty member has a section in her webpage where she lists courses that she teaches, and another section that lists students whom she advises. Thus, another useful correlation that we may wish to model is one between the types of two webpages that are both linked from a third, and whose links are in close proximity on the page. We can model this type of interaction using features of the form Close-Links$(W, W_1, W_2) \wedge T(W_1) = t_1 \wedge T(W_2) = t_2$, where Close-Links$(W, W_1, W_2)$ is derived directly from the structure of the page.

Finally, an extension of the same model can be used to label not only the entities (webpages) but also the links between them. For example, we might want to determine whether a student-professor (s, p) pair with a link from s to p represents an advising relationship, or whether a linked professor-course pair represents an instructor relationship. Once again, a standard classifier would make use of features such as words in the vicinity of the hyperlink. At the next level, we can use an extension of the model described earlier to classify jointly both the types of the entities and the types of the links that relate them. In a more interesting extension, a relational model can also utilize higher-level patterns; for example, using a template feature over triplets of template attributes $T(W)$, we can encode the fact that students and their advisors belong to the same research group, or that students often serve as teaching assistants in courses that their advisors teach.

6.6 Structural Uncertainty ⋆

structural uncertainty

The object-relational probabilistic models we described allow us to encode a very rich family of distributions over possible worlds. In addition to encoding distributions over the attributes of objects, these approaches can allow us to encode *structural uncertainty* — a probabilistic model over the actual structure of the worlds, both the set of objects they contain and the relations

between them. The different models we presented exhibit significant differences in the types of structural uncertainty that they naturally encompass. In this section, we discuss some of the major issues that arise when representing structural uncertainty, and how these issues are handled by the different models.

relational uncertainty

object uncertainty

There are two main types of structural uncertainty that we can consider: *relational uncertainty*, which models a distribution over the presence or absence of relations between objects; and *object uncertainty*, which models a distribution over the existence or number of actual objects in the domain. We discuss each in turn.

6.6.1 Relational Uncertainty

The template representations we have already developed already allow us to encode uncertainty about the relational structure. As in example 6.22, we can simply make the existence of a relationship a stochastic event. What types of probability distributions over relational structure can we encode using these representational tools? In example 6.22, each possible relation *Study-Pair*(s_1, s_2) is selected independently, at random, with probability p. Unfortunately, such graphs are not representative of most relational structures that we observe in real-world settings.

Example 6.23

Let us select an even simpler example, where the graph we are constructing is bipartite. Consider the relation Teaches(P, C)*, and assume that it takes the value true with probability* 0.1*. Consider a skeleton that contains 10 professors and 20 courses. Then the expected number of courses per professor is 2, and the expected number of professors per course is 1. So far, everything appears quite reasonable. However, the probability that, in the resulting graph, a particular professor teaches* ℓ *courses is distributed binomially:* $\binom{20}{\ell}0.1^{\ell}0.9^{20-\ell}$*. For example, the probability that any single professor teaches* 5 *or more courses is 4.3 percent, and the probability that at least one of them does is around 29 percent. This is much higher than is realistic in real-world graphs. The situation becomes much worse if we increase the number of professors and courses in our skeleton.*

Of course, we can add parents to this attribute. For example, we can let the presence of an edge depend on the research area of the professor and the topic of the course, so that this attribute is more likely to take the value true if the area and topic match. However, this solution does not address the fundamental problem: it is still the case that, given all of the research areas and topics, the relationship status for different pairs of objects (the edges in the relational graph) are chosen independently. ■

In this example, we wish to model certain global constraints on the distribution over the graph: the fact that each faculty member tends to teach only a small subset of courses. Unfortunately, it is far from clear how to incorporate this constraint into a template-level generative (directed) model over attributes corresponding to the presence of individual relations. Indeed, consider even the simpler case where we wish to encode the prior knowledge that each course has exactly one instructor. This model induces a correlation among all of the binary random variables corresponding to different instantiations *Teaches*(p, c) for different professors p and the same course c: once we have *Teaches*$(p, c) =$ *true*, we must have *Teaches*$(p', c) =$ *false* for all $p' \neq p$. In order to incorporate this correlation, we would have to define a generative process that "selects" the relation variables *Teaches*(p, c) in some sequence, in a way that allows each *Teaches*(p', c) to depend on all of the preceding variables *Teaches*(p, c). This induces

dependency models with dense connectivity, an arbitrary number of parents per variable (in the ground network), and a fairly complex dependency structure.

An alternative approach is to use a different encoding for the course-instructor relationship. In logical languages, an alternative mechanism for relating objects to each other is via functions.

object-valued attribute

A *function*, or *object-valued attribute* takes as argument a tuple of objects from a given set of classes, and returns a set of objects in another class. Thus, for example, rather than having a relation *Mother*(P_1, P_2), we might use a function *Mother-of*$(P_1) \mapsto$ Person that takes, as argument, a person-object p_1, and returns the person object p_2, which is p_1's mother. In this case, the return-value of the function is just a single object, but, in general, we can define functions that return an entire set of objects. In our University example, the relation *Teaches* defines the function *Courses-Of*, which maps from professors to the courses they teach, and the function *Instructor*, which maps from courses to the professors that teach them. We note that these functions are *inverses* of each other: We have that a professor p is in *Instructor*(c) if and only if c is in *Courses-Of*(p).

As we can see, we can easily convert between set-valued functions and relations. Indeed, as long as the relational structure is fixed, the decision on which representation to use is largely a matter of convenience (or convention). However, once we introduce probabilistic models over the relational structure, the two representations lend themselves more naturally to quite different types of model. Thus, for example, if we encode the course-instructor relationship as a function from professors to courses, then rather than select pairwise relations at random, we might select, for any professor p, the set of courses *Courses-Of*(p). We can define a distribution over sets in two components: a distribution over the size ℓ of the set, and a distribution that then selects ℓ distinct objects that will make up the set.

Example 6.24

Assume we want to define a probability distribution over the set Courses-Of(p) of courses taught by a professor p. We may first define a distribution over the number ℓ of courses c in Courses-Of(p). This distribution may depend on properties of the professor, such as her department or her level of seniority. Given the size ℓ, we now have to select the actual set of ℓ courses taught by p. We can define a model that selects ℓ courses independently from among the set of courses at the university. This choice can depend on properties of both the professor and the course. For example, if the professor's specialization is in artificial intelligence, she is more likely to teach a course in that area than in operating systems. Thus, the probability of selecting c to be in Courses-Of(p) depends both on Topic(c) and on Research-Area(p). Importantly, since we have already chosen $\ell = |$Courses-Of$(p)|$, we need to ensure that we actually select ℓ distinct courses, that is, we must sample from the courses without replacement. Thus, our ℓ sampling events for the different courses cannot be completely independent. ■

While useful in certain settings, this model does not solve the fundamental problem. For example, although it allows us to enforce that every professor teaches between two and four courses, it still leaves open the possibility that a single course is taught by ten professors. We can, of course, consider a model that reverses the direction of the function, encoding a distribution over the instructors of each course rather than the courses taught by a professor, but this solution would simply raise the converse problem of the possibility that a single professor teaches a large number of classes.

It follows from this discussion that **it is difficult, in generative (directed) representations, to define distributions over relational structures that guarantee (or prefer) certain structural**

properties of the relation. For example, there is no natural way in which we can construct a probabilistic model that exhibits (a preference for) transitivity, that is, one satisfying that if $R(u, v)$ and $R(v, w)$ then (it is more likely that) $R(u, w)$.

These problems have a natural solution within the undirected framework. For example, a preference for transitivity can be encoded simply as a template feature that ascribes a high value to the (template) event

$$R(U, V) = \textit{true}, R(V, W) = \textit{true}, R(U, W) = \textit{true}.$$

A (soft) constraint enforcing at most one instructor per course can be encoded similarly as a (very) low potential on the template event

$$\textit{Teaches}(P, C) = \textit{true}, \textit{Teaches}(P', C) = \textit{true}.$$

A constraint enforcing at least one instructor per course cannot be encoded in the framework of relational Markov networks, which allow only features with a bounded set of arguments. However, it is not difficult to extend the language to include potentials over unbounded sets of variables, as long as these potentials have a compact, finite-size representation. For example, we could incorporate an aggregator feature that counts the number t_p of objects c such that *Teaches*(p, c), and introduce a potential over the value of t_p. This extension would allow us to incorporate arbitrary preferences about the number of courses taught by a professor. At the same time, the model could also include potentials over the aggregator i_c that counts the number of instructors p for a course c. Thus, we can simultaneously include global preferences on both sides of the relation between courses and professors.

However, while this approach addresses the issue of expressing such constraints, it leaves unresolved the problem of the complexity of the resulting ground network. In all of these examples, the induced ground network is very densely connected, with a ground variable for every potential edge in the relational graph (for example, $R(u, v)$), and a factor relating every pair or even every triple of these variables. In the latter examples, involving the aggregator, we have potentials whose scope is unbounded, containing all of the ground variables $R(u, v)$.

6.6.2 Object Uncertainty

So far, we have focused our discussion on representing probabilistic models about the presence or absence of certain relations, given a set of base objects. One can also consider settings in which even the set of objects in the world is not predetermined, and so we wish to define a probability distribution over this set.

Perhaps the most common setting in which this type of reasoning arises is in situations where different objects in our domain may be equal to each other. This situation arises quite often. For example, a single person can be student #34 in CS101, student #57 in Econ203, the eldest daughter of John and Mary, the girlfriend of Tom, and so on.

One solution is to allow objects in the domain to correspond to different "names," or ways of referring to an object, but explicitly reason about the probability that some of these names refer to the same object. But how do we model a distribution over equality relationships between the objects playing different roles in the model?

The key insight is to introduce explicitly into the model the notion of a "reference" to an object, where the same object can be referred to in several different ways. That is, we include in

the model objects that correspond to the different "references" to the object. Thus, for example, we could have a class of "person objects" and another class for "person reference objects." We can use a relation-based representation in this setting, using a relation *Refers-to*(r, p) that is *true* whenever the reference r refers to a person p. However, we must also introduce uniqueness constraints to ensure that a reference r refers to precisely a single person p. Alternatively, a more natural approach is to use a function, or object-valued attribute, *Referent*(r), which designates the person to whom r refers. This approach automatically enforces the uniqueness constraints, and it is thus perhaps more appropriate to this application.

In either case, the relationship between references and the objects to which they refer is generally probabilistic and interacts probabilistically with other attributes in the domain. In particular, we would generally introduce factors that model the similarity of the properties of a "reference object" r and those of the true object p to which it refers. These *attribute similarity potentials* can be constructed to allow for noise and variation. For example, we can model the fact that a person whose name is "John Franklin Adams" may decide to go by "J.F. Adams" in one setting and "Frank Adams" in another, but is unlikely to go by the name "Peggy Smith." We can also model the fact that a person may decide to "round down" his or her reported age in some settings (for example, social interactions) but not in others (for example, tax forms). The problem of determining the correspondence between references and the entities to which they refer is an instance of the *correspondence* problem, which is described in detail in box 12.D. Box 6.D describes an application of this type of model to the problem of matching bibliographical citations.

correspondence

In an alternative approach, we might go one step further, we can eliminate any mention of the true underlying objects, and restrict the model only to object references. In this solution, the domain contains only "reference objects" (at least for some classes). Now, rather than mapping references to the object to which they refer, we simply allow for different references to "correspond" to each other. Specifically, we might include a binary predicate *Same-as*(r, r'), which asserts that r and r' both refer to the same underlying object (not included as an object in the domain).

To ensure that *Same-As* is consistent with the semantics of an equality relation, we need to introduce various constraints on its properties. (Because these constraints are standard axioms of equality, we can include them as part of the formalism rather than require each user to specify them.) First, using the ideas described in section 6.6.1, we can introduce undirected (hard) potentials to constrain the relation to satisfy:

- *Reflexivity* — *Same-As*(r, r);
- *Symmetry* — *Same-As*(r, r') if and only if *Same-As*(r', r);
- *Transitivity* — *Same-As*(r, r') and *Same-As*(r', r'') implies *Same-As*(r, r'').

These conditions imply that the *Same-As* relation defines an equivalence relation on reference objects, and thus partitions them into mutually exclusive and exhaustive equivalence classes. Importantly, however, these constraints can only be encoded in an undirected model, and therefore this approach to dealing with equality only applies in that setting. In addition, we include in the model attribute similarity potentials, as before, which indicate the extent to which we expect attributes or predicates for two *Same-As* reference objects r and r' to be similar to each other. This approach, applied to a set of named objects, tends to cluster them together

into groups whose attributes are similar and that participate in relations with objects that are also in equivalent groups.

There are, however, several problems with the reference-only solution. First, there is no natural place to put factors that should apply once per underlying entity.

Example 6.25

Suppose we are interested in inferring people's gender from their names. We might have a potential saying that someone named "Alex" is more likely to be male than female. But if we make this a template factor on $\{Name(R), Gender(R)\}$ *where* R *ranges over references, then the factor will apply many times to people with many references. Thus, the probability that a person named "Alex" is male will increase exponentially with the number of references to that person.* ■

A related but more subtle problem is the dependence of the outcome of our inference on the number of references.

Example 6.26

Consider a very simple example where we have only references to one type of object and only the attribute A*, which takes values* $1, 2, 3$*. For each pair of object references* r, r' *such that Same-As*(r, r') *holds, we have an attribute similarity potential relating* $A(r)$ *and* $A(r')$*: the cases of* $A(r) = A(r')$ *have the highest weight* w*;* $A(r) = 1,\ A(r') = 3$ *has very low weight; and* $A(r) = 2,\ A(r') = 1$ *and* $A(r) = 2,\ A(r') = 3$ *both have the same medium potential* q*. Now, consider the graph of people related by the Same-As relation: since Same-As is an equivalence relation, the graph is a set of mutually exclusive and exhaustive partitions, each corresponding to a set of references that correspond to the same object. Now, assume we have a configuration of evidence where we observe* k_i *references with* $A(r) = i$*, for* $i = 1, 2, 3$*. The most likely assignment relative to this model will have one cluster with all the* $A(r) = 1$ *references, and another with all the* $A(r) = 3$ *references. What about the references with* $A(r) = 2$*?*

Somewhat surprisingly, their disposition depends on the relative sizes of the clusters. To understand why, we first note that (assuming $w > 1$*) there are only three solutions with reasonably high probability: three separate clusters; a "1+2" and a "3" cluster; and a "1" and a "2+3" cluster. All other solutions have much lower probability, and the discrepancy decays exponentially with the size of the domain. Now, consider the case where* $k_2 = 1$*, so that there only one* r^* *with* $A(r) = 2$*. If we add* r^* *to the "1" cluster, we introduce an attribute similarity potential between* $A(r^*)$ *and all of the* $A(r)$*'s in the "1" cluster. This multiplies the overall probability of the configuration by* q^{k_1}*. Similarly, if we add* r^* *to the "3" cluster, the probability of the configuration is multiplied by* q^{k_3}*. Thus, if* $q < 1$*, the reference* r^* *is more likely to be placed in the smaller of the two clusters; if* $q > 1$*, it is more likely to be placed in the larger cluster. As* k_2 *grows, the optimal solution may now be one where we put the 2's into their own, separate cluster; the benefit of doing so depends on the relative sizes of the different parameters* q, w, k_1, k_2, k_3*.* ■

Thus, in this type of model, the resulting posterior is often highly peaked, and the probabilities of the different high-probability outcomes very sensitive to the parameters. By contrast, a model where each equivalence cluster is associated with a single actual object is a lot "smoother," for the number of attribute similarity potentials induced by a cluster of references grows linearly, not quadratically, in the size of the cluster.

Box 6.D — Case Study: Object Uncertainty and Citation Matching. *Being able to browse the network of citations between academic works is a valuable tool for research. For instance, given one citation to a relevant publication, one might want a list of other papers that cite the same work. There are several services that attempt to construct such lists automatically by extracting citations from online papers. This task is difficult because the citations come in a wide variety of formats, and often contain errors — owing both to the original author and to the vagaries of the extraction process. For example, consider the two citations:*

```
Elston R, Stewart A. A General Model for the Genetic Analysis of
Pedigree Data. Hum. Hered. 1971;21:523-542.

Elston RC, Stewart J (1971): A general model for the analysis of
pedigree data. Hum Hered 21523-542.
```

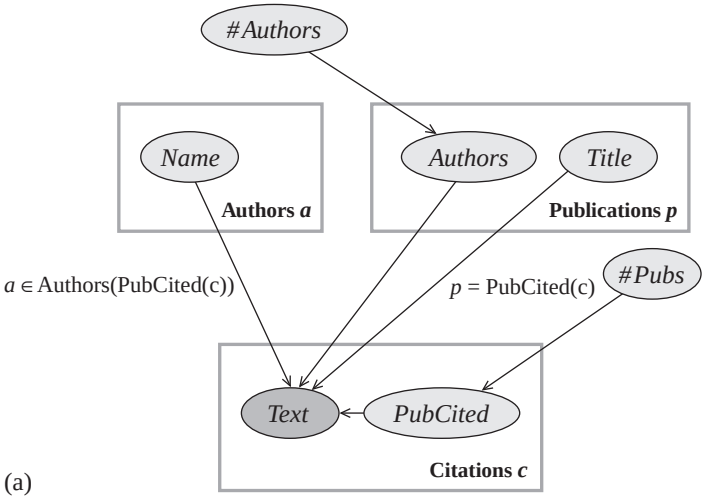
These citations refer to the same paper, but the first one gives the wrong first initial for J. Stewart, and the second one omits the word "genetic" in the title. The colon between the journal volume and page numbers has also been lost in the second citation. A citation matching system must handle this kind of variation, but must also avoid lumping together distinct papers that have similar titles and author lists.

Probabilistic object-relational models have proven to be an effective approach to this problem. One way to handle the inherent object uncertainty is to use a directed model with a Citation *class, as well as* Publication *and* Author *classes. The set of observed* Citation *objects can be included in the object skeleton, but the number of* Publication *and* Author *objects is unknown.*

A directed object-relational model for this problem (based roughly on the model of Milch et al. (2004)) is shown in figure 6.D.1a. The model includes random variables for the sizes of the Author *and* Publication *classes. The* Citation *class has an object-valued attribute PubCited(C), whose value is the* Publication *object that the citation refers to. The* Publication *class has a set-valued attribute Authors(P), indicating the set of authors on the publication. These attributes are given very simple CPDs: for PubCited(C), we use a uniform distribution over the set of* Publication *objects, and for Authors(P) we use a prior for the number of contributors along with a uniform selection distribution.*

To complete this model, we include string-valued attributes Name(A) and Title(P), whose CPDs encode prior distributions over name and title strings (for now, we ignore other attributes such as date and journal name). Finally, the Citation *class has an attribute Text(C), containing the observed text of the citation. The citation text attribute depends on the title and author names of the publication it refers to; its CPD encodes the way citation strings are formatted, and the probabilities of various errors and abbreviations.*

Thus, given observed values for all the Text(c_i) attributes, our goal is to infer an assignment of values to the PubCited attributes — which induces a partition of the citations into coreferring groups. To get a sense of how this process works, consider the two preceding citations. One hypothesis, H_1, is that the two citations c_1 and c_2 refer to a single publication p_1, which has "genetic" in its title. An alternative, H_2, is that there is an additional publication p_2 whose title is identical except for the omission of "genetic," and c_2 refers to p_2 instead. H_1 obviously involves an unlikely event — a word being left out of a citation; this is reflected in the probability of Text(c_2) given Title(p_1). But the probability of H_2 involves an additional factor for Title(p_2), reflecting the prior probability of the string "A general model for the analysis of pedigree data" under our model of academic paper titles. Since there are so many possible titles, this probability will be extremely small, allowing H_1 to win out. As this example shows, probabilistic models of this form exhibit



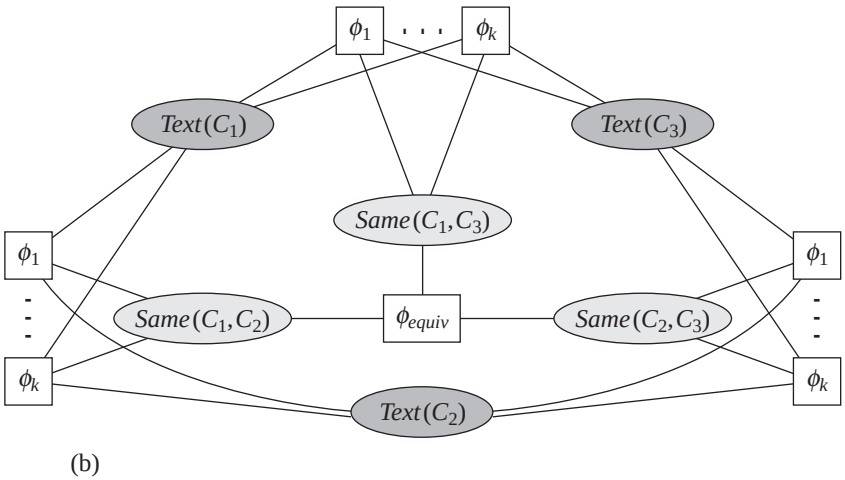


Figure 6.D.1 — Two template models for citation-matching (a) A directed model. (b) An undirected model instantiated for three citations.

a built-in Ockham's razor effect: the highest probability goes to hypotheses that do not include any more objects — and hence any more instantiated attributes — than necessary to explain the observed data.

Another line of work (for example, Wellner et al. (2004)) tackle the citation-matching problem using undirected template models, whose ground instantiation is a CRF (as in section 4.6.1). As we saw in the main text, one approach is to eliminate the Author *and* Publication *classes and simply reason about a relation Same(C, C') between citations (constrained to be an equivalence relation). Figure 6.D.1b shows an instantiation of such a model for three citations. For each pair of citations C, C', there is an array of factors $\phi_1, \ldots, \phi_k$ that look at various features of Text(C) and Text(C') — whether they have same surname for the first author, whether their titles are within an edit distance of two, and so on — and relate these features to Same(C_1, C_2). These factors encode preferences for and against coreference more explicitly than the factors in the directed model.*

However, as we have discussed, a reference-only model produces overly peaked posteriors that are very sensitive to parameters and to the number of mentions. Moreover, there are some examples where pairwise compatibility factors are insufficient for finding the right partition. For instance, suppose we have three references to people: "Jane," which is clearly a female's given name; "Smith," which is clearly a surname; and "Stanley," which could be a surname or a male's given name. Any pair of these references could refer to the same person: there could easily be a Jane Smith, a Stanley Smith, or a Jane Stanley. But it is unlikely that all three names corefer. Thus, a reasonable approach uses an undirected model that has explicit (hidden) variables for each entity and its attributes. The same potentials can be used as in the reference-only model. However, due to the use of undirected dependencies, we can allow the use of a much richer feature set, as described in box 4.E.

Systems that use template-based probabilistic models can now achieve accuracies in the high 90s for identifying coreferent citations. Identifying multiple mentions of the same author is harder; accuracies vary considerably depending on the data set, but tend to be around 70 percent. These models are also useful for segmenting citations into fields such as the title, author names, journal, and date. This is done by treating the citation text not as a single attribute but as a sequence of tokens (words and punctuation marks), each of which has an associated variable indicating which field it belongs to. These "field" variables can be thought of as the state variables in a hidden Markov model in the directed setting, or a conditional random field in the undirected setting (as in box 4.E). The resulting model can segment ambiguous citations more accurately than one that treats each citation in isolation, because it prefers for segmentations of coreferring citations to be consistent.

6.7 Summary

The representation languages discussed in earlier chapters — Bayesian networks and Markov networks — allow us to write down a model that encodes a specific probability distribution over a fixed, finite set of random variables. In this chapter, we have provided a general framework for defining *templates* for fragments of the probabilistic model. These templates can be reused both within a single model, and across multiple models of different structures. **Thus, a template-based representation language allows us to encode a potentially infinite set of distributions, over arbitrarily large probability spaces. The rich models that one can**

produce from such a representation can capture complex interactions between many interrelated objects, and thus utilize many pieces of evidence that we may otherwise ignore; as we have seen, these pieces of evidence can provide substantial improvements in the quality of our predictions.

We described several different representation languages: one specialized to temporal representations, and several that allow the specification of models over general object-relational domains. In the latter category, we first described two directed representations: plate models, and probabilistic relational models. The latter allow a considerably richer set of dependencies to be encoded, but at the cost of both conceptual and computational complexity. We also described an undirected representation, which, by avoiding the need to guarantee acyclicity and coherent local probability models, avoids some of the complexities of the directed models. As we discussed, the flexibility of undirected models is particularly valuable when we want to encode a probability distribution over richer representations, such as the structure of the relational graph.

There are, of course, other ways to produce these large, richly structured models. Most obviously, for any given application, we can define a procedural method that can take a skeleton, and produce a concrete model for that specific set of objects (and possibly relations). For example, we can easily build a program that takes a pedigree and produces a Bayesian network for genetic inheritance over that pedigree. The benefit of the template-based representations that we have described here is that they provide a uniform, modular, declarative language for models of this type. Unlike specialized representations, such a language allows the template-based model to be modified easily, whether by hand or as part of an automated learning algorithm. Indeed, learning is perhaps one of the key advantages of the template-based representations. In particular, as we will discuss, the model is learned at the template level, allowing a model to be learned from a domain with one set of objects, and applied seamlessly to a domain with a completely different set of objects (see section 17.5.1.2 and section 18.6.2).

In addition, by making objects and relations first-class citizens in the model, we have laid a foundation for the option of allowing probability distributions over probability spaces that are significantly richer than simply properties of objects. For example, as we saw, we can consider modeling uncertainty about the network of interrelationships between objects, and even about the actual set of objects included in our domain. These extensions raise many important and difficult questions regarding the appropriate type of distribution that one should use for such richly structured probability spaces. These questions become even more complex as we introduce more of the expressive power of relational languages, such as function symbols, quantifiers, and more. These issues are an active area of research.

These representations also raise important questions regarding inference. At first glance, the problem appears straightforward: The semantics for each of our representation languages depends on instantiating the template-based model to produce a specific ground network; clearly, we can simply run standard inference algorithms on the resulting network. This approach is has been called *knowledge-based model construction*, because a knowledge-base (or skeleton) is used to construct a model. However, this approach is problematic, because the models produced by this process can pose a significant challenge to inference algorithms. First, the network produced by this process is often quite large — much larger than models that one can reasonably construct by hand. Second, such models are often quite densely connected, due to the multiple interactions between variables. Finally, structural uncertainty, both about the relations and about the presence of objects, also makes for densely connected models. On the

knowledge-based model construction

other side, such models often have unique characteristics, such as multiple similar fragments across the network, or large amounts of context-specific independence, which could, perhaps, be exploited by an appropriate choice of inference algorithm. Chapter 15 presents some techniques for addressing the inference problems in temporal models. The question of inference in the models defined by the object-relational frameworks — and specifically of inference algorithms that exploit their special structure — is very much a topic of current work.

6.8 Relevant Literature

Probabilistic models of temporal processes go back many years. Hidden Markov models were discussed as early as Rabiner and Juang (1986), and expanded on in Rabiner (1989). Kalman filters were first described by Kalman (1960). The first temporal extension of probabilistic graphical models is due to Dean and Kanazawa (1989), who also coined the term *dynamic Bayesian network*. Much work has been done on defining various representations that are based on hidden Markov models or on dynamic Bayesian networks; these include generalizations of the basic framework, or special cases that allow more tractable inference. Examples include mixed-memory Markov models (Saul and Jordan 1999); variable-duration HMMs (Rabiner 1989) and their extension segment models (Ostendorf et al. 1996); factorial HMMs (Ghahramani and Jordan 1997); and hierarchical HMMs (Fine et al. 1998; Bui et al. 2001). Smyth, Heckerman, and Jordan (1997) is a review paper that was influential in providing a clear exposition of the connections between HMMs and DBNs. Murphy and Paskin (2001) show how hierarchical HMMs can be reduced to DBNs, a connection that provided a much faster inference algorithm than previously proposed for this representation. Murphy (2002) provides an excellent tutorial on the topics of dynamic Bayesian networks and related representations.

continuous time Bayesian network

Nodelman et al. (2002, 2003) build on continuous-time Markov processes to define *continuous time Bayesian networks*. As the name suggests, this representation is similar to a dynamic Bayesian network but encodes a probability distribution over trajectories over a continuum of time points.

The topic of integrating object-relational frameworks and probabilistic representations has received much attention over the past few years. Getoor and Taskar (2007) contains reviews of many of the important contributions, and citations to others. Work on this topic goes back to the idea of *knowledge-based model construction*, which was proposed in the early 1990s; Wellman, Breese, and Goldman (1992) review some of this earlier work. These ideas were then extended and formalized, using logic programming as a foundation (Poole 1993a; Ngo and Haddawy 1996; Kersting and De Raedt 2007).

knowledge-based model construction

Plate models were introduced by Gilks, Thomas, and Spiegelhalter (1994) and Buntine (1994) as a language for sharing parameters within and between models. Probabilistic relational models were proposed in Friedman et al. (1999); see Getoor et al. (2007) for a more detailed presentation. Heckerman, Meek, and Koller (2007) define a language that unifies plate models and probabilistic relational models, which was the inspiration for our presentation of PRMs in terms of contingent dependencies.

Undirected probabilistic models for relational domains originated with the framework of relational Markov networks of Taskar et al. (2002, 2007). Richardson and Domingos (2006) provide a particularly elegant representation of features, in terms of logical formulas. In a Markov logic

network (MLN), there is no separation between the specification of cliques and the specification of features in the potential. Rather, the model is defined in terms of a collection of logical formulas, each associated with a weight.

Getoor et al. (2002) discuss some strategies for modeling structural uncertainty in a directed setting. Taskar et al. (2002) investigate the same issues in an undirected setting, and demonstrate the advantages of the increased flexibility. Reasoning about object identity has been used in various applications, including data association (Pasula et al. 1999), coreference resolution in natural language text (McCallum and Wellner 2005; Culotta et al. 2007), and the citation matching application discussed in box 6.D (Pasula et al. 2002; Wellner et al. 2004; Milch et al. 2004; Poon and Domingos 2007). Milch et al. (2005, 2007) define BLOG (Bayesian Logic), a directed language explicitly designed to model uncertainty over the number of objects in the domain.

In addition to the logic-based representations we discuss in this chapter, a very different perspective on incorporating template-based structure in probabilistic models utilizes a programming-language framework. Here, we can view a random variable as a stochastic function from its inputs (its parents) to its output. If we explicitly define the stochastic function, one can then reuse it in in multiple places. More importantly, one can define functions that call other functions, or perhaps even functions that recursively call themselves. Important languages based on this framework include *probabilistic context-free grammars*, which play a key role in statistical models for natural language (see, for example, Manning and Schuetze (1999)) and in modeling RNA secondary structure (see, for example, Durbin et al. 1998), and object-oriented Bayesian networks (Koller and Pfeffer 1997; Pfeffer et al. 1999), which generalizes encapsulated Bayesian networks to allow for repeated elements.

probabilistic context-free grammar

6.9 Exercises

Exercise 6.1

Consider a temporal process where the state variables at time t depend directly not only on the variables at time $t-1$, but rather on the variables at time $t-1, \ldots, t-k$ for some fixed k. Such processes are called *semi-Markov* of order k.

semi-Markov order k

a. Extend definition 6.3 and definition 6.4 to richer notions, that encode such a kth order semi-Markov processes.

b. Show how you can convert a kth order Markov process to a regular (first-order) Markov process representable by a DBN over an extended set of state variables. Describe both the variables and the transition model.

Exercise 6.2⋆

Markov models of different orders are the standard representation of text sequences. For example, in a first-order Markov model, we define our distribution over word sequences in terms of a probability $P(W^{(t)} \mid W^{(t-1)})$. This model is also called a *bigram model*, because it requires that we collected statistics over pairs of words. A second-order Markov model, often called a *trigram model*, defines the distribution is terms of a probability $P(W^{(t)} \mid W^{(t-1)}, W^{(t-2)})$.

Unfortunately, because the set of words in our vocabulary is very large, trigram models define very large CPDs with very many parameters. These are very hard to estimate reliably from data (see section 17.2.3). One approach for producing more robust estimates while still making use of higher-order dependencies is *shrinkage*. Here, we define our transition model to be a weighted average of transition models of different

shrinkage

orders:

$$P(W^{(t)} \mid W^{(t-1)}, W^{(t-2)}) = \alpha_0(W^{(t-1)}, W^{(t-2)})Q_0(W^{(t)}) + \alpha_1(W^{(t-1)}, W^{(t-2)})Q_1(W^{(t)} \mid W^{(t-1)}) + \alpha_2(W^{(t-1)}, W^{(t-2)})Q_2(W^{(t)} \mid W^{(t-1)}, W^{(t-2)}),$$

where the Q_i's are different transition models, and the α_i's are nonnegative coefficients such that, for every $W^{(t-1)}, W^{(t-2)}$,

$$\alpha_0(W^{(t-1)}, W^{(t-2)}) + \alpha_1(W^{(t-1)}, W^{(t-2)}) + \alpha_2(W^{(t-1)}, W^{(t-2)}) = 1.$$

mixed-memory HMM

Show how we can construct a DBN model that gives rise to equivalent dynamics using standard CPDs, by introducing a new hidden variable $S^{(t)}$. This model is called *mixed-memory HMM*.

Exercise 6.3

In this exercise, we construct an HMM model that allows for a richer class of distributions over the duration for which the process stays in a given state.

a. Consider an HMM where the hidden variable has k states, and let $P(s'_j \mid s_i)$ denote the transition model. Assuming that the process is at state s_i at time t, what is the distribution over the number of steps until it first transitions out of state s_i (that is, the smallest number d such that $S^{(t+d)} \neq s_i$).

b. Construct a DBN model that allows us to incorporate an arbitrary distribution over the duration d_i that a process stays in state s_i after it first transitions to s_i. Your model should allow the distribution over d_i to depend on s_i. Do not worry about parameterizing the distribution over d_i. (Hint: Your model can include variables whose value changes deterministically.) This type of model is called a *duration HMM*.

duration HMM

Exercise 6.4⋆

segment HMM

A *segment HMM* is a Markov chain over the hidden states, but where each state emits not a single symbol as output, but rather a string of unknown length. Thus, at each state $S^{(t)} = s$, the model selects a segment length $L^{(t)}$, using a distribution that can depend on s. The model then emits a segment $Y^{(t,1)}, \ldots, Y^{(t,L^{(t)})}$ of length $L^{(t)}$. In this exercise, we assume that the distribution on the output segment is modeled by a separate HMM $\mathcal{H}_s$. Write down a 2-TBN model that encodes this model. (Hint: Use your answer to exercise 6.3.)

Exercise 6.5⋆

hierarchical HMM

A *hierarchical HMM* is similar to the segment HMM, except that there is no explicit selection of the segment length. Rather, the HMM at a state calls a "subroutine" HMM $\mathcal{H}_s$ that defines the output at the state s; when the "subroutine" HMM enters a finish-state, the control returns to the top-level HMM, which then transitions to its next state. This hierarchical HMM (with three levels) is precisely the framework used as the standard speech recognition architecture.

a. Show how a three-level hierarchical HMM can be represented as a DBN. (Hint: Use "finish variables" — binary variables that are true when a lower-level HMMs finishes its transition.)

b. Explain how you would modify the hierarchical HMM framework to deal with a motion tracking task, where, for example, the higher-level HMM represents motion between floors, the mid-level HMM motion between corridors, and the lowest-level HMM motion between rooms. (Hint: Consider situations where there are multiple staircases between floors.)

Exercise 6.6⋆

data association

Consider the following *data association* problem. We track K moving objects $u_1, \ldots, u_K$, using readings obtained over a trajectory of length T. Each object k has some (unknown) basic appearance A_k, and some position $X_k^{(t)}$ at every time point t. Our sensor provides, at each time point t, a set of L noisy sensor

readings, each corresponding to one object: for each $l = 1, \ldots, L$, it returns $B_l^{(t)}$ — the measured object appearance, and $Y_l^{(t)}$ — the measured object position. Unfortunately, our sensor cannot determine the identity of the sensed objects, so sensed object l does not generally correspond to the true object l. In fact, the labeling of the sensed objects is completely arbitrary — all labelings are equally likely.

Write down a DBN that represents the dynamics of this model.

Exercise 6.7

Consider a template-level CPD where $A(U)$ depends on $B(U, V)$, allowing for situations where the ground variable $A(u)$ can depend on unbounded number of ground variables $B(u, v)$. As discussed in the text, we can specify the parameterization for the resulting CPD in various ways: we can use a symmetric noisy-or or sigmoid model, or define a dependency of $A(u)$ on some aggregated statistics of the parent set $\{B(u, v)\}$. Assume that both $A(U)$ and $B(U, V)$ are binary-valued.

Show that both a symmetric noisy-or model and a symmetric logistic model can be formulated easily using an *aggregator CPDs*.

aggregator CPD

Exercise 6.8

Consider the template dependency graph for a model $\mathcal{M}_{PRM}$, as specified in definition 6.13. Show that if the template dependency graph is acyclic, then for any skeleton κ, the ground network $\mathcal{B}_\kappa^{\mathcal{M}_{PRM}}$ is also acyclic.

Exercise 6.9

Let $\mathcal{M}_{Plate}$ be a plate model, and assume that its template dependency graph contains a cycle. Let κ be any skeleton such that $\mathcal{O}^\kappa[\mathsf{Q}] \neq \emptyset$ for every class Q. Show that $\mathcal{B}_\kappa^{\mathcal{M}_{Plate}}$ is necessarily cyclic.

Exercise 6.10⋆⋆

Consider the cyclic dependency graph for the Genetics model shown in figure 6.9b. Clearly, for any valid pedigree — one where a person cannot be his or her own ancestor — the ground network is acyclic. We now describe a refinement of the dependency graph structure that would allow us to detect such acyclicity in this and other similar settings. Here, we assume for simplicity that all attributes in the guards are part of the relational skeleton, and therefore not part of the probabilistic model.

Let γ denote a tuple of objects from our skeleton. Assume that we have some prior knowledge about our domain in the following form: for any skeleton κ, there necessarily exists a partial ordering $\prec$ on tuples of objects γ that is transitive ($\gamma_1 \prec \gamma_2$ and $\gamma_2 \prec \gamma_3$ implies $\gamma_1 \prec \gamma_3$) and irreflexive ($\gamma \not\prec \gamma$). For example, in the Genetics example, we can use ancestry to define our ordering, where $u' \prec u$ whenever u' is an ancestor of u. We further assume that some of the guards used in the probabilistic model imply ordering constraints.

More precisely, let $B(\boldsymbol{U}') \in \mathrm{Pa}_{\boldsymbol{U}(A)}$. We say that a pair of assignments γ to $\boldsymbol{U}$ and γ' to $\boldsymbol{U}'$ is *valid* if they agree on the assignment to the overlapping variables in $\boldsymbol{U} \cap \boldsymbol{U}'$ and if they are consistent with the guard for A. The valid pairs are those that lead to actual edges $B(\gamma') \to A(\gamma)$ in the ground Bayesian network. (The definition here is slightly different than definition 6.12 because there γ' is an assignment to the variables in $\boldsymbol{U}'$ but not in $\boldsymbol{U}$.) We say that the dependence of A on B is *ordering-consistent* if, for any valid pair of assignments γ to $\boldsymbol{U}$ and γ' to $\boldsymbol{U}'$, we have that $\gamma' \prec \gamma$. Continuing our example, consider the dependence of *Genotype*(U) on *Genotype*(V) subject to the guard *Mother*(V, U). Here, for any pair of assignments u to U and v to V such that the guard *Mother*(v, u) holds, we have that $v \prec u$. Thus, this dependence is ordering-consistent.

We now define the following extension to our dependency graph. Let $\boldsymbol{U}'(B) \in \mathrm{Pa}_{\boldsymbol{U}(A)}$.

- If $\boldsymbol{U}' = \boldsymbol{U}$, we introduce an edge from B to A whose color is yellow.
- If the dependence is ordering-consistent, we introduce an edge from B to A whose color is green.
- Otherwise, we introduce an edge from B to A whose color is red.

Prove that if every cycle in the colored dependency graph for $\mathcal{M}_{PRM}$ has at least one green edge and no red edges, then for any skeleton satisfying the ordering constraints, the ground BN $\mathcal{B}_\kappa^{\mathcal{M}_{PRM}}$ is acyclic.

7 Gaussian Network Models

Although much of our presentation focuses on discrete variables, we mentioned in chapter 5 that the Bayesian network framework, and the associated results relating independencies to factorization of the distribution, also apply to continuous variables. The same statement holds for Markov networks. However, whereas table CPDs provide a general-purpose mechanism for describing any discrete distribution (albeit potentially not very compactly), the space of possible parameterizations in the case of continuous variables is essentially unbounded. In this chapter, we focus on a type of continuous distribution that is of particular interest: the class of multivariate Gaussian distributions. Gaussians are a particularly simple subclass of distributions that make very strong assumptions, such as the exponential decay of the distribution away from its mean, and the linearity of interactions between variables. While these assumptions are often invalid, Gaussians are nevertheless a surprisingly good approximation for many real-world distributions. Moreover, the Gaussian distribution has been generalized in many ways, to nonlinear interactions, or mixtures of Gaussians; many of the tools developed for Gaussians can be extended to that setting, so that the study of Gaussian provides a good foundation for dealing with a broad class of distributions.

In the remainder of this chapter, we first review the class of multivariate Gaussian distributions and some of its properties. We then discuss how a multivariate Gaussian can be encoded using probabilistic graphical models, both directed and undirected.

7.1 Multivariate Gaussians

7.1.1 Basic Parameterization

We have already described the univariate Gaussian distribution in chapter 2. We now describe its generalization to the multivariate case. As we discuss, there are two different parameterizations for a joint Gaussian density, with quite different properties.

The univariate Gaussian is defined in terms of two parameters: a mean and a variance. In its most common representation, a multivariate Gaussian distribution over $X_1, \ldots, X_n$ is characterized by an n-dimensional *mean vector* $\boldsymbol{\mu}$, and a symmetric $n \times n$ *covariance matrix* Σ; the density function is most often defined as:

mean vector

covariance matrix

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \tag{7.1}$$

where $|\Sigma|$ is the determinant of Σ.

standard Gaussian

We extend the notion of a *standard Gaussian* to the multidimensional case, defining it to be a Gaussian whose mean is the all-zero vector $\mathbf{0}$ and whose covariance matrix is the identity matrix I, which has 1's on the diagonal and zeros elsewhere. The multidimensional standard Gaussian is simply a product of independent standard Gaussians for each of the dimensions.

positive definite

In order for this equation to induce a well-defined density (that integrates to 1), the matrix Σ must be *positive definite*: for any $\boldsymbol{x} \in \mathbb{R}^n$ such that $\boldsymbol{x} \neq 0$, we have that $\boldsymbol{x}^T \Sigma \boldsymbol{x} > 0$. Positive definite matrices are guaranteed to be nonsingular, and hence have nonzero determinant, a necessary requirement for the coherence of this definition. A somewhat more complex definition can be used to generalize the multivariate Gaussian to the case of a *positive semi-definite* covariance matrix: for any $\boldsymbol{x} \in \mathbb{R}^n$, we have that $\boldsymbol{x}^T \Sigma \boldsymbol{x} \geq 0$. This extension is useful, since it allows for singular covariance matrices, which arise in several applications. For the remainder of our discussion, we focus our attention on Gaussians with positive definite covariance matrices.

positive semi-definite

information matrix

Because positive definite matrices are invertible, one can also utilize an alternative parameterization, where the Gaussian is defined in terms of its inverse covariance matrix $J = \Sigma^{-1}$, called *information matrix* (or *precision matrix*). This representation induces an alternative form for the Gaussian density. Consider the expression in the exponent of equation (7.1):

$$\begin{aligned} -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) &= -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T J (\boldsymbol{x} - \boldsymbol{\mu}) \\ &= -\frac{1}{2}\left[\boldsymbol{x}^T J \boldsymbol{x} - 2\boldsymbol{x}^T J \boldsymbol{\mu} + \boldsymbol{\mu}^T J \boldsymbol{\mu}\right]. \end{aligned}$$

The last term is constant, so we obtain:

$$p(\boldsymbol{x}) \propto \exp\left[-\frac{1}{2}\boldsymbol{x}^T J \boldsymbol{x} + (J\boldsymbol{\mu})^T \boldsymbol{x}\right]. \tag{7.2}$$

information form

This formulation of the Gaussian density is generally called the *information form*, and the vector $\boldsymbol{h} = J\boldsymbol{\mu}$ is called the *potential vector*. The information form defines a valid Gaussian density if and only if the information matrix is symmetric and positive definite, since Σ is positive definite if and only if Σ^{-1} is positive definite. The information form is useful in several settings, some of which are described here.

Intuitively, a multivariate Gaussian distribution specifies a set of ellipsoidal contours around the mean vector $\boldsymbol{\mu}$. The contours are parallel, and each corresponds to some particular value of the density function. The shape of the ellipsoid, as well as the "steepness" of the contours, are determined by the covariance matrix Σ. Figure 7.1 shows two multivariate Gaussians, one where the covariances are zero, and one where they are positive. As in the univariate case, the mean vector and covariance matrix correspond to the first two moments of the normal distribution. In matrix notation, $\boldsymbol{\mu} = \boldsymbol{E}[\boldsymbol{X}]$ and $\Sigma = \boldsymbol{E}[\boldsymbol{X}\boldsymbol{X}^T] - \boldsymbol{E}[\boldsymbol{X}]\boldsymbol{E}[\boldsymbol{X}]^T$. Breaking this expression down to the level of individual variables, we have that μ_i is the mean of X_i, $\Sigma_{i,i}$ is the variance of X_i, and $\Sigma_{i,j} = \Sigma_{j,i}$ (for $i \neq j$) is the *covariance* between X_i and X_j: $\boldsymbol{Cov}[X_i; X_j] = \boldsymbol{E}[X_i X_j] - \boldsymbol{E}[X_i]\boldsymbol{E}[X_j]$.

Example 7.1

Consider a particular joint distribution $p(X_1, X_2, X_3)$ over three random variables. We can

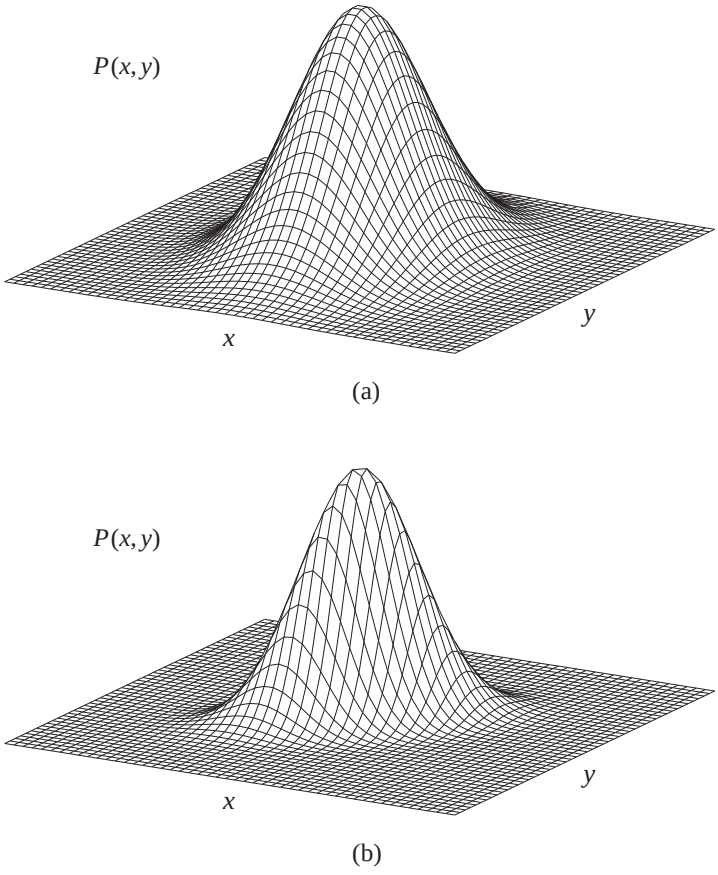


Figure 7.1 Gaussians over two variables X and Y. (a) X and Y uncorrelated. (b) X and Y correlated.

parameterize it via a mean vector $\boldsymbol{\mu}$ and a covariance matrix Σ:

$$\boldsymbol{\mu} = \begin{pmatrix} 1 \\ -3 \\ 4 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 4 & 2 & -2 \\ 2 & 5 & -5 \\ -2 & -5 & 8 \end{pmatrix}$$

As we can see, the covariances $\mathbf{Cov}[X_1; X_3]$ and $\mathbf{Cov}[X_2; X_3]$ are both negative. Thus, X_3 is negatively correlated with X_1: when X_1 goes up, X_3 goes down (and similarly for X_3 and X_2). ■

7.1.2 Operations on Gaussians

There are two main operations that we wish to perform on a distribution: compute the marginal distribution over some subset of the variables $\boldsymbol{Y}$, and conditioning the distribution on some assignment of values $\boldsymbol{Z} = \boldsymbol{z}$. It turns out that each of these operations is very easy to perform in one of the two ways of encoding a Gaussian, and not so easy in the other.

Marginalization is trivial to perform in the covariance form. Specifically, the marginal Gaussian distribution over any subset of the variables can simply be read from the mean and covariance matrix. For instance, in example 7.1, we can obtain the marginal Gaussian distribution over X_2 and X_3 by simply considering only the relevant entries in both the mean vector the covariance matrix. More generally, assume that we have a joint normal distribution over $\{\boldsymbol{X}, \boldsymbol{Y}\}$ where $\boldsymbol{X} \in I\!R^n$ and $\boldsymbol{Y} \in I\!R^m$. Then we can decompose the mean and covariance of this joint distribution as follows:

$$p(\boldsymbol{X}, \boldsymbol{Y}) = \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu_X} \\ \boldsymbol{\mu_Y} \end{pmatrix}; \begin{bmatrix} \Sigma_{\boldsymbol{XX}} & \Sigma_{\boldsymbol{XY}} \\ \Sigma_{\boldsymbol{YX}} & \Sigma_{\boldsymbol{YY}} \end{bmatrix}\right) \tag{7.3}$$

where $\boldsymbol{\mu_X} \in I\!R^n$, $\boldsymbol{\mu_Y} \in I\!R^m$, $\Sigma_{\boldsymbol{XX}}$ is a matrix of size $n \times n$, $\Sigma_{\boldsymbol{XY}}$ is a matrix of size $n \times m$, $\Sigma_{\boldsymbol{YX}} = \Sigma_{\boldsymbol{XT}}^T$ is a matrix of size $m \times n$ and $\Sigma_{\boldsymbol{YY}}$ is a matrix of size $m \times m$.

Lemma 7.1

Let $\{\boldsymbol{X}, \boldsymbol{Y}\}$ have a joint normal distribution defined in equation (7.3). Then the marginal distribution over $\boldsymbol{Y}$ is a normal distribution $\mathcal{N}(\boldsymbol{\mu_Y}; \Sigma_{\boldsymbol{YY}})$.

The proof follows directly from the definitions (see exercise 7.1).

On the other hand, conditioning a Gaussian on an observation $\boldsymbol{Z} = \boldsymbol{z}$ is very easy to perform in the information form. We simply assign the values $\boldsymbol{Z} = \boldsymbol{z}$ in equation (7.2). This process turns some of the quadratic terms into linear terms or even constant terms, and some of the linear terms into constant terms. The resulting expression, however, is still in the same form as in equation (7.2), albeit over a smaller subset of variables.

In summary, although the two representations both encode the same information, they have different computational properties. **To marginalize a Gaussian over a subset of the variables, one essentially needs to compute their pairwise covariances, which is precisely generating the distribution in its covariance form. Similarly, to condition a Gaussian on an observation, one essentially needs to invert the covariance matrix to obtain the information form. For small matrices, inverting a matrix may be feasible, but in high-dimensional spaces, matrix inversion may be far too costly.**

7.1.3 Independencies in Gaussians

For multivariate Gaussians, independence is easy to determine directly from the parameters of the distribution.

Theorem 7.1

Let $\boldsymbol{X} = X_1, ..., X_n$ have a joint normal distribution $\mathcal{N}(\boldsymbol{\mu}; \Sigma)$. Then X_i and X_j are independent if and only if $\Sigma_{i,j} = 0$.

The proof is left as an exercise (exercise 7.2).

Note that this property does not hold in general. In other words, if $p(X, Y)$ is not Gaussian, then it is possible that $\boldsymbol{Cov}[X; Y] = 0$ while X and Y are still dependent in p. (See exercise 7.2.)

At first glance, it seems that conditional independencies are not quite as apparent as marginal independencies. However, it turns out that the independence structure in the distribution is apparent not in the covariance matrix, but in the information matrix.

Theorem 7.2 *Consider a Gaussian distribution $p(X_1, \ldots, X_n) = \mathcal{N}(\boldsymbol{\mu}; \Sigma)$, and let $J = \Sigma^{-1}$ be the information matrix. Then $J_{i,j} = 0$ if and only if $p \models (X_i \perp X_j \mid \mathcal{X} - \{X_i, X_j\})$.*

The proof is left as an exercise (exercise 7.3).

Example 7.2 *Consider the covariance matrix of example 7.1. Simple algebraic operations allow us to compute its inverse:*

$$J = \begin{pmatrix} 0.3125 & -0.125 & 0 \\ -0.125 & 0.5833 & 0.3333 \\ 0 & 0.3333 & 0.3333 \end{pmatrix}$$

As we can see, the entry in the matrix corresponding to X_1, X_3 is zero, reflecting the fact that they are conditionally independent given X_2. ■

Theorem 7.2 asserts the fact that the information matrix captures independencies between pairs of variables, conditioned on all of the remaining variables in the model. These are precisely the same independencies as the pairwise Markov independencies of definition 4.10. Thus, we can view the information matrix J for a Gaussian density p as precisely capturing the pairwise Markov independencies in a Markov network representing p. Because a Gaussian density is a positive distribution, we can now use theorem 4.5 to construct a Markov network that is a unique minimal I-map for p: As stated in this theorem, the construction simply introduces an edge between X_i and X_j whenever $(X_i \perp X_j \mid \mathcal{X} - \{X_i, X_j\})$ does not hold in p. But this latter condition holds precisely when $J_{i,j} \neq 0$. **Thus, we can view the information matrix as directly defining a minimal I-map Markov network for p, whereby nonzero entries correspond to edges in the network.**

7.2 Gaussian Bayesian Networks

We now show how we can define a continuous joint distribution using a Bayesian network. This representation is based on the *linear Gaussian model*, which we defined in definition 5.14. Although this model can be used as a CPD within any network, it turns out that continuous networks defined solely in terms of linear Gaussian CPDs are of particular interest:

Definition 7.1 Gaussian Bayesian network

We define a Gaussian Bayesian network *to be a Bayesian network all of whose variables are continuous, and where all of the CPDs are linear Gaussians.* ■

An important and surprising result is that linear Gaussian Bayesian networks are an alternative representation for the class of multivariate Gaussian distributions. This result has two parts. The first is that a linear Gaussian network always defines a joint multivariate Gaussian distribution.

Theorem 7.3 *Let Y be a linear Gaussian of its parents $X_1, \ldots, X_k$:*

$$p(Y \mid \boldsymbol{x}) = \mathcal{N}\left(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}; \sigma^2\right).$$

Assume that $X_1, \ldots, X_k$ are jointly Gaussian with distribution $\mathcal{N}(\boldsymbol{\mu}; \Sigma)$. Then:

- *The distribution of Y is a normal distribution $p(Y) = \mathcal{N}\left(\mu_Y; \sigma_Y^2\right)$ where:*

$$\begin{aligned} \mu_Y &= \beta_0 + \boldsymbol{\beta}^T \boldsymbol{\mu} \\ \sigma_Y^2 &= \sigma^2 + \boldsymbol{\beta}^T \Sigma \boldsymbol{\beta}. \end{aligned}$$

- *The joint distribution over $\{\boldsymbol{X}, Y\}$ is a normal distribution where:*

$$\mathbf{Cov}[X_i; Y] = \sum_{j=1}^{k} \beta_j \Sigma_{i,j}.$$

From this theorem, it follows easily by induction that if $\mathcal{B}$ is a linear Gaussian Bayesian network, then it defines a joint distribution that is jointly Gaussian.

Example 7.3

Consider the linear Gaussian network $X_1 \to X_2 \to X_3$, where

$$\begin{aligned} p(X_1) &= \mathcal{N}(1; 4) \\ p(X_2 \mid X_1) &= \mathcal{N}(0.5X_1 - 3.5; 4) \\ p(X_3 \mid X_2) &= \mathcal{N}(-X_2 + 1; 3). \end{aligned}$$

Using the equations in theorem 7.3, we can compute the joint Gaussian distribution $p(X_1, X_2, X_3)$. For the mean, we have that:

$$\begin{aligned} \mu_2 &= 0.5\mu_1 - 3.5 = 0.5 \cdot 1 - 3.5 = -3 \\ \mu_3 &= (-1)\mu_2 + 1 = (-1) \cdot (-3) + 1 = 4. \end{aligned}$$

The variance of X_2 and X_3 can be computed as:

$$\begin{aligned} \Sigma_{22} &= 4 + (1/2)^2 \cdot 4 = 5 \\ \Sigma_{33} &= 3 + (-1)^2 \cdot 5 = 8. \end{aligned}$$

We see that the variance of the variable is a sum of two terms: the variance arising from its own Gaussian noise parameter, and the variance of its parent variables weighted by the strength of the dependence. Finally, we can compute the covariances as follows:

$$\begin{aligned} \Sigma_{12} &= (1/2) \cdot 4 = 2 \\ \Sigma_{23} &= (-1) \cdot \Sigma_{22} = -5 \\ \Sigma_{13} &= (-1) \cdot \Sigma_{12} = -2. \end{aligned}$$

The third equation shows that, although X_3 does not depend directly on X_1, they have a nonzero covariance. Intuitively, this is clear: X_3 depends on X_2, which depends on X_1; hence, we expect X_1 and X_3 to be correlated, a fact that is reflected in their covariance. As we can see, the covariance between X_1 and X_3 is the covariance between X_1 and X_2, weighted by the strength of the dependence of X_3 on X_2.

In general, putting these results together, we can see that the mean and covariance matrix for $p(X_1, X_2, X_3)$ is precisely our covariance matrix of example 7.1. ■

The converse to this theorem also holds: the result of conditioning is a normal distribution where there is a linear dependency on the conditioning variables. The expressions for converting a multivariate Gaussian to a linear Gaussian network appear complex, but they are based on simple algebra. They can be derived by taking the linear equations specified in theorem 7.3, and reformulating them as defining the parameters β_i in terms of the means and covariance matrix entries.

Theorem 7.4

Let $\{\boldsymbol{X}, Y\}$ have a joint normal distribution defined in equation (7.3). Then the conditional density

$$p(Y \mid \boldsymbol{X}) = \mathcal{N}\left(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{X}; \sigma^2\right),$$

is such that:

$$\begin{aligned} \beta_0 &= \mu_Y - \Sigma_{Y\boldsymbol{X}} \Sigma_{\boldsymbol{X}\boldsymbol{X}}^{-1} \boldsymbol{\mu}_{\boldsymbol{X}} \\ \boldsymbol{\beta} &= \Sigma_{\boldsymbol{X}\boldsymbol{X}}^{-1} \Sigma_{Y\boldsymbol{X}} \\ \sigma^2 &= \Sigma_{YY} - \Sigma_{Y\boldsymbol{X}} \Sigma_{\boldsymbol{X}\boldsymbol{X}}^{-1} \Sigma_{\boldsymbol{X}Y}. \end{aligned}$$

This result allows us to take a joint Gaussian distribution and produce a Bayesian network, using an identical process to our construction of a minimal I-map in section 3.4.1.

Theorem 7.5

Let $\mathcal{X} = \{X_1, \ldots, X_n\}$, and let p be a joint Gaussian distribution over $\mathcal{X}$. Given any ordering $X_1, \ldots, X_n$ over $\mathcal{X}$, we can construct a Bayesian network graph $\mathcal{G}$ and a Bayesian network $\mathcal{B}$ over $\mathcal{G}$ such that:

1. $\mathrm{Pa}_{X_i}^{\mathcal{G}} \subseteq \{X_1, \ldots, X_{i-1}\}$;
2. *the CPD of X_i in $\mathcal{B}$ is a linear Gaussian of its parents;*
3. *$\mathcal{G}$ is a minimal I-map for p.*

The proof is left as an exercise (exercise 7.4). As for the case of discrete networks, the minimal I-map is not unique: different choices of orderings over the variables will lead to different network structures. For example, the distribution in figure 7.1b can be represented either as the network where $X \rightarrow Y$ or as the network where $Y \rightarrow X$.

This equivalence between Gaussian distributions and linear Gaussian networks has important practical ramifications. On one hand, we can conclude that, for linear Gaussian networks, the joint distribution has a compact representation (one that is quadratic in the number of variables). Furthermore, the transformations from the network to the joint and back have a fairly simple and efficiently computable closed form. Thus, we can easily convert one representation to another, using whichever is more convenient for the current task. Conversely, **while the two representations are equivalent in their expressive power, there is not a one-to-one correspondence between their parameterizations. In particular, although in the worst case, the linear Gaussian representation and the Gaussian representation have the same number of parameters (exercise 7.6), there are cases where one representation can be significantly more compact than the other.**

Example 7.4

Consider a linear Gaussian network structured as a chain:

$$X_1 \rightarrow X_2 \rightarrow \cdots \rightarrow X_n.$$

Assuming the network parameterization is not degenerate (that is, the network is a minimal I-map of its distribution), we have that each pair of variables X_i, X_j are correlated. In this case, as shown in theorem 7.1, the covariance matrix would be dense — none of the entries would be zero. Thus, the representation of the covariance matrix would require a quadratic number of parameters. In the information matrix, however, for all X_i, X_j that are not neighbors in the chain, we have that X_i and X_j are conditionally independent given the rest of the variables in the network; hence, by theorem 7.2, $J_{i,j} = 0$. Thus, the information matrix has most of the entries being zero; the only nonzero entries are on the tridiagonal (the entries i, j for $j = i - 1, i, i + 1$). ■

However, not all structure in a linear Gaussian network is represented in the information matrix.

Example 7.5

In a v-structure $X \rightarrow Z \leftarrow Y$, we have that X and Y are marginally independent, but not conditionally independent given Z. Thus, according to theorem 7.2, the X, Y entry in the information matrix would not be 0. Conversely, because the variables are marginally independent, the X, Y entry in the covariance entry would be zero.

Complicating the example somewhat, assume that X and Y also have a joint parent W; that is, the network is structured as a diamond. In this case, X and Y are still not independent given the remaining network variables Z, W, and hence the X, Y entry in the information matrix is nonzero. Conversely, they are also not marginally independent, and thus the X, Y entry in the covariance matrix is also nonzero. ■

These examples simply recapitulate, in the context of Gaussian networks, the fundamental difference in expressive power between Bayesian networks and Markov networks.

7.3 Gaussian Markov Random Fields

We now turn to the representation of multivariate Gaussian distributions via an undirected graphical model. We first show how a Gaussian distribution can be viewed as an MRF. This formulation is derived almost immediately from the information form of the Gaussian. Consider again equation (7.2). We can break up the expression in the exponent into two types of terms: those that involve single variables X_i and those that involve pairs of variables X_i, X_j. The terms that involve only the variable X_i are:

$$-\frac{1}{2} J_{i,i} x_i^2 + h_i x_i, \tag{7.4}$$

where we recall that the potential vector $\boldsymbol{h} = J\boldsymbol{\mu}$. The terms that involve the pair X_i, X_j are:

$$-\frac{1}{2}[J_{i,j} x_i x_j + J_{j,i} x_j x_i] = -J_{i,j} x_i x_j, \tag{7.5}$$

due to the symmetry of the information matrix. Thus, the information form immediately induces a pairwise Markov network, whose node potentials are derived from the potential vector and the

diagonal elements of the information matrix, and whose edge potentials are derived from the off-diagonal entries of the information matrix. We also note that, when $J_{i,j} = 0$, there is no edge between X_i and X_j in the model, corresponding directly to the independence assumption of the Markov network.

Gaussian MRF

Thus, any Gaussian distribution can be represented as a pairwise Markov network with quadratic node and edge potentials. This Markov network is generally called a *Gaussian Markov random field (GMRF)*. Conversely, consider any pairwise Markov network with quadratic node and edge potentials. Ignoring constant factors, which can be assimilated into the partition function, we can write the node and edge energy functions (log-potentials) as:

$$\begin{aligned} \epsilon_i(x_i) &= d_0^i + d_1^i x_i + d_2^i x_i^2 \\ \epsilon_{i,j}(x_i, x_j) &= a_{00}^{i,j} + a_{01}^{i,j} x_i + a_{10}^{i,j} x_j + a_{11}^{i,j} x_i x_j + a_{02}^{i,j} x_i^2 + a_{20}^{i,j} x_j^2, \end{aligned} \tag{7.6}$$

where we used the log-linear notation of section 4.4.1.2. By aggregating like terms, we can reformulate any such set of potentials in the log-quadratic form:

$$p'(\boldsymbol{x}) = \exp(-\frac{1}{2}\boldsymbol{x}^T J \boldsymbol{x} + \boldsymbol{h}^T \boldsymbol{x}), \tag{7.7}$$

where we can assume without loss of generality that J is symmetric. This Markov network defines a valid Gaussian density if and only if J is a positive definite matrix. If so, then J is a legal information matrix, and we can take $\boldsymbol{h}$ to be a potential vector, resulting in a distribution in the form of equation (7.2).

However, unlike the case of Gaussian Bayesian networks, it is not the case that every set of quadratic node and edge potentials induces a legal Gaussian distribution. Indeed, the decomposition of equation (7.4) and equation (7.5) can be performed for any quadratic form, including one not corresponding to a positive definite matrix. For such matrices, the resulting function $\exp(\boldsymbol{x}^T A \boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x})$ will have an infinite integral, and cannot be normalized to produce a valid density. Unfortunately, **other than generating the entire information matrix and testing whether it is positive definite, there is no simple way to check whether the MRF is valid. In particular, there is no local test that can be applied to the network parameters that precisely characterizes valid Gaussian densities.** However, there *are* simple tests that are sufficient to induce a valid density. While these conditions are not necessary, they appear to cover many of the cases that occur in practice.

We first provide one very simple test that can be verified by direct examination of the information matrix.

Definition 7.2

diagonally dominant

A quadratic MRF parameterized by J is said to be diagonally dominant *if, for all i,*

$$\sum_{j \neq i} |J_{i,j}| < J_{i,i}.$$

■

For example, the information matrix in example 7.2 is diagonally dominant; for instance, for $i = 2$ we have:

$$|-0.125| + 0.3333 < 0.5833.$$

One can now show the following result:

Proposition 7.1 *Let $p'(\boldsymbol{x}) = \exp(-\frac{1}{2}\boldsymbol{x}^T J \boldsymbol{x} + \boldsymbol{h}^T \boldsymbol{x})$ be a quadratic pairwise MRF. If J is diagonally dominant, then p' defines a valid Gaussian MRF.*

The proof is straightforward algebra and is left as an exercise (exercise 7.8).

The following condition is less easily verified, since it cannot be tested by simple examination of the information matrix. Rather, it checks whether the distribution can be written as a quadratic pairwise MRF whose node and edge potentials satisfy certain conditions. Specifically, recall that a Gaussian MRF consists of a set of node potentials, which are log-quadratic forms in x_i, and a set of edge potentials, which are log-quadratic forms in x_i, x_j. We can state a condition in terms of the coefficients for the nonlinear components of this parameterization:

Definition 7.3 *A quadratic MRF parameterized as in equation (7.6) is said to be* pairwise normalizable *if:*

pairwise normalizable

- *for all i, $d_2^i > 0$;*
- *for all i, j, the 2×2 matrix*

$$\begin{pmatrix} a_{02}^{i,j} & a_{11}^{i,j}/2 \\ a_{11}^{i,j}/2 & a_{20}^{i,j} \end{pmatrix}$$

is positive semidefinite. ■

Intuitively, this definition states that each edge potential, considered in isolation, is normalizable (hence the name "pairwise-normalizable").

We can show the following result:

Proposition 7.2 *Let $p'(\boldsymbol{x})$ be a quadratic pairwise MRF, parameterized as in equation (7.6). If p' is pairwise normalizable, then it defines a valid Gaussian distribution.*

Once again, the proof follows from standard algebraic manipulations, and is left as an exercise (exercise 7.9).

We note that, like the preceding conditions, this condition is sufficient but not necessary:

Example 7.6 *Consider the following information matrix:*

$$\begin{pmatrix} 1 & 0.6 & 0.6 \\ 0.6 & 1 & 0.6 \\ 0.6 & 0.6 & 1 \end{pmatrix}$$

It is not difficult to show that this information matrix is positive definite, and hence defines a legal Gaussian distribution. However, it turns out that it is not possible to decompose this matrix into a set of three edge potentials, each of which is positive definite. ■

Unfortunately, evaluating whether pairwise normalizability holds for a given MRF is not always trivial, since it can be the case that one parameterization is not pairwise normalizable, yet a different parameterization that induces precisely the same density function is pairwise normalizable.

Example 7.7

Consider the information matrix of example 7.2, with a mean vector $\mathbf{0}$. *We can define this distribution using an MRF by simply choosing the node potential for* X_i *to be* $J_{i,i}x_i^2$ *and the edge potential for* X_i, X_j *to be* $2J_{i,j}x_ix_j$. *Clearly, the* X_1, X_2 *edge does* not *define a normalizable density over* X_1, X_2, *and hence this MRF is not pairwise normalizable. However, as we discussed in the context of discrete MRFs, the MRF parameterization is nonunique, and the same density can be induced using a continuum of different parameterizations. In this case, one alternative parameterization of the same density is to define all node potentials as* $\epsilon_i(x_i) = 0.05x_i^2$, *and the edge potentials to be* $\epsilon_{1,2}(x_1, x_2) = 0.2625x_1^2 + 0.0033x_2^2 - 0.25x_1x_2$, *and* $\epsilon_{2,3}(x_2, x_3) = 0.53x_2^2 + 0.2833x_3^2 + 0.6666x_2x_3$. *Straightforward arithmetic shows that this set of potentials induces the information matrix of example 7.2. Moreover, we can show that this formulation is pairwise normalizable: The three node potentials are all positive, and the two edge potentials are both positive definite. (This latter fact can be shown either directly or as a consequence of the fact that each of the edge potentials is diagonally dominant, and hence also positive definite.)* ■

This example illustrates that the pairwise normalizability condition is easily checked for a specific MRF parameterization. However, if our aim is to encode a particular Gaussian density as an MRF, we may have to actively search for a decomposition that satisfies the relevant constraints. If the information matrix is small enough to manipulate directly, this process is not difficult, but if the information matrix is large, finding an appropriate parameterization may incur a nontrivial computational cost.

7.4 Summary

This chapter focused on the representation and independence properties of Gaussian networks.

We showed an equivalence of expressive power between three representational classes: multivariate Gaussians, linear Gaussian Bayesian networks, and Gaussian MRFs. In particular, any distribution that can be represented in one of those forms can also be represented in another. We provided closed-form formulas that allow us convert between the multivariate Gaussian representation and the linear Gaussian Bayesian network. The conversion for Markov networks is simpler in some sense, inasmuch as there is a direct mapping between the entries in the information (inverse covariance) matrix of the Gaussian and the quadratic forms that parameterize the edge potentials in the Markov network. However, unlike the case of Bayesian networks, here we must take care, since not every quadratic parameterization of a pairwise Markov network induces a legal Gaussian distribution: The quadratic form that arises when we combine all the pairwise potentials may not have a finite integral, and therefore may not be normalizable. In general, there is no local way of determining whether a pairwise MRF with quadratic potentials is normalizable; however, we provided some easily checkable sufficient conditions that are often sufficient in practice.

The equivalence between the different representations is analogous to the equivalence of Bayesian networks, Markov networks, and discrete distributions: any discrete distribution can be encoded both as a Bayesian network and as a Markov network, and vice versa. However, as in the discrete case, this equivalence does *not* imply equivalence of expressive power with respect to independence assumptions. In particular, **the expressive power of the directed**

and undirected representations in terms of independence assumptions is exactly the same as in the discrete case: Directed models can encode the independencies associated with immoralities, whereas undirected models cannot; conversely, undirected models can encode a symmetric diamond, whereas directed models cannot. As we saw, the undirected models have a particularly elegant connection to the natural representation of the Gaussian distribution in terms of the information matrix; in particular, zeros in the information matrix for p correspond precisely to missing edges in the minimal I-map Markov network for p.

Finally, we note that the class of Gaussian distributions is highly restrictive, making strong assumptions that often do not hold in practice. Nevertheless, it is a very useful class, due to its compact representation and computational tractability (see section 14.2). Thus, in many cases, we may be willing to make the assumption that a distribution is Gaussian even when that is only a rough approximation. This approximation may happen a priori, in encoding a distribution as a Gaussian even when it is not. Or, in many cases, we perform the approximation as part of our inference process, representing intermediate results as a Gaussian, in order to keep the computation tractable. Indeed, as we will see, the Gaussian representation is ubiquitous in methods that perform inference in a broad range of continuous models.

7.5 Relevant Literature

The equivalence between the multivariate and linear Gaussian representations was first derived by Wermuth (1980), who also provided the one-to-one transformations between them. The introduction of linear Gaussian dependencies into a Bayesian network framework was first proposed by Shachter and Kenley (1989), in the context of influence diagrams.

Speed and Kiiveri (1986) were the first to make the connection between the structure of the information matrix and the independence assumptions in the distribution. Building on earlier results for discrete Markov networks, they also made the connection to the undirected graph as a representation. Lauritzen (1996, Chapter 5) and Malioutov et al. (2006) give a good overview of the properties of Gaussian MRFs.

7.6 Exercises

Exercise 7.1

Prove lemma 7.1. Note that you need to show both that the marginal distribution is a Gaussian, and that it is parameterized as $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y}}; \Sigma_{\boldsymbol{YY}})$.

Exercise 7.2

a. Show that, for any joint density function $p(X, Y)$, if we have $(X \perp Y)$ in p, then $\mathbf{\mathit{Cov}}[X; Y] = 0$.

b. Show that, if $p(X, Y)$ is Gaussian, and $\mathbf{\mathit{Cov}}[X; Y] = 0$, then $(X \perp Y)$ holds in p.

c. Show a counterexample to 2 for non-Gaussian distributions. More precisely, show a construction of a joint density function $p(X, Y)$ such that $\mathbf{\mathit{Cov}}[X; Y] = 0$, while $(X \perp Y)$ does not hold in p.

Exercise 7.3

Prove theorem 7.2.

Exercise 7.4

Prove theorem 7.5.

Exercise 7.5

Consider a Kalman filter whose transition model is defined in terms of a pair of matrices A, Q, and whose observation model is defined in terms of a pair of matrices H, R, as specified in equation (6.3) and equation (6.4). Describe how we can extract a 2-TBN structure representing the conditional independencies in this process from these matrices. (Hint: Use theorem 7.2.)

Exercise 7.6

In this question, we compare the number of independent parameters in a multivariate Gaussian distribution and in a linear Gaussian Bayesian network.

a. Show that the number of independent parameters in Gaussian distribution over $X_1, \ldots, X_n$ is the same as the number of independent parameters in a fully connected linear Gaussian Bayesian network over $X_1, \ldots, X_n$.

b. In example 7.4, we showed that the number of parameters in a linear Gaussian network can be substantially smaller than in its multivariate Gaussian representation. Show that the converse phenomenon can also happen. In particular, show an example of a distribution where the multivariate Gaussian representation requires a linear number of nonzero entries in the covariance matrix, while a corresponding linear Gaussian network (one that is a minimal I-map) requires a quadratic number of nonzero parameters. (Hint: The minimal I-map does not have to be the optimal one.)

Exercise 7.7

conditional covariance

Let p be a joint Gaussian density over $\mathcal{X}$ with mean vector $\boldsymbol{\mu}$ and information matrix J. Let $X_i \in \mathcal{X}$, and $\boldsymbol{Z} \subset \mathcal{X} - \{X_i\}$. We define the *conditional covariance* of X_i, X_j given $\boldsymbol{Z}$ as:

$$\mathbf{Cov}_p[X_i; X_j \mid \boldsymbol{Z}] = \mathbf{E}_p[(X_i - \mu_i)(X_j - \mu_j) \mid \boldsymbol{Z}] = \mathbf{E}_{\boldsymbol{z} \sim p(\boldsymbol{Z})}\left[\mathbf{E}_{p(X_i, X_j \mid \boldsymbol{z})}[(x_i - \mu_i)(x_j - \mu_j)]\right].$$

partial correlation coefficient

The conditional variance of X_i is defined by setting $j = i$. We now define the *partial correlation coefficient*

$$\rho_{i,j} = \frac{\mathbf{Cov}_p[X_i; X_j \mid \mathcal{X} - \{X_i, X_j\}]}{\sqrt{\mathbf{Var}_p[X_i \mid \mathcal{X} - \{X_i, X_j\}]\mathbf{Var}_p[X_j \mid \mathcal{X} - \{X_i, X_j\}]}}.$$

Show that

$$\rho_{i,j} = -\frac{J_{i,j}}{\sqrt{J_{i,i}J_{j,j}}}.$$

Exercise 7.8

Prove proposition 7.1.

Exercise 7.9

Prove proposition 7.2.

8 The Exponential Family

8.1 Introduction

In the previous chapters, we discussed several different representations of complex distributions. These included both representations of global structures (for example, Bayesian networks and Markov networks) and representations of local structures (for example, representations of CPDs and of potentials). In this chapter, we revisit these representations and view them from a different perspective. This view allows us to consider several basic questions and derive generic answers for these questions for a wide variety of representations. As we will see in later chapters, these solutions play a role in both inference and learning for the different representations we consider.

We note, however, that this chapter is somewhat abstract and heavily mathematical. Although the ideas described in this chapter are of central importance to understanding the theoretical foundations of learning and inference, the algorithms themselves can be understood even without the material presented in this chapter. Thus, this chapter can be skipped by readers who are interested primarily in the algorithms themselves.

8.2 Exponential Families

parametric family

Our discussion so far has focused on the representation of a single distribution (using, say, a Bayesian or Markov network). We now consider *families of distributions*. Intuitively, a family is a set of distributions that all share the same parametric form and differ only in choice of particular parameters (for example, the entries in table-CPDs). In general, once we choose the global structure and local structure of the network, we define a family of all distributions that can be attained by different parameters for this specific choice of CPDs.

Example 8.1

Consider the empty graph structure $\mathcal{G}_\emptyset$ over the variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. We can define the family $\mathcal{P}_\emptyset$ to be the set of distributions that are consistent with $\mathcal{G}_\emptyset$. If all the variables in $\mathcal{X}$ are binary, then we can specify a particular distribution in the family by using n parameters, $\boldsymbol{\theta} = \{P(x_i^1) : i = 1, \ldots, n\}$. ■

We will be interested in families that can be written in a particular form.

Definition 8.1

exponential family

Let $\mathcal{X}$ be a set of variables. An exponential family *$\mathcal{P}$ over $\mathcal{X}$ is specified by four components:*

sufficient statistic function

parameter space

legal parameter

natural parameter

- *A* sufficient statistics function τ *from assignments to* $\mathcal{X}$ *to* $\mathcal{R}^K$.
- *A* parameter space *that is a convex set* $\Theta \subseteq \mathcal{R}^M$ *of* legal parameters.
- *A* natural parameter *function* $\mathtt{t}$ *from* $\mathcal{R}^M$ *to* R^K.
- *An auxiliary measure* A *over* $\mathcal{X}$.

Each vector of parameters $\boldsymbol{\theta} \in \Theta$ *specifies a distribution* $P_{\boldsymbol{\theta}}$ *in the family as*

$$P_{\boldsymbol{\theta}}(\xi) = \frac{1}{Z(\boldsymbol{\theta})} A(\xi) \exp\{\langle \mathtt{t}(\boldsymbol{\theta}), \tau(\xi) \rangle\} \tag{8.1}$$

where $\langle \mathtt{t}(\boldsymbol{\theta}), \tau(\xi) \rangle$ *is the inner product of the vectors* $\mathtt{t}(\boldsymbol{\theta})$ *and* $\tau(\xi)$*, and*

$$Z(\boldsymbol{\theta}) = \sum_{\xi} A(\xi) \exp\{\langle \mathtt{t}(\boldsymbol{\theta}), \tau(\xi) \rangle\}$$

partition function

is the partition function *of* $\mathcal{P}$*, which must be finite. The parametric family* $\mathcal{P}$ *is defined as:*

$$\mathcal{P} = \{P_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\}.$$ ■

We see that an exponential family is a concise representation of a class of probability distributions that share a similar functional form. A member of the family is determined by the parameter vector $\boldsymbol{\theta}$ in the set of legal parameters. The sufficient statistic function τ summarizes the aspects of an instance that are relevant for assigning it a probability. The function $\mathtt{t}$ maps the parameters to space of the sufficient statistics.

The measure A assigns additional preferences among instances that do not depend on the parameters. However, in most of the examples we consider here A is a constant, and we will mention it explicitly only when it is not a constant.

Although this definition seems quite abstract, many distributions we already have encountered are exponential families.

Example 8.2

Consider a simple Bernoulli distribution. In this case, the distribution over a binary outcome (such as a coin toss) is controlled by a single parameter θ *that represents the probability of* x^1*. To show that this distribution is in the exponential family, we can set*

$$\tau(X) = \langle \mathbf{1}\{X = x^1\}, \mathbf{1}\{X = x^0\} \rangle, \tag{8.2}$$

a numerical vector representation of the value of X*, and*

$$\mathtt{t}(\theta) = \langle \ln \theta, \ln(1-\theta) \rangle. \tag{8.3}$$

It is easy to see that for $X = x^1$*, we have* $\tau(X) = \langle 1, 0 \rangle$*, and thus*

$$\exp\{\langle \mathtt{t}(\theta), \tau(X) \rangle\} = e^{1 \cdot \ln \theta + 0 \cdot \ln(1-\theta)} = \theta.$$

Similarly, for $X = x^0$*, we get that* $\exp\{\langle \mathtt{t}(\theta), \tau(X) \rangle\} = 1 - \theta$*. We conclude that, by setting* $Z(\theta) = 1$*, this representation is identical to the Bernoulli distribution.* ■

Example 8.3

Consider a Gaussian distribution over a single variable. Recall that

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}.$$

Define

$$\tau(x) = \langle x, x^2 \rangle \tag{8.4}$$

$$\mathsf{t}(\mu, \sigma^2) = \langle \frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2} \rangle \tag{8.5}$$

$$Z(\mu, \sigma^2) = \sqrt{2\pi}\sigma \exp\left\{\frac{\mu^2}{2\sigma^2}\right\}. \tag{8.6}$$

We can easily verify that

$$P(x) = \frac{1}{Z(\mu, \sigma^2)} \exp\left\{\langle \mathsf{t}(\theta), \tau(X) \rangle\right\}.$$

■

In fact, most of the parameterized distributions we encounter in probability textbooks can be represented as exponential families. This includes the Poisson distributions, exponential distributions, geometric distributions, Gamma distributions, and many others (see, for example, exercise 8.1).

We can often construct multiple exponential families that encode precisely the same class of distributions. There are, however, desiderata that we want from our representation of a class of distributions as an exponential family. First, we want the parameter space Θ to be "well-behaved," in particular, to be a convex, open subset of $\mathcal{R}^M$. Second, we want the parametric family to be *nonredundant* — to have each choice of parameters represent a unique distribution. More precisely, we want $\boldsymbol{\theta} \neq \boldsymbol{\theta}'$ to imply $P_{\boldsymbol{\theta}} \neq P_{\boldsymbol{\theta}'}$. It is easy check that a family is nonredundant if and only if the function t is invertible (over the set Θ). Such exponential families are called *invertible*. As we will discuss, these desiderata help us execute certain operations effectively, in particular, finding a distribution Q in some exponential family that is a "good approximation" to some other distribution P.

nonredundant parameterization

invertible exponential family

8.2.1 Linear Exponential Families

A special class of exponential families is made up of families where the function t is the identity function. This implies that the parameters are the same dimension K as the representation of the data. Such parameters are also called the *natural parameters* for the given sufficient statistic function. The name reflects that these parameters do not need to be modified in the exponential form. When using natural parameters, equation (8.1) simplifies to

natural parameter

$$P_{\boldsymbol{\theta}}(\xi) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left\{\langle \boldsymbol{\theta}, \tau(\xi) \rangle\right\}.$$

Clearly, for any given sufficient statistics function, we can reparameterize the exponential family using the natural parameters. However, as we discussed earlier, we want the space of parameters Θ to satisfy certain desiderata, which may not hold for the space of natural