

tracking uncertainty time series inference data
data mining statistics
decision **BAYESIAN**
finance kernels clustering
REASONING
sampling language trees
classification
and algorithms labels
networks recognition prediction
filtering
MACHINE control
modeling robotics MATLAB
graphs **LEARNING**
bioinformatics computational intelligence

David Barber

Bayesian Reasoning and Machine Learning

David Barber
University College London



CAMBRIDGE UNIVERSITY PRESS

Cambridge, New York, Melbourne, Madrid, Cape Town,
Singapore, São Paulo, Delhi, Tokyo, Mexico City

Cambridge University Press

The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org

Information on this title: www.cambridge.org/9780521518147

© D. Barber 2012

This publication is in copyright. Subject to statutory exception
and to the provisions of relevant collective licensing agreements,
no reproduction of any part may take place without the written
permission of Cambridge University Press.

First published 2012

Printed in the United Kingdom at the University Press, Cambridge

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing in Publication data

Barber, David, 1968–

Bayesian reasoning and machine learning / David Barber.

p. cm.

Includes bibliographical references and index.

ISBN 978-0-521-51814-7

1. Machine learning. 2. Bayesian statistical decision theory. I. Title.

QA267.B347 2012

006.3'1 – dc23 2011035553

ISBN 978-0-521-51814-7 Hardback

Additional resources for this publication at www.cambridge.org/brml and at www.cs.ucl.ac.uk/staff/D.Barber/brml

Cambridge University Press has no responsibility for the persistence or
accuracy of URLs for external or third-party internet websites referred to
in this publication, and does not guarantee that any content on such
websites is, or will remain, accurate or appropriate.

CONTENTS

Preface	xv		
List of notation	xx		
BRMLTOOLBOX	xxi		
I Inference in probabilistic models			
1 Probabilistic reasoning	3		
1.1 Probability refresher			
1.1.1 Interpreting conditional probability			
1.1.2 Probability tables			
1.2 Probabilistic reasoning			
1.3 Prior, likelihood and posterior			
1.3.1 Two dice: what were the individual scores?			
1.4 Summary			
1.5 Code			
1.6 Exercises			
2 Basic graph concepts	22		
2.1 Graphs			
2.2 Numerically encoding graphs			
2.2.1 Edge list			
2.2.2 Adjacency matrix			
2.2.3 Clique matrix			
2.3 Summary			
2.4 Code			
2.5 Exercises			
3 Belief networks	29		
3.1 The benefits of structure			
3.1.1 Modelling independencies			
3.1.2 Reducing the burden of specification			
3.2 Uncertain and unreliable evidence			
3.2.1 Uncertain evidence			
3.2.2 Unreliable evidence			
3.3 Belief networks			
3.3.1 Conditional independence			
3.3.2 The impact of collisions			
3.3.3 Graphical path manipulations for independence			
3.3.4 d-separation			
3.3.5 Graphical and distributional in/dependence			
3.3.6 Markov equivalence in belief networks			
3.3.7 Belief networks have limited expressibility			
3.4 Causality			
3.4.1 Simpson's paradox			
3.4.2 The do-calculus			
3.4.3 Influence diagrams and the do-calculus			
3.5 Summary			
3.6 Code			
3.7 Exercises			
4 Graphical models	58		
4.1 Graphical models			
4.2 Markov networks			
4.2.1 Markov properties			
4.2.2 Markov random fields			
4.2.3 Hammersley–Clifford theorem			
4.2.4 Conditional independence using Markov networks			
4.2.5 Lattice models			
4.3 Chain graphical models			
4.4 Factor graphs			
4.4.1 Conditional independence in factor graphs			
4.5 Expressiveness of graphical models			
4.6 Summary			
4.7 Code			
4.8 Exercises			

5	Efficient inference in trees	77	
5.1	Marginal inference		
5.1.1	Variable elimination in a Markov chain and message passing		
5.1.2	The sum-product algorithm on factor graphs		
5.1.3	Dealing with evidence		
5.1.4	Computing the marginal likelihood		
5.1.5	The problem with loops		
5.2	Other forms of inference		
5.2.1	Max-product		
5.2.2	Finding the N most probable states		
5.2.3	Most probable path and shortest path		
5.2.4	Mixed inference		
5.3	Inference in multiply connected graphs		
5.3.1	Bucket elimination		
5.3.2	Loop-cut conditioning		
5.4	Message passing for continuous distributions		
5.5	Summary		
5.6	Code		
5.7	Exercises		
6	The junction tree algorithm	102	
6.1	Clustering variables		
6.1.1	Reparameterisation		
6.2	Clique graphs		
6.2.1	Absorption		
6.2.2	Absorption schedule on clique trees		
6.3	Junction trees		
6.3.1	The running intersection property		
6.4	Constructing a junction tree for singly connected distributions		
6.4.1	Moralisation		
6.4.2	Forming the clique graph		
6.4.3	Forming a junction tree from a clique graph		
6.4.4	Assigning potentials to cliques		
6.5	Junction trees for multiply connected distributions		
6.5.1	Triangulation algorithms		
6.6	The junction tree algorithm		
6.6.1	Remarks on the JTA		
6.6.2	Computing the normalisation constant of a distribution		
6.6.3	The marginal likelihood		
6.6.4	Some small JTA examples		
6.6.5	Shafer–Shenoy propagation		
6.7	Finding the most likely state		
6.8	Reabsorption: converting a junction tree to a directed network		
6.9	The need for approximations		
6.9.1	Bounded width junction trees		
6.10	Summary		
6.11	Code		
6.12	Exercises		
7	Making decisions		127
7.1	Expected utility		
7.1.1	Utility of money		
7.2	Decision trees		
7.3	Extending Bayesian networks for decisions		
7.3.1	Syntax of influence diagrams		
7.4	Solving influence diagrams		
7.4.1	Messages on an ID		
7.4.2	Using a junction tree		
7.5	Markov decision processes		
7.5.1	Maximising expected utility by message passing		
7.5.2	Bellman's equation		
7.6	Temporally unbounded MDPs		
7.6.1	Value iteration		
7.6.2	Policy iteration		
7.6.3	A curse of dimensionality		
7.7	Variational inference and planning		
7.8	Financial matters		
7.8.1	Options pricing and expected utility		
7.8.2	Binomial options pricing model		
7.8.3	Optimal investment		
7.9	Further topics		
7.9.1	Partially observable MDPs		
7.9.2	Reinforcement learning		
7.10	Summary		
7.11	Code		
7.12	Exercises		

II Learning in probabilistic models

8	Statistics for machine learning	165	
8.1	Representing data		
8.1.1	Categorical		
8.1.2	Ordinal		
8.1.3	Numerical		
8.2	Distributions		
8.2.1	The Kullback–Leibler divergence $KL(q p)$		
8.2.2	Entropy and information		
8.3	Classical distributions		
8.4	Multivariate Gaussian		
8.4.1	Completing the square		
8.4.2	Conditioning as system reversal		
8.4.3	Whitening and centring		
8.5	Exponential family		
8.5.1	Conjugate priors		
8.6	Learning distributions		
8.7	Properties of maximum likelihood		
8.7.1	Training assuming the correct model class		
8.7.2	Training when the assumed model is incorrect		
8.7.3	Maximum likelihood and the empirical distribution		
8.8	Learning a Gaussian		
8.8.1	Maximum likelihood training		
8.8.2	Bayesian inference of the mean and variance		
8.8.3	Gauss-gamma distribution		
8.9	Summary		
8.10	Code		
8.11	Exercises		
9	Learning as inference	199	
9.1	Learning as inference		
9.1.1	Learning the bias of a coin		
9.1.2	Making decisions		
9.1.3	A continuum of parameters		
9.1.4	Decisions based on continuous intervals		
9.2	Bayesian methods and ML-II		
9.3	Maximum likelihood training of belief networks		
9.4	Bayesian belief network training		
9.4.1	Global and local parameter independence		
9.4.2	Learning binary variable tables using a Beta prior		
9.4.3	Learning multivariate discrete tables using a Dirichlet prior		
9.5	Structure learning		
9.5.1	PC algorithm		
9.5.2	Empirical independence		
9.5.3	Network scoring		
9.5.4	Chow–Liu trees		
9.6	Maximum likelihood for undirected models		
9.6.1	The likelihood gradient		
9.6.2	General tabular clique potentials		
9.6.3	Decomposable Markov networks		
9.6.4	Exponential form potentials		
9.6.5	Conditional random fields		
9.6.6	Pseudo likelihood		
9.6.7	Learning the structure		
9.7	Summary		
9.8	Code		
9.9	Exercises		
10	Naive Bayes	243	
10.1	Naive Bayes and conditional independence		
10.2	Estimation using maximum likelihood		
10.2.1	Binary attributes		
10.2.2	Multi-state variables		
10.2.3	Text classification		
10.3	Bayesian naive Bayes		
10.4	Tree augmented naive Bayes		
10.4.1	Learning tree augmented naive Bayes networks		
10.5	Summary		
10.6	Code		
10.7	Exercises		
11	Learning with hidden variables	256	
11.1	Hidden variables and missing data		
11.1.1	Why hidden/missing variables can complicate proceedings		
11.1.2	The missing at random assumption		

11.1.3	Maximum likelihood	
11.1.4	Identifiability issues	
11.2	Expectation maximisation	
11.2.1	Variational EM	
11.2.2	Classical EM	
11.2.3	Application to belief networks	
11.2.4	General case	
11.2.5	Convergence	
11.2.6	Application to Markov networks	
11.3	Extensions of EM	
11.3.1	Partial M-step	
11.3.2	Partial E-step	
11.4	A failure case for EM	
11.5	Variational Bayes	
11.5.1	EM is a special case of variational Bayes	
11.5.2	An example: VB for the Asbestos-Smoking-Cancer network	
11.6	Optimising the likelihood by gradient methods	
11.6.1	Undirected models	
11.7	Summary	
11.8	Code	
11.9	Exercises	
12	Bayesian model selection	284
12.1	Comparing models the Bayesian way	
12.2	Illustrations: coin tossing	
12.2.1	A discrete parameter space	
12.2.2	A continuous parameter space	
12.3	Occam's razor and Bayesian complexity penalisation	
12.4	A continuous example: curve fitting	
12.5	Approximating the model likelihood	
12.5.1	Laplace's method	
12.5.2	Bayes information criterion	
12.6	Bayesian hypothesis testing for outcome analysis	
12.6.1	Outcome analysis	
12.6.2	H_{indep} : model likelihood	
12.6.3	H_{same} : model likelihood	
12.6.4	Dependent outcome analysis	
12.6.5	Is classifier A better than B ?	
12.7	Summary	
12.8	Code	
12.9	Exercises	
III	Machine learning	
13	Machine learning concepts	305
13.1	Styles of learning	
13.1.1	Supervised learning	
13.1.2	Unsupervised learning	
13.1.3	Anomaly detection	
13.1.4	Online (sequential) learning	
13.1.5	Interacting with the environment	
13.1.6	Semi-supervised learning	
13.2	Supervised learning	
13.2.1	Utility and loss	
13.2.2	Using the empirical distribution	
13.2.3	Bayesian decision approach	
13.3	Bayes versus empirical decisions	
13.4	Summary	
13.5	Exercises	
14	Nearest neighbour classification	322
14.1	Do as your neighbour does	
14.2	K -nearest neighbours	
14.3	A probabilistic interpretation of nearest neighbours	
14.3.1	When your nearest neighbour is far away	
14.4	Summary	
14.5	Code	
14.6	Exercises	
15	Unsupervised linear dimension reduction	329
15.1	High-dimensional spaces – low-dimensional manifolds	
15.2	Principal components analysis	
15.2.1	Deriving the optimal linear reconstruction	
15.2.2	Maximum variance criterion	
15.2.3	PCA algorithm	
15.2.4	PCA and nearest neighbours classification	
15.2.5	Comments on PCA	

15.3	High-dimensional data		17.4.1	Logistic regression	
15.3.1	Eigen-decomposition for $N < D$		17.4.2	Beyond first-order gradient ascent	
15.3.2	PCA via singular value decomposition		17.4.3	Avoiding overconfident classification	
15.4	Latent semantic analysis		17.4.4	Multiple classes	
15.4.1	Information retrieval		17.4.5	The kernel trick for classification	
15.5	PCA with missing data		17.5	Support vector machines	
15.5.1	Finding the principal directions		17.5.1	Maximum margin linear classifier	
15.5.2	Collaborative filtering using PCA with missing data		17.5.2	Using kernels	
15.6	Matrix decomposition methods		17.5.3	Performing the optimisation	
15.6.1	Probabilistic latent semantic analysis		17.5.4	Probabilistic interpretation	
15.6.2	Extensions and variations		17.6	Soft zero-one loss for outlier robustness	
15.6.3	Applications of PLSA/NMF		17.7	Summary	
15.7	Kernel PCA		17.8	Code	
15.8	Canonical correlation analysis		17.9	Exercises	
15.8.1	SVD formulation				
15.9	Summary				
15.10	Code				
15.11	Exercises				
16	Supervised linear dimension reduction	359	18	Bayesian linear models	392
16.1	Supervised linear projections		18.1	Regression with additive Gaussian noise	
16.2	Fisher's linear discriminant		18.1.1	Bayesian linear parameter models	
16.3	Canonical variates		18.1.2	Determining hyperparameters: ML-II	
16.3.1	Dealing with the nullspace		18.1.3	Learning the hyperparameters using EM	
16.4	Summary		18.1.4	Hyperparameter optimisation: using the gradient	
16.5	Code		18.1.5	Validation likelihood	
16.6	Exercises		18.1.6	Prediction and model averaging	
17	Linear models	367	18.1.7	Sparse linear models	
17.1	Introduction: fitting a straight line		18.2	Classification	
17.2	Linear parameter models for regression		18.2.1	Hyperparameter optimisation	
17.2.1	Vector outputs		18.2.2	Laplace approximation	
17.2.2	Regularisation		18.2.3	Variational Gaussian approximation	
17.2.3	Radial basis functions		18.2.4	Local variational approximation	
17.3	The dual representation and kernels		18.2.5	Relevance vector machine for classification	
17.3.1	Regression in the dual space		18.2.6	Multi-class case	
17.4	Linear parameter models for classification		18.3	Summary	
			18.4	Code	
			18.5	Exercises	

19 Gaussian processes	412	
19.1 Non-parametric prediction		20.3.6 Bayesian mixture models
19.1.1 From parametric to non-parametric		20.3.7 Semi-supervised learning
19.1.2 From Bayesian linear models to Gaussian processes		20.4 Mixture of experts
19.1.3 A prior on functions		20.5 Indicator models
19.2 Gaussian process prediction		20.5.1 Joint indicator approach: factorised prior
19.2.1 Regression with noisy training outputs		20.5.2 Polya prior
19.3 Covariance functions		20.6 Mixed membership models
19.3.1 Making new covariance functions from old		20.6.1 Latent Dirichlet allocation
19.3.2 Stationary covariance functions		20.6.2 Graph-based representations of data
19.3.3 Non-stationary covariance functions		20.6.3 Dyadic data
19.4 Analysis of covariance functions		20.6.4 Monadic data
19.4.1 Smoothness of the functions		20.6.5 Cliques and adjacency matrices for monadic binary data
19.4.2 Mercer kernels		20.7 Summary
19.4.3 Fourier analysis for stationary kernels		20.8 Code
19.5 Gaussian processes for classification		20.9 Exercises
19.5.1 Binary classification		
19.5.2 Laplace's approximation		21 Latent linear models
19.5.3 Hyperparameter optimisation		462
19.5.4 Multiple classes		21.1 Factor analysis
19.6 Summary		21.1.1 Finding the optimal bias
19.7 Code		21.2 Factor analysis: maximum likelihood
19.8 Exercises		21.2.1 Eigen-approach likelihood optimisation
		21.2.2 Expectation maximisation
		21.3 Interlude: modelling faces
		21.4 Probabilistic principal components analysis
		21.5 Canonical correlation analysis and factor analysis
		21.6 Independent components analysis
		21.7 Summary
		21.8 Code
		21.9 Exercises
20 Mixture models	432	
20.1 Density estimation using mixtures		22 Latent ability models
20.2 Expectation maximisation for mixture models		479
20.2.1 Unconstrained discrete tables		22.1 The Rasch model
20.2.2 Mixture of product of Bernoulli distributions		22.1.1 Maximum likelihood training
20.3 The Gaussian mixture model		22.1.2 Bayesian Rasch models
20.3.1 EM algorithm		22.2 Competition models
20.3.2 Practical issues		22.2.1 Bradley–Terry–Luce model
20.3.3 Classification using Gaussian mixture models		22.2.2 Elo ranking model
20.3.4 The Parzen estimator		22.2.3 Glicko and TrueSkill
20.3.5 K-means		

28.6.1	The information maximisation algorithm	28.12	Code	
28.6.2	Linear Gaussian decoder	28.13	Exercises	
28.7	Loopy belief propagation	Appendix A: Background mathematics	655	
28.7.1	Classical BP on an undirected graph	A.1	Linear algebra	
28.7.2	Loopy BP as a variational procedure	A.2	Multivariate calculus	
28.8	Expectation propagation	A.3	Inequalities	
28.9	MAP for Markov networks	A.4	Optimisation	
28.9.1	Pairwise Markov networks	A.5	Multivariate optimisation	
28.9.2	Attractive binary Markov networks	A.6	Constrained optimisation using Lagrange multipliers	
28.9.3	Potts model	References		675
28.10	Further reading	Index		689
28.11	Summary			
		<i>Colour plate section between pp. 360 and 361</i>		

PREFACE

The data explosion

We live in a world that is rich in data, ever increasing in scale. This data comes from many different sources in science (bioinformatics, astronomy, physics, environmental monitoring) and commerce (customer databases, financial transactions, engine monitoring, speech recognition, surveillance, search). Possessing the knowledge as to how to process and extract value from such data is therefore a key and increasingly important skill. Our society also expects ultimately to be able to engage with computers in a natural manner so that computers can ‘talk’ to humans, ‘understand’ what they say and ‘comprehend’ the visual world around them. These are difficult large-scale information processing tasks and represent grand challenges for computer science and related fields. Similarly, there is a desire to control increasingly complex systems, possibly containing many interacting parts, such as in robotics and autonomous navigation. Successfully mastering such systems requires an understanding of the processes underlying their behaviour. Processing and making sense of such large amounts of data from complex systems is therefore a pressing modern-day concern and will likely remain so for the foreseeable future.

Machine learning

Machine learning is the study of data-driven methods capable of mimicking, understanding and aiding human and biological information processing tasks. In this pursuit, many related issues arise such as how to compress data, interpret and process it. Often these methods are not necessarily directed to mimicking directly human processing but rather to enhancing it, such as in predicting the stock market or retrieving information rapidly. In this probability theory is key since inevitably our limited data and understanding of the problem forces us to address uncertainty. In the broadest sense, machine learning and related fields aim to ‘learn something useful’ about the environment within which the agent operates. Machine learning is also closely allied with artificial intelligence, with machine learning placing more emphasis on using data to drive and adapt the model.

In the early stages of machine learning and related areas, similar techniques were discovered in relatively isolated research communities. This book presents a unified treatment via graphical models, a marriage between graph and probability theory, facilitating the transference of machine learning concepts between different branches of the mathematical and computational sciences.

Whom this book is for

The book is designed to appeal to students with only a modest mathematical background in undergraduate calculus and linear algebra. No formal computer science or statistical background is required to follow the book, although a basic familiarity with probability, calculus and linear algebra

would be useful. The book should appeal to students from a variety of backgrounds, including computer science, engineering, applied statistics, physics and bioinformatics that wish to gain an entry to probabilistic approaches in machine learning. In order to engage with students, the book introduces fundamental concepts in inference using only minimal reference to algebra and calculus. More mathematical techniques are postponed until as and when required, always with the concept as primary and the mathematics secondary.

The concepts and algorithms are described with the aid of many worked examples. The exercises and demonstrations, together with an accompanying MATLAB toolbox, enable the reader to experiment and more deeply understand the material. The ultimate aim of the book is to enable the reader to construct novel algorithms. The book therefore places an emphasis on skill learning, rather than being a collection of recipes. This is a key aspect since modern applications are often so specialised as to require novel methods. The approach taken throughout is to describe the problem as a graphical model, which is then translated into a mathematical framework, ultimately leading to an algorithmic implementation in the BRML_{TOOLBOX}.

The book is primarily aimed at final year undergraduates and graduates without significant experience in mathematics. On completion, the reader should have a good understanding of the techniques, practicalities and philosophies of probabilistic aspects of machine learning and be well equipped to understand more advanced research level material.

The structure of the book

The book begins with the basic concepts of graphical models and inference. For the independent reader Chapters 1, 2, 3, 4, 5, 9, 10, 13, 14, 15, 16, 17, 21 and 23 would form a good introduction to probabilistic reasoning, modelling and machine learning. The material in Chapters 19, 24, 25 and 28 is more advanced, with the remaining material being of more specialised interest. Note that in each chapter the level of material is of varying difficulty, typically with the more challenging material placed towards the end of each chapter. As an introduction to the area of probabilistic modelling, a course can be constructed from the material as indicated in the chart.

The material from Parts I and II has been successfully used for courses on graphical models. I have also taught an introduction to probabilistic machine learning using material largely from Part III, as indicated. These two courses can be taught separately and a useful approach would be to teach first the graphical models course, followed by a separate probabilistic machine learning course.

A short course on approximate inference can be constructed from introductory material in Part I and the more advanced material in Part V, as indicated. The exact inference methods in Part I can be covered relatively quickly with the material in Part V considered in more depth.

A timeseries course can be made by using primarily the material in Part IV, possibly combined with material from Part I for students that are unfamiliar with probabilistic modelling approaches. Some of this material, particularly in Chapter 25, is more advanced and can be deferred until the end of the course, or considered for a more advanced course.

The references are generally to works at a level consistent with the book material and which are in the most part readily available.

		Graphical models course	Probabilistic machine learning course	Approximate inference short course	Timeseries short course	Probabilistic modelling course
Part I: Inference in probabilistic models	1: Probabilistic reasoning	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
	2: Basic graph concepts	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
	3: Belief networks	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
	4: Graphical models	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
	5: Efficient inference in trees	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
	6: The junction tree algorithm	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
	7: Making decisions	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Part II: Learning in probabilistic models	8: Statistics for machine learning	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
	9: Learning as inference	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
	10: Naive Bayes	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	11: Learning with hidden variables	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
	12: Bayesian model selection	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Part III: Machine learning	13: Machine learning concepts	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	14: Nearest neighbour classification	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	15: Unsupervised linear dimension reduction	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	16: Supervised linear dimension reduction	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	17: Linear models	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	18: Bayesian linear models	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
	19: Gaussian processes	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	20: Mixture models	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
	21: Latent linear models	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
	22: Latent ability models	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Part IV: Dynamical models	23: Discrete-state Markov models	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
	24: Continuous-state Markov models	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
	25: Switching linear dynamical systems	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
	26: Distributed computation	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Part V: Approximate inference	27: Sampling	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
	28: Deterministic approximate inference	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

NOTATION

\mathcal{V}	A calligraphic symbol typically denotes a set of random variables	page 3
$\text{dom}(x)$	Domain of a variable	3
$x = x$	The variable x is in the state x	3
$p(x = \text{tr})$	Probability of event/variable x being in the state true	3
$p(x = \text{fa})$	Probability of event/variable x being in the state false	3
$p(x, y)$	Probability of x and y	4
$p(x \cap y)$	Probability of x and y	4
$p(x \cup y)$	Probability of x or y	4
$p(x y)$	The probability of x conditioned on y	4
$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mathcal{Z}$	Variables \mathcal{X} are independent of variables \mathcal{Y} conditioned on variables \mathcal{Z}	7
$\mathcal{X} \Pi \mathcal{Y} \mathcal{Z}$	Variables \mathcal{X} are dependent on variables \mathcal{Y} conditioned on variables \mathcal{Z}	7
$\int_x f(x)$	For continuous variables this is shorthand for $\int_x f(x)dx$ and for discrete variables means summation over the states of x , $\sum_x f(x)$	14
$\mathbb{I}[S]$	Indicator : has value 1 if the statement S is true, 0 otherwise	16
$\text{pa}(x)$	The parents of node x	24
$\text{ch}(x)$	The children of node x	24
$\text{ne}(x)$	Neighbours of node x	24
$\text{dim}(x)$	For a discrete variable x , this denotes the number of states x can take	34
$\langle f(x) \rangle_{p(x)}$	The average of the function $f(x)$ with respect to the distribution $p(x)$	170
$\delta(a, b)$	Delta function. For discrete a, b , this is the Kronecker delta, $\delta_{a,b}$ and for continuous a, b the Dirac delta function $\delta(a - b)$	172
$\text{dim}(\mathbf{x})$	The dimension of the vector/matrix \mathbf{x}	183
$\#(x = s, y = t)$	The number of times x is in state s and y in state t simultaneously	207
$\#_y^x$	The number of times variable x is in state y	293
\mathcal{D}	Dataset	303
n	Data index	303
N	Number of dataset training points	303
\mathbf{S}	Sample Covariance matrix	331
$\sigma(x)$	The logistic sigmoid $1/(1 + \exp(-x))$	371
$\text{erf}(x)$	The (Gaussian) error function	372
$x_{a:b}$	x_a, x_{a+1}, \dots, x_b	372
$i \sim j$	The set of unique neighbouring edges on a graph	624
\mathbf{I}_m	The $m \times m$ identity matrix	644

BRMLTOOLBOX

The BRMLTOOLBOX is a lightweight set of routines that enables the reader to experiment with concepts in graph theory, probability theory and machine learning. The code contains basic routines for manipulating discrete variable distributions, along with more limited support for continuous variables. In addition there are many hard-coded standard machine learning algorithms. The website contains also a complete list of all the teaching demos and related exercise material.

BRMLTOOLKIT

Graph theory

<code>ancestors</code>	- Return the ancestors of nodes x in DAG A
<code>ancestralorder</code>	- Return the ancestral order or the DAG A (oldest first)
<code>descendents</code>	- Return the descendents of nodes x in DAG A
<code>children</code>	- Return the children of variable x given adjacency matrix A
<code>edges</code>	- Return edge list from adjacency matrix A
<code>elimtri</code>	- Return a variable elimination sequence for a triangulated graph
<code>connectedComponents</code>	- Find the connected components of an adjacency matrix
<code>istree</code>	- Check if graph is singly connected
<code>neigh</code>	- Find the neighbours of vertex v on a graph with adjacency matrix G
<code>noselfpath</code>	- Return a path excluding self-transitions
<code>parents</code>	- Return the parents of variable x given adjacency matrix A
<code>spantree</code>	- Find a spanning tree from an edge list
<code>triangulate</code>	- Triangulate adjacency matrix A
<code>triangulatePorder</code>	- Triangulate adjacency matrix A according to a partial ordering

Potential manipulation

<code>condpot</code>	- Return a potential conditioned on another variable
<code>changevar</code>	- Change variable names in a potential
<code>dag</code>	- Return the adjacency matrix (zeros on diagonal) for a belief network
<code>deltapot</code>	- A delta function potential
<code>disptable</code>	- Print the table of a potential
<code>divpots</code>	- Divide potential pot_a by pot_b
<code>drawFG</code>	- Draw the factor graph A
<code>drawID</code>	- Plot an influence diagram
<code>drawJTree</code>	- Plot a junction tree
<code>drawNet</code>	- Plot network
<code>evalpot</code>	- Evaluate the table of a potential when variables are set
<code>exppot</code>	- Exponential of a potential
<code>eyepot</code>	- Return a unit potential
<code>grouppot</code>	- Form a potential based on grouping variables together
<code>groupstate</code>	- Find the state of the group variables corresponding to a given ungrouped state
<code>logpot</code>	- Logarithm of the potential
<code>markov</code>	- Return a symmetric adjacency matrix of Markov network in pot
<code>maxpot</code>	- Maximise a potential over variables
<code>maxsumpot</code>	- Maximise or sum a potential over variables
<code>multpots</code>	- Multiply potentials into a single potential

<code>numstates</code>	- Number of states of the variables in a potential
<code>orderpot</code>	- Return potential with variables reordered according to order
<code>orderpotfields</code>	- Order the fields of the potential, creating blank entries where necessary
<code>potssample</code>	- Draw sample from a single potential
<code>potsscontainingonly</code>	- Returns those potential numbers that contain only the required variables
<code>potvariables</code>	- Returns information about all variables in a set of potentials
<code>setevpot</code>	- Sets variables in a potential into evidential states
<code>setpot</code>	- Sets potential variables to specified states
<code>setstate</code>	- Set a potential's specified joint state to a specified value
<code>squeezepots</code>	- Eliminate redundant potentials (those contained wholly within another)
<code>sumpot</code>	- Sum potential pot over variables
<code>sumpotID</code>	- Return the summed probability and utility tables from an ID
<code>sumpots</code>	- Sum a set of potentials
<code>table</code>	- Return the potential table
<code>ungrouppot</code>	- Form a potential based on ungrouping variables
<code>uniquepots</code>	- Eliminate redundant potentials (those contained wholly within another)
<code>whichpot</code>	- Returns potentials that contain a set of variables

Routines also extend the toolbox to deal with Gaussian potentials: `multpotsGaussianMoment.m`, `sumpotGaussianCanonical.m`, `sumpotGaussianMoment.m`, `multpotsGaussianCanonical.m` See `demoSumprodGaussCanon.m`, `demoSumprodGaussCanonLDS.m`, `demoSumprodGaussMoment.m`

Inference

<code>absorb</code>	- Update potentials in absorption message passing on a junction tree
<code>absorption</code>	- Perform full round of absorption on a junction tree
<code>absorptionID</code>	- Perform full round of absorption on an influence diagram
<code>ancestralsample</code>	- Ancestral sampling from a belief network
<code>binaryMRFmap</code>	- Get the MAP assignment for a binary MRF with positive W
<code>bucketelim</code>	- Bucket elimination on a set of potentials
<code>condindep</code>	- Conditional independence check using graph of variable interactions
<code>condindepEmp</code>	- Compute the empirical log Bayes factor and MI for independence/dependence
<code>condindepPot</code>	- Numerical conditional independence measure
<code>condMI</code>	- Conditional mutual information $I(x,y z)$ of a potential
<code>FactorConnectingVariable</code>	- Factor nodes connecting to a set of variables
<code>FactorGraph</code>	- Returns a factor graph adjacency matrix based on potentials
<code>IDvars</code>	- Probability and decision variables from a partial order
<code>jtassignpot</code>	- Assign potentials to cliques in a junction tree
<code>jtree</code>	- Setup a junction tree based on a set of potentials
<code>jtreeID</code>	- Setup a junction tree based on an influence diagram
<code>LoopyBP</code>	- Loopy belief propagation using sum-product algorithm
<code>MaxFlow</code>	- Ford Fulkerson max-flow min-cut algorithm (breadth first search)
<code>maxNpot</code>	- Find the N most probable values and states in a potential
<code>maxNprodFG</code>	- N-max-product algorithm on a factor graph (returns the Nmax most probable states)
<code>maxprodFG</code>	- Max-product algorithm on a factor graph
<code>MDPemDeterministicPolicy</code>	- Solve MDP using EM with deterministic policy
<code>MDPsolve</code>	- Solve a Markov decision process
<code>MesstoFact</code>	- Returns the message numbers that connect into factor potential
<code>metropolis</code>	- Metropolis sample
<code>mostprobablepath</code>	- Find the most probable path in a Markov chain
<code>mostprobablepathmult</code>	- Find the all source all sink most probable paths in a Markov chain
<code>sumprodFG</code>	- Sum-product algorithm on a factor graph represented by A

Specific models

<code>ARlds</code>	- Learn AR coefficients using a linear dynamical system
<code>ARtrain</code>	- Fit auto-regressive (AR) coefficients of order L to v.
<code>BayesLinReg</code>	- Bayesian linear regression training using basis functions $\phi(x)$
<code>BayesLogRegressionRVM</code>	- Bayesian logistic regression with the relevance vector machine
<code>CanonVar</code>	- Canonical variates (no post rotation of variates)

cca	- Canonical correlation analysis
covfnGE	- Gamma exponential covariance function
FA	- Factor analysis
GMMem	- Fit a mixture of Gaussian to the data X using EM
GPclass	- Gaussian process binary classification
GPreg	- Gaussian process regression
HebbML	- Learn a sequence for a Hopfield network
HMMbackward	- HMM backward pass
HMMbackwardSAR	- Backward pass (beta method) for the switching Auto-regressive HMM
HMMem	- EM algorithm for HMM
HMMforward	- HMM forward pass
HMMforwardSAR	- Switching auto-regressive HMM with switches updated only every Tskip timesteps
HMMgamma	- HMM posterior smoothing using the Rauch–Tung–Striebel correction method
yHMMsmooth	- Smoothing for a hidden Markov model (HMM)
HMMsmoothSAR	- Switching auto-regressive HMM smoothing
HMMviterbi	- Viterbi most likely joint hidden state of HMM
kernel	- A kernel evaluated at two points
Kmeans	- K-means clustering algorithm
LDSbackward	- Full backward pass for a latent linear dynamical system (RTS correction method)
LDSbackwardUpdate	- Single backward update for a latent linear dynamical system (RTS smoothing update)
LDSforward	- Full forward pass for a latent linear dynamical system (Kalman filter)
LDSforwardUpdate	- Single forward update for a latent linear dynamical system (Kalman filter)
LDSsmooth	- Linear dynamical system: filtering and smoothing
LDSsubspace	- Subspace method for identifying linear dynamical system
LogReg	- Learning logistic linear regression using gradient ascent
MIXprodBern	- EM training of a mixture of a product of Bernoulli distributions
mixMarkov	- EM training for a mixture of Markov models
NaiveBayesDirichletTest	- Naive Bayes prediction having used a Dirichlet prior for training
NaiveBayesDirichletTrain	- Naive Bayes training using a Dirichlet prior
NaiveBayesTest	- Test Naive Bayes Bernoulli distribution after max likelihood training
NaiveBayesTrain	- Train Naive Bayes Bernoulli distribution using max likelihood
nearNeigh	- Nearest neighbour classification
pca	- Principal components analysis
plsa	- Probabilistic latent semantic analysis
plsaCond	- Conditional PLSA (probabilistic latent semantic analysis)
rbf	- Radial basis function output
SARlearn	- EM training of a switching AR model
SLDSbackward	- Backward pass using a mixture of Gaussians
SLDSforward	- Switching latent linear dynamical system Gaussian sum forward pass
SLDSmargGauss	- Compute the single Gaussian from a weighted SLDS mixture
softloss	- Soft loss function
svdm	- Singular value decomposition with missing values
SVMtrain	- Train a support vector machine

General

argmax	- Performs argmax returning the index and value
assign	- Assigns values to variables
betaXbiggerY	- $p(x>y)$ for $x\sim\text{Beta}(a,b)$, $y\sim\text{Beta}(c,d)$
bar3zcolor	- Plot a 3D bar plot of the matrix Z
avsigmaGauss	- Average of a logistic sigmoid under a Gaussian
cap	- Cap x at absolute value c
chi2test	- Inverse of the chi square cumulative density
count	- For a data matrix (each column is a datapoint), return the state counts
condexp	- Compute normalised p proportional to $\exp(\log p)$
condp	- Make a conditional distribution from the matrix
dirrnd	- Samples from a Dirichlet distribution
field2cell	- Place the field of a structure in a cell
GaussCond	- Return the mean and covariance of a conditioned Gaussian

<code>hinton</code>	- Plot a Hinton diagram
<code>ind2subv</code>	- Subscript vector from linear index
<code>ismember_sorted</code>	- True for member of sorted set
<code>lengthcell</code>	- Length of each cell entry
<code>logdet</code>	- Log determinant of a positive definite matrix computed in a numerically stable manner
<code>logeps</code>	- $\log(x+\text{eps})$
<code>logGaussGamma</code>	- Unnormalised log of the Gauss-Gamma distribution
<code>logsumexp</code>	- Compute $\log(\sum(\exp(a).*b))$ valid for large a
<code>logZdirichlet</code>	- Log normalisation constant of a Dirichlet distribution with parameter u
<code>majority</code>	- Return majority values in each column on a matrix
<code>maxarray</code>	- Maximise a multi-dimensional array over a set of dimensions
<code>maxNarray</code>	- Find the highest values and states of an array over a set of dimensions
<code>mix2mix</code>	- Fit a mixture of Gaussians with another mixture of Gaussians
<code>mvrands</code>	- Samples from a multivariate Normal (Gaussian) distribution
<code>mygamrnd</code>	- Gamma random variate generator
<code>mynanmean</code>	- Mean of values that are not nan
<code>mynansum</code>	- Sum of values that are not nan
<code>mynchoosek</code>	- Binomial coefficient v choose k
<code>myones</code>	- Same as <code>ones(x)</code> , but if x is a scalar, interprets as <code>ones([x 1])</code>
<code>myrand</code>	- Same as <code>rand(x)</code> but if x is a scalar interprets as <code>rand([x 1])</code>
<code>myzeros</code>	- Same as <code>zeros(x)</code> but if x is a scalar interprets as <code>zeros([x 1])</code>
<code>normp</code>	- Make a normalised distribution from an array
<code>randgen</code>	- Generates discrete random variables given the pdf
<code>replace</code>	- Replace instances of a value with another value
<code>sigma</code>	- $1./(1+\exp(-x))$
<code>sigmoid</code>	- $1./(1+\exp(-\text{beta}*x))$
<code>sqdist</code>	- Square distance between vectors in x and y
<code>subv2ind</code>	- Linear index from subscript vector.
<code>sumlog</code>	- $\sum(\log(x))$ with a cutoff at $10e-200$

Miscellaneous

<code>compat</code>	- Compatibility of object F being in position h for image v on grid Gx,Gy
<code>logp</code>	- The logarithm of a specific non-Gaussian distribution
<code>placeobject</code>	- Place the object F at position h in grid Gx,Gy
<code>plotCov</code>	- Return points for plotting an ellipse of a covariance
<code>pointsCov</code>	- Unit variance contours of a 2D Gaussian with mean m and covariance S
<code>setup</code>	- Run me at initialisation – checks for bugs in matlab and initialises path
<code>validgridposition</code>	- Returns 1 if point is on a defined grid

1

Probabilistic reasoning

We have intuition about how uncertainty works in simple cases. To reach sensible conclusions in complicated situations, however – where there may be many (possibly) related events and many possible outcomes – we need a formal ‘calculus’ that extends our intuitive notions. The concepts, mathematical language and rules of probability give us the formal framework we need. In this chapter we review basic concepts in probability – in particular, conditional probability and Bayes’ rule, the workhorses of machine learning. Another strength of the language of probability is that it structures problems in a form consistent for computer implementation. We also introduce basic features of the `BRMLTOOLBOX` that support manipulating probability distributions.

1.1 Probability refresher

Variables, states and notational shortcuts

Variables will be denoted using either upper case X or lower case x and a set of variables will typically be denoted by a calligraphic symbol, for example $\mathcal{V} = \{a, B, c\}$.

The *domain* of a variable x is written $\text{dom}(x)$, and denotes the states x can take. States will typically be represented using sans-serif font. For example, for a coin c , $\text{dom}(c) = \{\text{heads}, \text{tails}\}$ and $p(c = \text{heads})$ represents the probability that variable c is in state heads. The meaning of $p(\text{state})$ will often be clear, without specific reference to a variable. For example, if we are discussing an experiment about a coin c , the meaning of $p(\text{heads})$ is clear from the context, being shorthand for $p(c = \text{heads})$. When summing over a variable $\sum_x f(x)$, the interpretation is that all states of x are included, i.e. $\sum_x f(x) \equiv \sum_{s \in \text{dom}(x)} f(x = s)$. Given a variable, x , its domain $\text{dom}(x)$ and a full specification of the probability values for each of the variable states, $p(x)$, we have a *distribution* for x . Sometimes we will not fully specify the distribution, only certain properties, such as for variables x, y , $p(x, y) = p(x)p(y)$ for some unspecified $p(x)$ and $p(y)$. When clarity on this is required we will say distributions with structure $p(x)p(y)$, or a distribution class $p(x)p(y)$.

For our purposes, *events* are expressions about random variables, such as *Two heads in six coin tosses*. Two events are *mutually exclusive* if they cannot both be true. For example the events *The coin is heads* and *The coin is tails* are mutually exclusive. One can think of defining a new variable named by the event so, for example, $p(\text{The coin is tails})$ can be interpreted as $p(\text{The coin is tails} = \text{true})$. We use the shorthand $p(x = \text{tr})$ for the probability of event/variable x being in the state true and $p(x = \text{fa})$ for the probability of variable x being in the state false.

Definition 1.1 Rules of probability for discrete variables The probability $p(x = x)$ of variable x being in state x is represented by a value between 0 and 1. $p(x = x) = 1$ means that we are certain x

is in state x . Conversely, $p(x = x) = 0$ means that we are certain x is not in state x . Values between 0 and 1 represent the degree of certainty of state occupancy.

The summation of the probability over all the states is 1:

$$\sum_{x \in \text{dom}(x)} p(x = x) = 1. \quad (1.1.1)$$

This is called the normalisation condition. We will usually more conveniently write $\sum_x p(x) = 1$.

Two variables x and y can interact through

$$p(x = a \text{ or } y = b) = p(x = a) + p(y = b) - p(x = a \text{ and } y = b). \quad (1.1.2)$$

Or, more generally, we can write

$$p(x \text{ or } y) = p(x) + p(y) - p(x \text{ and } y). \quad (1.1.3)$$

We will use the shorthand $p(x, y)$ for $p(x \text{ and } y)$. Note that $p(y, x) = p(x, y)$ and $p(x \text{ or } y) = p(y \text{ or } x)$.

Definition 1.2 Set notation An alternative notation in terms of set theory is to write

$$p(x \text{ or } y) \equiv p(x \cup y), \quad p(x, y) \equiv p(x \cap y). \quad (1.1.4)$$

Definition 1.3 Marginals Given a *joint distribution* $p(x, y)$ the distribution of a single variable is given by

$$p(x) = \sum_y p(x, y). \quad (1.1.5)$$

Here $p(x)$ is termed a *marginal* of the joint probability distribution $p(x, y)$. The process of computing a marginal from a joint distribution is called *marginalisation*. More generally, one has

$$p(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = \sum_{x_i} p(x_1, \dots, x_n). \quad (1.1.6)$$

Definition 1.4 Conditional probability/Bayes' rule The probability of event x conditioned on knowing event y (or more shortly, the probability of x given y) is defined as

$$p(x|y) \equiv \frac{p(x, y)}{p(y)}. \quad (1.1.7)$$

If $p(y) = 0$ then $p(x|y)$ is not defined. From this definition and $p(x, y) = p(y, x)$ we immediately arrive at Bayes' rule

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}. \quad (1.1.8)$$

Since Bayes' rule trivially follows from the definition of conditional probability, we will sometimes be loose in our language and use the terms Bayes' rule and conditional probability as synonymous.

As we shall see throughout this book, Bayes' rule plays a central role in probabilistic reasoning since it helps us 'invert' probabilistic relationships, translating between $p(y|x)$ and $p(x|y)$.

Definition 1.5 Probability density functions For a continuous variable x , the probability density $f(x)$ is defined such that

$$f(x) \geq 0, \quad \int_{-\infty}^{\infty} f(x)dx = 1, \quad (1.1.9)$$

and the probability that x falls in an interval $[a, b]$ is given by

$$p(a \leq x \leq b) = \int_a^b f(x)dx. \quad (1.1.10)$$

As shorthand we will sometimes write $\int_x f(x)$, particularly when we want an expression to be valid for either continuous or discrete variables. The multivariate case is analogous with integration over all real space, and the probability that x belongs to a region of the space defined accordingly. Unlike probabilities, probability densities can take positive values greater than 1.

Formally speaking, for a continuous variable, one should not speak of the probability that $x = 0.2$ since the probability of a single value is always zero. However, we shall often write $p(x)$ for continuous variables, thus not distinguishing between probabilities and probability density function values. Whilst this may appear strange, the nervous reader may simply replace our $p(x)$ notation for $\int_{x \in \Delta} f(x)dx$, where Δ is a small region centred on x . This is well defined in a probabilistic sense and, in the limit Δ being very small, this would give approximately $\Delta f(x)$. If we consistently use the same Δ for all occurrences of pdfs, then we will simply have a common prefactor Δ in all expressions. Our strategy is to simply ignore these values (since in the end only relative probabilities will be relevant) and write $p(x)$. In this way, all the standard rules of probability carry over, including Bayes' rule.

Remark 1.1 (Subjective probability) Probability is a contentious topic and we do not wish to get bogged down by the debate here, apart from pointing out that it is not necessarily the rules of probability that are contentious, rather what interpretation we should place on them. In some cases potential repetitions of an experiment can be envisaged so that the 'long run' (or frequentist) definition of probability in which probabilities are defined with respect to a potentially infinite repetition of experiments makes sense. For example, in coin tossing, the probability of heads might be interpreted as 'If I were to repeat the experiment of flipping a coin (at "random"), the limit of the number of heads that occurred over the number of tosses is defined as the probability of a head occurring.'

Here's a problem that is typical of the kind of scenario one might face in a machine learning situation. A film enthusiast joins a new online film service. Based on expressing a few films a user likes and dislikes, the online company tries to estimate the probability that the user will like each of the 10 000 films in their database. If we were to define probability as a limiting case of infinite repetitions of the same experiment, this wouldn't make much sense in this case since we can't repeat the experiment. However, if we assume that the user behaves in a manner consistent with other users, we should be able to exploit the large amount of data from other users' ratings to make a reasonable 'guess' as to what this consumer likes. This *degree of belief* or *Bayesian* subjective interpretation of probability sidesteps non-repeatability issues – it's just a framework for manipulating real values consistent with our intuition about probability [158].

1.1.1 Interpreting conditional probability

Conditional probability matches our intuitive understanding of uncertainty. For example, imagine a circular dart board, split into 20 equal sections, labelled from 1 to 20. Randy, a dart thrower, hits any one of the 20 sections uniformly at random. Hence the probability that a dart thrown by Randy occurs in any one of the 20 regions is $p(\text{region } i) = 1/20$. A friend of Randy tells him that he hasn't hit the 20 region. What is the probability that Randy has hit the 5 region? Conditioned on this information, only regions 1 to 19 remain possible and, since there is no preference for Randy to hit any of these regions, the probability is $1/19$. The conditioning means that certain states are now

inaccessible, and the original probability is subsequently distributed over the remaining accessible states. From the rules of probability:

$$p(\text{region 5}|\text{not region 20}) = \frac{p(\text{region 5, not region 20})}{p(\text{not region 20})} = \frac{p(\text{region 5})}{p(\text{not region 20})} = \frac{1/20}{19/20} = \frac{1}{19}$$

giving the intuitive result. An important point to clarify is that $p(A = a|B = b)$ should not be interpreted as ‘Given the event $B = b$ has occurred, $p(A = a|B = b)$ is the probability of the event $A = a$ occurring’. In most contexts, no such explicit temporal causality is implied¹ and the correct interpretation should be ‘ $p(A = a|B = b)$ is the probability of A being in state a under the constraint that B is in state b ’.

The relation between the conditional $p(A = a|B = b)$ and the joint $p(A = a, B = b)$ is just a normalisation constant since $p(A = a, B = b)$ is not a distribution in A – in other words, $\sum_a p(A = a, B = b) \neq 1$. To make it a distribution we need to divide: $p(A = a, B = b) / \sum_a p(A = a, B = b)$ which, when summed over a does sum to 1. Indeed, this is just the definition of $p(A = a|B = b)$.

Definition 1.6 Independence Variables x and y are independent if knowing the state (or value in the continuous case) of one variable gives no extra information about the other variable. Mathematically, this is expressed by

$$p(x, y) = p(x)p(y). \quad (1.1.11)$$

Provided that $p(x) \neq 0$ and $p(y) \neq 0$ independence of x and y is equivalent to

$$p(x|y) = p(x) \Leftrightarrow p(y|x) = p(y). \quad (1.1.12)$$

If $p(x|y) = p(x)$ for all states of x and y , then the variables x and y are said to be independent. If

$$p(x, y) = kf(x)g(y) \quad (1.1.13)$$

for some constant k , and positive functions $f(\cdot)$ and $g(\cdot)$ then x and y are independent and we write $x \perp\!\!\!\perp y$.

Example 1.1 Independence

Let x denote the day of the week in which females are born, and y denote the day in which males are born, with $\text{dom}(x) = \text{dom}(y) = \{1, \dots, 7\}$. It is reasonable to expect that x is independent of y . We randomly select a woman from the phone book, Alice, and find out that she was born on a Tuesday. We also select a male at random, Bob. Before phoning Bob and asking him, what does knowing Alice’s birthday add to which day we think Bob is born on? Under the independence assumption, the answer is nothing. Note that this doesn’t mean that the distribution of Bob’s birthday is necessarily uniform – it just means that knowing when Alice was born doesn’t provide any extra information than we already knew about Bob’s birthday, $p(y|x) = p(y)$. Indeed, the distribution of birthdays $p(y)$ and $p(x)$ are non-uniform (statistically fewer babies are born on weekends), though there is nothing to suggest that x and y are dependent.

Deterministic dependencies

Sometimes the concept of independence is perhaps a little strange. Consider the following: variables x and y are both binary (their domains consist of two states). We define the distribution such that x

¹ We will discuss issues related to causality further in Section 3.4.

and y are always both in a certain joint state:

$$p(x = a, y = 1) = 1, \quad p(x = a, y = 2) = 0, \quad p(x = b, y = 2) = 0, \quad p(x = b, y = 1) = 0.$$

Are x and y dependent? The reader may show that $p(x = a) = 1$, $p(x = b) = 0$ and $p(y = 1) = 1$, $p(y = 2) = 0$. Hence $p(x)p(y) = p(x, y)$ for all states of x and y , and x and y are therefore independent. This may seem strange – we know for sure the relation between x and y , namely that they are always in the same joint state, yet they are independent. Since the distribution is trivially concentrated in a single joint state, knowing the state of x tells you nothing that you didn't anyway know about the state of y , and vice versa. This potential confusion comes from using the term 'independent' which may suggest that there is no relation between objects discussed. The best way to think about statistical independence is to ask whether or not knowing the state of variable y tells you something more than you knew before about variable x , where 'knew before' means working with the joint distribution of $p(x, y)$ to figure out what we can know about x , namely $p(x)$.

Definition 1.7 Conditional independence

$$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z} \tag{1.1.14}$$

denotes that the two sets of variables \mathcal{X} and \mathcal{Y} are independent of each other provided we know the state of the set of variables \mathcal{Z} . For conditional independence, \mathcal{X} and \mathcal{Y} must be independent given *all* states of \mathcal{Z} . Formally, this means that

$$p(\mathcal{X}, \mathcal{Y} | \mathcal{Z}) = p(\mathcal{X} | \mathcal{Z})p(\mathcal{Y} | \mathcal{Z}) \tag{1.1.15}$$

for all states of $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$. In case the conditioning set is empty we may also write $\mathcal{X} \perp\!\!\!\perp \mathcal{Y}$ for $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \emptyset$, in which case \mathcal{X} is (unconditionally) independent of \mathcal{Y} .

If \mathcal{X} and \mathcal{Y} are not conditionally independent, they are conditionally dependent. This is written

$$\mathcal{X} \not\perp\!\!\!\perp \mathcal{Y} | \mathcal{Z} \tag{1.1.16}$$

Similarly $\mathcal{X} \not\perp\!\!\!\perp \mathcal{Y} | \emptyset$ can be written as $\mathcal{X} \not\perp\!\!\!\perp \mathcal{Y}$.

Intuitively, if x is conditionally independent of y given z , this means that, given z , y contains no additional information about x . Similarly, given z , knowing x does not tell me anything more about y . Note that $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z} \Rightarrow \mathcal{X}' \perp\!\!\!\perp \mathcal{Y}' | \mathcal{Z}$ for $\mathcal{X}' \subseteq \mathcal{X}$ and $\mathcal{Y}' \subseteq \mathcal{Y}$.

Remark 1.2 (Independence implications) It's tempting to think that if a is independent of b and b is independent of c then a must be independent of c :

$$\{a \perp\!\!\!\perp b, b \perp\!\!\!\perp c\} \Rightarrow a \perp\!\!\!\perp c. \tag{1.1.17}$$

However, this does not follow. Consider for example a distribution of the form

$$p(a, b, c) = p(b)p(a, c). \tag{1.1.18}$$

From this

$$p(a, b) = \sum_c p(a, b, c) = p(b) \sum_c p(a, c). \tag{1.1.19}$$

Hence $p(a, b)$ is a function of b multiplied by a function of a so that a and b are independent. Similarly, one can show that b and c are independent. However, a is not necessarily independent of c since the distribution $p(a, c)$ can be set arbitrarily.

prior criminal knowledge can be formulated mathematically as follows:

$$\text{dom}(B) = \text{dom}(M) = \{\text{murderer, not murderer}\}, \text{dom}(K) = \{\text{knife used, knife not used}\} \quad (1.2.4)$$

$$p(B = \text{murderer}) = 0.6, \quad p(M = \text{murderer}) = 0.2 \quad (1.2.5)$$

$$\begin{aligned} p(\text{knife used} | B = \text{not murderer}, M = \text{not murderer}) &= 0.3 \\ p(\text{knife used} | B = \text{not murderer}, M = \text{murderer}) &= 0.2 \\ p(\text{knife used} | B = \text{murderer}, M = \text{not murderer}) &= 0.6 \\ p(\text{knife used} | B = \text{murderer}, M = \text{murderer}) &= 0.1. \end{aligned} \quad (1.2.6)$$

In addition $p(K, B, M) = p(K|B, M)p(B)p(M)$. Assuming that the knife is the murder weapon, what is the probability that the Butler is the murderer? (Remember that it might be that neither is the murderer.) Using b for the two states of B and m for the two states of M ,

$$\begin{aligned} p(B|K) &= \sum_m p(B, m|K) = \sum_m \frac{p(B, m, K)}{p(K)} = \frac{\sum_m p(K|B, m)p(B, m)}{\sum_{m,b} p(K|b, m)p(b, m)} \\ &= \frac{p(B) \sum_m p(K|B, m)p(m)}{\sum_b p(b) \sum_m p(K|b, m)p(m)}. \end{aligned} \quad (1.2.7)$$

where we used the fact that in our model $p(B, M) = p(B)p(M)$. Plugging in the values we have (see also `demoClouseau.m`)

$$p(B = \text{murderer} | \text{knife used}) = \frac{\frac{6}{10} \left(\frac{2}{10} \times \frac{1}{10} + \frac{8}{10} \times \frac{6}{10} \right)}{\frac{6}{10} \left(\frac{2}{10} \times \frac{1}{10} + \frac{8}{10} \times \frac{6}{10} \right) + \frac{4}{10} \left(\frac{2}{10} \times \frac{2}{10} + \frac{8}{10} \times \frac{3}{10} \right)} = \frac{300}{412} \approx 0.73. \quad (1.2.8)$$

Hence knowing that the knife was the murder weapon strengthens our belief that the butler did it.

Remark 1.3 The role of $p(\text{knife used})$ in the Inspector Clouseau example can cause some confusion. In the above,

$$p(\text{knife used}) = \sum_b p(b) \sum_m p(\text{knife used} | b, m) p(m) \quad (1.2.9)$$

is computed to be 0.456. But surely, $p(\text{knife used}) = 1$, since this is given in the question! Note that the quantity $p(\text{knife used})$ relates to the *prior* probability the model assigns to the knife being used (in the absence of any other information). If we know that the knife is used, then the *posterior*

$$p(\text{knife used} | \text{knife used}) = \frac{p(\text{knife used, knife used})}{p(\text{knife used})} = \frac{p(\text{knife used})}{p(\text{knife used})} = 1 \quad (1.2.10)$$

which, naturally, must be the case.

Example 1.4 Who's in the bathroom?

Consider a household of three people, Alice, Bob and Cecil. Cecil wants to go to the bathroom but finds it occupied. He then goes to Alice's room and sees she is there. Since Cecil knows that only either Alice or Bob can be in the bathroom, from this he infers that Bob must be in the bathroom.

To arrive at the same conclusion in a mathematical framework, we define the following events

$$A = \text{Alice is in her bedroom}, \quad B = \text{Bob is in his bedroom}, \quad O = \text{Bathroom occupied.} \quad (1.2.11)$$

We can encode the information that if either Alice or Bob are not in their bedrooms, then they must be in the bathroom (they might both be in the bathroom) as

$$p(O = \text{tr}|A = \text{fa}, B) = 1, \quad p(O = \text{tr}|A, B = \text{fa}) = 1. \quad (1.2.12)$$

The first term expresses that the bathroom is occupied if Alice is not in her bedroom, wherever Bob is. Similarly, the second term expresses bathroom occupancy as long as Bob is not in his bedroom. Then

$$\begin{aligned} p(B = \text{fa}|O = \text{tr}, A = \text{tr}) &= \frac{p(B = \text{fa}, O = \text{tr}, A = \text{tr})}{p(O = \text{tr}, A = \text{tr})} \\ &= \frac{p(O = \text{tr}|A = \text{tr}, B = \text{fa})p(A = \text{tr}, B = \text{fa})}{p(O = \text{tr}, A = \text{tr})} \end{aligned} \quad (1.2.13)$$

where

$$p(O = \text{tr}, A = \text{tr}) = p(O = \text{tr}|A = \text{tr}, B = \text{fa})p(A = \text{tr}, B = \text{fa}) + p(O = \text{tr}|A = \text{tr}, B = \text{tr})p(A = \text{tr}, B = \text{tr}). \quad (1.2.14)$$

Using the fact $p(O = \text{tr}|A = \text{tr}, B = \text{fa}) = 1$ and $p(O = \text{tr}|A = \text{tr}, B = \text{tr}) = 0$, which encodes that if Alice is in her room and Bob is not, the bathroom must be occupied, and similarly, if both Alice and Bob are in their rooms, the bathroom cannot be occupied,

$$p(B = \text{fa}|O = \text{tr}, A = \text{tr}) = \frac{p(A = \text{tr}, B = \text{fa})}{p(A = \text{tr}, B = \text{fa})} = 1. \quad (1.2.15)$$

This example is interesting since we are not required to make a full probabilistic model in this case thanks to the limiting nature of the probabilities (we don't need to specify $p(A, B)$). The situation is common in limiting situations of probabilities being either 0 or 1, corresponding to traditional logic systems.

Example 1.5 Aristotle: Resolution

We can represent the statement 'All apples are fruit' by $p(F = \text{tr}|A = \text{tr}) = 1$. Similarly, 'All fruits grow on trees' may be represented by $p(T = \text{tr}|F = \text{tr}) = 1$. Additionally we assume that whether or not something grows on a tree depends only on whether or not it is a fruit, $p(T|A, F) = P(T|F)$. From this we can compute

$$\begin{aligned} p(T = \text{tr}|A = \text{tr}) &= \sum_F p(T = \text{tr}|F, A = \text{tr})p(F|A = \text{tr}) = \sum_F p(T = \text{tr}|F)p(F|A = \text{tr}) \\ &= p(T = \text{tr}|F = \text{fa}) \underbrace{p(F = \text{fa}|A = \text{tr})}_{=0} + \underbrace{p(T = \text{tr}|F = \text{tr})}_{=1} \underbrace{p(F = \text{tr}|A = \text{tr})}_{=1} = 1. \end{aligned} \quad (1.2.16)$$

In other words we have deduced that 'All apples grow on trees' is a true statement, based on the information presented. (This kind of reasoning is called resolution and is a form of transitivity: from the statements $A \Rightarrow F$ and $F \Rightarrow T$ we can infer $A \Rightarrow T$.)

Example 1.6 Aristotle: Inverse Modus Ponens

According to Logic, from the statement: ‘If A is true then B is true’, one may deduce that ‘If B is false then A is false’. To see how this fits in with a probabilistic reasoning system we can first express the statement: ‘If A is true then B is true’ as $p(B = \text{tr}|A = \text{tr}) = 1$. Then we may infer

$$\begin{aligned} p(A = \text{fa}|B = \text{fa}) &= 1 - p(A = \text{tr}|B = \text{fa}) \\ &= 1 - \frac{p(B = \text{fa}|A = \text{tr})p(A = \text{tr})}{p(B = \text{fa}|A = \text{tr})p(A = \text{tr}) + p(B = \text{fa}|A = \text{fa})p(A = \text{fa})} = 1. \end{aligned} \quad (1.2.17)$$

This follows since $p(B = \text{fa}|A = \text{tr}) = 1 - p(B = \text{tr}|A = \text{tr}) = 1 - 1 = 0$, annihilating the second term.

Both the above examples are intuitive expressions of deductive logic. The standard rules of Aristotelian logic are therefore seen to be limiting cases of probabilistic reasoning.

Example 1.7 Soft XOR gate

A standard XOR logic gate is given by the table on the right. If we observe that the output of the XOR gate is 0, what can we say about A and B ? In this case, either A and B were both 0, or A and B were both 1. This means we don’t know which state A was in – it could equally likely have been 1 or 0.

A	B	$A \text{ xor } B$
0	0	0
0	1	1
1	0	1
1	1	0

Consider a ‘soft’ version of the XOR gate given on the right, with additionally $A \perp B$ and $p(A = 1) = 0.65$, $p(B = 1) = 0.77$. What is $p(A = 1|C = 0)$?

A	B	$p(C = 1 A, B)$
0	0	0.1
0	1	0.99
1	0	0.8
1	1	0.25

$$\begin{aligned} p(A = 1, C = 0) &= \sum_B p(A = 1, B, C = 0) = \sum_B p(C = 0|A = 1, B)p(A = 1)p(B) \\ &= p(A = 1)(p(C = 0|A = 1, B = 0)p(B = 0) \\ &\quad + p(C = 0|A = 1, B = 1)p(B = 1)) \\ &= 0.65 \times (0.2 \times 0.23 + 0.75 \times 0.77) = 0.405\,275. \end{aligned} \quad (1.2.18)$$

$$\begin{aligned} p(A = 0, C = 0) &= \sum_B p(A = 0, B, C = 0) = \sum_B p(C = 0|A = 0, B)p(A = 0)p(B) \\ &= p(A = 0)(p(C = 0|A = 0, B = 0)p(B = 0) \\ &\quad + p(C = 0|A = 0, B = 1)p(B = 1)) \\ &= 0.35 \times (0.9 \times 0.23 + 0.01 \times 0.77) = 0.075\,145. \end{aligned}$$

Then

$$p(A = 1|C = 0) = \frac{p(A = 1, C = 0)}{p(A = 1, C = 0) + p(A = 0, C = 0)} = \frac{0.405\,275}{0.405\,275 + 0.075\,145} = 0.8436. \quad (1.2.19)$$

Example 1.8 Larry

Larry is typically late for school. If Larry is late, we denote this with $L = \text{late}$, otherwise, $L = \text{not late}$. When his mother asks whether or not he was late for school he never admits to being late. The response Larry gives R_L is represented as follows

$$p(R_L = \text{not late} | L = \text{not late}) = 1, \quad p(R_L = \text{late} | L = \text{late}) = 0. \quad (1.2.20)$$

The remaining two values are determined by normalisation and are

$$p(R_L = \text{late} | L = \text{not late}) = 0, \quad p(R_L = \text{not late} | L = \text{late}) = 1. \quad (1.2.21)$$

Given that $R_L = \text{not late}$, what is the probability that Larry was late, i.e. $p(L = \text{late} | R_L = \text{not late})$?

Using Bayes' we have

$$\begin{aligned} p(L = \text{late} | R_L = \text{not late}) &= \frac{p(L = \text{late}, R_L = \text{not late})}{p(R_L = \text{not late})} \\ &= \frac{p(L = \text{late}, R_L = \text{not late})}{p(L = \text{late}, R_L = \text{not late}) + p(L = \text{not late}, R_L = \text{not late})}. \end{aligned} \quad (1.2.22)$$

In the above

$$p(L = \text{late}, R_L = \text{not late}) = \underbrace{p(R_L = \text{not late} | L = \text{late})}_{=1} p(L = \text{late}) \quad (1.2.23)$$

and

$$p(L = \text{not late}, R_L = \text{not late}) = \underbrace{p(R_L = \text{not late} | L = \text{not late})}_{=1} p(L = \text{not late}). \quad (1.2.24)$$

Hence

$$p(L = \text{late} | R_L = \text{not late}) = \frac{p(L = \text{late})}{p(L = \text{late}) + p(L = \text{not late})} = p(L = \text{late}). \quad (1.2.25)$$

Where we used normalisation in the last step, $p(L = \text{late}) + p(L = \text{not late}) = 1$. This result is intuitive – Larry's mother knows that he never admits to being late, so her belief about whether or not he really was late is unchanged, regardless of what Larry actually says.

Example 1.9 Larry and Sue

Continuing the example above, Larry's sister Sue always tells the truth to her mother as to whether or not Larry was late for school.

$$p(R_S = \text{not late} | L = \text{not late}) = 1, \quad p(R_S = \text{late} | L = \text{late}) = 1. \quad (1.2.26)$$

The remaining two values are determined by normalisation and are

$$p(R_S = \text{late} | L = \text{not late}) = 0, \quad p(R_S = \text{not late} | L = \text{late}) = 0. \quad (1.2.27)$$

We also assume $p(R_S, R_L | L) = p(R_S | L)p(R_L | L)$. We can then write

$$p(R_L, R_S, L) = p(R_L | L)p(R_S | L)p(L). \quad (1.2.28)$$

Given that $R_S = \text{late}$ and $R_L = \text{not late}$, what is the probability that Larry was late?

Using Bayes' rule, we have

$$\begin{aligned} p(L = \text{late} | R_L = \text{not late}, R_S = \text{late}) \\ = \frac{1}{Z} p(R_S = \text{late} | L = \text{late}) p(R_L = \text{not late} | L = \text{late}) p(L = \text{late}) \end{aligned} \quad (1.2.29)$$

where the normalisation Z is given by

$$\begin{aligned} p(R_S = \text{late} | L = \text{late}) p(R_L = \text{not late} | L = \text{late}) p(L = \text{late}) \\ + p(R_S = \text{late} | L = \text{not late}) p(R_L = \text{not late} | L = \text{not late}) \times p(L = \text{not late}). \end{aligned} \quad (1.2.30)$$

Hence

$$p(L = \text{late} | R_L = \text{not late}, R_S = \text{late}) = \frac{1 \times 1 \times p(L = \text{late})}{1 \times 1 \times p(L = \text{late}) + 0 \times 1 \times p(L = \text{not late})} = 1. \quad (1.2.31)$$

This result is also intuitive – since Larry's mother knows that Sue always tells the truth, no matter what Larry says, she knows he was late.

Example 1.10 Luke

Luke has been told he's lucky and has won a prize in the lottery. There are five prizes available of value £10, £100, £1000, £10 000, £1 000 000. The prior probabilities of winning these five prizes are p_1, p_2, p_3, p_4, p_5 , with p_0 being the prior probability of winning no prize. Luke asks eagerly 'Did I win £1 000 000?!'. 'I'm afraid not sir', is the response of the lottery phone operator. 'Did I win £10 000?!' asks Luke. 'Again, I'm afraid not sir'. What is the probability that Luke has won £1000?

Note first that $p_0 + p_1 + p_2 + p_3 + p_4 + p_5 = 1$. We denote $W = 1$ for the first prize of £10, and $W = 2, \dots, 5$ for the remaining prizes and $W = 0$ for no prize. We need to compute

$$\begin{aligned} p(W = 3 | W \neq 5, W \neq 4, W \neq 0) &= \frac{p(W = 3, W \neq 5, W \neq 4, W \neq 0)}{p(W \neq 5, W \neq 4, W \neq 0)} \\ &= \frac{p(W = 3)}{p(W = 1 \text{ or } W = 2 \text{ or } W = 3)} = \frac{p_3}{p_1 + p_2 + p_3} \end{aligned} \quad (1.2.32)$$

where the term in the denominator is computed using the fact that the events W are mutually exclusive (one can only win one prize). This result makes intuitive sense: once we have removed the impossible states of W , the probability that Luke wins the prize is proportional to the prior probability of that prize, with the normalisation being simply the total set of possible probability remaining.

1.3 Prior, likelihood and posterior

Much of science deals with problems of the form: tell me something about the variable θ given that I have observed data \mathcal{D} and have some knowledge of the underlying data generating mechanism.

The posterior is then given by,

$$p(s_a, s_b | t = 9) = \frac{p(t = 9 | s_a, s_b) p(s_a) p(s_b)}{p(t = 9)} \quad (1.3.10)$$

where

$$p(t = 9) = \sum_{s_a, s_b} p(t = 9 | s_a, s_b) p(s_a) p(s_b). \quad (1.3.11)$$

The term $p(t = 9) = \sum_{s_a, s_b} p(t = 9 | s_a, s_b) p(s_a) p(s_b) = 4 \times 1/36 = 1/9$. Hence the posterior is given by equal mass in only four non-zero elements, as shown.

$p(s_a, s_b | t = 9)$:

	$s_a = 1$	$s_a = 2$	$s_a = 3$	$s_a = 4$	$s_a = 5$	$s_a = 6$
$s_b = 1$	0	0	0	0	0	0
$s_b = 2$	0	0	0	0	0	0
$s_b = 3$	0	0	0	0	0	1/4
$s_b = 4$	0	0	0	0	1/4	0
$s_b = 5$	0	0	0	1/4	0	0
$s_b = 6$	0	0	1/4	0	0	0

1.4 Summary

- The standard rules of probability are a consistent, logical way to reason with uncertainty.
- Bayes' rule mathematically encodes the process of inference.

A useful introduction to probability is given in [292]. The interpretation of probability is contentious and we refer the reader to [158, 197, 193] for detailed discussions. The website understandinguncertainty.org contains entertaining discussions on reasoning with uncertainty.

1.5 Code

The `BRMLTOOLBOX` code accompanying this book is intended to give the reader some insight into representing discrete probability tables and performing simple inference. We provide here only the briefest of descriptions of the code and the reader is encouraged to experiment with the demos to understand better the routines and their purposes.

1.5.1 Basic probability code

At the simplest level, we only need two basic routines. One for multiplying probability tables together (called potentials in the code), and one for summing a probability table. Potentials are represented using a structure. For example, in the code corresponding to the Inspector Clouseau example `demoClouseau.m`, we define a probability table as

```
>> pot(1)
ans =
    variables: [1 3 2]
    table: [2x2x2 double]
```

This says that the potential depends on the variables 1, 3, 2 and the entries are stored in the array given by the `table` field. The size of the array informs how many states each variable takes in the order given by `variables`. The order in which the variables are defined in a potential is irrelevant provided that one indexes the array consistently. A routine that can help with setting table entries is `setstate.m`. For example,

```
>> pot(1) = setstate(pot(1), [2 1 3], [2 1 1], 0.3)
```

means that for potential 1, the table entry for variable 2 being in state 2, variable 1 being in state 1 and variable 3 being in state 1 should be set to value 0.3.

The philosophy of the code is to keep the information required to perform computations to a minimum. Additional information about the labels of variables and their domains can be useful to interpret results, but is not actually required to carry out computations. One may also specify the name and domain of each variable, for example

```
>>variable(3)
ans =
    domain: {'murderer' 'not murderer'}
    name: 'butler'
```

The variable name and domain information in the Clouseau example is stored in the structure `variable`, which can be helpful to display the potential table:

```
>> disptable(pot(1),variable);
knife = used      maid = murderer      butler = murderer      0.100000
knife = not used  maid = murderer      butler = murderer      0.900000
knife = used      maid = not murderer  butler = murderer      0.600000
knife = not used  maid = not murderer  butler = murderer      0.400000
knife = used      maid = murderer      butler = not murderer  0.200000
knife = not used  maid = murderer      butler = not murderer  0.800000
knife = used      maid = not murderer  butler = not murderer  0.300000
knife = not used  maid = not murderer  butler = not murderer  0.700000
```

Multiplying potentials

In order to multiply potentials, (as for arrays) the tables of each potential must be dimensionally consistent – that is the number of states of variable i must be the same for all potentials. This can be checked using `potvariables.m`. This consistency is also required for other basic operations such as summing potentials.

`multpots.m`: Multiplying two or more potentials
`divpots.m`: Dividing a potential by another

Summing a potential

`sumpot.m`: Sum (marginalise) a potential over a set of variables
`sumpots.m`: Sum a set of potentials together

Making a conditional potential

`condpot.m`: Make a potential conditioned on variables

Setting a potential

`setpot.m`: Set variables in a potential to given states
`setevpot.m`: Set variables in a potential to given states and return also an identity potential on the given states

The philosophy of `BRMLTOOLBOX` is that all information about variables is local and is read off from a potential. Using `setevpot.m` enables one to set variables in a state whilst maintaining information about the number of states of a variable.

Maximising a potential

`maxpot.m`: Maximise a potential over a set of variables

See also `maxNarray.m` and `maxNpot.m` which return the N -highest values and associated states.

Other potential utilities

`setstate.m`: Set a potential state to a given value

`table.m`: Return a table from a potential

`whichpot.m`: Return potentials which contain a set of variables

`potvariables.m`: Variables and their number of states in a set of potentials

`orderpotfields.m`: Order the fields of a potential structure

`uniquepots.m`: Merge redundant potentials by multiplication and return only unique ones

`numstates.m`: Number of states of a variable in a domain

`squeezepots.m`: Find unique potentials and rename the variables 1,2,...

`normpot.m`: Normalise a potential to form a distribution

1.5.2 General utilities

`condp.m`: Return a table $p(x|y)$ from $p(x, y)$

`condexp.m`: Form a conditional distribution from a log value

`logsumexp.m`: Compute the log of a sum of exponentials in a numerically precise way

`normp.m`: Return a normalised table from an unnormalised table

`assign.m`: Assign values to multiple variables

`maxarray.m`: Maximise a multi-dimensional array over a subset

1.5.3 An example

The following code highlights the use of the above routines in solving the Inspector Clouseau, Example 1.3, and the reader is invited to examine the code to become familiar with how to numerically represent probability tables

`demoClouseau.m`: Solving the Inspector Clouseau example

1.6 Exercises

1.1 Prove

$$p(x, y|z) = p(x|z)p(y|x, z) \quad (1.6.1)$$

and also

$$p(x|y, z) = \frac{p(y|x, z)p(x|z)}{p(y|z)}. \quad (1.6.2)$$

1.2 Prove the *Bonferroni inequality*

$$p(a, b) \geq p(a) + p(b) - 1. \quad (1.6.3)$$

1.3 (Adapted from [181]) There are two boxes. Box 1 contains three red and five white balls and box 2 contains two red and five white balls. A box is chosen at random $p(\text{box} = 1) = p(\text{box} = 2) = 0.5$ and a ball chosen at random from this box turns out to be red. What is the posterior probability that the red ball came from box 1?

1.4 (Adapted from [181]) Two balls are placed in a box as follows: A fair coin is tossed and a white ball is placed in the box if a head occurs, otherwise a red ball is placed in the box. The coin is tossed again and a red ball is placed in the box if a tail occurs, otherwise a white ball is placed in the box. Balls are drawn from the box three times in succession (always with replacing the drawn ball back in the box). It is found that on all three occasions a red ball is drawn. What is the probability that both balls in the box are red?

1.5 (From David Spiegelhalter understandinguncertainty.org) A secret government agency has developed a scanner which determines whether a person is a terrorist. The scanner is fairly reliable; 95% of all scanned terrorists are identified as terrorists, and 95% of all upstanding citizens are identified as such. An informant tells the agency that exactly one passenger of 100 aboard an aeroplane in which you are seated is a terrorist. The agency decides to scan each passenger and the shifty-looking man sitting next to you is the first to test positive. What are the chances that this man is a terrorist?

1.6 Consider three variable distributions which admit the factorisation

$$p(a, b, c) = p(a|b)p(b|c)p(c) \quad (1.6.4)$$

where all variables are binary. How many parameters are needed to specify distributions of this form?

1.7 Repeat the Inspector Clouseau scenario, Example 1.3, but with the restriction that either the Maid or the Butler is the murderer, but not both. Explicitly, the probability of the Maid being the murderer and not the Butler is 0.04, the probability of the Butler being the murderer and not the Maid is 0.64. Modify `demoClouseau.m` to implement this.

1.8 Prove

$$p(a, (b \text{ or } c)) = p(a, b) + p(a, c) - p(a, b, c). \quad (1.6.5)$$

1.9 Prove

$$p(x|z) = \sum_y p(x|y, z)p(y|z) = \sum_{y,w} p(x|w, y, z)p(w|y, z)p(y|z). \quad (1.6.6)$$

1.10 As a young man Mr Gott visits Berlin in 1969. He's surprised that he cannot cross into East Berlin since there is a wall separating the two halves of the city. He's told that the wall was erected eight years previously. He reasons that: The wall will have a finite lifespan; his ignorance means that he arrives uniformly at random at some time in the lifespan of the wall. Since only 5% of the time one would arrive in the first or last 2.5% of the lifespan of the wall he asserts that with 95% confidence the wall will survive between $8/0.975 \approx 8.2$ and $8/0.025 = 320$ years. In 1989 the now Professor Gott is pleased to find that his prediction was correct and promotes his prediction method in prestigious journals. This 'delta-t' method is widely adopted and used to form predictions in a range of scenarios about which researchers are 'totally ignorant'. Would you 'buy' a prediction from Professor Gott? Explain carefully your reasoning.

1.11 Implement the soft XOR gate, Example 1.7 using BRMLTOOLBOX. You may find `condpot.m` of use.

1.12 Implement the hamburgers, Example 1.2 (both scenarios) using BRMLTOOLBOX. To do so you will need to define the joint distribution $p(\text{hamburgers}, KJ)$ in which $\text{dom}(\text{hamburgers}) = \text{dom}(KJ) = \{\text{tr}, \text{fa}\}$.

- 1.13** Implement the two-dice example, Section 1.3.1 using `BRMLTOOLBOX`.
- 1.14** A redistribution lottery involves picking the correct four numbers from 1 to 9 (without replacement, so 3,4,4,1 for example is not possible). The order of the picked numbers is irrelevant. Every week a million people play this game, each paying £1 to enter, with the numbers 3,5,7,9 being the most popular (1 in every 100 people chooses these numbers). Given that the million pounds prize money is split equally between winners, and that any four (different) numbers come up at random, what is the expected amount of money each of the players choosing 3,5,7,9 will win each week? The least popular set of numbers is 1,2,3,4 with only 1 in 10 000 people choosing this. How much do they profit each week, on average? Do you think there is any ‘skill’ involved in playing this lottery?
- 1.15** In a test of ‘psychometry’ the car keys and wristwatches of five people are given to a medium. The medium then attempts to match the wristwatch with the car key of each person. What is the expected number of correct matches that the medium will make (by chance)? What is the probability that the medium will obtain at least one correct match?
- 1.16** 1. Show that for any function f
- $$\sum_x p(x|y)f(y) = f(y). \quad (1.6.7)$$
2. Explain why, in general,
- $$\sum_x p(x|y)f(x, y) \neq \sum_x f(x, y). \quad (1.6.8)$$
- 1.17** (Inspired by singingbanana.com). Seven friends decide to order pizzas by telephone from Pizza4U based on a flyer pushed through their letterbox. Pizza4U has only four kinds of pizza, and each person chooses a pizza independently. Bob phones Pizza4U and places the combined pizza order, simply stating how many pizzas of each kind are required. Unfortunately, the precise order is lost, so the chef makes seven randomly chosen pizzas and then passes them to the delivery boy.
1. How many different combined orders are possible?
 2. What is the probability that the delivery boy has the right order?
- 1.18** Sally is new to the area and listens to some friends discussing about another female friend. Sally knows that they are talking about either Alice or Bella but doesn’t know which. From previous conversations Sally knows some independent pieces of information: She’s 90% sure that Alice has a white car, but doesn’t know if Bella’s car is white or black. Similarly, she’s 90% sure that Bella likes sushi, but doesn’t know if Alice likes sushi. Sally hears from the conversation that the person being discussed hates sushi and drives a white car. What is the probability that the friends are talking about Alice?
- 1.19** The weather in London can be summarised as: if it rains one day there’s a 70% chance it will rain the following day; if it’s sunny one day there’s a 40% chance it will be sunny the following day.
1. Assuming that the prior probability it rained yesterday is 0.5, what is the probability that it was raining yesterday given that it’s sunny today?
 2. If the weather follows the same pattern as above, day after day, what is the probability that it will rain on any day (based on an effectively infinite number of days of observing the weather)?
 3. Use the result from part 2 above as a new prior probability of rain yesterday and recompute the probability that it was raining yesterday given that it’s sunny today.

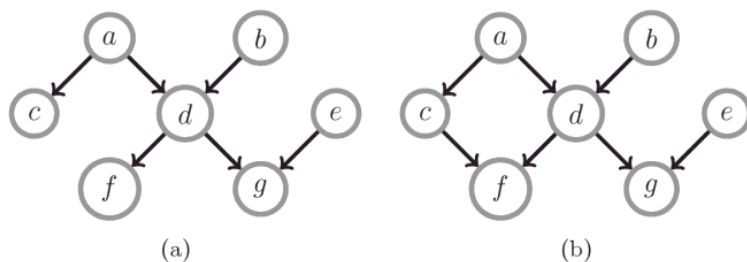


Figure 2.1 (a) Singly connected graph. (b) Multiply connected graph.

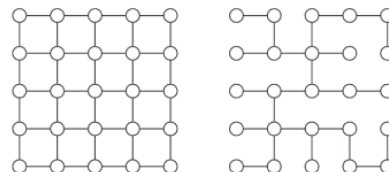
Cliques play a central role in both modelling and inference. In modelling they will describe variables that are all dependent on each other, see Chapter 4. In inference they describe sets of variables with no simpler structure describing the relationship between them and hence for which no simpler efficient inference procedure is likely to exist. We will discuss this issue in detail in Chapter 5 and Chapter 6.

Definition 2.8 Connected graph An undirected graph is *connected* if there is a path between every pair of nodes (i.e. there are no isolated islands). For a graph which is not connected, the connected components are those subgraphs which are connected.

Definition 2.9 Singly-connected graph A graph is *singly connected* if there is only one path from any node A to any other node B . Otherwise the graph is *multiply connected* (see Fig. 2.1). This definition applies regardless of whether or not the edges in the graph are directed. An alternative name for a singly connected graph is a *tree*. A multiply connected graph is also called *loopy*.

Definition 2.10 Spanning tree

A spanning tree of an undirected graph G is a singly connected subset of the existing edges such that the resulting singly connected graph covers all nodes of G . On the right is a graph and an associated spanning tree. A maximum weight spanning tree is a spanning tree such that the sum of all weights on the edges of the tree is at least as large as any other spanning tree of G .



Procedure 2.1 (Finding a maximal weight spanning tree) An algorithm to find a spanning tree with maximal weight is as follows: Start by picking the edge with the largest weight and add this to the edge set. Then pick the next candidate edge which has the largest weight and add this to the edge set – if this results in an edge set with cycles, then reject the candidate edge and propose the next largest edge weight. Note that there may be more than one maximal weight spanning tree.

2.2 Numerically encoding graphs

Our ultimate goal is to make computational implementations of inference. Therefore, if we want to incorporate graph structure into our models, we need to express graphs in a way that a computer can understand and manipulate. There are several equivalent possibilities.

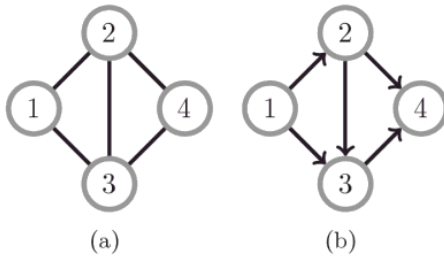


Figure 2.2 (a) An undirected graph can be represented as a symmetric adjacency matrix. (b) A directed graph with nodes labelled in ancestral order corresponds to a triangular adjacency matrix.

2.2.1 Edge list

As the name suggests, an *edge list* simply lists which node-node pairs are in the graph. For Fig. 2.2(a), an edge list is $L = \{(1, 2), (2, 1), (1, 3), (3, 1), (2, 3), (3, 2), (2, 4), (4, 2), (3, 4), (4, 3)\}$. Undirected edges are listed twice, once for each direction.

2.2.2 Adjacency matrix

An alternative is to use an *adjacency matrix*

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad (2.2.1)$$

where $A_{ij} = 1$ if there is an edge from node i to node j in the graph, and 0 otherwise. Some authors include self-connections and place 1's on the diagonal in this definition. An undirected graph has a symmetric adjacency matrix.

Provided that the nodes are labelled in *ancestral order* (parents always come before children) a directed graph Fig. 2.2(b) can be represented as a triangular adjacency matrix:

$$\mathbf{T} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \quad (2.2.2)$$

Adjacency matrix powers

Adjacency matrices may seem wasteful since many of the entries are zero. However, they have a useful property that more than redeems them. For an $N \times N$ adjacency matrix \mathbf{A} , powers of the adjacency matrix $[\mathbf{A}^k]_{ij}$ specify how many paths there are from node i to node j in k edge hops. If we include 1's on the diagonal of \mathbf{A} then $[\mathbf{A}^{N-1}]_{ij}$ is non-zero when there is a path connecting i to j in the graph. If \mathbf{A} corresponds to a DAG the non-zero entries of the j th row of $[\mathbf{A}^{N-1}]$ correspond to the descendants of node j .

2.2.3 Clique matrix

For an undirected graph with N nodes and maximal cliques C_1, \dots, C_K a clique matrix is an $N \times K$ matrix in which each column c_k has zeros except for ones on entries describing the clique. For example

$$\mathbf{C} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \quad (2.2.3)$$

is a clique matrix for Fig. 2.2(a). A cliquo matrix relaxes the constraint that cliques are required to be maximal. A cliquo matrix containing only two-node cliques is called an *incidence matrix*. For example

$$\mathbf{C}_{inc} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad (2.2.4)$$

is an incidence matrix for Fig. 2.2(a). It is straightforward to show that $\mathbf{C}_{inc}\mathbf{C}_{inc}^T$ is equal to the adjacency matrix except that the diagonals now contain the *degree* of each node (the number of edges it touches). Similarly, for any cliquo matrix the diagonal entry of $[\mathbf{C}\mathbf{C}^T]_{ii}$ expresses the number of cliques (columns) that node i occurs in. Off-diagonal elements $[\mathbf{C}\mathbf{C}^T]_{ij}$ contain the number of cliques that nodes i and j jointly inhabit.

Remark 2.1 (Graph confusions) Graphs are widely used, but differ markedly in what they represent. Two potential pitfalls are described below.

State-transition diagrams Such representations are used in Markov chains and finite state automata. Each state is a node and a directed edge between node i and node j (with an associated weight p_{ij}) represents that a transition from state i to state j can occur with probability p_{ij} . From the graphical models perspective we use a directed graph $x(t) \rightarrow x(t+1)$ to represent this Markov chain. The state-transition diagram provides a more detailed graphical description of the conditional probability table $p(x(t+1)|x(t))$.

Neural networks Neural networks also have nodes and edges. In general, however, neural networks are graphical representations of *functions*, whereas graphical models are representations of distributions.

2.3 Summary

- A graph is made of nodes and edges, which we will use to represent variables and relations between them.
- A DAG is an acyclic graph and will be useful for representing 'causal' relationships between variables.
- Neighbouring nodes on an undirected graph will represent dependent variables.
- A graph is singly connected if there is only one path from any node to any other – otherwise the graph is multiply connected.

- A clique is group of nodes all of which are connected to each other.
- The adjacency matrix is a machine-readable description of a graph. Powers of the adjacency matrix give information on the paths between nodes.

Good references for graphs, associated theories and their uses are [86, 121].

2.4 Code

2.4.1 Utility routines

`drawNet.m`: Draw a graph based on an adjacency matrix
`ancestors.m`: Find the ancestors of a node in a DAG
`edges.m`: Edge list from an adjacency matrix
`ancestralorder.m`: Ancestral order from a DAG
`connectedComponents.m`: Connected components
`parents.m`: Parents of a node given an adjacency matrix
`children.m`: Children of a node given an adjacency matrix
`neigh.m`: Neighbours of a node given an adjacency matrix

A connected graph is a tree if the number of edges plus 1 is equal to the number of nodes. However, for a disconnected graph this is not the case. The code `istree.m` below deals with the disconnected case. The routine is based on the observation that any singly connected graph must always possess a simplicial node (a leaf node) which can be eliminated to reveal a smaller singly connected graph.

`istree.m`: If graph is singly connected return 1 and elimination sequence
`spantree.m`: Return a spanning tree from an ordered edge list
`singleparenttree.m`: Find a directed tree with at most one parent from an undirected tree

Additional routines for basic graph manipulations are given at the end of Chapter 6.

2.5 Exercises

- 2.1 Consider an adjacency matrix \mathbf{A} with elements $[\mathbf{A}]_{ij} = 1$ if one can reach state i from state j in one timestep, and 0 otherwise. Show that the matrix $[\mathbf{A}^k]_{ij}$ represents the number of paths that lead from state j to i in k timesteps. Hence derive an algorithm that will find the minimum number of steps to get from state j to state i .
- 2.2 For an $N \times N$ symmetric adjacency matrix \mathbf{A} , describe an algorithm to find the connected components. You may wish to examine `connectedComponents.m`.
- 2.3 Show that for a connected graph that is singly connected, the number of edges E must be equal to the number of nodes minus 1, $E = V - 1$. Give an example graph with $E = V - 1$ that is not singly connected. Hence the condition $E = V - 1$ is a necessary but not sufficient condition for a graph to be singly connected.
- 2.4 Describe a procedure to determine if a graph is singly connected.
- 2.5 Describe a procedure to determine all the ancestors of a set of nodes in a DAG.
- 2.6 `WikiAdjSmall.mat` contains a random selection of 1000 Wiki authors, with a link between two authors if they 'know' each other (see snap.stanford.edu/data/wiki-Vote.html). Plot a histogram of the

separation (the length of the path between two users on the graph corresponding to the adjacency matrix) between all users based on separations from 1 to 20. That is the bin $n(s)$ in the histogram contains the number of pairs with separation s .

- 2.7** The file `cliques.mat` contains a list of 100 non-maximal cliques defined on a graph of 10 nodes. Your task is to return a set of unique maximal cliques, eliminating cliques that are wholly contained within another. Once you have found a clique, you can represent it in binary form as, for example

(1110011110)

which says that this clique contains variables 1, 2, 3, 6, 7, 8, 9, reading from left to right. Converting this binary representation to decimal (with the rightmost bit being the units and the leftmost 2^9) this corresponds to the number 926. Using this decimal representation, write the list of unique cliques, ordered from lowest decimal representation to highest. Describe fully the stages of the algorithm you use to find these unique cliques. Hint: you may find examining `uniquepots.m` useful.

- 2.8** Explain how to construct a graph with N nodes, where N is even, that contains at least $(N/2)^2$ maximal cliques.
- 2.9** Let N be divisible by 3. Construct a graph with N nodes by partitioning the nodes into $N/3$ subsets, each subset containing 3 nodes. Then connect all nodes, provided they are not in the same subset. Show that such a graph has $3^{N/3}$ maximal cliques. This shows that a graph can have an exponentially large number of maximal cliques [217].

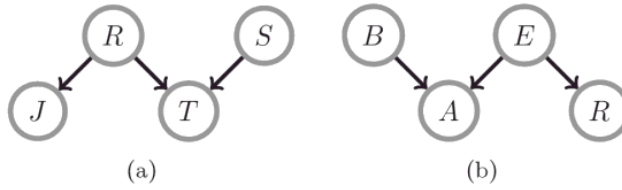


Figure 3.1 (a) Belief network structure for the ‘wet grass’ example. Each node in the graph represents a variable in the joint distribution, and the variables which feed in (the parents) to another variable represent which variables are to the right of the conditioning bar. (b) Belief network for the Burglar model.

Similarly, we assume that Jack’s grass is wet is influenced only directly by whether or not it has been raining, and write

$$p(J|R, S) = p(J|R). \quad (3.1.5)$$

Furthermore, we assume the rain is not directly influenced by the sprinkler,

$$p(R|S) = p(R) \quad (3.1.6)$$

which means that our model equation now becomes

$$p(T, J, R, S) = p(T|R, S)p(J|R)p(R)p(S). \quad (3.1.7)$$

We can represent these conditional independencies graphically, as in Fig. 3.1(a). This reduces the number of values that we need to specify to $4 + 2 + 1 + 1 = 8$, a saving over the previous 15 values in the case where no conditional independencies had been assumed.

To complete the model, we need to numerically specify the values of each Conditional Probability Table (CPT). Let the prior probabilities for R and S be $p(R = 1) = 0.2$ and $p(S = 1) = 0.1$. We set the remaining probabilities to $p(J = 1|R = 1) = 1$, $p(J = 1|R = 0) = 0.2$ (sometimes Jack’s grass is wet due to unknown effects, other than rain), $p(T = 1|R = 1, S = 0) = 1$, $p(T = 1|R = 1, S = 1) = 1$, $p(T = 1|R = 0, S = 1) = 0.9$ (there’s a small chance that even though the sprinkler was left on, it didn’t wet the grass noticeably), $p(T = 1|R = 0, S = 0) = 0$.

Inference

Now that we’ve made a model of an environment, we can perform inference. Let’s calculate the probability that the sprinkler was on overnight, given that Tracey’s grass is wet: $p(S = 1|T = 1)$. To do this we use:

$$p(S = 1|T = 1) = \frac{p(S = 1, T = 1)}{p(T = 1)} = \frac{\sum_{J,R} p(T = 1, J, R, S = 1)}{\sum_{J,R,S} p(T = 1, J, R, S)} \quad (3.1.8)$$

$$= \frac{\sum_{J,R} p(J|R)p(T = 1|R, S = 1)p(R)p(S = 1)}{\sum_{J,R,S} p(J|R)p(T = 1|R, S)p(R)p(S)} \quad (3.1.9)$$

$$= \frac{\sum_R p(T = 1|R, S = 1)p(R)p(S = 1)}{\sum_{R,S} p(T = 1|R, S)p(R)p(S)} \quad (3.1.10)$$

$$= \frac{0.9 \times 0.8 \times 0.1 + 1 \times 0.2 \times 0.1}{0.9 \times 0.8 \times 0.1 + 1 \times 0.2 \times 0.1 + 0 \times 0.8 \times 0.9 + 1 \times 0.2 \times 0.9} = 0.3382 \quad (3.1.11)$$

so the (posterior) belief that the sprinkler is on increases above the prior probability 0.1, due to the evidence that the grass is wet. Note that in Equation (3.1.9), the summation over J in the numerator is unity since, for any function $f(R)$, a summation of the form $\sum_J p(J|R)f(R)$ equals $f(R)$. This

follows from the definition that a distribution $p(J|R)$ must sum to one, and the fact that $f(R)$ does not depend on J . A similar effect occurs for the summation over J in the denominator.

Let us now calculate the probability that Tracey's sprinkler was on overnight, given that her grass is wet and that Jack's grass is also wet, $p(S = 1|T = 1, J = 1)$. We use conditional probability again:

$$p(S = 1|T = 1, J = 1) = \frac{p(S = 1, T = 1, J = 1)}{p(T = 1, J = 1)} \quad (3.1.12)$$

$$= \frac{\sum_R p(T = 1, J = 1, R, S = 1)}{\sum_{R,S} p(T = 1, J = 1, R, S)} \quad (3.1.13)$$

$$= \frac{\sum_R p(J = 1|R)p(T = 1|R, S = 1)p(R)p(S = 1)}{\sum_{R,S} p(J = 1|R)p(T = 1|R, S)p(R)p(S)} \quad (3.1.14)$$

$$= \frac{0.0344}{0.2144} = 0.1604. \quad (3.1.15)$$

The probability that the sprinkler is on, given the extra evidence that Jack's grass is wet, is *lower* than the probability that the grass is wet given only that Tracey's grass is wet. This occurs since the fact that Jack's grass is also wet increases the chance that the rain has played a role in making Tracey's grass wet.

Naturally, we don't wish to carry out such inference calculations by hand all the time. General purpose algorithms exist for this, such as the junction tree algorithm, Chapter 6.

Example 3.1 Was it the burglar?

Here's another example using binary variables, adapted from [236]. Sally comes home to find that the burglar alarm is sounding ($A = 1$). Has she been burgled ($B = 1$), or was the alarm triggered by an earthquake ($E = 1$)? She turns the car radio on for news of earthquakes and finds that the radio broadcasts an earthquake alert ($R = 1$).

Using Bayes' rule, we can write, without loss of generality,

$$p(B, E, A, R) = p(A|B, E, R)p(B, E, R). \quad (3.1.16)$$

We can repeat this for $p(B, E, R)$, and continue

$$p(B, E, A, R) = p(A|B, E, R)p(R|B, E)p(E|B)p(B). \quad (3.1.17)$$

However, the alarm is surely not directly influenced by any report on the radio – that is, $p(A|B, E, R) = p(A|B, E)$. Similarly, we can make other conditional independence assumptions such that

$$p(B, E, A, R) = p(A|B, E)p(R|E)p(E)p(B) \quad (3.1.18)$$

as depicted in Fig. 3.1(b).

Specifying conditional probability tables

Alarm = 1	Burglar	Earthquake
0.9999	1	1
0.99	1	0
0.99	0	1
0.0001	0	0

Radio = 1	Earthquake
1	1
0	0

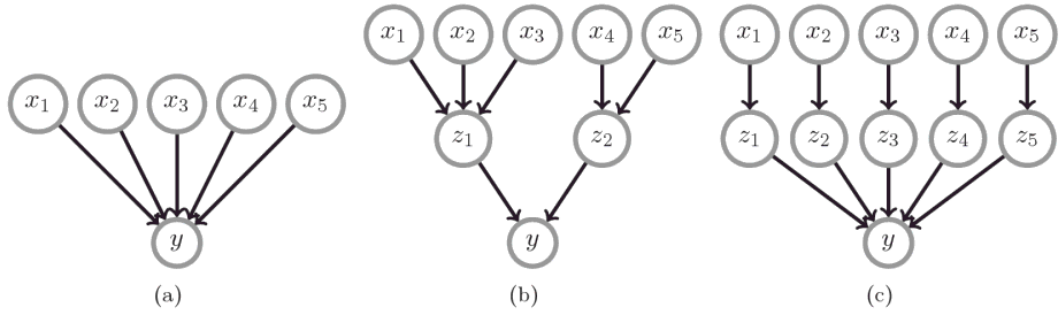


Figure 3.2 (a) If all variables are binary $2^5 = 32$ states are required to specify $p(y|x_1, \dots, x_5)$. (b) Here only 16 states are required. (c) Noisy logic gates.

The remaining tables are $p(B = 1) = 0.01$ and $p(E = 1) = 0.000001$. The tables and graphical structure fully specify the distribution. Now consider what happens as we observe evidence.

Initial evidence: the alarm is sounding

$$p(B = 1|A = 1) = \frac{\sum_{E,R} p(B = 1, E, A = 1, R)}{\sum_{B,E,R} p(B, E, A = 1, R)} \quad (3.1.19)$$

$$= \frac{\sum_{E,R} p(A = 1|B = 1, E)p(B = 1)p(E)p(R|E)}{\sum_{B,E,R} p(A = 1|B, E)p(B)p(E)p(R|E)} \approx 0.99. \quad (3.1.20)$$

Additional evidence: the radio broadcasts an earthquake warning: A similar calculation gives $p(B = 1|A = 1, R = 1) \approx 0.01$. Thus, initially, because the alarm sounds, Sally thinks that she's been burgled. However, this probability drops dramatically when she hears that there has been an earthquake. That is, the earthquake 'explains away' to an extent the fact that the alarm is ringing. See `demoBurglar.m`.

Remark 3.1 (Causal intuitions) Belief networks as we've defined them are ways to express independence statements. Nevertheless, in expressing these independencies it can be useful (though also potentially misleading) to think of 'what causes what'. In Example 3.1 we chose the ordering of the variables as (reading from right to left) B, E, R, A in Equation (3.1.17) since B and E can be considered root 'causes' and A and R as 'effects'.

3.1.2 Reducing the burden of specification

Consider a discrete variable y with many discrete parental variables x_1, \dots, x_n , Fig. 3.2(a). Formally, the structure of the graph implies nothing about the form of the parameterisation of the table $p(y|x_1, \dots, x_n)$. If each parent x_i has $\dim(x_i)$ states, and there is no constraint on the table, then the table $p(y|x_1, \dots, x_n)$ contains $(\dim(y) - 1) \prod_i \dim(x_i)$ entries. If stored explicitly for each state, this would require potentially huge storage. An alternative is to constrain the table to have a simpler parametric form. For example, one might write a decomposition in which only a limited number of parental interactions are required (this is called *divorcing parents* in [161]). For example, in Fig. 3.2(b), we have

$$p(y|x_1, \dots, x_5) = \sum_{z_1, z_2} p(y|z_1, z_2)p(z_1|x_1, x_2, x_3)p(z_2|x_4, x_5). \quad (3.1.21)$$

Assuming all variables are binary, the number of states requiring specification is $2^3 + 2^2 + 2^2 = 16$, compared to the $2^5 = 32$ states in the unconstrained case.

Logic gates

Another technique to constrain tables uses simple classes of conditional tables. For example, in Fig. 3.2(c), one could use a logical OR gate on binary z_i , say

$$p(y|z_1, \dots, z_5) = \begin{cases} 1 & \text{if at least one of the } z_i \text{ is in state 1} \\ 0 & \text{otherwise.} \end{cases} \quad (3.1.22)$$

We can then make a table $p(y|x_1, \dots, x_5)$ by including the additional terms $p(z_i = 1|x_i)$. When each x_i is binary there are in total only $2 + 2 + 2 + 2 + 2 = 10$ quantities required for specifying $p(y|x)$. In this case, Fig. 3.2(c) can be used to represent any *noisy logic gate*, such as the *noisy OR* or *noisy AND*, where the number of parameters required to specify the noisy gate is linear in the number of parents.

The noisy-OR is particularly common in disease–symptom networks in which many diseases x can give rise to the same symptom y – provided that at least one of the diseases is present, the probability that the symptom will be present is high.

3.2 Uncertain and unreliable evidence

In the following we make a distinction between evidence that is uncertain, and evidence that is unreliable.

3.2.1 Uncertain evidence

In soft or *uncertain evidence*, the evidence variable is in more than one state, with the strength of our belief about each state being given by probabilities. For example, if x has the states $\text{dom}(x) = \{\text{red, blue, green}\}$ the vector $(0.6, 0.1, 0.3)$ represents the belief in the respective states. In contrast, for *hard-evidence* we are certain that a variable is in a particular state. In this case, all the probability mass is in one of the vector components, for example $(0, 0, 1)$.

Performing inference with soft-evidence is straightforward and can be achieved using Bayes' rule. For example, for a model $p(x, y)$, consider that we have some soft-evidence \tilde{y} about the variable y , and wish to know what effect this has on the variable x – that is we wish to compute $p(x|\tilde{y})$. From Bayes' rule, and the assumption $p(x|y, \tilde{y}) = p(x|y)$, we have

$$p(x|\tilde{y}) = \sum_y p(x, y|\tilde{y}) = \sum_y p(x|y, \tilde{y})p(y|\tilde{y}) = \sum_y p(x|y)p(y|\tilde{y}) \quad (3.2.1)$$

where $p(y = i|\tilde{y})$ represents the probability that y is in state i under the soft-evidence. This is a generalisation of hard-evidence in which the vector $p(y|\tilde{y})$ has all zero component values, except for all but a single component. This procedure in which we first define the model conditioned on the evidence, and then average over the distribution of the evidence is also known as Jeffrey's rule.

In the BN we use a dashed circle to represent that a variable is in a soft-evidence state.



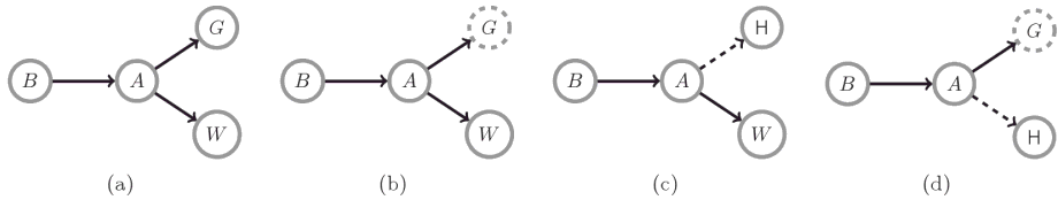


Figure 3.3 (a) Mr Holmes' burglary worries as given in [236]: (B)urglar, (A)larm, (W)atson, Mrs (G)ibbon. (b) Mrs Gibbon's uncertain evidence represented by a dashed circle. (c) Virtual evidence or the replacement of unreliable evidence can be represented by a dashed line. (d) Mrs Gibbon is uncertain in her evidence. Holmes also replaces the unreliable Watson with his own interpretation.

Example 3.2 Soft-evidence

Revisiting the burglar scenario, Example 3.1, imagine that we are only 70 per cent sure we heard the burglar alarm sounding. For this binary variable case we represent this soft-evidence for the states $(1, 0)$ as $\tilde{A} = (0.7, 0.3)$. What is the probability of a burglary under this soft-evidence?

$$p(B = 1|\tilde{A}) = \sum_A p(B = 1|A)p(A|\tilde{A}) = p(B = 1|A = 1) \times 0.7 + p(B = 1|A = 0) \times 0.3. \quad (3.2.2)$$

The probabilities $p(B = 1|A = 1) \approx 0.99$ and $p(B = 1|A = 0) \approx 0.0001$ are calculated using Bayes' rule as before to give

$$p(B = 1|\tilde{A}) \approx 0.6930. \quad (3.2.3)$$

This is lower than 0.99, the probability of having been burgled when we are sure we heard the alarm.

Holmes, Watson and Mrs Gibbon

An entertaining example of uncertain evidence is given by Pearl [236] that we adapt for our purposes here. The environment contains four variables:

- $B \in \{\text{tr}, \text{fa}\}$ $B = \text{tr}$ means that Holmes' house has been burgled
- $A \in \{\text{tr}, \text{fa}\}$ $A = \text{tr}$ means that Holmes' house alarm went off
- $W \in \{\text{tr}, \text{fa}\}$ $W = \text{tr}$ means that Watson heard the alarm
- $G \in \{\text{tr}, \text{fa}\}$ $G = \text{tr}$ means that Mrs Gibbon heard the alarm.

The BN below for this scenario is depicted in Fig. 3.3(a)

$$p(B, A, G, W) = p(A|B)p(B)p(W|A)p(G|A). \quad (3.2.4)$$

Watson states that he heard the alarm is sounding. Mrs Gibbon is a little deaf and cannot be sure herself that she heard the alarm, being 80 per cent sure she heard it. This can be dealt with using the soft-evidence technique, Fig. 3.3(b). From Jeffrey's rule, one uses the original model



Figure 3.4 Two BNs for a four-variable distribution. Both graphs (a) and (b) represent the *same* distribution $p(x_1, x_2, x_3, x_4)$. Strictly speaking they represent the same (lack of) independence assumptions – the graphs say nothing about the content of the tables. The extension of this ‘cascade’ to many variables is clear and always results in a directed acyclic graph.

to deleting one of the edges. More formally, this corresponds to an ordering of the variables which, without loss of generality, we may write as x_1, \dots, x_n . Then, from Bayes’ rule, we have

$$p(x_1, \dots, x_n) = p(x_1|x_2, \dots, x_n)p(x_2, \dots, x_n) \quad (3.3.4)$$

$$= p(x_1|x_2, \dots, x_n)p(x_2|x_3, \dots, x_n)p(x_3, \dots, x_n) \quad (3.3.5)$$

$$= p(x_n) \prod_{i=1}^{n-1} p(x_i|x_{i+1}, \dots, x_n). \quad (3.3.6)$$

The representation of any BN is therefore a *directed acyclic graph*.

Every probability distribution can be written as a BN, even though it may correspond to a fully connected ‘cascade’ DAG. The particular role of a BN is that the structure of the DAG corresponds to a set of conditional independence assumptions, namely which ancestral parental variables are sufficient to specify each conditional probability table. Note that this does not mean that non-parental variables have no influence. For example, for distribution $p(x_1|x_2)p(x_2|x_3)p(x_3)$ with DAG $x_1 \leftarrow x_2 \leftarrow x_3$, this does not imply $p(x_2|x_1, x_3) = p(x_2|x_3)$. The DAG specifies conditional independence statements of variables on their ancestors – namely which ancestors are direct ‘causes’ for the variable. The ‘effects’, given by the descendants of the variable, will generally be dependent on the variable. See also Remark 3.3.

Remark 3.3 (Dependencies and the Markov blanket) Consider a distribution on a set of variables \mathcal{X} . For a variable $x_i \in \mathcal{X}$ and corresponding belief network represented by a DAG G , let $MB(x_i)$ be the variables in the Markov blanket of x_i . Then for any other variable y that is also not in the Markov blanket of x_i ($y \in \mathcal{X} \setminus \{x_i \cup MB(x_i)\}$), then $x_i \perp\!\!\!\perp y | MB(x_i)$. That is, the Markov blanket of x_i carries all information about x_i . As an example, for Fig. 3.2(b), $MB(z_1) = \{x_1, x_2, x_3, y, z_2\}$ and $z_1 \perp\!\!\!\perp x_4 | MB(z_1)$.

The DAG corresponds to a statement of conditional independencies in the model. To complete the specification of the BN we need to define all elements of the conditional probability tables $p(x_i|pa(x_i))$. Once the graphical structure is defined, the entries of the Conditional Probability Tables (CPTs) $p(x_i|pa(x_i))$ can be expressed. For every possible state of the parental variables $pa(x_i)$, a value for each of the states of x_i needs to be specified (except one, since this is determined by normalisation). For a large number of parents, writing out a table of values is intractable, and the tables are usually parameterised in a low-dimensional manner. This will be a central topic of our discussion on the application of BNs in machine learning.

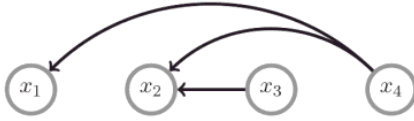


Figure 3.5 $p(x_1, x_2, x_3, x_4) = p(x_1|x_4)p(x_2|x_3, x_4)p(x_3)p(x_4)$.

3.3.1 Conditional independence

Whilst a BN corresponds to a set of conditional independence assumptions, it is not always immediately clear from the DAG whether a set of variables is conditionally independent of a set of other variables (see Definition 1.7). For example, in Fig. 3.5 are x_1 and x_2 independent, given the state of x_4 ? The answer is yes, since we have

$$p(x_1, x_2|x_4) = \frac{1}{p(x_4)} \sum_{x_3} p(x_1, x_2, x_3, x_4) = \frac{1}{p(x_4)} \sum_{x_3} p(x_1|x_4)p(x_2|x_3, x_4)p(x_3)p(x_4) \quad (3.3.7)$$

$$= p(x_1|x_4) \sum_{x_3} p(x_2|x_3, x_4)p(x_3). \quad (3.3.8)$$

Now

$$p(x_2|x_4) = \frac{1}{p(x_4)} \sum_{x_1, x_3} p(x_1, x_2, x_3, x_4) = \frac{1}{p(x_4)} \sum_{x_1, x_3} p(x_1|x_4)p(x_2|x_3, x_4)p(x_3)p(x_4) \quad (3.3.9)$$

$$= \sum_{x_3} p(x_2|x_3, x_4)p(x_3). \quad (3.3.10)$$

Combining the two results above we have

$$p(x_1, x_2|x_4) = p(x_1|x_4)p(x_2|x_4) \quad (3.3.11)$$

so that x_1 and x_2 are indeed independent conditioned on x_4 .

We would like to have a general algorithm that will allow us to avoid doing such tedious manipulations by reading the result directly from the graph. To help develop intuition towards constructing such an algorithm, consider the three-variable distribution $p(x_1, x_2, x_3)$. We may write this in any of the six ways

$$p(x_1, x_2, x_3) = p(x_{i_1}|x_{i_2}, x_{i_3})p(x_{i_2}|x_{i_3})p(x_{i_3}) \quad (3.3.12)$$

where (i_1, i_2, i_3) is any of the six permutations of $(1, 2, 3)$. Whilst each factorisation produces a different DAG, all represent the same distribution, namely one that makes no independence statements. If the DAGs are of the cascade form, no independence assumptions have been made. The minimal independence assumptions then correspond to dropping a single link in the cascade graph. This gives rise to the four DAGs in Fig. 3.6. Are any of these graphs equivalent, in the sense that they represent the same distribution? Applying Bayes' rule gives:

$$\underbrace{p(x_2|x_3)p(x_3|x_1)p(x_1)}_{\text{graph}(c)} = p(x_2, x_3)p(x_3, x_1)/p(x_3) = p(x_1|x_3)p(x_2, x_3) \quad (3.3.13)$$

$$= \underbrace{p(x_1|x_3)p(x_3|x_2)p(x_2)}_{\text{graph}(d)} = \underbrace{p(x_1|x_3)p(x_2|x_3)p(x_3)}_{\text{graph}(b)} \quad (3.3.14)$$

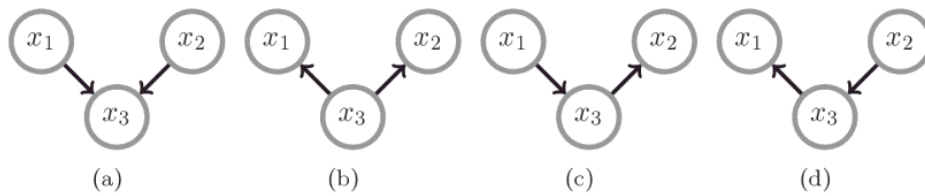


Figure 3.6 By dropping say the connection between variables x_1 and x_2 , we reduce the six possible BN graphs amongst three variables to four. (The six fully connected ‘cascade’ graphs correspond to (a) with $x_1 \rightarrow x_2$, (a) with $x_2 \rightarrow x_1$, (b) with $x_1 \rightarrow x_2$, (b) with $x_2 \rightarrow x_1$, (c) with $x_1 \rightarrow x_3$ and (d) with $x_2 \rightarrow x_1$. Any other graphs would be cyclic and therefore not distributions.)

so that DAGs (b), (c) and (d) represent the same conditional independence (CI) assumptions – given the state of variable x_3 , variables x_1 and x_2 are independent, $x_1 \perp\!\!\!\perp x_2 \mid x_3$.

However, graph (a) represents something fundamentally different, namely: $p(x_1, x_2) = p(x_1)p(x_2)$. There is no way to transform the distribution $p(x_3|x_1, x_2)p(x_1)p(x_2)$ into any of the others.

Remark 3.4 (Graphical dependence) Belief network (graphs) are good for encoding conditional independence but are not well suited for encoding dependence. For example, consider the graph $a \rightarrow b$. This may appear to encode the relation that a and b are dependent. However, a specific numerical instance of a belief network distribution could be such that $p(b|a) = p(b)$, for which $a \perp\!\!\!\perp b$. The lesson is that even when the DAG appears to show ‘graphical’ dependence, there can be instances of the distributions for which dependence does not follow. The same caveat holds for Markov networks, Section 4.2. We discuss this issue in more depth in Section 3.3.5.

3.3.2 The impact of collisions

Definition 3.2 Given a path \mathcal{P} , a collider is a node c on \mathcal{P} with neighbours a and b on \mathcal{P} such that $a \rightarrow c \leftarrow b$. Note that a collider is path specific, see Fig. 3.8.

In a general BN, how can we check if $x \perp\!\!\!\perp y \mid z$? In Fig. 3.7(a), x and y are independent when conditioned on z since

$$p(x, y|z) = p(x|z)p(y|z). \quad (3.3.15)$$

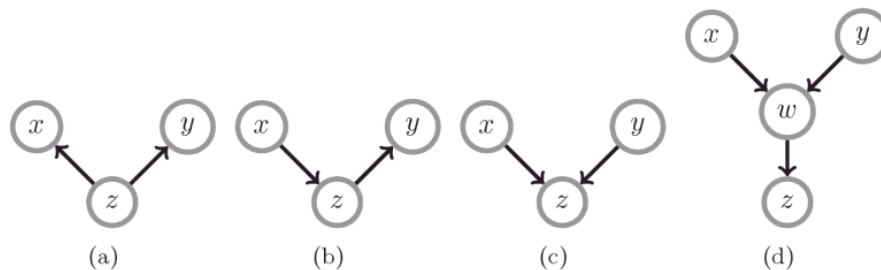
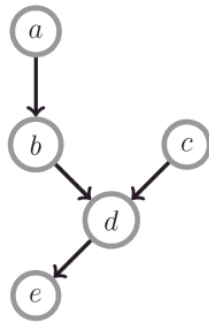
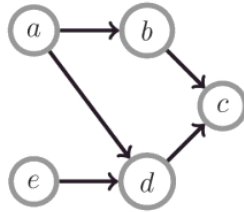


Figure 3.7 In graphs (a) and (b), variable z is not a collider. (c) Variable z is a collider. Graphs (a) and (b) represent conditional independence $x \perp\!\!\!\perp y \mid z$. In graphs (c) and (d), x and y are ‘graphically’ conditionally dependent given variable z .



(a) The variable d is a collider along the path $a - b - d - c$, but not along the path $a - b - d - e$. Is $a \perp\!\!\!\perp e | b$? Variables a and e are *not* d -connected since there are no colliders on the only path between a and e , and since there is a non-collider b which is in the conditioning set. Hence a and e are d -separated by b , $\Rightarrow a \perp\!\!\!\perp e | b$.



(b) The variable d is a collider along the path $a - d - e$, but not along the path $a - b - c - d - e$. Is $a \perp\!\!\!\perp e | c$? There are two paths between a and e , namely $a - d - e$ and $a - b - c - d - e$. The path $a - d - e$ is not blocked since although d is a collider on this path and d is not in the conditioning set, we have a descendant of the collider d in the conditioning set, namely c . For the path $a - b - c - d - e$, the node c is a collider on this path and c is in the conditioning set. For this path d is not a collider. Hence this path is not blocked and a and e are (graphically) dependent given c .

Figure 3.8 Collider examples for d -separation and d -connection.

Similarly, for Fig. 3.7(b), x and y are independent conditioned on z .

$$p(x, y|z) \propto p(z|x)p(x)p(y|z) \quad (3.3.16)$$

which is a function of x multiplied by a function of y . In Fig. 3.7(c), however, x and y are graphically dependent since $p(x, y|z) \propto p(z|x, y)p(x)p(y)$; in this situation, variable z is called a *collider* – the arrows of its neighbours are pointing towards it. What about Fig. 3.7(d)? In (d), when we condition on z , x and y will be graphically dependent, since

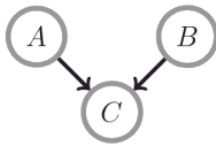
$$p(x, y|z) = \frac{p(x, y, z)}{p(z)} = \frac{1}{p(z)} \sum_w p(z|w)p(w|x, y)p(x)p(y) \neq p(x|z)p(y|z) \quad (3.3.17)$$

– intuitively, variable w becomes dependent on the value of z , and since x and y are conditionally dependent on w , they are also conditionally dependent on z .

If there is a non-collider z which is conditioned along the path between x and y (as in Fig. 3.7(a,b)), then this path cannot induce dependence between x and y . Similarly, if there is a path between x and y which contains a collider, provided that this collider is not in the conditioning set (and neither are any of its descendants) then this path does not make x and y dependent. If there is a path between x and y which contains no colliders and no conditioning variables, then this path ‘ d -connects’ x and y . Note that a collider is defined *relative to a path*. In Fig. 3.8(a), the variable d is a collider along the path $a - b - d - c$, but not along the path $a - b - d - e$ (since, relative to this path, the two arrows do not point inwards to d).

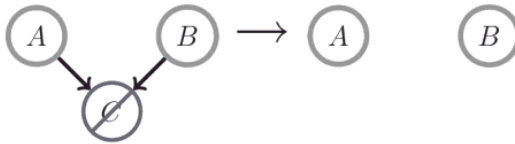
Consider the BN: $A \rightarrow B \leftarrow C$. Here A and C are (unconditionally) independent. However, conditioning of B makes them ‘graphically’ dependent. Intuitively, whilst we believe the root causes are independent, given the value of the observation, this tells us something about the state of *both* the causes, coupling them and making them (generally) dependent. In Definition 3.3 below we describe the effect that conditioning/marginalisation has on the graph of the remaining variables.

Definition 3.3 Some properties of belief networks It is useful to understand what effect conditioning or marginalising a variable has on a belief network. We state here how these operations effect the remaining variables in the graph and use this intuition to develop a more complete description in Section 3.3.4.

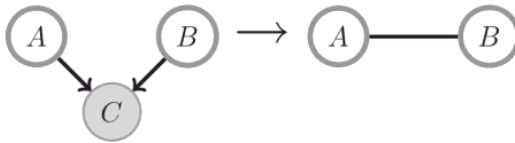


$$p(A, B, C) = p(C|A, B)p(A)p(B) \quad (3.3.18)$$

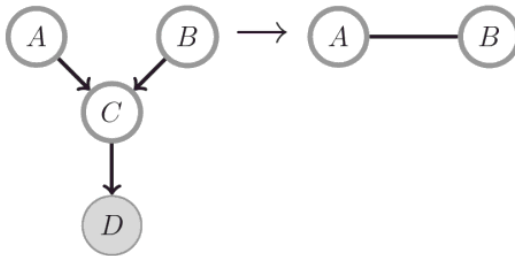
From a 'causal' perspective, this models the 'causes' A and B as a priori independent, both determining the effect C .



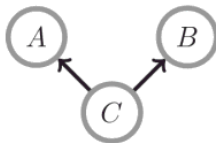
Marginalising over C makes A and B independent. A and B are (unconditionally) independent: $p(A, B) = p(A)p(B)$. In the absence of any information about the effect C , we retain this belief.



Conditioning on C makes A and B (graphically) dependent – in general $p(A, B|C) \neq p(A|C)p(B|C)$. Although the causes are a priori independent, knowing the effect C in general tells us something about how the causes colluded to bring about the effect observed.

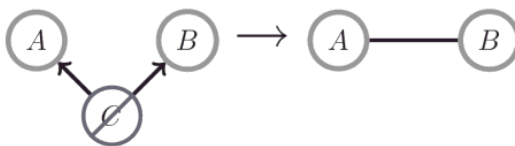


Conditioning on D , a descendent of a collider C , makes A and B (graphically) dependent – in general $p(A, B|D) \neq p(A|D)p(B|D)$.



$$p(A, B, C) = p(A|C)p(B|C)p(C) \quad (3.3.19)$$

Here there is a 'cause' C and independent 'effects' A and B .



Marginalising over C makes A and B (graphically) dependent. In general, $p(A, B) \neq p(A)p(B)$. Although we don't know the 'cause', the 'effects' will nevertheless be dependent.

Remark 3.5 (Bayes ball) The Bayes ball algorithm [258] provides a linear time complexity algorithm which given a set of nodes \mathcal{X} and \mathcal{Z} determines the set of nodes \mathcal{Y} such that $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{Z}$. \mathcal{Y} is called the set of irrelevant nodes for \mathcal{X} given \mathcal{Z} .

3.3.5 Graphical and distributional in/dependence

We have shown

\mathcal{X} and \mathcal{Y} d-separated by $\mathcal{Z} \Rightarrow \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{Z}$ in *all* distributions consistent with the belief network structure.

In other words, if one takes any instance of a distribution P which factorises according to the belief network structure and then writes down a list \mathcal{L}_P of all the conditional independence statements that can be obtained from P , if \mathcal{X} and \mathcal{Y} are d-separated by \mathcal{Z} then this list must contain the statement $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{Z}$. Note that the list \mathcal{L}_P could contain more statements than those obtained from the graph. For example for the belief network graph

$$p(a, b, c) = p(c|a, b)p(a)p(b) \quad (3.3.20)$$

which is representable by the DAG $a \rightarrow c \leftarrow b$, then $a \perp\!\!\!\perp b$ is the only graphical independence statement we can make. Consider a distribution consistent with Equation (3.3.20), for example, on binary variables $\text{dom}(a) = \text{dom}(b) = \text{dom}(c) = \{0, 1\}$

$$p_{[1]}(c = 1|a, b) = (a - b)^2, \quad p_{[1]}(a = 1) = 0.3, \quad p_{[1]}(b = 1) = 0.4 \quad (3.3.21)$$

then numerically we must have $a \perp\!\!\!\perp b$ for this distribution $p_{[1]}$. Indeed the list $\mathcal{L}_{[1]}$ contains only the statement $a \perp\!\!\!\perp b$. On the other hand, we can also consider the distribution

$$p_{[2]}(c = 1|a, b) = 0.5, \quad p_{[2]}(a = 1) = 0.3, \quad p_{[2]}(b = 1) = 0.4 \quad (3.3.22)$$

from which $\mathcal{L}_{[2]} = \{a \perp\!\!\!\perp b, a \perp\!\!\!\perp c, b \perp\!\!\!\perp c\}$. In this case $\mathcal{L}_{[2]}$ contains more statements than $a \perp\!\!\!\perp b$.

An interesting question is whether or not d-connection similarly implies dependence? That is, do *all* distributions P consistent with the belief network possess the dependencies implied by the graph? If we consider the belief network structure Equation (3.3.20) above, a and b are d-connected by c , so that graphically a and b are dependent, conditioned on c . For the specific instance $p_{[1]}$ we have numerically $a \perp\!\!\!\perp b \mid c$ so that the list of dependence statements for $p_{[1]}$ contains the graphical dependence statement. Now consider $p_{[2]}$. The list of dependence statements for $p_{[2]}$ is empty. Hence the graphical dependence statements are not necessarily found in all distributions consistent with the belief network. Hence

\mathcal{X} and \mathcal{Y} d-connected by $\mathcal{Z} \not\Rightarrow \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{Z}$ in *all* distributions consistent with the belief network structure.

See also Exercise 3.17. This shows that belief networks are powerful in ensuring that distributions necessarily obey the independence assumptions we expect from the graph. However, belief networks are not suitable for ensuring that distributions obey desired dependency statements.

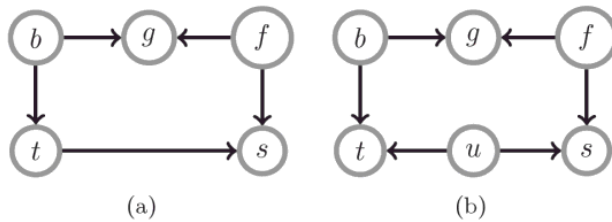


Figure 3.10 (a) t and f are d-connected by g . (b) b and f are d-separated by u .

Example 3.3

Consider the graph in Fig. 3.10(a).

1. Are the variables t and f unconditionally independent, i.e. $t \perp\!\!\!\perp f | \emptyset$? Here there are two colliders, namely g and s – however, these are not in the conditioning set (which is empty), and hence t and f are d-separated and therefore unconditionally independent.
2. What about $t \perp\!\!\!\perp f | g$? There is a path between t and f for which all colliders are in the conditioning set. Hence t and f are d-connected by g , and therefore t and f are graphically dependent conditioned on g .

Example 3.4

Is $\{b, f\} \perp\!\!\!\perp u | \emptyset$ in Fig. 3.10(b)? Since the conditioning set is empty and every path from either b or f to u contains a collider, ‘ b and f ’ are unconditionally independent of u .

3.3.6 Markov equivalence in belief networks

We have invested a lot of effort in learning how to read conditional independence relations from a DAG. Happily, we can determine whether two DAGs represent the same set of conditional independence statements (even when we don’t know what they are) by using a relatively simple rule.

Definition 3.5 Markov equivalence Two graphs are Markov equivalent if they both represent the same set of conditional independence statements. This definition holds for both directed and undirected graphs.

Example 3.5

Consider the belief network with edges $A \rightarrow C \leftarrow B$, from which the set of conditional statements is $A \perp\!\!\!\perp B | \emptyset$. For another belief network with edges $A \rightarrow C \leftarrow B$ and $A \rightarrow B$, the set of conditional independence statements is empty. In this case, the two belief networks are not Markov equivalent.

Procedure 3.1 (Determining Markov equivalence) Define an *immorality* in a DAG as a configuration of three nodes, A, B, C such that C is a child of both A and B , with A and B not directly connected. Define the *skeleton* of a graph by removing the directions on the arrows. Two DAGs represent the

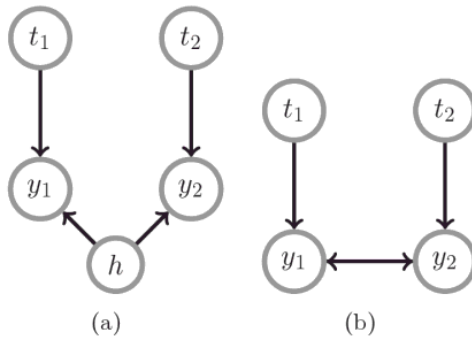


Figure 3.11 (a) Two treatments t_1 , t_2 and corresponding outcomes y_1 , y_2 . The health of a patient is represented by h . This DAG embodies the conditional independence statements $t_1 \perp\!\!\!\perp t_2$, $y_2 \perp\!\!\!\perp \emptyset$, $t_2 \perp\!\!\!\perp t_1$, $y_1 \perp\!\!\!\perp \emptyset$, namely that the treatments have no effect on each other. (b) One could represent the effect of marginalising over h using a bi-directional edge.

same set of independence assumptions (they are *Markov equivalent*) if and only if they have the same skeleton and the same set of immoralities [78].

Using Procedure 3.1 we see that in Fig. 3.6, BNs (b,c,d) have the same skeleton with no immoralities and are therefore equivalent. However BN (a) has an immorality and is therefore not equivalent to BNs (b,c,d).

3.3.7 Belief networks have limited expressibility

Belief networks fit well with our intuitive notion of modelling ‘causal’ independencies. However, formally speaking they cannot necessarily graphically represent all the independence properties of a given distribution.

Consider the DAG in Fig. 3.11(a) (from [249]). This DAG could be used to represent two successive experiments where t_1 and t_2 are two treatments and y_1 and y_2 represent two outcomes of interest; h is the underlying health status of the patient; the first treatment has no effect on the second outcome hence there is no edge from y_1 to y_2 . Now consider the implied independencies in the marginal distribution $p(t_1, t_2, y_1, y_2)$, obtained by marginalising the full distribution over h . There is no DAG containing only the vertices t_1, y_1, t_2, y_2 which represents the independence relations and does not also imply some other independence relation that is not implied by Fig. 3.11(a). Consequently, any DAG on vertices t_1, y_1, t_2, y_2 alone will either fail to represent an independence relation of $p(t_1, t_2, y_1, y_2)$, or will impose some additional independence restriction that is not implied by the DAG. In the above example

$$p(t_1, t_2, y_1, y_2) = p(t_1)p(t_2) \sum_h p(y_1|t_1, h)p(y_2|t_2, h)p(h) \quad (3.3.23)$$

cannot in general be expressed as a product of functions defined on a limited set of the variables. However, it is the case that the conditional independence conditions $t_1 \perp\!\!\!\perp (t_2, y_2)$, $t_2 \perp\!\!\!\perp (t_1, y_1)$ hold in $p(t_1, t_2, y_1, y_2)$ – they are there, encoded in the form of the conditional probability tables. It is just that we cannot ‘see’ this independence since it is not present in the structure of the marginalised graph (though one can naturally infer this in the larger graph $p(t_1, t_2, y_1, y_2, h)$). For example, for the BN with link from y_2 to y_1 , we have $t_1 \perp\!\!\!\perp t_2|y_2$, which is not true for the distribution in (3.3.23). Similarly, for the BN with link from y_1 to y_2 , the implied statement $t_1 \perp\!\!\!\perp t_2|y_1$ is also not true for (3.3.23).

This example demonstrates that BNs cannot express all the conditional independence statements that could be made on that set of variables (the set of conditional independence statements can be

increased by considering additional variables however). This situation is rather general in the sense that any graphical model has limited expressibility in terms of independence statements [281]. It is worth bearing in mind that BNs may not always be the most appropriate framework to express one's independence assumptions and intuitions.

A natural consideration is to use a bi-directional arrow when a variable is marginalised. For Fig. 3.11(a), one could depict the marginal distribution using a bi-directional edge, Fig. 3.11(b). For a discussion of extensions of BNs using bi-directional edges see [249].

3.4 Causality

Causality is a contentious topic and the purpose of this section is to make the reader aware of some pitfalls that can occur and which may give rise to erroneous inferences. The reader is referred to [237] and [78] for further details.

The word 'causal' is contentious particularly in cases where the model of the data contains no explicit temporal information, so that formally only correlations or dependencies can be inferred. For a distribution $p(a, b)$, we could write this as either (i) $p(a|b)p(b)$ or (ii) $p(b|a)p(a)$. In (i) we might think that b 'causes' a , and in (ii) a 'causes' b . Clearly, this is not very meaningful since they both represent exactly the same distribution, see Fig. 3.12. Formally BNs only make independence statements, not causal ones. Nevertheless, in constructing BNs, it can be helpful to think about dependencies in terms of causation since our intuitive understanding is usually framed in how one variable 'influences' another. First we discuss a classic conundrum that highlights potential pitfalls that can arise.

3.4.1 Simpson's paradox

Simpson's 'paradox' is a cautionary tale in causal reasoning in BNs. Consider a medical trial in which patient treatment and outcome are recovered. Two trials were conducted, one with 40 females and one with 40 males. The data is summarised in Table 3.1. The question is: Does the drug cause increased recovery? According to the table for males, the answer is no, since more males recovered when they were not given the drug than when they were. Similarly, more females recovered when not given the drug than recovered when given the drug. The conclusion appears that the drug cannot be beneficial since it aids neither subpopulation.

However, ignoring the gender information, and collating both the male and female data into one combined table, we find that more people recovered when given the drug than when not. Hence, even though the drug doesn't seem to work for either males or females, it does seem to work overall! Should we therefore recommend the drug or not?



Figure 3.12 Both (a) and (b) represent the same distribution $p(a, b) = p(a|b)p(b) = p(b|a)p(a)$. (c) The graph represents $p(\text{rain}, \text{grasswet}) = p(\text{grasswet}|\text{rain})p(\text{rain})$. (d) We could equally have written $p(\text{rain}|\text{grasswet})p(\text{grasswet})$, although this appears to be causally non-sensical.

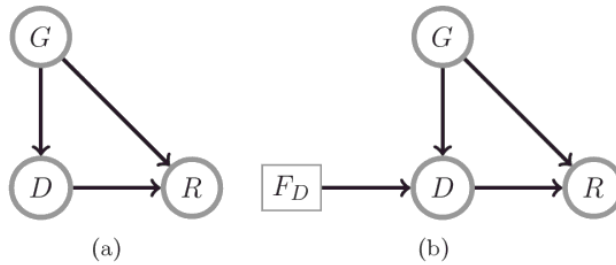


Figure 3.13 (a) A DAG for the relation between Gender (G), Drug (D) and Recovery (R), see Table 3.1. (b) Influence diagram. No decision variable is required for G since G has no parents.

Resolution of the paradox

The ‘paradox’ occurs because we are asking a *causal* (interventional) question – If we give someone the drug, what happens? – but we are performing an observational calculation. Pearl [237] would remind us that there is a difference between ‘given that we see’ (observational evidence) and ‘given that we do’ (interventional evidence). We want to model a causal experiment in which we first intervene, setting the drug state, and then observe what effect this has on recovery.

A model of the Gender, Drug and Recovery data (which makes no conditional independence assumptions) is, Fig. 3.13(a),

$$p(G, D, R) = p(R|G, D)p(D|G)p(G). \quad (3.4.1)$$

In a causal interpretation, however, if we intervene and give the drug, then the term $p(D|G)$ in Equation (3.4.1) should play no role in the experiment – we decide to give the drug or not independent of gender. The term $p(D|G)$ therefore needs to be replaced by a term that reflects the set-up of the experiment. We use the idea of an atomic intervention, in which a single variable is set in a particular state. In our atomic causal intervention, where we set D , we deal with the modified distribution

$$\tilde{p}(G, R|D) = p(R|G, D)p(G) \quad (3.4.2)$$

where the terms on the right-hand side of this equation are taken from the original BN of the data. To denote an intervention we use $\|\|$:

$$p(R\|\|G, D) \equiv \tilde{p}(R|G, D) = \frac{p(R|G, D)p(G)}{\sum_R p(R|G, D)p(G)} = p(R|G, D). \quad (3.4.3)$$

Table 3.1 Table for Simpson’s Paradox (from [237])

	Recovered	Not recovered	Rec. rate
Males			
Given drug	18	12	60%
Not given drug	7	3	70%
Females			
Given drug	2	8	20%
Not given drug	9	21	30%
Combined			
Given drug	20	20	50%
Not given drug	16	24	40%

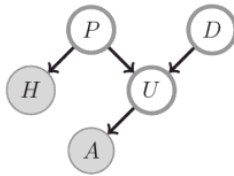


Figure 3.14 Party animal. Here all variables are binary. P = Been to party, H = Got a headache, D = Demotivated at work, U = Underperform at work, A = Boss angry. Shaded variables are observed in the true state.

3.6 Code

3.6.1 Naive inference demo

`demoBurglar.m`: Was it the burglar demo

`demoChestClinic.m`: Naive inference on chest clinic. See Exercise 3.4.

3.6.2 Conditional independence demo

The following demo determines whether $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{Z}$ for the Chest Clinic network, Fig. 3.15, and checks the result numerically.⁴ The independence test is based on the Markov method of Section 4.2.4. This is an alternative to the d-separation method and also more general in that it deals also with conditional independence in Markov Networks as well as belief networks. Running the demo code below, it may happen that the numerical dependence is very small – that is

$$p(\mathcal{X}, \mathcal{Y} \mid \mathcal{Z}) \approx p(\mathcal{X} \mid \mathcal{Z})p(\mathcal{Y} \mid \mathcal{Z}) \quad (3.6.1)$$

even though $\mathcal{X} \not\perp\!\!\!\perp \mathcal{Y} \mid \mathcal{Z}$. This highlights the difference between ‘structural’ and ‘numerical’ independence.

`condindepPot.m`: Numerical measure of conditional independence

`demoCondindep.m`: Demo of conditional independence (using Markov method)

3.6.3 Utility routines

`dag.m`: Find the DAG structure for a belief network

3.7 Exercises

- 3.1** (Party animal) The party animal problem corresponds to the network in Fig. 3.14. The boss is angry and the worker has a headache – what is the probability the worker has been to a party? To complete the specifications, the probabilities are given as follows:

$$\begin{array}{lll} p(U = \text{tr} \mid P = \text{tr}, D = \text{tr}) = 0.999 & p(U = \text{tr} \mid P = \text{fa}, D = \text{tr}) = 0.9 & p(H = \text{tr} \mid P = \text{tr}) = 0.9 \\ p(U = \text{tr} \mid P = \text{tr}, D = \text{fa}) = 0.9 & p(U = \text{tr} \mid P = \text{fa}, D = \text{fa}) = 0.01 & p(H = \text{tr} \mid P = \text{fa}) = 0.2 \\ p(A = \text{tr} \mid U = \text{tr}) = 0.95 & p(A = \text{tr} \mid U = \text{fa}) = 0.5 & p(P = \text{tr}) = 0.2, p(D = \text{tr}) = 0.4 \end{array}$$

- 3.2** Consider the distribution $p(a, b, c) = p(c \mid a, b)p(a)p(b)$. (i) Is $a \perp\!\!\!\perp b \mid \emptyset$? (ii) Is $a \perp\!\!\!\perp b \mid c$?

⁴ The code for graphical conditional independence is given in Chapter 4.

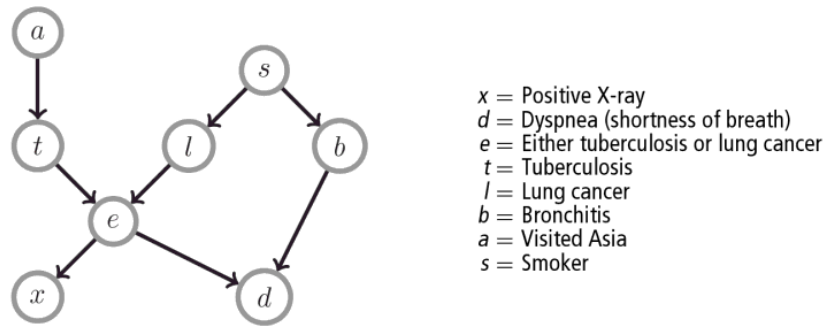


Figure 3.15 Belief network structure for the Chest Clinic example.

- 3.3 The Chest Clinic network [184] concerns the diagnosis of lung disease (tuberculosis, lung cancer, or both, or neither), see Fig. 3.15. In this model a visit to Asia is assumed to increase the probability of tuberculosis. State if the following conditional independence relationships are true or false
1. tuberculosis $\perp\!\!\!\perp$ smoking | shortness of breath
 2. lung cancer $\perp\!\!\!\perp$ bronchitis | smoking
 3. visit to Asia $\perp\!\!\!\perp$ smoking | lung cancer
 4. visit to Asia $\perp\!\!\!\perp$ smoking | lung cancer, shortness of breath

- 3.4 Consider the Chest Clinic belief network in Fig. 3.15 [184]. Calculate by hand the values for $p(d)$, $p(d|s = \text{tr})$, $p(d|s = \text{fa})$. The table values are:

$p(a = \text{tr})$	= 0.01	$p(s = \text{tr})$	= 0.5
$p(t = \text{tr} a = \text{tr})$	= 0.05	$p(t = \text{tr} a = \text{fa})$	= 0.01
$p(l = \text{tr} s = \text{tr})$	= 0.1	$p(l = \text{tr} s = \text{fa})$	= 0.01
$p(b = \text{tr} s = \text{tr})$	= 0.6	$p(b = \text{tr} s = \text{fa})$	= 0.3
$p(x = \text{tr} e = \text{tr})$	= 0.98	$p(x = \text{tr} e = \text{fa})$	= 0.05
$p(d = \text{tr} e = \text{tr}, b = \text{tr})$	= 0.9	$p(d = \text{tr} e = \text{tr}, b = \text{fa})$	= 0.7
$p(d = \text{tr} e = \text{fa}, b = \text{tr})$	= 0.8	$p(d = \text{tr} e = \text{fa}, b = \text{fa})$	= 0.1

$p(e = \text{tr}|t, l) = 0$ only if both t and l are fa , 1 otherwise.

- 3.5 If we interpret the Chest Clinic network Exercise 3.4 causally, how can we help a doctor answer the question ‘If I could cure my patients of bronchitis, how would this affect my patients’ chance of being short of breath?’. How does this compare with $p(d = \text{tr}|b = \text{fa})$ in a non-causal interpretation, and what does this mean?
- 3.6 ([140]) The network in Fig. 3.16 concerns the probability of a car starting, with

$p(b = \text{bad}) = 0.02$	$p(f = \text{empty}) = 0.05$
$p(g = \text{empty} b = \text{good}, f = \text{not empty}) = 0.04$	$p(g = \text{empty} b = \text{good}, f = \text{empty}) = 0.97$
$p(g = \text{empty} b = \text{bad}, f = \text{not empty}) = 0.1$	$p(g = \text{empty} b = \text{bad}, f = \text{empty}) = 0.99$
$p(t = \text{fa} b = \text{good}) = 0.03$	$p(t = \text{fa} b = \text{bad}) = 0.98$
$p(s = \text{fa} t = \text{tr}, f = \text{not empty}) = 0.01$	$p(s = \text{fa} t = \text{tr}, f = \text{empty}) = 0.92$
$p(s = \text{fa} t = \text{fa}, f = \text{not empty}) = 1.0$	$p(s = \text{fa} t = \text{fa}, f = \text{empty}) = 0.99$

Calculate $P(f = \text{empty}|s = \text{no})$, the probability of the fuel tank being empty conditioned on the observation that the car does not start.

- 3.7 There is a synergistic relationship between Asbestos (A) exposure, Smoking (S) and Cancer (C). A model describing this relationship is given by

$$p(A, S, C) = p(C|A, S)p(A)p(S). \quad (3.7.1)$$

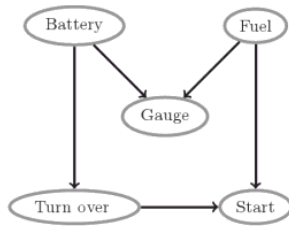
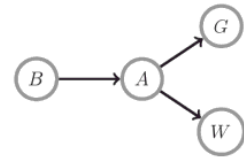


Figure 3.16 Belief network of car starting, see Exercise 3.6.

1. Is $A \perp\!\!\!\perp S \mid \emptyset$?
2. Is $A \perp\!\!\!\perp S \mid C$?
3. How could you adjust the model to account for the fact that people who work in the building industry have a higher likelihood to also be smokers and also a higher likelihood to be exposed to asbestos?

- 3.8 Consider the belief network on the right which represents Mr Holmes' burglary worries as given in Fig. 3.3(a): (B)urglar, (A)larm, (W)atson, Mrs (G)ibbon.



All variables are binary with states {tr, fa}. The table entries are

$$\begin{array}{ll}
 p(B = \text{tr}) & = 0.01 \\
 p(A = \text{tr} \mid B = \text{tr}) & = 0.99 & p(A = \text{tr} \mid B = \text{fa}) & = 0.05 \\
 p(W = \text{tr} \mid A = \text{tr}) & = 0.9 & p(W = \text{tr} \mid A = \text{fa}) & = 0.5 \\
 p(G = \text{tr} \mid A = \text{tr}) & = 0.7 & p(G = \text{tr} \mid A = \text{fa}) & = 0.2
 \end{array} \tag{3.7.2}$$

1. Compute 'by hand' (i.e. show your working):
 - (a) $p(B = \text{tr} \mid W = \text{tr})$
 - (b) $p(B = \text{tr} \mid W = \text{tr}, G = \text{fa})$
2. Consider the same situation as above, except that now the evidence is uncertain. Mrs Gibbon thinks that the state is $G = \text{fa}$ with probability 0.9. Similarly, Dr Watson believes in the state $W = \text{fa}$ with value 0.7. Compute 'by hand' the posteriors under these uncertain (soft) evidences:
 - (a) $p(B = \text{tr} \mid \tilde{W})$
 - (b) $p(B = \text{tr} \mid \tilde{W}, \tilde{G})$

- 3.9 A doctor gives a patient a (D)rug (drug or no drug) dependent on their (A)ge (old or young) and (G)ender (male or female). Whether or not the patient (R)ecovers (recovers or doesn't recover) depends on all D, A, G . In addition $A \perp\!\!\!\perp G \mid \emptyset$.

1. Write down the belief network for the above situation.
2. Explain how to compute $p(\text{recover} \mid \text{drug})$.
3. Explain how to compute $p(\text{recover} \mid \text{do}(\text{drug}), \text{young})$.

- 3.10 Implement the Wet Grass scenario in Section 3.1.1 using the BRMLTOOLBOX.

- 3.11 (LA Burglar) Consider the Burglar scenario, Example 3.1. We now wish to model the fact that in Los Angeles the probability of being burgled increases if there is an earthquake. Explain how to include this effect in the model.

- 3.12 Given two belief networks represented as DAGs with associated adjacency matrices \mathbf{A} and \mathbf{B} , write a MATLAB function `MarkovEquiv(A,B).m` that returns 1 if \mathbf{A} and \mathbf{B} are Markov equivalent, and zero otherwise.

- 3.13** The adjacency matrices of two belief networks are given below (see `ABmatrices.mat`). State if they are Markov equivalent.

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (3.7.3)$$

- 3.14** There are three computers indexed by $i \in \{1, 2, 3\}$. Computer i can send a message in one timestep to computer j if $C_{ij} = 1$, otherwise $C_{ij} = 0$. There is a fault in the network and the task is to find out some information about the communication matrix \mathbf{C} (\mathbf{C} is not necessarily symmetric). To do this, Thomas, the engineer, will run some tests that reveal whether or not computer i can send a message to computer j in t timesteps, $t \in \{1, 2\}$. This is expressed as $C_{ij}(t)$, with $C_{ij}(1) \equiv C_{ij}$. For example, he might know that $C_{13}(2) = 1$, meaning that according to his test, a message sent from computer 1 will arrive at computer 3 in at most two timesteps. Note that this message could go via different routes – it might go directly from 1 to 3 in one timestep, or indirectly from 1 to 2 and then from 2 to 3, or both. You may assume $C_{ii} = 1$. A priori Thomas thinks there is a 10% probability that $C_{ij} = 1$, $i \neq j$, and assumes that each such connection is independent of the rest. Given the test information $\mathcal{C} = \{C_{12}(2) = 1, C_{23}(2) = 0\}$, compute the a posteriori probability vector

$$[p(C_{12} = 1|\mathcal{C}), p(C_{13} = 1|\mathcal{C}), p(C_{23} = 1|\mathcal{C}), p(C_{32} = 1|\mathcal{C}), p(C_{21} = 1|\mathcal{C}), p(C_{31} = 1|\mathcal{C})]. \quad (3.7.4)$$

- 3.15** A belief network models the relation between the variables oil, inf, eh, bp, rt which stand for the price of oil, inflation rate, economy health, British Petroleum Stock price, retailer stock price. Each variable takes the states low, high, except for bp which has states low, high, normal. The belief network model for these variables has tables

$p(eh=low)=0.2$	
$p(bp=low oil=low)=0.9$	$p(bp=normal oil=low)=0.1$
$p(bp=low oil=high)=0.1$	$p(bp=normal oil=high)=0.4$
$p(oil=low eh=low)=0.9$	$p(oil=low eh=high)=0.05$
$p(rt=low inf=low,eh=low)=0.9$	$p(rt=low inf=low,eh=high)=0.1$
$p(rt=low inf=high,eh=low)=0.1$	$p(rt=low inf=high,eh=high)=0.01$
$p(inf=low oil=low,eh=low)=0.9$	$p(inf=low oil=low,eh=high)=0.1$
$p(inf=low oil=high,eh=low)=0.1$	$p(inf=low oil=high,eh=high)=0.01$

1. Draw a belief network for this distribution.
2. Given that the BP stock price is normal and the retailer stock price is high, what is the probability that inflation is high?

- 3.16** There is a set of C potentials with potential c defined on a subset of variables \mathcal{X}_c . If $\mathcal{X}_c \subseteq \mathcal{X}_d$ we can merge (multiply) potentials c and d since the variables in potential c are contained within potential d . With reference to suitable graph structures, describe an efficient algorithm to merge a set of potentials so that for the new set of potentials no potential is contained within the other.

- 3.17** This exercise explores the distinction between d-connection and dependence. Consider the distribution class

$$p(a, b, c) = p(c|b)p(b|a)p(a) \quad (3.7.5)$$

for which a is d-connected to c . One might expect that this means that a and c are dependent, $a \not\perp\!\!\!\perp c$. Our interest is to show that there are non-trivial distributions for which $a \perp\!\!\!\perp c$.

1. Consider $\text{dom}(a) = \text{dom}(c) = \{1, 2\}$ and $\text{dom}(b) = \{1, 2, 3\}$. For

$$p(a) = \begin{pmatrix} 3/5 \\ 2/5 \end{pmatrix}, \quad p(b|a) = \begin{pmatrix} 1/4 & 15/40 \\ 1/12 & 1/8 \\ 2/3 & 1/2 \end{pmatrix}, \quad p(c|b) = \begin{pmatrix} 1/3 & 1/2 & 15/40 \\ 2/3 & 1/2 & 5/8 \end{pmatrix} \quad (3.7.6)$$

show that $a \perp\!\!\!\perp c$.

2. Consider

$$p(a, b, c) = \frac{1}{Z} \phi(a, b) \psi(b, c) \quad (3.7.7)$$

for positive function ϕ, ψ and $Z = \sum_{a,b,c} \phi(a, b) \psi(b, c)$. Defining matrices \mathbf{M} and \mathbf{N} with elements

$$M_{ij} = \phi(a = i, b = j), \quad N_{kj} = \psi(b = j, c = k) \quad (3.7.8)$$

show that the marginal distribution $p(a = i, c = k)$ is represented by the matrix elements

$$p(a = i, c = k) = \frac{1}{Z} [\mathbf{M}\mathbf{N}^T]_{ik}. \quad (3.7.9)$$

3. Show that if

$$\mathbf{M}\mathbf{N}^T = \mathbf{m}_0 \mathbf{n}_0^T \quad (3.7.10)$$

for some vectors \mathbf{m}_0 and \mathbf{n}_0 , then $a \perp\!\!\!\perp c$.

4. Writing

$$\mathbf{M} = [\mathbf{m}_1 \quad \mathbf{m}_2 \quad \mathbf{m}_3], \quad \mathbf{N} = [\mathbf{n}_1 \quad \mathbf{n}_2 \quad \mathbf{n}_3] \quad (3.7.11)$$

for two-dimensional vectors $\mathbf{m}_i, \mathbf{n}_i, i = 1, \dots, 3$, show that

$$\mathbf{M}\mathbf{N}^T = \mathbf{m}_1 \mathbf{n}_1^T + \mathbf{m}_2 \mathbf{n}_2^T + \mathbf{m}_3 \mathbf{n}_3^T. \quad (3.7.12)$$

5. Show that by setting

$$\mathbf{m}_2 = \lambda \mathbf{m}_1, \quad \mathbf{n}_3 = \gamma (\mathbf{n}_1 + \lambda \mathbf{n}_2) \quad (3.7.13)$$

for scalar λ, γ then $\mathbf{M}\mathbf{N}^T$ can be written as $\mathbf{m}_0 \mathbf{n}_0^T$ where

$$\mathbf{m}_0 \equiv \mathbf{m}_1 + \gamma \mathbf{m}_3, \quad \mathbf{n}_0 \equiv \mathbf{n}_1 + \lambda \mathbf{n}_2. \quad (3.7.14)$$

6. Hence construct example tables $p(a), p(b|a), p(c|b)$ for which $a \perp\!\!\!\perp c$. Verify your examples explicitly using BRMLTOOLBOX.

- 3.18** Alice and Bob share a bank account which contains an a priori unknown total amount of money T . Whenever Alice goes to the cash machine, the available amount for withdrawal A for Alice is always 10% of the total T . Similarly, when Bob goes to the cash machine the available amount for withdrawal B for Bob is 10% of the total T . Whatever the amount in the bank, Alice and Bob check their available amounts for withdrawal independently. Draw a belief network that expresses this situation and show that $A \perp\!\!\!\perp B$.

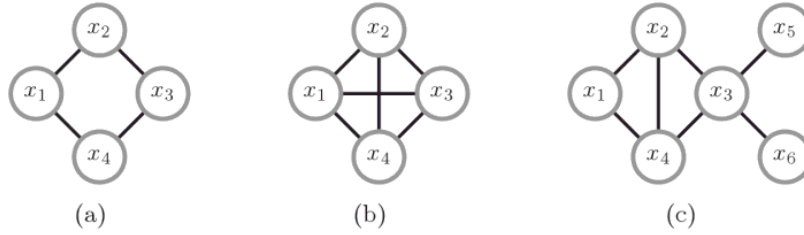


Figure 4.1 (a) $\phi(x_1, x_2)\phi(x_2, x_3)\phi(x_3, x_4)\phi(x_4, x_1)/Z_a$. (b) $\phi(x_1, x_2, x_3, x_4)/Z_b$. (c) $\phi(x_1, x_2, x_4)\phi(x_2, x_3, x_4) \times \phi(x_3, x_5)\phi(x_3, x_6)/Z_c$.

Whilst not a strict separation, GMs tend to fall into two broad classes – those useful in modelling, and those useful in representing inference algorithms. For modelling, belief networks, Markov networks, chain graphs and influence diagrams are some of the most popular. For inference one typically ‘compiles’ a model into a suitable GM for which an algorithm can be readily applied. Such inference GMs include factor graphs and junction trees.

4.2 Markov networks

Belief networks correspond to a special kind of factorisation of the joint probability distribution in which each of the factors is itself a distribution. An alternative factorisation is, for example

$$p(a, b, c) = \frac{1}{Z} \phi(a, b)\phi(b, c) \quad (4.2.1)$$

where $\phi(a, b)$ and $\phi(b, c)$ are *potentials* (see below) and Z is a constant which ensures normalisation, called the *partition function*

$$Z = \sum_{a,b,c} \phi(a, b)\phi(b, c). \quad (4.2.2)$$

Definition 4.1 Potential A potential $\phi(x)$ is a non-negative function of the variable x , $\phi(x) \geq 0$. A joint potential $\phi(x_1, \dots, x_n)$ is a non-negative function of the set of variables. A distribution is a special case of a potential satisfying normalisation, $\sum_x \phi(x) = 1$. This holds similarly for continuous variables, with summation replaced by integration.

We will typically use the convention that the ordering of the variables in the potential is not relevant (as for a distribution) – the joint variables simply index an element of the potential table. Markov Networks (MNs) are defined as products of potentials defined on maximal cliques of an undirected graph – see below and Fig. 4.1.

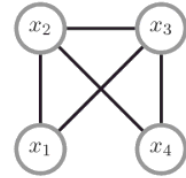
Definition 4.2 Markov network For a set of variables $\mathcal{X} = \{x_1, \dots, x_n\}$ a Markov network is defined as a product of potentials on subsets of the variables $\mathcal{X}_c \subseteq \mathcal{X}$:

$$p(x_1, \dots, x_n) = \frac{1}{Z} \prod_{c=1}^C \phi_c(\mathcal{X}_c). \quad (4.2.3)$$

The constant Z ensures the distribution is normalised. Graphically this is represented by an undirected graph G with $\mathcal{X}_c, c = 1, \dots, C$ being the maximal cliques of G . For the case in which clique potentials are strictly positive, this is called a *Gibbs distribution*.

Definition 4.3 Pairwise Markov network In the special case that the graph contains cliques of only size 2, the distribution is called a *pairwise Markov Network*, with potentials defined on each link between two variables.

Whilst a Markov network is formally defined on maximal cliques, in practice authors often use the term to refer to non-maximal cliques. For example, in the graph on the right, the maximal cliques are x_1, x_2, x_3 and x_2, x_3, x_4 , so that the graph describes a distribution $p(x_1, x_2, x_3, x_4) = \phi(x_1, x_2, x_3)\phi(x_2, x_3, x_4)/Z$. In a pairwise network though the potentials are assumed to be over two cliques, giving $p(x_1, x_2, x_3, x_4) = \phi(x_1, x_2)\phi(x_1, x_3)\phi(x_2, x_3)\phi(x_2, x_4)\phi(x_3, x_4)/Z$.



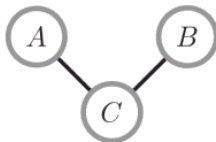
Example 4.1 Boltzmann machine

A Boltzmann machine is an MN on binary variables $\text{dom}(x_i) = \{0, 1\}$ of the form

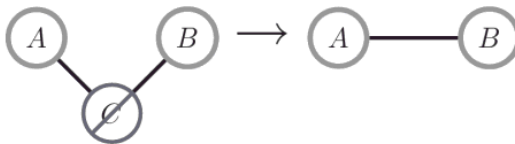
$$p(\mathbf{x}) = \frac{1}{Z(\mathbf{w}, b)} e^{\sum_{i < j} w_{ij} x_i x_j + \sum_i b_i x_i} \quad (4.2.4)$$

where the interactions w_{ij} are the ‘weights’ and the b_i the ‘biases’. This model has been studied in the machine learning community as a basic model of distributed memory and computation [2]. The graphical model of the BM is an undirected graph with a link between nodes i and j for $w_{ij} \neq 0$. Consequently, for all but specially constrained \mathbf{W} , the graph is multiply connected and inference will be typically intractable.

Definition 4.4 Properties of Markov networks



$$p(A, B, C) = \phi_{AC}(A, C)\phi_{BC}(B, C)/Z \quad (4.2.5)$$



Marginalising over C makes A and B (graphically) dependent. In general $p(A, B) \neq p(A)p(B)$.



Conditioning on C makes A and B independent: $p(A, B|C) = p(A|C)p(B|C)$.

4.2.1 Markov properties

We consider here informally the properties of Markov networks and the reader is referred to [182] for detailed proofs. Consider the MN in Fig. 4.2(a) in which we use the shorthand $p(1) \equiv p(x_1)$,

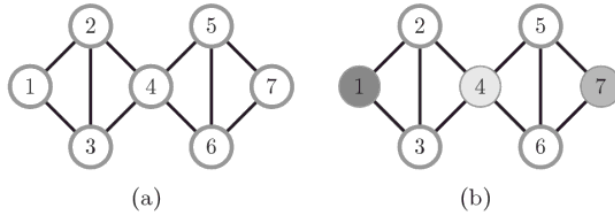


Figure 4.2 (a) $\phi(1, 2, 3)\phi(2, 3, 4)\phi(4, 5, 6)\phi(5, 6, 7)$. (b) By the global Markov property, since every path from 1 to 7 passes through 4, then $1 \perp\!\!\!\perp 7 \mid 4$.

$\phi(1, 2, 3) \equiv \phi(x_1, x_2, x_3)$ etc. We will use this undirected graph to demonstrate conditional independence properties. Note that throughout we will be often dividing by potentials and, in order to ensure this is well defined, we assume the potentials are positive. For positive potentials the following local, pairwise and global Markov properties are all equivalent.

Definition 4.5 Separation A subset \mathcal{S} separates a subset \mathcal{A} from a subset \mathcal{B} (for disjoint \mathcal{A} and \mathcal{B}) if every path from any member of \mathcal{A} to any member of \mathcal{B} passes through \mathcal{S} . If there is no path from a member of \mathcal{A} to a member of \mathcal{B} then \mathcal{A} is separated from \mathcal{B} . If $\mathcal{S} = \emptyset$ then provided no path exists from \mathcal{A} to \mathcal{B} , \mathcal{A} and \mathcal{B} are separated.

Definition 4.6 Global Markov property For disjoint sets of variables, $(\mathcal{A}, \mathcal{B}, \mathcal{S})$ where \mathcal{S} separates \mathcal{A} from \mathcal{B} in G , then $\mathcal{A} \perp\!\!\!\perp \mathcal{B} \mid \mathcal{S}$.

As an example of the global Markov property, consider

$$p(1, 7 \mid 4) \propto \sum_{2,3,5,6} p(1, 2, 3, 4, 5, 6, 7) \tag{4.2.6}$$

$$= \sum_{2,3,5,6} \phi(1, 2, 3)\phi(2, 3, 4)\phi(4, 5, 6)\phi(5, 6, 7) \tag{4.2.7}$$

$$= \left\{ \sum_{2,3} \phi(1, 2, 3)\phi(2, 3, 4) \right\} \left\{ \sum_{5,6} \phi(4, 5, 6)\phi(5, 6, 7) \right\}. \tag{4.2.8}$$

This implies that $p(1, 7 \mid 4) = p(1 \mid 4)p(7 \mid 4)$. This can be inferred since all paths from 1 to 7 pass through 4, see Fig. 4.2(a).

Procedure 4.1 (An algorithm for independence) The separation property implies a simple algorithm for deciding $\mathcal{A} \perp\!\!\!\perp \mathcal{B} \mid \mathcal{S}$. We simply remove all links that neighbour the set of variables \mathcal{S} . If there is no path from any member of \mathcal{A} to any member of \mathcal{B} , then $\mathcal{A} \perp\!\!\!\perp \mathcal{B} \mid \mathcal{S}$ is true – see also Section 4.2.4.

For positive potentials, the so-called local Markov property holds

$$p(x \mid \mathcal{X} \setminus x) = p(x \mid \text{ne}(x)). \tag{4.2.9}$$

That is, when conditioned on its neighbours, x is independent of the remaining variables of the graph. In addition, the so-called pairwise Markov property holds that for any non-adjacent vertices x and y

$$x \perp\!\!\!\perp y \mid \mathcal{X} \setminus \{x, y\}. \tag{4.2.10}$$

4.2.2 Markov random fields

A Markov Random Field (MRF) is a set of conditional distributions, one for each indexed ‘location’.

Definition 4.7 Markov random field A MRF is defined by a set of distributions $p(x_i | \text{ne}(x_i))$ where $i \in \{1, \dots, n\}$ indexes the distributions and $\text{ne}(x_i)$ are the neighbours of variable x_i , namely that subset of the variables x_1, \dots, x_n that the distribution of variable x_i depends on. The term Markov indicates that this is a proper subset of the variables.

A distribution is an MRF with respect to an undirected graph G if

$$p(x_i | x_{\setminus i}) = p(x_i | \text{ne}(x_i)) \quad (4.2.11)$$

where $\text{ne}(x_i)$ are the neighbouring variables of variable x_i , according to the undirected graph G . The notation $x_{\setminus i}$ is shorthand for the set of all variables \mathcal{X} excluding variable x_i , namely $\mathcal{X} \setminus x_i$ in set notation.

4.2.3 Hammersley–Clifford theorem

An undirected graph G specifies a set of independence statements. An interesting challenge is to find the most general functional form of a distribution that satisfies these independence statements. A trivial example is the graph $x_1 - x_2 - x_3$, from which we have $x_1 \perp\!\!\!\perp x_3 | x_2$. From this requirement we must have

$$p(x_1 | x_2, x_3) = p(x_1 | x_2). \quad (4.2.12)$$

Hence

$$p(x_1, x_2, x_3) = p(x_1 | x_2, x_3) p(x_2, x_3) = p(x_1 | x_2) p(x_2, x_3) = \phi_{12}(x_1, x_2) \phi_{23}(x_2, x_3) \quad (4.2.13)$$

where the ϕ are potentials.

More generally, for any decomposable graph G , see Definition 6.9, we can start at the edge and work inwards to reveal that the functional form must be a product of potentials on the cliques of G . For example, for Fig. 4.2(a), we can start with the variable x_1 and the corresponding local Markov statement $x_1 \perp\!\!\!\perp x_4, x_5, x_6, x_6 | x_2, x_3$ to write

$$p(x_1, \dots, x_7) = p(x_1 | x_2, x_3) p(x_2, x_3, x_4, x_5, x_6, x_7). \quad (4.2.14)$$

Now we consider x_1 eliminated and move to the neighbours of x_1 , namely x_2, x_3 . The graph specifies that x_1, x_2, x_3 are independent of x_5, x_6, x_7 given x_4 :

$$p(x_1, x_2, x_3 | x_4, x_5, x_6, x_7) = p(x_1, x_2, x_3 | x_4). \quad (4.2.15)$$

By summing both sides above over x_1 we have that $p(x_2, x_3 | x_4, x_5, x_6, x_7) = p(x_2, x_3 | x_4)$. Hence

$$p(x_2, x_3, x_4, x_5, x_6, x_7) = p(x_2, x_3 | x_4, x_5, x_6, x_7) p(x_4, x_5, x_6, x_7) = p(x_2, x_3 | x_4) p(x_4, x_5, x_6, x_7) \quad (4.2.16)$$

and

$$p(x_1, \dots, x_7) = p(x_1 | x_2, x_3) p(x_2, x_3 | x_4) p(x_4, x_5, x_6, x_7). \quad (4.2.17)$$

Having eliminated x_2, x_3 , we now move to their neighbour(s) on the remaining graph, namely x_4 . Continuing in this way, we necessarily end up with a distribution of the form

$$p(x_1, \dots, x_7) = p(x_1 | x_2, x_3) p(x_2, x_3 | x_4) p(x_4 | x_5, x_6) p(x_5, x_6 | x_7) p(x_7). \quad (4.2.18)$$

The pattern here is clear and shows that the Markov conditions mean that the distribution is expressible as a product of potentials defined on the cliques of the graph. That is $G \Rightarrow F$ where F is a factorisation into clique potentials on G . The converse is easily shown, namely that given a factorisation into clique potentials, the Markov conditions on G are implied. Hence $G \Leftrightarrow F$. It is clear that for any decomposable G , this always holds since we can always work inwards from the edges of the graph.

The Hammersley–Clifford theorem is a stronger result and shows that this factorisation property holds for any undirected graph, provided that the potentials are positive. For a formal proof, the reader is referred to [182, 36, 219]. An informal argument can be made by considering a specific example, and we take the 4-cycle $x_1 - x_2 - x_3 - x_4 - x_1$ from Fig. 4.1(a). The theorem states that for positive potentials ϕ , the Markov conditions implied by the graph mean that the distribution must be of the form

$$p(x_1, x_2, x_3, x_4) = \phi_{12}(x_1, x_2)\phi_{23}(x_2, x_3)\phi_{34}(x_3, x_4)\phi_{41}(x_4, x_1). \quad (4.2.19)$$

One may readily verify that for any distribution of this form $x_1 \perp\!\!\!\perp x_3 | x_2, x_4$. Consider including an additional term that links x_1 to a variable not a member of the cliques that x_1 inhabits. That is we include a term $\phi_{13}(x_1, x_3)$. Our aim is to show that a distribution of the form

$$p(x_1, x_2, x_3, x_4) = \phi_{12}(x_1, x_2)\phi_{23}(x_2, x_3)\phi_{34}(x_3, x_4)\phi_{41}(x_4, x_1)\phi_{13}(x_1, x_3) \quad (4.2.20)$$

cannot satisfy the Markov property $x_1 \perp\!\!\!\perp x_3 | x_2, x_4$. To do so we examine

$$p(x_1 | x_2, x_3, x_4) = \frac{\phi_{12}(x_1, x_2)\phi_{23}(x_2, x_3)\phi_{34}(x_3, x_4)\phi_{41}(x_4, x_1)\phi_{13}(x_1, x_3)}{\sum_{x_1} \phi_{12}(x_1, x_2)\phi_{23}(x_2, x_3)\phi_{34}(x_3, x_4)\phi_{41}(x_4, x_1)\phi_{13}(x_1, x_3)} \quad (4.2.21)$$

$$= \frac{\phi_{12}(x_1, x_2)\phi_{41}(x_4, x_1)\phi_{13}(x_1, x_3)}{\sum_{x_1} \phi_{12}(x_1, x_2)\phi_{41}(x_4, x_1)\phi_{13}(x_1, x_3)}. \quad (4.2.22)$$

If we assume that the potential ϕ_{13} is weakly dependent on x_1 and x_3 ,

$$\phi_{13}(x_1, x_3) = 1 + \epsilon \psi(x_1, x_3) \quad (4.2.23)$$

where $\epsilon \ll 1$, then $p(x_1 | x_2, x_3, x_4)$ is given by

$$\frac{\phi_{12}(x_1, x_2)\phi_{41}(x_4, x_1)}{\sum_{x_1} \phi_{12}(x_1, x_2)\phi_{41}(x_4, x_1)} (1 + \epsilon \psi(x_1, x_3)) \left(1 + \epsilon \frac{\sum_{x_1} \phi_{12}(x_1, x_2)\phi_{41}(x_4, x_1)\psi(x_1, x_3)}{\sum_{x_1} \phi_{12}(x_1, x_2)\phi_{41}(x_4, x_1)} \right)^{-1}. \quad (4.2.24)$$

By expanding $(1 + \epsilon f)^{-1} = 1 - \epsilon f + O(\epsilon^2)$ and retaining only terms that are first order in ϵ , we obtain

$$p(x_1 | x_2, x_3, x_4) = \frac{\phi_{12}(x_1, x_2)\phi_{41}(x_4, x_1)}{\sum_{x_1} \phi_{12}(x_1, x_2)\phi_{41}(x_4, x_1)} \times \left(1 + \epsilon \left[\psi(x_1, x_3) - \frac{\sum_{x_1} \phi_{12}(x_1, x_2)\phi_{41}(x_4, x_1)\psi(x_1, x_3)}{\sum_{x_1} \phi_{12}(x_1, x_2)\phi_{41}(x_4, x_1)} \right] \right) + O(\epsilon^2). \quad (4.2.25)$$

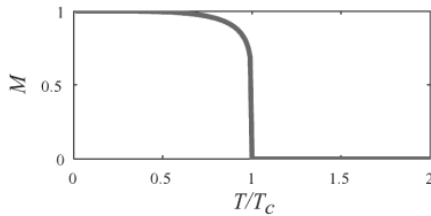
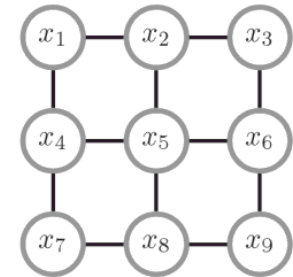


Figure 4.5 Onsagar magnetisation. As the temperature T decreases towards the critical temperature T_c a phase transition occurs in which a large fraction of the variables become aligned in the same state.

Consider a model in which our desire is that states of the binary valued variables x_1, \dots, x_9 , arranged on a lattice (right) should prefer their neighbouring variables to be in the same state

$$p(x_1, \dots, x_9) = \frac{1}{Z} \prod_{i \sim j} \phi_{ij}(x_i, x_j) \quad (4.2.28)$$

where $i \sim j$ denotes the set of indices where i and j are neighbours in the undirected graph.



The Ising model

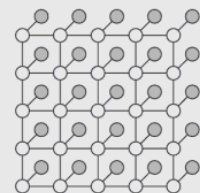
A set of potentials for Equation (4.2.28) that encourages neighbouring variables to have the same state is

$$\phi_{ij}(x_i, x_j) = e^{-\frac{1}{2T}(x_i - x_j)^2}, x_i \in \{-1, +1\}. \quad (4.2.29)$$

This corresponds to a well-known model of the physics of magnetic systems, called the *Ising model* which consists of ‘mini-magnets’ which prefer to be aligned in the same state, depending on the temperature T . For high T the variables behave independently so that no global magnetisation appears. For low T , there is a strong preference for neighbouring mini-magnets to become aligned, generating a strong macro-magnet. Remarkably, one can show that, in a very large two-dimensional lattice, below the so-called Curie temperature, $T_c \approx 2.269$ (for ± 1 variables), the system admits a phase change in that a large fraction of the variables become aligned – above T_c , on average, the variables are unaligned. This is depicted in Fig. 4.5 where $M = \left| \sum_{i=1}^N x_i \right| / N$ is the average alignment of the variables. That this phase change happens for non-zero temperature has driven considerable research in this and related areas [41]. Global coherence effects such as this that arise from weak local constraints are present in systems that admit *emergent behaviour*. Similar local constraints are popular in image restoration algorithms to clean up noise, under the assumption that noise will not show any local spatial coherence, whilst ‘signal’ will.

Example 4.2 Cleaning up images

Consider a binary image defined on a set of pixels $x_i \in \{-1, +1\}, i = 1, \dots, D$. We observe a noise-corrupted version y_i of each pixel x_i , in which the state of $y_i \in \{-1, +1\}$ is opposite to x_i with some probability. Here the filled nodes indicate observed noisy pixels and the unshaded nodes the latent clean pixels. Our interest is to ‘clean up’ the observed dirty image \mathcal{Y} , and find the most likely joint clean image \mathcal{X} .



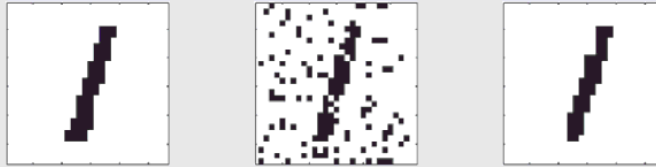
A model for this situation is

$$p(\mathcal{X}, \mathcal{Y}) = \frac{1}{Z} \left[\prod_{i=1}^D \phi(x_i, y_i) \right] \left[\prod_{i \sim j} \psi(x_i, x_j) \right], \quad \phi(x_i, y_i) = e^{\beta x_i y_i}, \quad \psi(x_i, x_j) = e^{\alpha x_i x_j} \quad (4.2.30)$$

here $i \sim j$ indicates the set of latent variables that are neighbours. The potential ϕ encourages the noisy and clean pixel to be in the same state. Similarly, the potential $\psi(x_i, x_j)$ encourages neighbouring pixels to be in the same state. To find the most likely clean image, we need to compute

$$\operatorname{argmax}_{\mathcal{X}} p(\mathcal{X}|\mathcal{Y}) = \operatorname{argmax}_{\mathcal{X}} p(\mathcal{X}, \mathcal{Y}). \quad (4.2.31)$$

This is a computationally difficult task but can be approximated using iterative methods, see Section 28.9.



On the left is the clean image, from which a noisy corrupted image \mathcal{Y} is formed (middle). The most likely restored image is given on the right. See `demoMRFclean.m`. Note that the parameter β is straightforward to set, given knowledge of the corruption probability p_{corrupt} , since $p(y_i \neq x_i | x_i) = \sigma(2\beta)$, so that $\beta = \frac{1}{2} \sigma^{-1}(p_{\text{corrupt}})$. Setting α is more complex since relating $p(x_i = x_j)$ to α is not straightforward, see Section 28.4.1. In the demonstration we set $\alpha = 10$, $p_{\text{corrupt}} = 0.15$.

4.3 Chain graphical models

Chain Graphs (CGs) contain both directed and undirected links. To develop the intuition, consider Fig. 4.6(a). The only terms that we can unambiguously specify from this depiction are $p(a)$ and $p(b)$ since there is no mixed interaction of directed and undirected edges at the a and b vertices. By probability, therefore, we must have

$$p(a, b, c, d) = p(a)p(b)p(c, d|a, b). \quad (4.3.1)$$

Looking at the graph, we might expect the interpretation to be

$$p(c, d|a, b) = \phi(c, d)p(c|a)p(d|b). \quad (4.3.2)$$

However, to ensure normalisation, and also to retain generality, we interpret this as

$$p(c, d|a, b) = \phi(c, d)p(c|a)p(d|b)\phi(a, b), \quad \text{with } \phi(a, b) \equiv \left(\sum_{c,d} \phi(c, d)p(c|a)p(d|b) \right)^{-1}. \quad (4.3.3)$$

This leads to the interpretation of a CG as a DAG over the chain components see below.

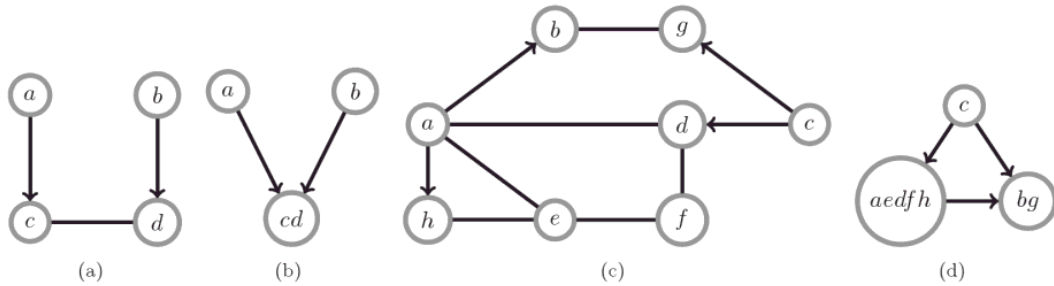


Figure 4.6 Chain graphs. The chain components are identified by deleting the directed edges and identifying the remaining connected components. (a) Chain components are (a) , (b) , (c, d) , which can be written as a BN on the cluster variables in (b). (c) Chain components are (a, e, d, f, h) , (b, g) , (c) , which has the cluster BN representation (d).

Definition 4.8 Chain component The chain components of a graph G are obtained by:

1. Forming a graph G' with directed edges removed from G .
2. Then each connected component in G' constitutes a chain component.

Each chain component represents a distribution over the variables of the component, conditioned on the parental components. The conditional distribution is itself a product over the cliques of the undirected component and moralised parental components, including also a factor to ensure normalisation over the chain component.

Definition 4.9 Chain graph distribution The distribution associated with a chain graph G is found by first identifying the chain components, τ . Then

$$p(x) = \prod_{\tau} p(\mathcal{X}_{\tau} | \text{pa}(\mathcal{X}_{\tau})) \quad (4.3.4)$$

and

$$p(\mathcal{X}_{\tau} | \text{pa}(\mathcal{X}_{\tau})) \propto \prod_{c \in \mathcal{C}_{\tau}} \phi(\mathcal{X}_c) \quad (4.3.5)$$

where \mathcal{C}_{τ} denotes the union of the cliques in component τ together with the moralised parental components of τ , with ϕ being the associated functions defined on each clique. The proportionality factor is determined implicitly by the constraint that the distribution sums to 1.

BNs are CGs in which the connected components are singletons. MNs are CGs in which the chain components are simply the connected components of the undirected graph. Chain graphs can be useful since they are more expressive of CI statements than either belief networks or Markov networks alone. The reader is referred to [182] and [106] for further details.

Example 4.3 Chain graphs are more expressive than belief or Markov networks

Consider the chain graph in Fig. 4.7(a), which has chain component decomposition

$$p(a, b, c, d, e, f) = p(a)p(b)p(c, d, e, f|a, b) \quad (4.3.6)$$

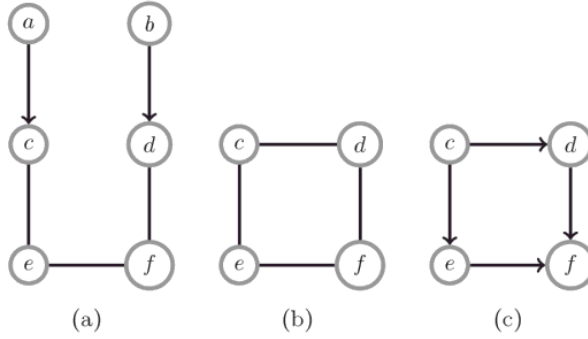


Figure 4.7 The CG (a) expresses $a \perp\!\!\!\perp b \mid \emptyset$ and $d \perp\!\!\!\perp e \mid (c, f)$. No directed graph could express both these conditions since the marginal distribution $p(c, d, e, f)$ is an undirected 4-cycle, (b). Any DAG on a 4-cycle must contain a collider, as in (c) and therefore express a different set of CI statements than (b). Similarly, no connected Markov network can express unconditional independence and hence (a) expresses CI statements that no belief network or Markov network alone can express.

where

$$p(c, d, e, f \mid a, b) = \phi(a, c)\phi(c, e)\phi(e, f)\phi(d, f)\phi(d, b)\phi(a, b) \quad (4.3.7)$$

with the normalisation requirement

$$\phi(a, b) \equiv \left(\sum_{c, d, e, f} \phi(a, c)\phi(c, e)\phi(e, f)\phi(d, f)\phi(d, b) \right)^{-1}. \quad (4.3.8)$$

The marginal $p(c, d, e, f)$ is given by

$$\phi(c, e)\phi(e, f)\phi(d, f) \underbrace{\sum_{a, b} \phi(a, b)p(a)p(b)\phi(a, c)\phi(d, b)}_{\phi(c, d)}. \quad (4.3.9)$$

Since the marginal distribution of $p(c, d, e, f)$ is an undirected 4-cycle, no DAG can express the CI statements contained in the marginal $p(c, d, e, f)$. Similarly no undirected distribution on the same skeleton as Fig. 4.7(a) could express that a and b are independent (unconditionally), i.e. $p(a, b) = p(a)p(b)$.

4.4 Factor graphs

Factor Graphs (FGs) are mainly used as part of inference algorithms.¹

Definition 4.10 Factor graph Given a function

$$f(x_1, \dots, x_n) = \prod_i \psi_i(\mathcal{X}_i), \quad (4.4.1)$$

the FG has a node (represented by a square) for each factor ψ_i , and a variable node (represented by a circle) for each variable x_j . For each $x_j \in \mathcal{X}_i$ an undirected link is made between factor ψ_i and variable x_j .

¹ Formally a FG is an alternative graphical depiction of a hypergraph [86] in which the vertices represent variables, and a hyperedge a factor as a function of the variables associated with the hyperedge. A FG is therefore a hypergraph with the additional interpretation that the graph represents a function defined as products over the associated hyperedges. Many thanks to Robert Cowell for this observation.

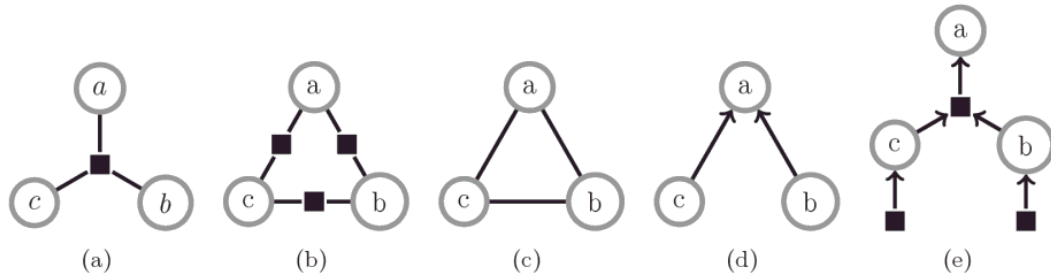


Figure 4.8 (a) $\phi(a, b, c)$. (b) $\phi(a, b)\phi(b, c)\phi(c, a)$. (c) $\phi(a, b, c)$. Both (a) and (b) have the same undirected graphical model, (c). (d) (a) is an undirected FG of (d). (e) Directed FG of the BN in (d). A directed factor represents a term p (children|parents). The advantage of (e) over (a) is that information regarding the marginal independence of variables b and c is clear from graph (e), whereas one could only ascertain this by examination of the numerical entries of the factors in graph (a).

When used to represent a distribution

$$p(x_1, \dots, x_n) = \frac{1}{Z} \prod_i \psi_i(\mathcal{X}_i) \quad (4.4.2)$$

a normalisation constant $Z = \sum_{\mathcal{X}} \prod_i \psi_i(\mathcal{X}_i)$ is assumed.

For a factor $\psi_i(\mathcal{X}_i)$ which is a conditional distribution $p(x_i | \text{pa}(x_i))$, we may use directed links from the parents to the factor node, and a directed link from the factor node to the child x_i . This has the same structure as an (undirected) FG, but preserves the information that the factors are distributions.

Factor graphs are useful since they can preserve more information about the form of the distribution than either a belief network or a Markov network (or chain graph) can do alone. Consider the distribution

$$p(a, b, c) = \phi(a, b)\phi(a, c)\phi(b, c). \quad (4.4.3)$$

Represented as an MN, this must have a single clique, as given in Fig. 4.8(c). However, Fig. 4.8(c) could equally represent some unfactored clique potential $\phi(a, b, c)$ so that the factorised structure within the clique is lost. In this sense, the FG representation in Fig. 4.8(b) more precisely conveys the form of distribution equation (4.4.3). An unfactored clique potential $\phi(a, b, c)$ is represented by the FG Fig. 4.8(a). Hence different FGs can have the same MN since information regarding the structure of the clique potential is lost in the MN. Similarly, for a belief network, as in Fig. 4.8(d) one can represent this using a standard undirected FG, although more information about the independence is preserved by using a directed FG representation, as in Fig. 4.8(e). One can also consider partially directed FGs which contain both directed and undirected edges; this requires a specification of how the structure is normalised, one such being to use an approach analogous to the chain graph – see [103] for details.

4.4.1 Conditional independence in factor graphs

Conditional independence questions can be addressed using a rule which works with directed, undirected and partially directed FGs [103]. To determine whether two variables are independent given a set of conditioned variables, consider all paths connecting the two variables. If all paths

For this we have

$$\mathcal{L}_G = \{t_2 \perp\!\!\!\perp (t_1, y_1)\}. \quad (4.5.10)$$

Hence $\mathcal{L}_G \subset \mathcal{L}_P$ so that the BN is an I-map for (4.5.7) since every independence statement in the BN is true for the distribution class in Equation (4.5.7). However, it is not a D-map since $\mathcal{L}_P \not\subseteq \mathcal{L}_G$. In this case no perfect map (a BN or an MN) can represent (4.5.7).

Remark 4.1 (Forcing dependencies?) Whilst graphical models as we have defined them ensure specified independencies, they seem to be inappropriate for ensuring specified dependencies. Consider the undirected graph $x - y - z$. Graphically this expresses that x and z are dependent. However, there are numerical instances of distributions for which this does not hold, for example

$$p_1(x, y, z) = \phi(x, y)\phi(y, z)/Z_1 \quad (4.5.11)$$

with $\phi(x, y) = \text{const}$. One might complain that this is a pathological case since any graphical representation of this particular instance contains no link between x and y . Maybe one should therefore ‘force’ potentials to be non-trivial functions of their arguments and thereby ensure dependency? Consider

$$\phi(x, y) = \frac{x}{y}, \quad \phi(y, z) = yz. \quad (4.5.12)$$

In this case both potentials are non-trivial in the sense that they are truly functionally dependent on their arguments. Hence, the undirected network contains ‘genuine’ links $x - y$ and $y - z$. Nevertheless,

$$p_2(x, y, z) = \phi(x, y)\phi(y, z)/Z_2 \propto \frac{x}{y}yz = xz. \quad (4.5.13)$$

Hence $p_2(x, z) \propto xz \Rightarrow x \perp\!\!\!\perp z$. So ‘forcing’ local non-trivial functions does not guarantee dependence of path-connected variables. In this case, the algebraic cancellation is clear and the problem is again rather trivial since for p_2 , $x \perp\!\!\!\perp y$ and $y \perp\!\!\!\perp z$, so one might assume that $x \perp\!\!\!\perp z$ (see however, Remark 1.2). However, there may be cases where such algebraic simplifications are highly non-trivial, though nevertheless true. See, for example, Exercise 3.17 in which we construct $p(x, y, z) \propto \phi(x, y)\phi(y, z)$ for which $x \perp\!\!\!\perp y$ and $y \perp\!\!\!\perp z$, yet $x \perp\!\!\!\perp z$.

4.6 Summary

- Graphical modelling is the discipline of representing probability models graphically.
- Belief networks intuitively describe which variables ‘causally’ influence others and are represented using directed graphs.
- A Markov network is represented by an undirected graph.
- Intuitively, linked variables in a Markov network are graphically dependent, describing local cliques of graphically dependent variables.
- Markov networks are historically important in physics and may be used to understand how global collaborative phenomena can emerge from only local dependencies.
- Graphical models are generally limited in their ability to represent all the possible logical consequences of a probabilistic model.
- Some special probabilistic models can be ‘perfectly’ mapped graphically.

- Factor graphs describe the factorisation of functions and are not necessarily related to probability distributions.

A detailed discussion of the axiomatic and logical basis of conditional independence is given in [47] and [280].

4.7 Code

`condindep.m`: Conditional independence test $p(X, Y|Z) = p(X|Z)p(Y|Z)$?

4.8 Exercises

- 4.1 1. Consider the pairwise Markov network,

$$p(x) = \phi(x_1, x_2)\phi(x_2, x_3)\phi(x_3, x_4)\phi(x_4, x_1). \quad (4.8.1)$$

Express in terms of ϕ the following:

$$p(x_1|x_2, x_4), \quad p(x_2|x_1, x_3), \quad p(x_3|x_2, x_4), \quad p(x_4|x_1, x_3). \quad (4.8.2)$$

2. For a set of local distributions defined as

$$p_1(x_1|x_2, x_4), \quad p_2(x_2|x_1, x_3), \quad p_3(x_3|x_2, x_4), \quad p_4(x_4|x_1, x_3) \quad (4.8.3)$$

is it always possible to find a joint distribution $p(x_1, x_2, x_3, x_4)$ consistent with these local conditional distributions?

- 4.2 Consider the Markov network

$$p(a, b, c) = \phi_{ab}(a, b)\phi_{bc}(b, c). \quad (4.8.4)$$

Nominally, by summing over b , the variables a and c are dependent. For binary b , explain a situation in which this is not the case, so that marginally, a and c are independent.

- 4.3 Show that for the Boltzmann machine defined on binary variables x_i with

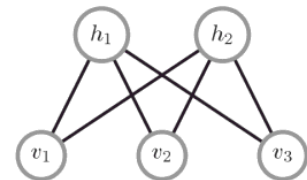
$$p(\mathbf{x}) = \frac{1}{Z(\mathbf{W}, \mathbf{b})} \exp(\mathbf{x}^T \mathbf{W} \mathbf{x} + \mathbf{x}^T \mathbf{b}) \quad (4.8.5)$$

one may assume, without loss of generality, $\mathbf{W} = \mathbf{W}^T$.

- 4.4 The *restricted Boltzmann machine* (or *Harmonium* [269]) is a constrained Boltzmann machine on a bipartite graph, consisting of a layer of visible variables $\mathbf{v} = (v_1, \dots, v_V)$ and hidden variables $\mathbf{h} = (h_1, \dots, h_H)$:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z(\mathbf{W}, \mathbf{a}, \mathbf{b})} \exp(\mathbf{v}^T \mathbf{W} \mathbf{h} + \mathbf{a}^T \mathbf{v} + \mathbf{b}^T \mathbf{h}) \quad (4.8.6)$$

All variables are binary taking states 0, 1.



1. Show that the distribution of hidden units conditional on the visible units factorises as

$$p(\mathbf{h}|\mathbf{v}) = \prod_i p(h_i|\mathbf{v}), \quad \text{with } p(h_i = 1|\mathbf{v}) = \sigma\left(b_i + \sum_j W_{ji} v_j\right) \quad (4.8.7)$$


where $\sigma(x) = e^x / (1 + e^x)$.

2. By symmetry arguments, write down the form of the conditional $p(\mathbf{v}|\mathbf{h})$.
3. Is $p(\mathbf{h})$ factorised?
4. Can the partition function $Z(\mathbf{W}, \mathbf{a}, \mathbf{b})$ be computed efficiently for the RBM?

4.5 You are given that

$$x \perp\!\!\!\perp y | (z, u), \quad u \perp\!\!\!\perp z | \emptyset. \quad (4.8.8)$$

Derive the most general form of probability distribution $p(x, y, z, u)$ consistent with these statements. Does this distribution have a simple graphical model?

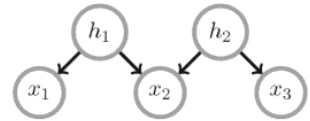
- 4.6 The undirected graph  represents a Markov network with nodes x_1, x_2, x_3, x_4, x_5 , counting clockwise around the pentagon with potentials $\phi(x_i, x_j)$. Show that the joint distribution can be written as

$$p(x_1, x_2, x_3, x_4, x_5) = \frac{p(x_1, x_2, x_5)p(x_2, x_4, x_5)p(x_2, x_3, x_4)}{p(x_2, x_5)p(x_2, x_4)} \quad (4.8.9)$$

and express the marginal probability tables explicitly as functions of the potentials $\phi(x_i, x_j)$.

4.7 Consider the belief network on the right.

1. Write down a Markov network of $p(x_1, x_2, x_3)$.
2. Is your Markov network a perfect map of $p(x_1, x_2, x_3)$?



4.8 Two research labs work independently on the relationship between discrete variables x and y . Lab A proudly announces that they have ascertained distribution $p_A(x|y)$ from data. Lab B proudly announces that they have ascertained $p_B(y|x)$ from data.

1. Is it always possible to find a joint distribution $p(x, y)$ consistent with the results of both labs?
2. Is it possible to define consistent marginals $p(x)$ and $p(y)$, in the sense that $p(x) = \sum_y p_A(x|y)p(y)$ and $p(y) = \sum_x p_B(y|x)p(x)$? If so, explain how to find such marginals. If not, explain why not.

4.9 Research lab A states its findings about a set of variables x_1, \dots, x_n as a list L_A of conditional independence statements. Lab B similarly provides a list of conditional independence statements L_B .

1. Is it always possible to find a distribution which is consistent with L_A and L_B ?
2. If the lists also contain dependence statements, how could one attempt to find a distribution that is consistent with both lists?

4.10 Consider the distribution

$$p(x, y, w, z) = p(z|w)p(w|x, y)p(x)p(y). \quad (4.8.10)$$

1. Write $p(x|z)$ using a formula involving (all or some of) $p(z|w)$, $p(w|x, y)$, $p(x)$, $p(y)$.
2. Write $p(y|z)$ using a formula involving (all or some of) $p(z|w)$, $p(w|x, y)$, $p(x)$, $p(y)$.
3. Using the above results, derive an explicit condition for $x \perp\!\!\!\perp y | z$ and explain if this is satisfied for this distribution.

4.11 Consider the distribution

$$p(t_1, t_2, y_1, y_2, h) = p(y_1|y_2, t_1, t_2, h)p(y_2|t_2, h)p(t_1)p(t_2)p(h). \quad (4.8.11)$$

1. Draw a belief network for this distribution.
2. Does the distribution

$$p(t_1, t_2, y_1, y_2) = \sum_h p(y_1|y_2, t_1, t_2, h)p(y_2|t_2, h)p(t_1)p(t_2)p(h) \quad (4.8.12)$$

have a perfect map belief network?

3. Show that for $p(t_1, t_2, y_1, y_2)$ as defined above $t_1 \perp\!\!\!\perp y_2 \mid \emptyset$.

4.12 Consider the distribution

$$p(a, b, c, d) = \phi_{ab}(a, b)\phi_{bc}(b, c)\phi_{cd}(c, d)\phi_{da}(d, a) \quad (4.8.13)$$

where the ϕ are potentials.

1. Draw a Markov network for this distribution.
 2. Explain if the distribution can be represented as a ('non-complete') belief network.
 3. Derive explicitly if $a \perp\!\!\!\perp c \mid \emptyset$.
- 4.13** Show how for any singly connected Markov network, one may construct a Markov equivalent belief network.
- 4.14** Consider a pairwise binary Markov network defined on variables $s_i \in \{0, 1\}$, $i = 1, \dots, N$, with $p(s) = \prod_{i,j \in \mathcal{E}} \phi_{ij}(s_i, s_j)$, where \mathcal{E} is a given edge set and the potentials ϕ_{ij} are arbitrary. Explain how to translate such a Markov network into a Boltzmann machine.

5

Efficient inference in trees

In previous chapters we discussed how to set up models. Inference then corresponds to operations such as summing over subsets of variables. In machine learning and related areas we will often deal with distributions containing hundreds of variables. In general inference it is computationally very expensive and it is useful to understand for which graphical structures this could be cheap in order that we may make models which we can subsequently compute with. In this chapter we discuss inference in a cheap case, namely trees, which has links to classical algorithms in many different fields from computer science (dynamic programming) to physics (transfer matrix methods).

5.1 Marginal inference

Given a distribution $p(x_1, \dots, x_n)$, *inference* is the process of computing functions of the distribution. *Marginal inference* is concerned with the computation of the distribution of a subset of variables, possibly conditioned on another subset. For example, given a joint distribution $p(x_1, x_2, x_3, x_4, x_5)$ and evidence $x_1 = \text{tr}$, a marginal inference calculation is

$$p(x_5 | x_1 = \text{tr}) \propto \sum_{x_2, x_3, x_4} p(x_1 = \text{tr}, x_2, x_3, x_4, x_5). \quad (5.1.1)$$

Marginal inference for discrete models involves summation and will be the focus of our development. In principle the algorithms carry over to continuous variable models although the lack of closure of most continuous distributions under marginalisation (the Gaussian being a notable exception) can make the direct transference of these algorithms to the continuous domain problematic. The focus here is on efficient inference algorithms for marginal inference in singly connected structures. An efficient algorithm for multiply connected graphs will be considered in Chapter 6.

5.1.1 Variable elimination in a Markov chain and message passing

A key concept in efficient inference is *message passing* in which information from the graph is summarised by local edge information. To develop this idea, consider the four-variable Markov chain (Markov chains are discussed in more depth in Section 23.1)

$$p(a, b, c, d) = p(a|b)p(b|c)p(c|d)p(d) \quad (5.1.2)$$

where M_{ij} is an element of the transition matrix

$$\mathbf{M} = \begin{pmatrix} 0.7 & 0.5 & 0 \\ 0.3 & 0.3 & 0.5 \\ 0 & 0.2 & 0.5 \end{pmatrix}. \quad (5.1.12)$$

The matrix \mathbf{M} is called ‘stochastic’ meaning that, as required of a conditional probability table, its columns sum to 1, $\sum_{i=1}^3 M_{ij} = 1$. Given that the fly is in room 1 at time $t = 1$, what is the probability of room occupancy at time $t = 5$? Assume a Markov chain which is defined by the joint distribution

$$p(x_1, \dots, x_T) = p(x_1) \prod_{t=1}^{T-1} p(x_{t+1}|x_t). \quad (5.1.13)$$

We are asked to compute $p(x_5|x_1 = 1)$ which is given by

$$\sum_{x_4, x_3, x_2} p(x_5|x_4)p(x_4|x_3)p(x_3|x_2)p(x_2|x_1 = 1). \quad (5.1.14)$$

Since the graph of the distribution is a Markov chain, we can easily distribute the summation over the terms. This is most easily done using the transfer matrix method, giving

$$p(x_5 = i|x_1 = 1) = [\mathbf{M}^4 \mathbf{v}]_i \quad (5.1.15)$$

where \mathbf{v} is a vector with components $(1, 0, 0)^\top$, reflecting the evidence that at time $t = 1$ the fly is in room 1. Computing this we have (to four decimal places of accuracy)

$$\mathbf{M}^4 \mathbf{v} = \begin{pmatrix} 0.5746 \\ 0.3180 \\ 0.1074 \end{pmatrix}. \quad (5.1.16)$$

Similarly, at time $t = 6$, the occupancy probabilities are $(0.5612, 0.3215, 0.1173)$. The room occupancy probability is converging to a particular distribution – the *stationary* distribution of the Markov chain. One might ask where the fly is after an *infinite* number of timesteps. That is, we are interested in the large t behaviour of

$$p(x_{t+1}) = \sum_{x_t} p(x_{t+1}|x_t)p(x_t). \quad (5.1.17)$$

At convergence $p(x_{t+1}) = p(x_t)$. Writing \mathbf{p} for the vector describing the stationary distribution, this means

$$\mathbf{p} = \mathbf{M}\mathbf{p}. \quad (5.1.18)$$

In other words, \mathbf{p} is the eigenvector of \mathbf{M} with eigenvalue 1 [134]. Computing this numerically, the stationary distribution is $(0.5435, 0.3261, 0.1304)$. Note that software packages usually return eigenvectors with $\sum_i e_i^2 = 1$ – the unit eigenvector therefore will usually require normalisation to make this a probability with $\sum_i e_i = 1$.

5.1.2 The sum-product algorithm on factor graphs

Both Markov and belief networks can be represented using factor graphs. For this reason it is convenient to derive a marginal inference algorithm for FGs since this then applies to both Markov



Figure 5.2 For singly connected structures without branches, simple messages from one variable to its neighbour may be defined to form an efficient marginal inference scheme.

and belief networks. This is termed the sum-product algorithm since to compute marginals we need to distribute the sum over variable states over the product of factors. In other texts, this is also referred to as *belief propagation*.

Non-branching graphs: variable-to-variable messages

Consider the distribution

$$p(a, b, c, d) = f_1(a, b) f_2(b, c) f_3(c, d) f_4(d) \quad (5.1.19)$$

which has the factor graph represented in Fig. 5.2. To compute the marginal $p(a, b, c)$, since the variable d only occurs locally, we use

$$\begin{aligned} p(a, b, c) &= \sum_d p(a, b, c, d) = \sum_d f_1(a, b) f_2(b, c) f_3(c, d) f_4(d) \\ &= f_1(a, b) f_2(b, c) \underbrace{\sum_d f_3(c, d) f_4(d)}_{\mu_{d \rightarrow c}(c)}. \end{aligned} \quad (5.1.20)$$

Here $\mu_{d \rightarrow c}(c)$ defines a message from node d to node c and is a function of the variable c . Similarly,

$$p(a, b) = \sum_c p(a, b, c) = f_1(a, b) \underbrace{\sum_c f_2(b, c) \mu_{d \rightarrow c}(c)}_{\mu_{c \rightarrow b}(b)}. \quad (5.1.21)$$

Hence

$$\mu_{c \rightarrow b}(b) = \sum_c f_2(b, c) \mu_{d \rightarrow c}(c). \quad (5.1.22)$$

It is clear how one can recurse this definition of messages so that for a chain of n variables the marginal of the first node can be computed in time linear in n . The term $\mu_{c \rightarrow b}(b)$ can be interpreted as carrying marginal information from the graph beyond c . For simple linear structures with no branching, messages from variables to variables are sufficient. However, as we will see below in more general structures with branching, it is useful to consider two types of messages, namely those from variables to factors and vice versa.

General singly connected factor graphs

The slightly more complex example,

$$p(a|b)p(b|c, d)p(c)p(d)p(e|d) \quad (5.1.23)$$

has the factor graph depicted in Fig. 5.3

$$f_1(a, b) f_2(b, c, d) f_3(c) f_4(d, e) f_5(d). \quad (5.1.24)$$

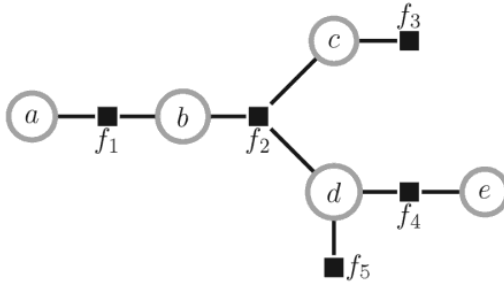


Figure 5.3 For a branching singly connected graph, it is useful to define messages from both factors to variables, and variables to factors.

The marginal $p(a, b)$ can be represented by an amputated graph with a message, since

$$p(a, b) = f_1(a, b) \underbrace{\sum_{c,d} f_2(b, c, d) f_3(c) f_5(d) \sum_e f_4(d, e)}_{\mu_{f_2 \rightarrow b}(b)} \quad (5.1.25)$$

where $\mu_{f_2 \rightarrow b}(b)$ is a message from a factor to a variable. This message can be constructed from messages arriving from the two branches through c and d , namely

$$\mu_{f_2 \rightarrow b}(b) = \sum_{c,d} f_2(b, c, d) \underbrace{f_3(c)}_{\mu_{c \rightarrow f_2}(c)} \underbrace{\sum_e f_4(d, e)}_{\mu_{d \rightarrow f_2}(d)}. \quad (5.1.26)$$

Similarly, we can interpret

$$\mu_{d \rightarrow f_2}(d) = \underbrace{f_5(d)}_{\mu_{f_5 \rightarrow d}(d)} \underbrace{\sum_e f_4(d, e)}_{\mu_{f_4 \rightarrow d}(d)}. \quad (5.1.27)$$

To complete the interpretation we identify $\mu_{c \rightarrow f_2}(c) \equiv \mu_{f_3 \rightarrow c}(c)$. In a non-branching link, one can more simply use a variable-to-variable message. To compute the marginal $p(a)$, we then have

$$p(a) = \underbrace{\sum_b f_1(a, b) \mu_{f_2 \rightarrow b}(b)}_{\mu_{f_1 \rightarrow a}(a)}. \quad (5.1.28)$$

For consistency of interpretation, one also can view the above as

$$\mu_{f_1 \rightarrow a}(a) = \sum_b f_1(a, b) \underbrace{\mu_{f_2 \rightarrow b}(b)}_{\mu_{b \rightarrow f_1}(b)}. \quad (5.1.29)$$

We can now see how a message from a factor to a node is formed from summing the product of incoming node-to-factor messages. Similarly, a message from a node to a factor is given by the product of incoming factor-to-node messages.

A convenience of this approach is that the messages can be reused to evaluate other marginal inferences. For example, it is clear that $p(b)$ is given by

$$p(b) = \underbrace{\sum_a f_1(a, b) \mu_{f_2 \rightarrow b}(b)}_{\mu_{f_1 \rightarrow b}(b)}. \quad (5.1.30)$$

If we additionally desire $p(c)$, we need to define the message from f_2 to c ,

$$\mu_{f_2 \rightarrow c}(c) = \sum_{b,d} f_2(b, c, d) \mu_{b \rightarrow f_2}(b) \mu_{d \rightarrow f_2}(d) \quad (5.1.31)$$

where $\mu_{b \rightarrow f_2}(b) \equiv \mu_{f_1 \rightarrow b}(b)$. This demonstrates the reuse of already computed message from d to f_2 to compute the marginal $p(c)$.

Definition 5.1 Message schedule A message schedule is a specified sequence of message updates. A valid schedule is that a message can be sent from a node only when that node has received all requisite messages from its neighbours. In general, there is more than one valid updating schedule.

Sum-product algorithm

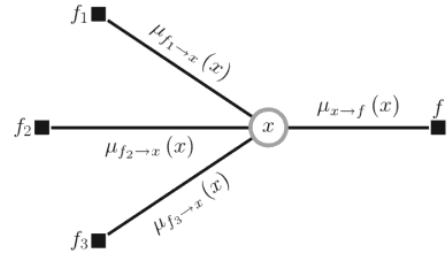
The sum-product algorithm is described below in which messages are updated as a function of incoming messages. One then proceeds by computing the messages in a schedule that allows the computation of a new message based on previously computed messages, until all messages from all factors to variables and vice versa have been computed.

Procedure 5.1 (Sum-product messages on factor graphs) Given a distribution defined as a product on subsets of the variables, $p(\mathcal{X}) = \frac{1}{Z} \prod_f \phi_f(\mathcal{X}_f)$, provided the factor graph is singly connected we can carry out summation over the variables efficiently.

Initialisation Messages from leaf node factors are initialised to the factor. Messages from leaf variable nodes are set to unity.

Variable-to-factor message

$$\mu_{x \rightarrow f}(x) = \prod_{g \in \{ne(x) \setminus f\}} \mu_{g \rightarrow x}(x)$$



Factor-to-variable message

$$\mu_{f \rightarrow x}(x) = \sum_{\mathcal{X}_f \setminus x} \phi_f(\mathcal{X}_f) \prod_{y \in \{ne(f) \setminus x\}} \mu_{y \rightarrow f}(y)$$

We write $\sum_{\mathcal{X}_f \setminus x}$ to denote summation over all states in the set of variables $\mathcal{X}_f \setminus x$.

