Bayesian Reasoning and Machine Learning

Notation List

$\mathcal{V}$	a calligraphic symbol typically denotes a set of random variables 7
$\text{dom}(x)$	Domain of a variable 7
$x = \mathsf{x}$	The variable x is in the state x 7
$p(x = \mathsf{tr})$	probability of event/variable x being in the state true 7
$p(x = \mathsf{fa})$	probability of event/variable x being in the state false 7
$p(x, y)$	probability of x and y 8
$p(x \cap y)$	probability of x and y 8
$p(x \cup y)$	probability of x or y 8
$p(x\|y)$	The probability of x conditioned on y 8
$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \| \mathcal{Z}$	Variables $\mathcal{X}$ are independent of variables $\mathcal{Y}$ conditioned on variables $\mathcal{Z}$. 11
$\mathcal{X} \top\!\!\!\top \mathcal{Y} \| \mathcal{Z}$	Variables $\mathcal{X}$ are dependent on variables $\mathcal{Y}$ conditioned on variables $\mathcal{Z}$ .. 11
$\int_x f(x)$	For continuous variables this is shorthand for $\int f(x)dx$ and for discrete variables means summation over the states of x, $\sum_x f(x)$ 18
$\mathbb{I}[S]$	Indicator : has value 1 if the statement S is true, 0 otherwise 19
$\text{pa}(x)$	The parents of node x 30
$\text{ch}(x)$	The children of node x 30
$\text{ne}(x)$	Neighbours of node x 30
$\dim(x)$	For a discrete variable x, this denotes the number of states x can take .. 40
$\langle f(x) \rangle_{p(x)}$	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
$\dim \mathbf{x}$	The dimension of the vector/matrix $\mathbf{x}$ 183
$\sharp(x = s, y = t)$	The number of times x is in state s and y in state t simultaneously ... 209
$\sharp_y^x$	The number of times variable x is in state y 290
$\mathcal{D}$	Dataset 303
n	Data index 303
N	Number of dataset training points 303
$\mathbf{S}$	Sample Covariance matrix 327
$\sigma(x)$	The logistic sigmoid $1/(1 + \exp(-x))$ 367
$\text{erf}(x)$	The (Gaussian) error function 367
$x_{a:b}$	$x_a, x_{a+1}, \ldots, x_b$ 469
$i \sim j$	The set of unique neighbouring edges on a graph 601
$\mathbf{I}_m$	The $m \times m$ identity matrix 623

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 enhance 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 BRMLtoolbox.

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 in 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.

Accompanying code

The BRMLtoolbox is provided to help readers see how mathematical models translate into actual MATLAB code. There are a large number of demos that a lecturer may wish to use or adapt to help illustrate the material. In addition many of the exercises make use of the code, helping the reader gain confidence in the concepts and their application. Along with complete routines for many Machine Learning methods, the philosophy is to provide low level routines whose composition intuitively follows the mathematical description of the algorithm. In this way students may easily match the mathematics with the corresponding algorithmic implementation.

Website

The BRMLtoolbox along with an electronic version of the book is available from

`www.cs.ucl.ac.uk/staff/D.Barber/brml`

Instructors seeking solutions to the exercises can find information at the website, along with additional teaching materials.

Other books in this area

The literature on Machine Learning is vast with much relevant literature also contained in statistics, engineering and other physical sciences. A small list of more specialised books that may be referred to for deeper treatments of specific topics is:

- Graphical models
  - *Graphical models* by S. Lauritzen, Oxford University Press, 1996.
  - *Bayesian Networks and Decision Graphs* by F. Jensen and T. D. Nielsen, Springer Verlag, 2007.
  - *Probabilistic Networks and Expert Systems* by R. G. Cowell, A. P. Dawid, S. L. Lauritzen and D. J. Spiegelhalter, Springer Verlag, 1999.
  - *Probabilistic Reasoning in Intelligent Systems* by J. Pearl, Morgan Kaufmann, 1988.
  - *Graphical Models in Applied Multivariate Statistics* by J. Whittaker, Wiley, 1990.
  - *Probabilistic Graphical Models: Principles and Techniques* by D. Koller and N. Friedman, MIT Press, 2009.
- Machine Learning and Information Processing
  - *Information Theory, Inference and Learning Algorithms* by D. J. C. MacKay, Cambridge University Press, 2003.
  - *Pattern Recognition and Machine Learning* by C. M. Bishop, Springer Verlag, 2006.
  - *An Introduction To Support Vector Machines*, N. Cristianini and J. Shawe-Taylor, Cambridge University Press, 2000.
  - *Gaussian Processes for Machine Learning* by C. E. Rasmussen and C. K. I. Williams, MIT press, 2006.

Acknowledgements

Many people have helped this book along the way either in terms of reading, feedback, general insights, allowing me to present their work, or just plain motivation. Amongst these I would like to thank Dan Cornford, Massimiliano Pontil, Mark Herbster, John Shawe-Taylor, Vladimir Kolmogorov, Yuri Boykov, Tom Minka, Simon Prince, Silvia Chiappa, Bertrand Mesot, Robert Cowell, Ali Taylan Cemgil, David Blei, Jeff Bilmes, David Cohn, David Page, Peter Sollich, Chris Williams, Marc Toussaint, Amos Storkey, Zakria Hussain, Le Chen, Serafín Moral, Milan Studený, Luc De Raedt, Tristan Fletcher, Chris Vryonides, Yannis Haralambous (and particularly for his help with example 1.5), Tom Furmston, Ed Challis and Chris Bracegirdle. I would also like to thank the many students that have helped improve the material during lectures over the years. I'm particularly grateful to Taylan Cemgil for allowing his GraphLayout package to be bundled with the BRMLtoolbox. ++

The staff at Cambridge University Press have been a delight to work with and I would especially like to thank Heather Bergman for her initial endeavors and the wonderful Diana Gillooly for her continued enthusiasm.

A heartfelt thankyou to my parents and sister – I hope this small token will make them proud. I'm also fortunate to be able to acknowledge the support and generosity of friends throughout. Finally, I'd like to thank Silvia who made it all worthwhile.

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

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

Potential manipulation

```
condpot                 - Return a potential conditioned on another variable
changevar               - Change variable names in a potential
dag                     - Return the adjacency matrix (zeros on diagonal) for a Belief Network
deltapot                - A delta function potential
disptable               - Print the table of a potential
divpots                 - Divide potential pota by potb
drawFG                  - Draw the Factor Graph A
drawID                  - plot an Influence Diagram
drawJTree               - plot a Junction Tree
drawNet                 - plot network
evalpot                 - Evaluate the table of a potential when variables are set
exppot                  - exponential of a potential
eyepot                  - Return a unit potential
grouppot                - Form a potential based on grouping variables together
groupstate              - Find the state of the group variables corresponding to a given ungrouped state
logpot                  - logarithm of the potential
markov                  - Return a symmetric adjacency matrix of Markov Network in pot
maxpot                  - Maximise a potential over variables
maxsumpot               - Maximise or Sum a potential over variables
multpots                - Multiply potentials into a single potential
numstates               - Number of states of the variables in a potential
```

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

```
absorb                  - Update potentials in absorption message passing on a Junction Tree
absorption              - Perform full round of absorption on a Junction Tree
absorptionID            - Perform full round of absorption on an Influence Diagram
ancestralsample         - Ancestral sampling from a Belief Network
binaryMRFmap            - get the MAP assignment for a binary MRF with positive W
bucketelim              - Bucket Elimination on a set of potentials
condindep               - Conditional Independence check using graph of variable interactions
condindepEmp            - Compute the empirical log Bayes Factor and MI for independence/dependence
condindepPot            - Numerical conditional independence measure
condMI                  - conditional mutual information I(x,y|z) of a potential.
FactorConnectingVariable - Factor nodes connecting to a set of variables
FactorGraph             - Returns a Factor Graph adjacency matrix based on potentials
IDvars                  - probability and decision variables from a partial order
jtassignpot             - Assign potentials to cliques in a Junction Tree
jtree                   - Setup a Junction Tree based on a set of potentials
jtreeID                 - Setup a Junction Tree based on an Influence Diagram
LoopyBP                 - loopy Belief Propagation using sum-product algorithm
MaxFlow                 - Ford Fulkerson max flow - min cut algorithm (breadth first search)
maxNpot                 - Find the N most probable values and states in a potential
maxNprodFG              - N-Max-Product algorithm on a Factor Graph (Returns the Nmax most probable States)
maxprodFG               - Max-Product algorithm on a Factor Graph
MDPemDeterministicPolicy - Solve MDP using EM with deterministic policy
MDPsolve                - Solve a Markov Decision Process
MesstoFact              - Returns the message numbers that connect into factor potential
metropolis              - Metropolis sample
mostprobablepath        - Find the most probable path in a Markov Chain
mostprobablepathmult    - Find the all source all sink most probable paths in a Markov Chain
sumprodFG               - Sum-Product algorithm on a Factor Graph represented by A
```

Specific Models

```
ARlds                   - Learn AR coefficients using a Linear Dynamical System
ARtrain                 - Fit autoregressive (AR) coefficients of order L to v.
BayesLinReg             - Bayesian Linear Regression training using basis functions phi(x)
BayesLogRegressionRVM   - Bayesian Logistic Regression with the Relevance Vector Machine
CanonVar                - Canonical Variates (no post rotation of variates)
cca                     - canonical correlation analysis
covfnGE                 - Gamma Exponential Covariance Function
EMbeliefnet             - train a Belief Network using Expectation Maximisation
EMminimizeKL            - MDP deterministic policy solver. Finds optimal actions
EMqTranMarginal         - EM marginal transition in MDP
EMqUtilMarginal         - Returns term proportional to the q marginal for the utility term
EMTotalBetaMessage      - backward information needed to solve the MDP process using message passing
EMvalueTable            - MDP solver calculates the value function of the MDP with the current policy
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 Autoregressive HMM
HMMem                    - EM algorithm for HMM
HMMforward               - HMM Forward Pass
HMMforwardSAR            - Switching Autoregressive HMM with switches updated only every Tskip timesteps
HMMgamma                 - HMM Posterior smoothing using the Rauch-Tung-Striebel correction method
HMMsmooth                - Smoothing for a Hidden Markov Model (HMM)
HMMsmoothSAR             - Switching Autoregressive HMM smoothing
HMMviterbi               - Viterbi most likely joint hidden state of a 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 (BATCH VERSION)
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~Beta(a,b), y~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(logp);
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
hinton                   - Plot a Hinton diagram
ind2subv                 - Subscript vector from linear index
ismember_sorted          - True for member of sorted set
lengthcell               - Length of each cell entry
logdet                   - Log determinant of a positive definite matrix computed in a numerically stable manner
logeps                   - log(x+eps)
logGaussGamma            - unnormalised log of the Gauss-Gamma distribution
logsumexp                - Compute log(sum(exp(a).*b)) valid for large a
logZdirichlet            - Log Normalisation constant of a Dirichlet distribution with parameter u
majority                 - Return majority values in each column on a matrix
maxarray                 - Maximise a multi-dimensional array over a set of dimensions
maxNarray                - Find the highest values and states of an array over a set of dimensions
```

```
mix2mix                 - Fit a mixture of Gaussians with another mixture of Gaussians
mvrandn                 - Samples from a multi-variate Normal(Gaussian) distribution
mygamrnd                - Gamma random variate generator
mynanmean               - mean of values that are not nan
mynansum                - sum of values that are not nan
mynchoosek              - binomial coefficient v choose k
myones                  - same as ones(x), but if x is a scalar, interprets as ones([x 1])
myrand                  - same as rand(x) but if x is a scalar interprets as rand([x 1])
myzeros                 - same as zeros(x) but if x is a scalar interprets as zeros([x 1])
normp                   - Make a normalised distribution from an array
randgen                 - Generates discrete random variables given the pdf
replace                 - Replace instances of a value with another value
sigma                   - 1./(1+exp(-x))
sigmoid                 - 1./(1+exp(-beta*x))
sqdist                  - Square distance between vectors in x and y
subv2ind                - Linear index from subscript vector.
sumlog                  - sum(log(x)) with a cutoff at 10e-200
```

Miscellaneous

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

Contents

Part I

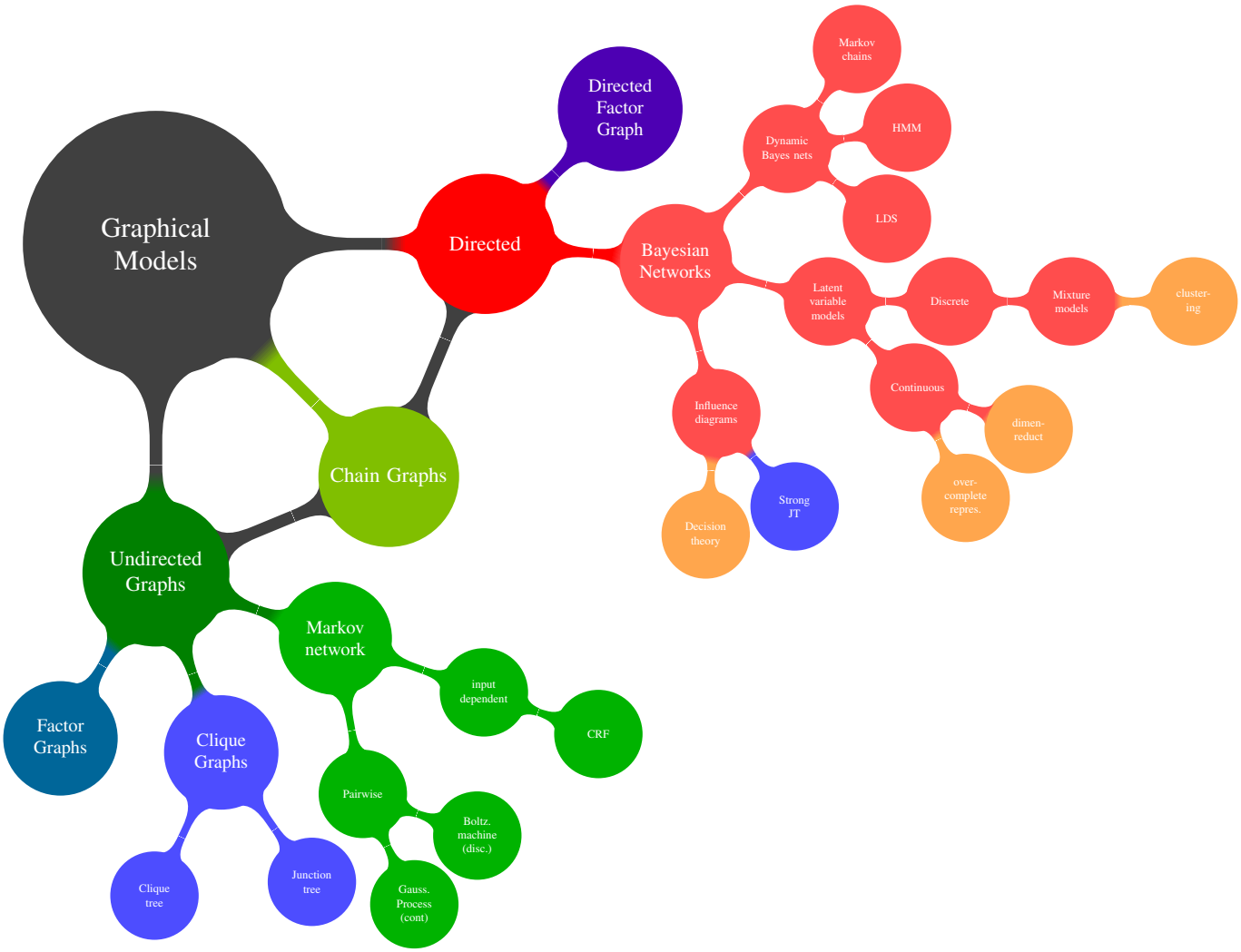
Inference in Probabilistic Models

Introduction to Part I

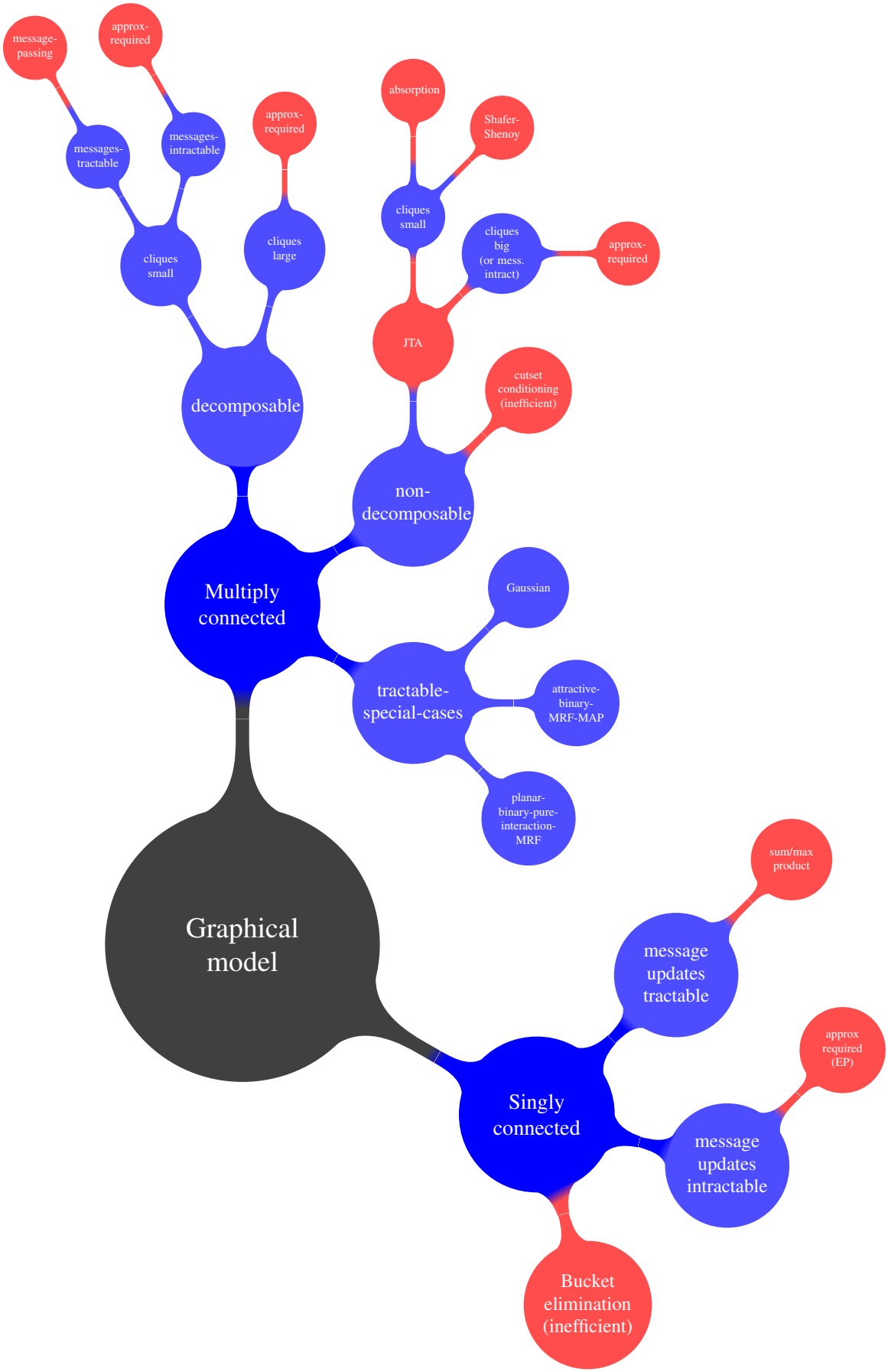
Probabilistic models explicitly take into account uncertainty and deal with our imperfect knowledge of the world. Such models are of fundamental significance in Machine Learning since our understanding of the world will always be limited by our observations and understanding. We will focus initially on using probabilistic models as a kind of expert system.

In Part I, we assume that the model is fully specified. That is, given a model of the environment, how can we use it to answer questions of interest. We will relate the complexity of inferring quantities of interest to the structure of the graph describing the model. In addition, we will describe operations in terms of manipulations on the corresponding graphs. As we will see, provided the graphs are simple tree-like structures, most quantities of interest can be computed efficiently.

Part I deals with manipulating mainly discrete variable distributions and forms the background to all the later material in the book.



Some members of the graphical models family and their uses. Nodes further from the Graphical Models root node are loosely speaking specialised versions of their parents. We discuss many of these models in Part I, although some of the more specialised models are deferred to later Parts of the book.



Graphical models and associated (marginal) inference methods. Specific inference methods are highlighted in red. Loosely speaking, provided the graph corresponding to the model is singly-connected most of the standard (marginal) inference methods are tractable. Multiply-connected graphs are generally more problematic, although there are special cases which remain tractable.

CHAPTER 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 $\mathrm{dom}(x)$, and denotes the states x can take. States will typically be represented using sans-serif font. For example, for a coin c, $\mathrm{dom}(c) = \{\mathsf{heads}, \mathsf{tails}\}$ and $p(c = \mathsf{heads})$ represents the probability that variable c is in state heads. The meaning of $p(\mathsf{state})$ 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 $p(\mathsf{heads})$ is clear from the context, being shorthand for $p(c = \mathsf{heads})$. When summing over a variable $\sum_x f(x)$, the interpretation is that all states of x are included, *i.e.* $\sum_x f(x) \equiv \sum_{\mathsf{s} \in \mathrm{dom}(x)} f(x = \mathsf{s})$. Given a variable, x, its domain $\mathrm{dom}(x)$ 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 6 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, $p(\textit{The coin is tails})$ can be interpreted as $p(\textit{The coin is tails} = \mathsf{true})$. We use the shorthand $p(x = \mathsf{tr})$ for the probability of event/variable x being in the state true and $p(x = \mathsf{fa})$ for the probability of variable x being in the state false.

Definition 1.1 (Rules of Probability for Discrete Variables)**.**

The probability $p(x = \mathsf{x})$ of variable x being in state x is represented by a value between 0 and 1. $p(x = \mathsf{x}) = 1$ means that we are certain x is in state x. Conversely, $p(x = \mathsf{x}) = 0$ 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:

$$\sum_{\mathsf{x} \in \mathrm{dom}(x)} p(x = \mathsf{x}) = 1 \tag{1.1.1}$$

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

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

Or, more generally, we can write

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

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

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

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

$$p(x) = \sum_y p(x, y) \tag{1.1.5}$$

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

$$p(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \sum_{x_i} p(x_1, \ldots, x_n) \tag{1.1.6}$$

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

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

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

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

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

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

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

$$p(a \leq x \leq b) = \int_a^b f(x)dx \tag{1.1.10}$$

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 10000 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[159].

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 $p(\textit{region } i) = 1/20$. 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 :

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

giving the intuitive result. An important point to clarify is that $p(A = \mathsf{a}|B = \mathsf{b})$ should not be interpreted as 'Given the event $B = \mathsf{b}$ has occurred, $p(A = \mathsf{a}|B = \mathsf{b})$ is the probability of the event $A = \mathsf{a}$ occurring'. In most contexts, no such explicit temporal causality is implied[1] and the correct interpretation should be ' $p(A = \mathsf{a}|B = \mathsf{b})$ is the probability of A being in state a under the constraint that B is in state b'.

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

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

$$p(x, y) = p(x)p(y) \tag{1.1.11}$$

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

$$p(x|y) = p(x) \Leftrightarrow p(y|x) = p(y) \tag{1.1.12}$$

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

$$p(x, y) = kf(x)g(y) \tag{1.1.13}$$

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 $\text{dom}(x) = \text{dom}(y) = \{1, \ldots, 7\}$. 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 randomly select a male at random, Bob. Before phoning Bob and asking him, what does knowing Alice's birth day 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 and y are always both in a certain joint state:

$$p(x = \mathsf{a}, y = 1) = 1, \quad p(x = \mathsf{a}, y = 2) = 0, \quad p(x = \mathsf{b}, y = 2) = 0, \quad p(x = \mathsf{b}, y = 1) = 0$$

[1] We will discuss issues related to causality further in section(3.4).

Are x and y dependent? The reader may show that $p(x = \mathsf{a}) = 1$, $p(x = \mathsf{b}) = 0$ 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.

Similarly, it's tempting to think that if a and b are dependent, and b and c are dependent, then a and c must be dependent:

$$\{a \not\perp\!\!\!\perp b, b \not\perp\!\!\!\perp c\} \Rightarrow a \not\perp\!\!\!\perp c \tag{1.1.20}$$

However, this also does not follow. We give an explicit numerical example in exercise(3.17).

Finally, note that conditional independence $x \perp\!\!\!\perp y | z$ does not imply marginal independence $x \perp\!\!\!\perp y$. See also exercise(3.20).

1.1.2 Probability Tables

Based on the populations 60776238, 5116900 and 2980700 of England (E), Scotland (S) and Wales (W), the a priori probability that a randomly selected person from the combined three countries would live in England, Scotland or Wales, is approximately 0.88, 0.08 and 0.04 respectively. We can write this as a vector (or probability table) :

$$\begin{pmatrix} p(Cnt = \mathsf{E}) \\ p(Cnt = \mathsf{S}) \\ p(Cnt = \mathsf{W}) \end{pmatrix} = \begin{pmatrix} 0.88 \\ 0.08 \\ 0.04 \end{pmatrix} \tag{1.1.21}$$

whose component values sum to 1. The ordering of the components in this vector is arbitrary, as long as it is consistently applied.

For the sake of simplicity, we assume that only three Mother Tongue languages exist : English (Eng), Scottish (Scot) and Welsh (Wel), with conditional probabilities given the country of residence, England (E), Scotland (S) and Wales (W). We write a (fictitious) conditional probability table

$$\begin{array}{lll} p(MT = \mathsf{Eng}|Cnt = \mathsf{E}) = 0.95 & p(MT = \mathsf{Eng}|Cnt = \mathsf{S}) = 0.7 & p(MT = \mathsf{Eng}|Cnt = \mathsf{W}) = 0.6 \\ p(MT = \mathsf{Scot}|Cnt = \mathsf{E}) = 0.04 & p(MT = \mathsf{Scot}|Cnt = \mathsf{S}) = 0.3 & p(MT = \mathsf{Scot}|Cnt = \mathsf{W}) = 0.0 \\ p(MT = \mathsf{Wel}|Cnt = \mathsf{E}) = 0.01 & p(MT = \mathsf{Wel}|Cnt = \mathsf{S}) = 0.0 & p(MT = \mathsf{Wel}|Cnt = \mathsf{W}) = 0.4 \end{array} \tag{1.1.22}$$

From this we can form a joint distribution $p(Cnt, MT) = p(MT|Cnt)p(Cnt)$. This could be written as a 3×3 matrix with columns indexed by country and rows indexed by Mother Tongue:

$$\begin{pmatrix} 0.95 \times 0.88 & 0.7 \times 0.08 & 0.6 \times 0.04 \\ 0.04 \times 0.88 & 0.3 \times 0.08 & 0.0 \times 0.04 \\ 0.01 \times 0.88 & 0.0 \times 0.08 & 0.4 \times 0.04 \end{pmatrix} = \begin{pmatrix} 0.836 & 0.056 & 0.024 \\ 0.0352 & 0.024 & 0 \\ 0.0088 & 0 & 0.016 \end{pmatrix} \tag{1.1.23}$$

The joint distribution contains all the information about the model of this environment. By summing within each column of this table, we have the marginal $p(Cnt)$. Summing within each row gives the marginal $p(MT)$. Similarly, one could easily infer $p(Cnt|MT) \propto p(MT|Cnt)p(Cnt)$ from this joint distribution by dividing an entry of equation (1.1.23) by its row sum. @@ @@

For joint distributions over a larger number of variables, $x_i, i = 1, \ldots, D$, with each variable x_i taking K_i states, the table describing the joint distribution is an array with $\prod_{i=1}^{D} K_i$ entries. Explicitly storing tables therefore requires space exponential in the number of variables, which rapidly becomes impractical for a large number of variables. We discuss how to deal with this issue in chapter(3) and chapter(4).

A probability distribution assigns a value to each of the joint states of the variables. For this reason, $p(T, J, R, S)$ is considered equivalent to $p(J, S, R, T)$ (or any such reordering of the variables), since in each case the joint setting of the variables is simply a different index to the same probability. This situation is more clear in the set theoretic notation $p(J \cap S \cap T \cap R)$. We abbreviate this set theoretic notation by using the commas – however, one should be careful not to confuse the use of this indexing type notation with functions $f(x, y)$ which are in general dependent on the variable order. Whilst the variables to the left of the conditioning bar may be written in any order, and equally those to the right of the conditioning bar may be written in any order, moving variables across the bar is not generally equivalent, so that $p(x_1|x_2) \neq p(x_2|x_1)$.

1.2 Probabilistic Reasoning

The central paradigm of probabilistic reasoning is to identify all relevant variables $x_1, \ldots, x_N$ in the environment, and make a probabilistic model $p(x_1, \ldots, x_N)$ of their interaction. Reasoning (inference) is then performed by introducing *evidence* that sets variables in known states, and subsequently computing probabilities of interest, conditioned on this evidence. The rules of probability, combined with Bayes' rule make for a complete reasoning system, one which includes traditional deductive logic as a special case[159]. In the examples below, the number of variables in the environment is very small. In chapter(3) we will discuss

reasoning in networks containing many variables, for which the graphical notations of chapter(2) will play a central role.

Example 1.2 (Hamburgers)**.** Consider the following fictitious scientific information: Doctors find that people with Kreuzfeld-Jacob disease (KJ) almost invariably ate hamburgers, thus $p(\textit{Hamburger Eater}|KJ) = 0.9$. The probability of an individual having KJ is currently rather low, about one in 100,000.

1. Assuming eating lots of hamburgers is rather widespread, say $p(\textit{Hamburger Eater}) = 0.5$, what is the probability that a hamburger eater will have Kreuzfeld-Jacob disease?

   This may be computed as

$$p(KJ\,|\textit{Hamburger Eater}) = \frac{p(\textit{Hamburger Eater}, KJ)}{p(\textit{Hamburger Eater})} = \frac{p(\textit{Hamburger Eater}|KJ)p(KJ)}{p(\textit{Hamburger Eater})} \tag{1.2.1}$$

$$= \frac{\frac{9}{10} \times \frac{1}{100000}}{\frac{1}{2}} = 1.8 \times 10^{-5} \tag{1.2.2}$$

2. If the fraction of people eating hamburgers was rather small, $p(\textit{Hamburger Eater}) = 0.001$, what is the probability that a regular hamburger eater will have Kreuzfeld-Jacob disease? Repeating the above calculation, this is given by

$$\frac{\frac{9}{10} \times \frac{1}{100000}}{\frac{1}{1000}} \approx 1/100 \tag{1.2.3}$$

   This is much higher than in scenario (1) since here we can be more sure that eating hamburgers is related to the illness.

Example 1.3 (Inspector Clouseau)**.** Inspector Clouseau arrives at the scene of a crime. The victim lies dead in the room alongside the possible murder weapon, a knife. The Butler (B) and Maid (M) are the inspector's main suspects and the inspector has a prior belief of 0.6 that the Butler is the murderer, and a prior belief of 0.2 that the Maid is the murderer. These beliefs are independent in the sense that $p(B, M) = p(B)p(M)$. (It is possible that both the Butler and the Maid murdered the victim or neither). The inspector's *prior* criminal knowledge can be formulated mathematically as follows:

$$\mathrm{dom}(B) = \mathrm{dom}(M) = \{\mathsf{murderer}, \mathsf{not\ murderer}\}, \mathrm{dom}(K) = \{\mathsf{knife\ used}, \mathsf{knife\ not\ used}\} \tag{1.2.4}$$

$$p(B = \mathsf{murderer}) = 0.6, \qquad p(M = \mathsf{murderer}) = 0.2 \tag{1.2.5}$$

$$\begin{array}{lll} p(\mathsf{knife\ used}|B = \mathsf{not\ murderer}, & M = \mathsf{not\ murderer}) & = 0.3 \\ p(\mathsf{knife\ used}|B = \mathsf{not\ murderer}, & M = \mathsf{murderer}) & = 0.2 \\ p(\mathsf{knife\ used}|B = \mathsf{murderer}, & M = \mathsf{not\ murderer}) & = 0.6 \\ p(\mathsf{knife\ used}|B = \mathsf{murderer}, & M = \mathsf{murderer}) & = 0.1 \end{array} \tag{1.2.6}$$

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,

$$p(B|K) = \sum_m p(B, m|K) = \sum_m \frac{p(B, m, K)}{p(K)} = \frac{\sum_m p(K|B, m)p(B, m)}{\sum_{m,b} p(K|b, m)p(b, m)} = \frac{p(B)\sum_m p(K|B, m)p(m)}{\sum_b p(b)\sum_m p(K|b, m)p(m)} \tag{1.2.7}$$

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 0.412. 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*

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

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

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

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

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

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 : Modus Ponens)**.** According to logic, the statements 'All apples are fruit' and 'All fruits grow on trees' lead to the conclusion that 'All apples grow on trees'. This kind of reasoning is a form of transitivity : from the statements $A \Rightarrow F$ and $F \Rightarrow T$ we can infer $A \Rightarrow T$.

To see how this might be deduced using Bayesian, we assume that 'All apples are fruit' corresponds to $p(F = \mathsf{tr}|A = \mathsf{tr}) = 1$ and 'All fruit grows on trees' corresponds to $p(T = \mathsf{tr}|F = \mathsf{tr}) = 1$. We then want to show that this implies $p(T = \mathsf{tr}|A = \mathsf{tr}) = 1$. Showing this is equivalent to showing $p(T = \mathsf{fa}|A = \mathsf{tr}) = 0$ which (assuming $p(A = \mathsf{tr}) > 0$) is in turn equivalent to showing that $p(T = \mathsf{fa}, A = \mathsf{tr}) = 0$. Consider

$$p(T = \mathsf{fa}, A = \mathsf{tr}) = p(T = \mathsf{fa}, A = \mathsf{tr}, F = \mathsf{tr}) + p(T = \mathsf{fa}, A = \mathsf{tr}, F = \mathsf{fa}) \tag{1.2.16}$$

We can show that both terms on the right are zero. First, consider

$$p(T = \mathsf{fa}, A = \mathsf{tr}, F = \mathsf{tr}) \leq p(T = \mathsf{fa}, F = \mathsf{tr}) = p(T = \mathsf{fa}|F = \mathsf{tr})p(F = \mathsf{tr}) \tag{1.2.17}$$

This is zero since, by assumption, $p(T = \mathsf{fa}|F = \mathsf{tr}) = 1 - p(T = \mathsf{tr}|F = \mathsf{tr}) = 1 - 1 = 0$. Similarly,

$$p(T = \mathsf{fa}, A = \mathsf{tr}, F = \mathsf{fa}) \leq p(A = \mathsf{tr}, F = \mathsf{fa}) = p(F = \mathsf{fa}|A = \mathsf{tr})p(A = \mathsf{tr}) \tag{1.2.18}$$

where again, by assumption, $p(F = \mathsf{fa}|A = \mathsf{tr}) = 0$.

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 $p(B = \mathsf{tr}|A = \mathsf{tr}) = 1$. Then we may infer

$$\begin{aligned} p(A = \mathsf{fa}|B = \mathsf{fa}) &= 1 - p(A = \mathsf{tr}|B = \mathsf{fa}) \\ &= 1 - \frac{p(B = \mathsf{fa}|A = \mathsf{tr})p(A = \mathsf{tr})}{p(B = \mathsf{fa}|A = \mathsf{tr})p(A = \mathsf{tr}) + p(B = \mathsf{fa}|A = \mathsf{fa})p(A = \mathsf{fa})} = 1 \end{aligned} \tag{1.2.19}$$

This follows since $p(B = \mathsf{fa}|A = \mathsf{tr}) = 1 - p(B = \mathsf{tr}|A = \mathsf{tr}) = 1 - 1 = 0$, 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 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, so that the gate stochastically outputs $C = 1$ depending on its inputs, with additionally $A \perp\!\!\!\perp B$ 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

$$\begin{aligned} p(A = 1, C = 0) &= \sum_B p(A = 1, B, C = 0) = \sum_B p(C = 0|A = 1, B)p(A = 1)p(B) \\ &= p(A = 1)\left(p(C = 0|A = 1, B = 0)p(B = 0) + p(C = 0|A = 1, B = 1)p(B = 1)\right) \\ &= 0.65 \times (0.2 \times 0.23 + 0.75 \times 0.77) = 0.405275 \end{aligned} \tag{1.2.20}$$

$$\begin{aligned} p(A = 0, C = 0) &= \sum_B p(A = 0, B, C = 0) = \sum_B p(C = 0|A = 0, B)p(A = 0)p(B) \\ &= p(A = 0)\left(p(C = 0|A = 0, B = 0)p(B = 0) + p(C = 0|A = 0, B = 1)p(B = 1)\right) \\ &= 0.35 \times (0.9 \times 0.23 + 0.01 \times 0.77) = 0.075145 \end{aligned}$$

Then

$$p(A = 1|C = 0) = \frac{p(A = 1, C = 0)}{p(A = 1, C = 0) + p(A = 0, C = 0)} = \frac{0.405275}{0.405275 + 0.075145} = 0.8436 \tag{1.2.21}$$

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

$$p(R_L = \text{not late}|L = \text{not late}) = 1, \qquad p(R_L = \text{late}|L = \text{late}) = 0 \tag{1.2.22}$$

The remaining two values are determined by normalisation and are

$$p(R_L = \text{late}|L = \text{not late}) = 0, \qquad p(R_L = \text{not late}|L = \text{late}) = 1 \tag{1.2.23}$$

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

$$\begin{aligned} p(L = \text{late}|R_L = \text{not late}) &= \frac{p(L = \text{late}, R_L = \text{not late})}{p(R_L = \text{not late})} \\ &= \frac{p(L = \text{late}, R_L = \text{not late})}{p(L = \text{late}, R_L = \text{not late}) + p(L = \text{not late}, R_L = \text{not late})} \end{aligned} \tag{1.2.24}$$

In the above

$$p(L = \text{late}, R_L = \text{not late}) = \underbrace{p(R_L = \text{not late}|L = \text{late})}_{=1} p(L = \text{late}) \tag{1.2.25}$$

and

$$p(L = \text{not late}, R_L = \text{not late}) = \underbrace{p(R_L = \text{not late}|L = \text{not late})}_{=1} p(L = \text{not late}) \tag{1.2.26}$$

Hence

$$p(L = \text{late}|R_L = \text{not late}) = \frac{p(L = \text{late})}{p(L = \text{late}) + p(L = \text{not late})} = p(L = \text{late}) \tag{1.2.27}$$

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.

$$p(R_S = \text{not late}|L = \text{not late}) = 1, \quad p(R_S = \text{late}|L = \text{late}) = 1 \tag{1.2.28}$$

The remaining two values are determined by normalisation and are

$$p(R_S = \text{late}|L = \text{not late}) = 0, \quad p(R_S = \text{not late}|L = \text{late}) = 0 \tag{1.2.29}$$

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

$$p(R_L, R_S, L) = p(R_L|L)p(R_S|L)p(L) \tag{1.2.30}$$

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

$$\begin{aligned} &p(L = \text{late}|R_L = \text{not late}, R_S = \text{late}) \\ &\qquad = \frac{1}{Z} p(R_S = \text{late}|L = \text{late}) p(R_L = \text{not late}|L = \text{late}) p(L = \text{late}) \end{aligned} \tag{1.2.31}$$

where the normalisation Z is given by

$$\begin{aligned} &p(R_S = \text{late}|L = \text{late}) p(R_L = \text{not late}|L = \text{late}) p(L = \text{late}) \\ &\qquad + p(R_S = \text{late}|L = \text{not late}) p(R_L = \text{not late}|L = \text{not late}) p(L = \text{not late}) \end{aligned} \tag{1.2.32}$$

Hence

$$p(L = \text{late}|R_L = \text{not late}, R_S = \text{late}) = \frac{1 \times 1 \times p(L = \text{late})}{1 \times 1 \times p(L = \text{late}) + 0 \times 1 \times p(L = \text{not late})} = 1 \tag{1.2.33}$$

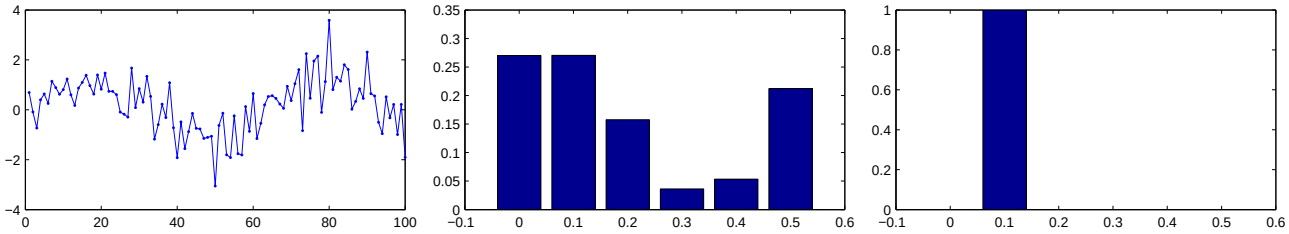
This result is also intuitive – Since Larry's mother knows that Sue always tells the truth, no matter what Larry says, she knows he was late.

Example 1.10 (Luke)**.** Luke has been told he's lucky and has won a prize in the lottery. There are 5 prizes available of value £10, £100, £1000, £10000, £1000000. The prior probabilities of winning these 5 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 £1000000?!'. 'I'm afraid not sir', is the response of the lottery phone operator. 'Did I win £10000?!' 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 $W = 2, \ldots, 5$ for the remaining prizes and $W = 0$ for no prize. We need to compute

$$\begin{aligned} p(W = 3|W \neq 5, W \neq 4, W \neq 0) &= \frac{p(W = 3, W \neq 5, W \neq 4, W \neq 0)}{p(W \neq 5, W \neq 4, W \neq 0)} \\ &= \frac{p(W = 3)}{p(W = 1 \text{ or } W = 2 \text{ or } W = 3)} = \frac{p_3}{p_1 + p_2 + p_3} \end{aligned} \tag{1.2.34}$$

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.



(a) (b) (c)

Figure 1.1: **(a)**: Noisy observations of displacements $x_1, \ldots, x_{100}$ for a pendulum. **(b)**: The prior belief on 5 possible values of θ. **(c)**: The posterior belief on θ.

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. Our interest is then the quantity

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int_\theta p(\mathcal{D}|\theta)p(\theta)} \tag{1.3.1}$$

This shows how from a forward or *generative model* $p(\mathcal{D}|\theta)$ of the dataset, and coupled with a *prior* belief $p(\theta)$ about which variable values are appropriate, we can infer the *posterior* distribution $p(\theta|\mathcal{D})$ of the variable in light of the observed data. The *most probable a posteriori* (*MAP*) setting is that which maximises the posterior, $\theta_* = \arg\max_\theta p(\theta|\mathcal{D})$. For a 'flat prior', $p(\theta)$ being a constant, not changing with θ, the MAP solution is equivalent to the *maximum likelihood*, namely that θ that maximises the likelihood $p(\mathcal{D}|\theta)$ of the model generating the observed data. We will return to a discussion of such summaries of the posterior and parameter learning in chapter(9).

This use of a generative model sits well with physical models of the world which typically postulate how to generate observed phenomena, assuming we know the model. For example, one might postulate how to generate a time-series of displacements for a swinging pendulum but with unknown mass, length and damping constant. Using this generative model, and given only the displacements, we could infer the unknown physical properties of the pendulum.

Example 1.11 (Pendulum)**.** As a prelude to scientific inference and the use of continuous variables, we consider an idealised pendulum for which x_t is the angular displacement of the pendulum at time t. Assuming that the measurements are independent, given the knowledge of the parameter of the problem, θ, we have that the likelihood of a sequence of observations $x_1, \ldots, x_T$ is given by

$$p(x_1, \ldots, x_T|\theta) = \prod_{t=1}^{T} p(x_t|\theta) \tag{1.3.2}$$

If the model is correct and our measurement of the displacements x is perfect, then the physical model is

$$x_t = \sin(\theta t) \tag{1.3.3}$$

where θ represents the unknown physical constants of the pendulum ($\sqrt{g/L}$, where g is the gravitational attraction and L the length of the pendulum). If, however, we assume that we have a rather poor instrument to measure the displacements, with a known variance of σ^2 (see chapter(8)), then

$$x_t = \sin(\theta t) + \epsilon_t \tag{1.3.4}$$

where ϵ_t is zero mean Gaussian noise with variance σ^2. We can also consider a set of possible parameters θ and place a prior $p(\theta)$ over them, expressing our prior belief (before seeing the measurements) in the appropriateness of the different values of θ. The posterior distribution is then given by

$$p(\theta|x_1,\ldots,x_T) \propto p(\theta)\prod_{t=1}^{T}\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{1}{2\sigma^2}(x_t-\sin(\theta t))^2} \tag{1.3.5}$$

Despite noisy measurements, the posterior over the assumed possible values for θ becomes strongly peaked for a large number of measurements, see fig(1.1).

1.3.1 Two dice : what were the individual scores?

Two fair dice are rolled. Someone tells you that the sum of the two scores is 9. What is the posterior distribution of the dice scores[2]?

The score of die a is denoted s_a with $\mathrm{dom}(s_a) = \{1,2,3,4,5,6\}$ and similarly for s_b. The three variables involved are then s_a, s_b and the total score, $t = s_a + s_b$. A model of these three variables naturally takes the form

$$p(t,s_a,s_b) = \underbrace{p(t|s_a,s_b)}_{\text{likelihood}}\underbrace{p(s_a,s_b)}_{\text{prior}} \tag{1.3.6}$$

The prior $p(s_a,s_b)$ is the joint probability of score s_a and score s_b without knowing anything else. Assuming no dependency in the rolling mechanism,

$$p(s_a,s_b) = p(s_a)p(s_b) \tag{1.3.7}$$

Since the dice are fair both $p(s_a)$ and $p(s_b)$ are uniform distributions, $p(s_a) = p(s_b) = 1/6$.

Here the likelihood term is

$$p(t|s_a,s_b) = \mathbb{I}\left[t = s_a + s_b\right] \tag{1.3.8}$$

which states that the total score is given by $s_a + s_b$. Here $\mathbb{I}\left[A\right]$ is the *indicator function* defined as $\mathbb{I}\left[A\right] = 1$ if the statement A is true and 0 otherwise.

Hence, our complete model is

$$p(t,s_a,s_b) = p(t|s_a,s_b)p(s_a)p(s_b) \tag{1.3.9}$$

where the terms on the right are explicitly defined.

The posterior is then given by,

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

where

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

$p(s_a)p(s_b)$:

	$s_a=1$	$s_a=2$	$s_a=3$	$s_a=4$	$s_a=5$	$s_a=6$
$s_b=1$	1/36	1/36	1/36	1/36	1/36	1/36
$s_b=2$	1/36	1/36	1/36	1/36	1/36	1/36
$s_b=3$	1/36	1/36	1/36	1/36	1/36	1/36
$s_b=4$	1/36	1/36	1/36	1/36	1/36	1/36
$s_b=5$	1/36	1/36	1/36	1/36	1/36	1/36
$s_b=6$	1/36	1/36	1/36	1/36	1/36	1/36

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

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

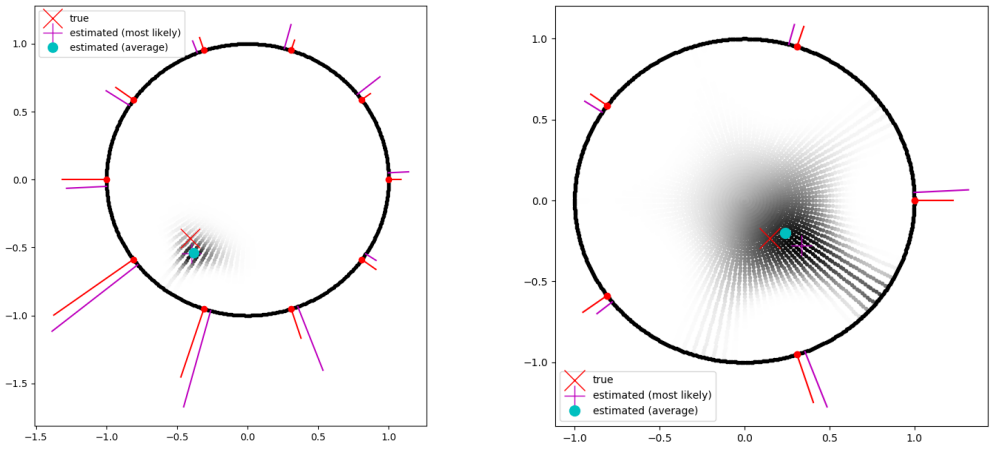
$p(t=9|s_a,s_b)p(s_a)p(s_b)$:

	$s_a=1$	$s_a=2$	$s_a=3$	$s_a=4$	$s_a=5$	$s_a=6$
$s_b=1$	0	0	0	0	0	0
$s_b=2$	0	0	0	0	0	0
$s_b=3$	0	0	0	0	0	1/36
$s_b=4$	0	0	0	0	1/36	0
$s_b=5$	0	0	0	1/36	0	0
$s_b=6$	0	0	1/36	0	0	0

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

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

[2]This example is due to Taylan Cemgil.



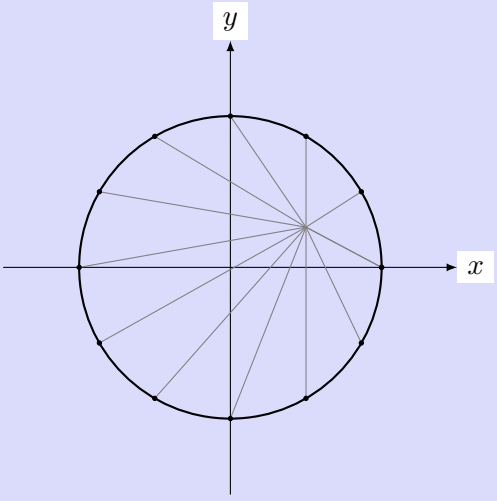
(a) (b)

Figure 1.2: The posterior distribution for the location of the explosion (darker corresponds to a higher probability). This is based on a spiral coordinate system (see earthquake.jl) with $\sigma = 0.2$. The red points are the locations of the surface sensors. The observed (noisy) measurements at each sensor are represented by the magenta lines, and the true (unknown) blast values are denoted by the red lines. (a) Using 10 sensors. (b) Using 5 sensors. Note how the uncertainty in the posterior is larger in the case of having fewer sensors.

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 4 non-zero elements, as shown.

Example 1.12 (Explosions)**.** We consider a modified and simplified form of Stuart Russell's 'earthquake/nuclear blast' detection problem [9]. We assume that there is an 'explosion event' somewhere within the earth and we wish to estimate where the explosion happened based on surface measurements of the explosion. For simplicity we assume a two dimensional earth.

There are N sensors evenly spread out on the surface of the earth with locations $(x_i, y_i), i = 1, \ldots, N$, indicated by the dots below:



The explosion happens at some (unknown) location (e_x, e_y) within the earth, from which a wave of energy propagates outwards and reaches the sensors on the surface. The blast signal that each sensor receives is

given by

$$\frac{1}{d_i^2 + 0.1}$$

where d_i^2 is the squared distance from the explosion to sensor i and

$$d_i^2 = (x_i - e_x)^2 + (y_i - e_y)^2$$

This means that the signal strength of the explosion decreases as the distance from the explosion to a sensor increases.

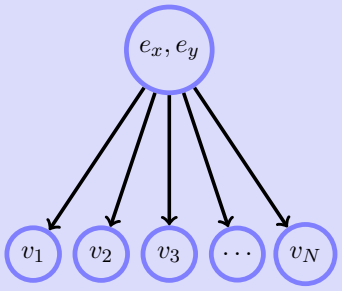
The sensors are not able to perfectly detect the strength of the signal and the signal is measured with Gaussian noise with a standard deviation of σ. This means that the observed value v_i at sensor i follows a Gaussian distribution:

$$p(v_i|d_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}\left(v_i - \frac{1}{d_i^2+0.1}\right)^2}$$

Assuming that the observed sensor values are independent (given the explosion location), our simple generative model is

$$p(v_1, \ldots, v_N, e_x, e_y) = p(e_x, e_y) \prod_{i=1}^{N} p(v_i|d_i)$$

which has the belief network description below (see chapter(3))



Given an observed set of values $v_1, \ldots, v_N$, our interest is then the posterior distribution

$$p(e_x, e_y|v_1, \ldots, v_N)$$

For a uniform prior $p(e_x, e_y) = \mathsf{const.}$, then

$$p(e_x, e_y|v_1, \ldots, v_N) \propto \prod_{i=1}^{N} p(v_i|d_i)$$

In fig(1.2) we plot a representation of the posterior and also plot the most likely point

$$\arg\max p(e_x, e_y|v_1, \ldots, v_N)$$

One can also extend the method to deal with multiple explosions, see fig(1.3) for a two-sources example.

1.4 Summary

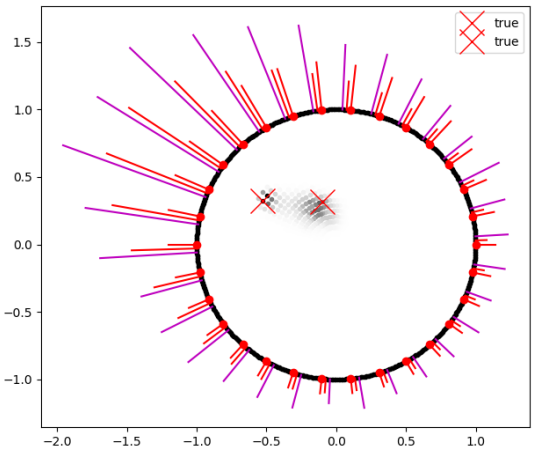


Figure 1.3: The posterior distribution for the location of two explosions (darker corresponds to a higher probability). This is based on a spiral coordinate system (see earthquake.jl) with $\sigma = 0.2$. The red points are the locations of the surface sensors. The observed (noisy) measurements at each sensor are represented by the magenta lines, and the two true (unknown) blast values are denoted by the red lines.

- 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 [293]. The interpretation of probability is contentious and we refer the reader to [159, 198, 194] 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`: Maximize 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

Exercise 1.1. *Prove*

$$p(x, y|z) = p(x|z)p(y|x, z) \tag{1.6.1}$$

and also

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

Exercise 1.2. *Prove the* Bonferroni inequality

$$p(a, b) \geq p(a) + p(b) - 1 \tag{1.6.3}$$

Exercise 1.3 (Adapted from [182])**.** *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* $p(box = 1) = p(box = 2) = 0.5$ *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?*

Exercise 1.4 (Adapted from [182])**.** *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?*

Exercise 1.5 (Adapted 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 police haul off the plane the first person for which the scanner tests positive. What is the probability that this person is a terrorist?

Exercise 1.6. *Consider three variable distributions which admit the factorisation*

$$p(a, b, c) = p(a|b)p(b|c)p(c) \tag{1.6.4}$$

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

Exercise 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.*

Exercise 1.8. *Prove*

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

Exercise 1.9. *Prove*

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

Exercise 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 8 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 Prof. Gott? Explain carefully your reasoning.*

Exercise 1.11. *Implement the soft XOR gate, example(1.7) using* BRMLtoolbox. *You may find* `condpot.m` *of use.*

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

Exercise 1.13. *Implement the two-dice example, section(1.3.1) using* BRMLtoolbox.

Exercise 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?*

Exercise 1.15. *In a test of 'psychometry' the car keys and wrist watches of 5 people are given to a medium. The medium then attempts to match the wrist watch 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 1 correct match?*

Exercise 1.16. *1. Show that for any function* f

$$\sum_x p(x|y)f(y) = f(y) \tag{1.6.7}$$

2. *Explain why, in general,*

$$\sum_x p(x|y)f(x,y) \neq \sum_x f(x,y) \tag{1.6.8}$$

Exercise 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 4 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?*

Exercise 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? Assume maximal uncertainty in the absence of any knowledge of the probabilities.* @@

Exercise 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.*

Exercise 1.20. *A game of Battleships is played on a 10×10 pixel grid. There are two 5-pixel length ships placed uniformly at random on the grid, subject to the constraints that (i) the ships cannot overlap and (ii) one ship is vertical and the other horizontal. After 10 unsuccessful 'misses' in locations $(1,10),(2,2),(3,8),(4,4),(5,6),(6,5),(7,4),(7,7),(9,2),(9,9)$ calculate which pixel has the highest probability of containing a ship. State this pixel and the value of the highest probability.* ++

Exercise 1.21. *A game of Battleships is played on a 8×8 grid. There are two 5-pixel length ships placed horizontally and two 5-pixel length ships placed vertically, subject to the same constraints as in the previous question. Given 'misses' in locations $(1,1),(2,2)$ and a 'hit' in location $(5,5)$, which pixel most likely contains a ship and what is that probability?* ++

Exercise 1.22. *We consider an extension of the explosion example. In this extension there are two explosions at locations s_1 and s_2 and the observed value at sensor i is* ++

$$v_i = \frac{1}{d_i^2(1)+0.1} + \frac{1}{d_i^2(2)+0.1} + \sigma\epsilon_i$$

where $d_i(1)$, $d_i(2)$ is the distance from explosion $1,2$ to the sensor respectively; σ is the standard deviation of the Gaussian sensor noise and the noise ϵ_i is drawn from a zero mean unit variance Gaussian independently for each sensor. The data in the file `EarthquakeExerciseData.txt` represents the observed sensor values v_i and the coordinate system setup is given in `earthquakeExerciseSetup.jl`. Assuming that the prior locations of the explosions are independent and uniform (according to the spiral coordinate system):

1. *Calculate the posterior $p(s_1|\mathbf{v})$ and draw an image similar to fig(1.3) that visualises the posterior.*

2. *Writing $\mathcal{H}_2$ for the hypothesis that there are two explosions and $\mathcal{H}_1$ for the hypothesis that there is only one explosion, report the value of $\log p(\mathbf{v}|\mathcal{H}_2) - \log p(\mathbf{v}|\mathcal{H}_1)$.*

3. *Assuming we have no prior preference, that is* $p(\mathcal{H}_1) = p(\mathcal{H}_2) = \mathsf{const.}$*, explain why* $\log p(\mathbf{v}|\mathcal{H}_2) - \log p(\mathbf{v}|\mathcal{H}_1)$ *relates to the probability that there are two explosions compared to only 1.*

4. *If we assumed that there were* K *explosions, explain the computational complexity of calculating* $\log p(\mathbf{v}|\mathcal{H}_K)$*.*

++ **Exercise 1.23.** *Similar to the above exercise, a different kind of explosion sensor measures the mean of two incoming explosions, so that*

$$v_i = \frac{0.5}{d_i^2(1) + 0.1} + \frac{0.5}{d_i^2(2) + 0.1} + \sigma\epsilon_i$$

The observed sensor data is given in `EarthquakeExerciseMeanData.txt`*.*

1. *Calculate the posterior* $p(s_1|\mathbf{v})$ *and draw an image similar to fig(1.3) that visualises the posterior.*

2. *Writing* $\mathcal{H}_2$ *for the hypothesis that there are two explosions and* $\mathcal{H}_1$ *for the hypothesis that there is only one explosion, report the value of* $\log p(\mathbf{v}|\mathcal{H}_2) - \log p(\mathbf{v}|\mathcal{H}_1)$*.*

3. *Explain why in this case it will generally be more difficult (compared to the previous exercise) to accurately estimate the location of the two explosions and determine the number of explosions.*

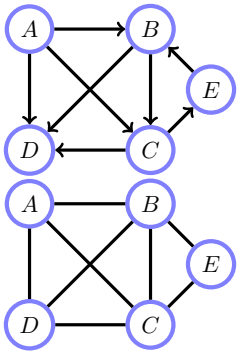
CHAPTER 2

Basic Graph Concepts

Often we have good reason to believe that one event affects another, or conversely that some events are independent. Incorporating such knowledge can produce models that are better specified and computationally more efficient. Graphs describe how objects are linked and provide a convenient picture for describing related objects. We will ultimately introduce a graph structure among the variables of a probabilistic model to produce a 'graphical model' that captures relations among the variables as well as their uncertainties. In this chapter, we introduce the required basic concepts from graph theory.

2.1 Graphs

Definition 2.1 (Graph)**.** A *graph* G consists of nodes (also called vertices) and edges (also called links) between the nodes. Edges may be directed (they have an arrow in a single direction) or undirected. Edges can also have associated weights. A graph with all edges directed is called a *directed graph*, and one with all edges undirected is called an *undirected graph*.



An directed graph G consists of directed edges between nodes.

An undirected graph G consists of undirected edges between nodes.

Graphs with edge weights are often used to model networks and flows along 'pipes', or distances between cities, where each node represents a city. We will also make use of these concepts in chapter(5) and chapter(28). Our main use of graphs though will be to endow them with a probabilistic interpretation and we develop a connection between directed graphs and probability in chapter(3). Undirected graphs also play a central role in modelling and reasoning with uncertainty. Essentially, two variables will be independent if they are not linked by a path on the graph. We will discuss this in more detail when we consider Markov networks in chapter(4).

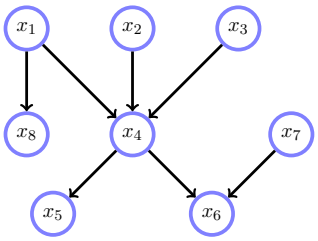
Definition 2.2 (Path, ancestors, descendants)**.** A *path* $A \mapsto B$ from node A to node B is a sequence of nodes that connects A to B. That is, a path is of the form $A_0, A_1, \ldots, A_{n-1}, A_n$, with $A_0 = A$ and $A_n = B$

and each edge (A_{k-1}, A_k), $k = 1, \ldots, n$ being in the graph. A directed path is a sequence of nodes which when we follow the direction of the arrows leads us from A to B. In directed graphs, the nodes A such that $A \mapsto B$ and $B \not\mapsto A$ are the *ancestors* of B. The nodes B such that $A \mapsto B$ and $B \not\mapsto A$ are the *descendants* of A.

Definition 2.3 (Cycle, loop and chord)**.** A *cycle* is a directed path that starts and returns to the same node $a \to b \to \ldots \to z \to a$. A *loop* is a path containing more than two nodes, irrespective of edge direction, that starts and returns to the same node. For example in fig(2.2b) $1 - 2 - 4 - 3 - 1$ forms a loop, but the graph is *acyclic* (contains no cycles). A *chord* is an edge that connects two non-adjacent nodes in a loop – for example, the $2 - 3$ edge is a chord in the $1 - 2 - 4 - 3 - 1$ loop of fig(2.2a).

Definition 2.4 (Directed Acyclic Graph (DAG))**.** A DAG is a graph G with directed edges (arrows on each link) between the nodes such that by following a path of nodes from one node to another along the direction of each edge no path will revisit a node. In a DAG the ancestors of B are those nodes who have a directed path ending at B. Conversely, the descendants of A are those nodes who have a directed path starting at A.

Definition 2.5 (Relationships in a DAG)**.**

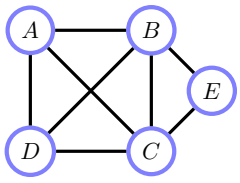


The *parents* of x_4 are $\text{pa}(x_4) = \{x_1, x_2, x_3\}$. The *children* of x_4 are $\text{ch}(x_4) = \{x_5, x_6\}$. The *family* of a node is itself and its parents. The *Markov blanket* of a node is its parents, children and the parents of its children (excluding itself). In this case, the Markov blanket of x_4 is $x_1, x_2, x_3, x_5, x_6, x_7$. @@

DAGs will play a central role in modelling environments with many variables, in particular they are used for the belief networks that we describe in the following chapter. One can view the directed links on a graph as 'direct dependencies' between parent and child variables. Naively, the acyclic condition prevents circular reasoning. These connections are discussed in detail in chapter(3).

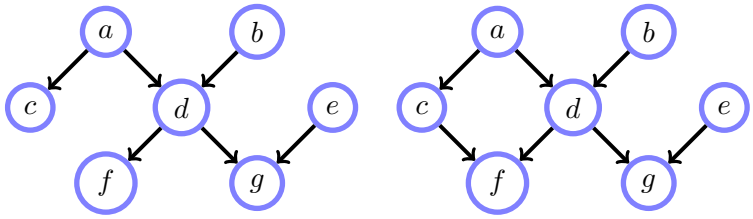
Definition 2.6 (Neighbour)**.** For an undirected graph G the neighbours of x, $\text{ne}(x)$ are those nodes directly connected to x.

Definition 2.7 (Clique)**.**



Given an undirected graph, a clique is a fully connected subset of nodes. All the members of the clique are neighbours; for a maximal clique there is no larger clique that contains the clique. For example this graph has two maximal cliques, $\mathcal{C}_1 = \{A, B, C, D\}$ and $\mathcal{C}_2 = \{B, C, E\}$. Whilst A, B, C are fully connected, this is a non-maximal clique since there is a larger fully connected set, A, B, C, D that contains this. A non-maximal clique is sometimes called a *cliquo*.

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).



(a) (b)

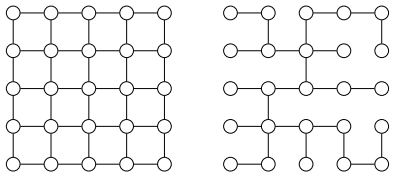
Figure 2.1: **(a)**: Singly-connected graph. **(b)**: Multiply-connected graph.

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*. 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.

2.2.1 Edge list

As the name suggests, an *edge list* simply lists which node-node pairs are in the graph. For fig(2.2a), 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.

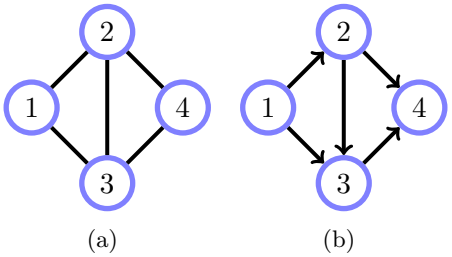


Figure 2.2: **(a)**: An undirected graph can be represented as a symmetric adjacency matrix. **(b)**: A directed acyclic graph with nodes labelled in ancestral order corresponds to a triangular adjacency matrix.

2.2.2 Adjacency matrix

An alternative is to use an *adjacency matrix*

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \tag{2.2.1}$$

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(2.2b) can be represented as a triangular adjacency matrix:

$$\mathbf{T} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.2.2}$$

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 $\left[\mathbf{A}^k\right]_{ij}$ 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 $\left[\mathbf{A}^{N-1}\right]_{ij}$ 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 $\left[\mathbf{A}^{N-1}\right]$ correspond to the descendants of node j.

2.2.3 Clique matrix

For an undirected graph with N nodes and maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_K$ 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

$$\mathbf{C} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \tag{2.2.3}$$

is a clique matrix for fig(2.2a). A cliquo matrix relaxes the constraint that cliques are required to be maximal[1]. A cliquo matrix containing only two-node cliques is called an *incidence matrix*. For example

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

[1]The term 'cliquo' for a non-maximal clique is attributed to Julian Besag.

is an incidence matrix for fig(2.2a). It is straightforward to show that $\mathbf{C}_{inc}\mathbf{C}_{inc}^{\mathsf{T}}$ 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 $[\mathbf{C}\mathbf{C}^{\mathsf{T}}]_{ii}$ expresses the number of cliquos (columns) that node i occurs in. Off diagonal elements $[\mathbf{C}\mathbf{C}^{\mathsf{T}}]_{ij}$ contain the number of cliquos 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 $x(t) \to x(t+1)$ 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 reference for graphs, associated theories and their uses are [87, 122].

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 possibly 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

Exercise 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* $\left[\mathbf{A}^k\right]_{ij}$ *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*.*

Exercise 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`*.*

Exercise 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.*

Exercise 2.4. *Describe a procedure to determine if a graph is singly-connected.*

Exercise 2.5. *Describe a procedure to determine all the ancestors of a set of nodes in a DAG.*

Exercise 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`*). Assume that if* i *'knows'* j*, then* j *'knows'* i*. Plot a histogram of the separation (the length of the shortest 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*.* ++ ++

Exercise 2.7. *The file* `cliques.mat` *contains a list of 100 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.*

Exercise 2.8. *Explain how to construct a graph with* N *nodes, where* N *is even, that contains at least* $(N/2)^2$ *maximal cliques.*

Exercise 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[218].*

Exercise 2.10. *A jewel was stolen from a room during a party*[2]*. Each of the guests (A,B,C,D,E,F) enters the room, stays for some time, and leaves. The testimonies they gave to the police are:* ++

[2]This question is a modified version of a well known problem, disguised here to make looking up the solution harder. Apologies to the original author for the lack of a citation.

1. *A: I was present in the room with E, B*
2. *B: I was present in the room with A, F and E*
3. *C: I was present in the room with F and D*
4. *D: I was present in the room with A and F*
5. *E: I was present in the room with C*
6. *F: I was present in the room with C and E*

One of the statements in the testimonies is false. Which is it?

Notes:

If any two people are present in the room at the same time, then at least one of them will report the fact that he was present in the room with the other person. Each above testimony should be interpreted as, for example:

A: I was present in the room with E. I was present in the room with B. It does not necessarily mean that A was simultaneously present in the room with both E and B (so A and B could have been present in the room at some time t_1, and A and E present in the room at some other time t_2). The other testimonies have a similar interpretation.

The testimonies could be partially false. If X says that he was present in the room with Y and Z, it could be that the statement X was present in the room with Y is true, but X was present in the room with Z is false.

CHAPTER 3

Belief Networks

We can now make a first connection between probability and graph theory. A belief network introduces structure into a probabilistic model by using graphs to represent independence assumptions among the variables. Probability operations such as marginalizing and conditioning then correspond to simple operations on the graph, and details about the model can be 'read' from the graph. There is also a benefit in terms of computational efficiency. Belief networks cannot capture all possible relations among variables. However, they are natural for representing 'causal' relations, and they are a part of the family of graphical models we study further in chapter(4).

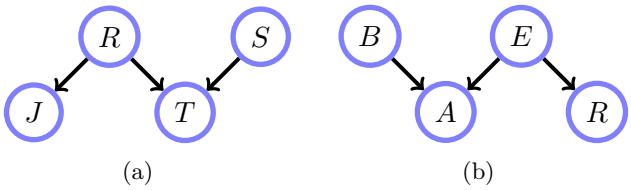
3.1 The Benefits of Structure

It's tempting to think of feeding a mass of undigested data and probability distributions into a computer and getting back good predictions and useful insights in extremely complex environments. However, unfortunately, such a naive approach is likely to fail. The possible ways variables can interact is extremely large, so that without some sensible assumptions we are unlikely to make a useful model. Independently specifying all the entries of a table $p(x_1, \ldots, x_N)$ over binary variables x_i takes $O(2^N)$ space, which is impractical for more than a handful of variables. This is clearly infeasible in many machine learning and related application areas where we need to deal with distributions on potentially hundreds if not millions of variables. Structure is also important for computational tractability of inferring quantities of interest. Given a distribution on N binary variables, $p(x_1, \ldots, x_N)$, computing a marginal such as $p(x_1)$ requires summing over the 2^{N-1} states of the other variables. Even on the most optimistically fast supercomputer this would take far too long, even for a $N = 100$ variable system.

The only way to deal with such large distributions is to constrain the nature of the variable interactions in some manner, both to render specification and ultimately inference in such systems tractable. The key idea is to specify which variables are independent of others, leading to a structured factorisation of the joint probability distribution. For a distribution on a chain, $p(x_1, \ldots, x_{100}) = \prod_{i=1}^{99} \phi(x_i, x_{i+1})$, computing a marginal $p(x_1)$ can be computed in the blink of an eye on modern computers. Belief networks are a convenient framework for representing such independence assumptions. We will discuss belief networks more formally in section(3.3), first discussing their natural role as 'causal' models.

Belief networks (also called Bayes' networks or Bayesian belief networks) are a way to depict the independence assumptions made in a distribution [162, 183]. Their application domain is widespread, ranging from troubleshooting[54] and expert reasoning under uncertainty to machine learning. Before we more formally define a Belief Network (BN), an example will help motivate the development[1].

[1]The scenario is adapted from [237].



(a) (b)

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.

3.1.1 Modelling independencies

One morning Tracey leaves her house and realises that her grass is wet. Is it due to overnight rain or did she forget to turn off the sprinkler last night? Next she notices that the grass of her neighbour, Jack, is also wet. This *explains away* to some extent the possibility that her sprinkler was left on, and she concludes therefore that it has probably been raining.

We can model the above situation by first defining the variables we wish to include in our model. In the above situation, the natural variables are

$R \in \{0, 1\}$ $R = 1$ means that it has been raining, and 0 otherwise
$S \in \{0, 1\}$ $S = 1$ means that Tracey has forgotten to turn off the sprinkler, and 0 otherwise
$J \in \{0, 1\}$ $J = 1$ means that Jack's grass is wet, and 0 otherwise
$T \in \{0, 1\}$ $T = 1$ means that Tracey's Grass is wet, and 0 otherwise

A model of Tracey's world then corresponds to a probability distribution on the joint set of the variables of interest $p(T, J, R, S)$ (the order of the variables is irrelevant).

Since each of the variables in this example can take one of two states, it would appear that we naively have to specify the values for each of the $2^4 = 16$ states, *e.g.* $p(T = 1, J = 0, R = 0, S = 1) = 0.057$ *etc.* @@ However, since there are normalisation conditions for probabilities, we do not need to specify all the state probabilities. To see how many states need to be specified, consider the following decomposition. Without loss of generality and repeatedly using the definition of conditional probability, we may write

$$p(T, J, R, S) = p(T|J, R, S)p(J, R, S) \tag{3.1.1}$$
$$= p(T|J, R, S)p(J|R, S)p(R, S) \tag{3.1.2}$$
$$= p(T|J, R, S)p(J|R, S)p(R|S)p(S) \tag{3.1.3}$$

That is, we may write the joint distribution as a product of conditional distributions. The first term $p(T|J, R, S)$ requires us to specify $2^3 = 8$ values – we need $p(T = 1|J, R, S)$ for the 8 joint states of J, R, S. The other value $p(T = 0|J, R, S)$ is given by normalisation : $p(T = 0|J, R, S) = 1 - p(T = 1|J, R, S)$. Similarly, we need $4 + 2 + 1$ values for the other factors, making a total of 15 values in all. In general, for a distribution on n binary variables, we need to specify $2^n - 1$ values in the range $[0, 1]$. The important point here is that the number of values that need to be specified in general scales exponentially with the number of variables in the model – this is impractical in general and motivates simplifications.

Conditional independence

The modeler often knows constraints on the system. For example, in the scenario above, we may assume that Tracey's grass is wet depends only directly on whether or not it has been raining and whether or not her sprinkler was on. That is, we make a *conditional independence* assumption

$$p(T|J, R, S) = p(T|R, S) \tag{3.1.4}$$

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

$$p(J|R, S) = p(J|R) \tag{3.1.5}$$

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

$$p(R|S) = p(R) \tag{3.1.6}$$

which means that our model equation (3.1.3) now becomes

$$p(T, J, R, S) = p(T|R, S)p(J|R)p(R)p(S) \tag{3.1.7}$$

We can represent these conditional independencies graphically, as in fig(3.1a). 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:

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

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

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

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

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:

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

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

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

$$= \frac{0.0344}{0.2144} = 0.1604 \tag{3.1.15}$$

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 [237]. 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,

$$p(B, E, A, R) = p(A|B, E, R)p(B, E, R) \tag{3.1.16}$$

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

$$p(B, E, A, R) = p(A|B, E, R)p(R|B, E)p(E|B)p(B) \tag{3.1.17}$$

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

$$p(B, E, A, R) = p(A|B, E)p(R|E)p(E)p(B) \tag{3.1.18}$$

as depicted in fig(3.1b).

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

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

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

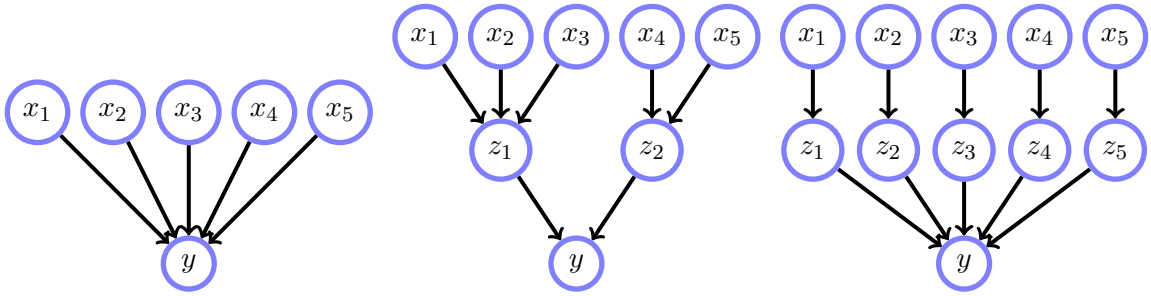
$$= \frac{\sum_{E,R} p(A = 1|B = 1, E)p(B = 1)p(E)p(R|E)}{\sum_{B,E,R} p(A = 1|B, E)p(B)p(E)p(R|E)} \approx 0.99 \tag{3.1.20}$$

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 $x_1, \ldots, x_n$, fig(3.2a). Formally, the structure of the graph implies nothing about the form of the parameterisation of the table $p(y|x_1, \ldots, x_n)$. If each parent x_i has $\dim(x_i)$ states, and there is no constraint on the table, then the table $p(y|x_1, \ldots, x_n)$



(a) (b) (c)

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

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 [162]). For example, in fig(3.2b), we have

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

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(3.2c), one could use a logical OR gate on binary z_i, say

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

We can then make a table $p(y|x_1, \ldots, x_5)$ 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(3.2c) 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 $\text{dom}(x) = \{\text{red}, \text{blue}, \text{green}\}$ 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 $(0, 0, 1)$.

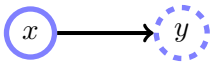
Performing inference with soft-evidence is straightforward and can be achieved using Bayes' rule. For example, for a model $p(x, y)$, consider that we have some soft evidence $\tilde{y}$ about the variable y, and wish to

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

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

where $p(y = \mathsf{i}|\tilde{y})$ represents the probability that y is in state i under the soft-evidence. The assumption here is that, if we know the certain evidence y, we don't need to know the uncertain evidence $\tilde{y}$, hence $p(x|y,\tilde{y}) = p(x|y)$. This is a generalisation of hard-evidence in which the vector $p(y|\tilde{y})$ has all zero component values, except for 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.



Example 3.2 (soft-evidence)**.** Revisiting the burglar scenario, example(3.1), imagine that we are only 70% 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?

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

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

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

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[237] that we adapt for our purposes here. The environment contains four variables

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

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

$$p(B, A, G, W) = p(A|B)p(B)p(W|A)p(G|A). \tag{3.2.4}$$

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% sure she heard it. This can be dealt with using the soft evidence technique, fig(3.3b). From Jeffrey's rule, one uses the original model equation (3.2.4) to first compute the model conditioned on the evidence

$$p(B = \mathsf{tr}|W = \mathsf{tr}, G) = \frac{p(B = \mathsf{tr}, W = \mathsf{tr}, G)}{p(W = \mathsf{tr}, G)} = \frac{\sum_A p(G|A)p(W = \mathsf{tr}|A)p(A|B = \mathsf{tr})p(B = \mathsf{tr})}{\sum_{B,A} p(G|A)p(W = \mathsf{tr}|A)p(A|B)p(B)} \tag{3.2.5}$$

and then uses the soft-evidence

$$p(G|\tilde{G}) = \begin{cases} 0.8 & G = \mathsf{tr} \\ 0.2 & G = \mathsf{fa} \end{cases} \tag{3.2.6}$$

to compute

$$p(B = \mathsf{tr}|W = \mathsf{tr}, \tilde{G}) = p(B = \mathsf{tr}|W = \mathsf{tr}, G = \mathsf{tr})p(G = \mathsf{tr}|\tilde{G}) + p(B = \mathsf{tr}|W = \mathsf{tr}, G = \mathsf{fa})p(G = \mathsf{fa}|\tilde{G}) \tag{3.2.7}$$

A full calculation requires us to numerically specify all the terms in equation (3.2.4); see for example exercise(3.8).

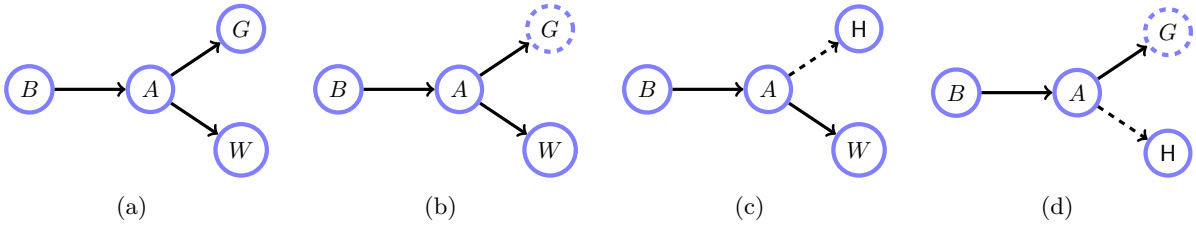


Figure 3.3: **(a)**: Mr Holmes' burglary worries as given in [237]: (B)urglar, (A)larm, (W)atson, Mrs (G)ibbon. **(b)**: Mrs Gibbon's uncertain evidence represented by a dashed circle. **(c)**: Virtual evidence or the replacement of unreliable evidence can be represented by a dashed line. **(d)**: Mrs Gibbon is uncertain in her evidence. Holmes also replaces the unreliable Watson with his own interpretation.

3.2.2 Unreliable evidence

Holmes telephones Mrs Gibbon and realises that he doesn't trust her evidence (he suspects that she's been @@ drinking) and his interpretation is that if the alarm sounded it was 80% probable to have resulted in Mrs Gibbon stating that she heard it; if the alarm didn't sound, there is a 20% chance that Mrs Gibbon would have stated that she heard it. Note that this is not the same as Mrs Gibbon being 80% sure herself that she heard the alarm – this would be soft evidence whose effect on our calculations would also contain the term $p(G|A)$, as in equation (3.2.5). Holmes rather wishes to discard all of this and simply replace it with his own interpretation of events. Mr Holmes can achieve this by replacing the term $p(G|A)$ by a so-called *virtual evidence* term

$$p(G|A) \to p(\mathsf{H}|A), \qquad \text{where} \quad p(\mathsf{H}|A) = \begin{cases} 0.8 & A = \mathsf{tr} \\ 0.2 & A = \mathsf{fa} \end{cases} \tag{3.2.8}$$

Here the state H is arbitrary and fixed. This is used to modify the joint distribution to

$$p(B, A, \mathsf{H}, W) = p(A|B)p(B)p(W|A)p(\mathsf{H}|A), \tag{3.2.9}$$

see fig(3.3c). When we then compute $p(B = \mathsf{tr}|W = \mathsf{tr}, \mathsf{H})$ the effect of Mr Holmes' judgement will count for a factor of 4 times more in favour of the alarm sounding than not. The values of the table entries are irrelevant up to normalisation since any constants can be absorbed into the proportionality constant. Note also that $p(\mathsf{H}|A)$ is not a distribution in A, and hence no normalisation is required. This form of evidence is also called *likelihood evidence*.

Uncertain and unreliable evidence

To demonstrate how to combine such effects as unreliable and uncertain evidence, consider the situation in which Mrs Gibbon is uncertain in her evidence, and Mr Holmes feels that Watson's evidence is unreliable and wishes to replaces it with his own interpretation, see fig(3.3d). To account for this we first deal with the unreliable evidence

$$p(B, A, W, G) \to p(B, A, \mathsf{H}, G) = p(B)p(A|B)p(G|A)p(\mathsf{H}|A) \tag{3.2.10}$$

Using this modified model, we can now use Jeffrey's rule to compute the model conditioned on the evidence

$$p(B, A|\mathsf{H}, G) = \frac{p(B)p(A|B)p(G|A)p(\mathsf{H}|A)}{\sum_{A,B} p(B)p(A|B)p(G|A)p(\mathsf{H}|A)} \tag{3.2.11}$$

We now include the uncertain evidence $\tilde{G}$ to form the final model

$$p(B, A|\mathsf{H}, \tilde{G}) = \sum_G p(B, A|\mathsf{H}, G)p(G|\tilde{G}) \tag{3.2.12}$$

from which we may compute the marginal $p(B|\mathsf{H}, \tilde{G})$

$$p(B|\mathsf{H}, \tilde{G}) = \sum_A p(B, A|\mathsf{H}, \tilde{G}) \tag{3.2.13}$$

3.3 Belief Networks

Definition 3.1 (Belief network)**.** A belief network is a distribution of the form

$$p(x_1, \ldots, x_D) = \prod_{i=1}^{D} p(x_i|\text{pa}\,(x_i)) \tag{3.3.1}$$

where $\text{pa}\,(x_i)$ represent the *parental* variables of variable x_i. Represented as a directed graph, with an arrow pointing from a parent variable to child variable, a belief network corresponds to a Directed Acyclic Graph (DAG), with the i^{th} node in the graph corresponding to the factor $p(x_i|\text{pa}\,(x_i))$.

Remark 3.2 (Graphs and distributions)**.** A somewhat subtle point is whether or not a belief network corresponds to a specific instance of a distribution (as given in definition(3.1)) requiring also the numerical specification of the conditional probability tables, or whether or not it refers to any distribution which is consistent with the specified structure. In this one can potentially distinguish between a belief network distribution (containing a numerical specification) and a belief network graph (which contains no numerical specification). Normally this issue will not arise much throughout the book, but is potentially important in clarifying the scope of independence/dependence statements.

In the Wet Grass and Burglar examples, we had a choice as to how we recursively used Bayes' rule. In a general 4 variable case we could choose the factorisation,

$$p(x_1, x_2, x_3, x_4) = p(x_1|x_2, x_3, x_4)p(x_2|x_3, x_4)p(x_3|x_4)p(x_4) \tag{3.3.2}$$

An equally valid choice is (see fig(3.4))

$$p(x_1, x_2, x_3, x_4) = p(x_3|x_4, x_1, x_2)p(x_4|x_1, x_2)p(x_1|x_2)p(x_2). \tag{3.3.3}$$

In general, two different graphs may represent the same independence assumptions, as we will discuss further in section(3.3.1). If one wishes to make independence assumptions, then the choice of factorisation becomes significant.

The observation that any distribution may be written in the *cascade* form, fig(3.4), gives an algorithm for constructing a BN on variables $x_1, \ldots, x_n$: write down the $n-$node cascade graph; label the nodes with the variables in any order; now each successive independence statement corresponds 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 $x_1, \ldots, x_n$. Then, from Bayes' rule, we have

$$p(x_1, \ldots, x_n) = p(x_1|x_2, \ldots, x_n)p(x_2, \ldots, x_n) \tag{3.3.4}$$

$$= p(x_1|x_2, \ldots, x_n)p(x_2|x_3, \ldots, x_n)p(x_3, \ldots, x_n) \tag{3.3.5}$$

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

The representation of any BN is therefore a *Directed Acyclic Graph (DAG)*.

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).

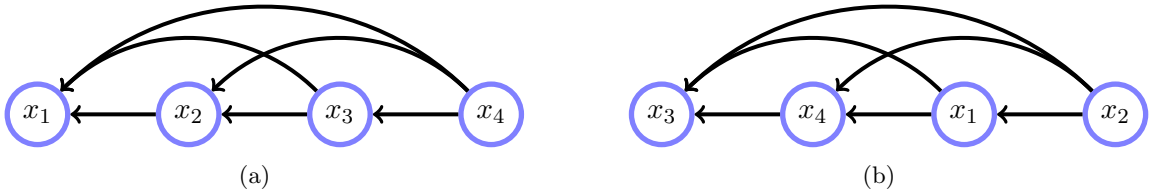


Figure 3.4: Two BNs for a 4 variable distribution. Both graphs (a) and (b) represent the *same* distribution $p(x_1, x_2, x_3, x_4)$. Strictly speaking they represent the same (lack of) independence assumptions – the graphs say nothing about the content of the tables. The extension of this 'cascade' to many variables is clear and always results in a Directed Acyclic Graph.

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(3.2b), $MB(z_1) = \{x_1, x_2, x_3, y, z_2\}$ 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 $p(x_i|\text{pa}(x_i))$. Once the graphical structure is defined, the entries of the conditional probability tables (CPTs) $p(x_i|\text{pa}(x_i))$ can be expressed. For every possible state of the parental variables $\text{pa}(x_i)$, 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.

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

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

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

Now

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

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

Combining the two results above we have

$$p(x_1, x_2|x_4) = p(x_1|x_4)p(x_2|x_4) \tag{3.3.11}$$

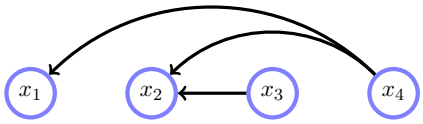


Figure 3.5: $p(x_1, x_2, x_3, x_4) = p(x_1|x_4)p(x_2|x_3, x_4)p(x_3)p(x_4)$.

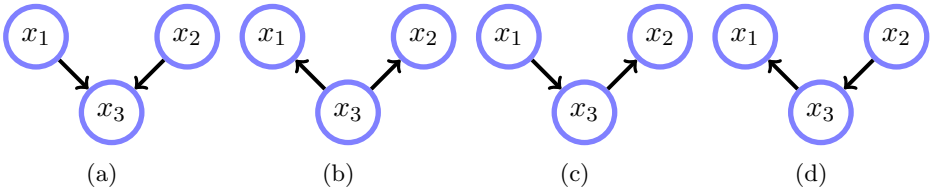


Figure 3.6: By dropping say the connection between variables x_1 and x_2, we reduce the 6 possible BN graphs amongst three variables to 4. (The 6 fully connected 'cascade' graphs correspond to (a) with $x_1 \to x_2$, (a) with $x_2 \to x_1$, (b) with $x_1 \to x_2$, (b) with $x_2 \to x_1$, (c) with $x_1 \to x_2$ and (d) with $x_2 \to x_1$. Any other graphs would be cyclic and therefore not distributions).

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 6 ways

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

where (i_1, i_2, i_3) is any of the 6 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 4 DAGs in fig(3.6). Are any of these graphs equivalent, in the sense that they represent the same distribution? Applying Bayes' rule gives :

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

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

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, $x_1 \perp\!\!\!\perp x_2 | x_3$.

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 $a \to b$. 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 @@ $a \to c \leftarrow b$. Note that a collider is path specific, see fig(3.8).

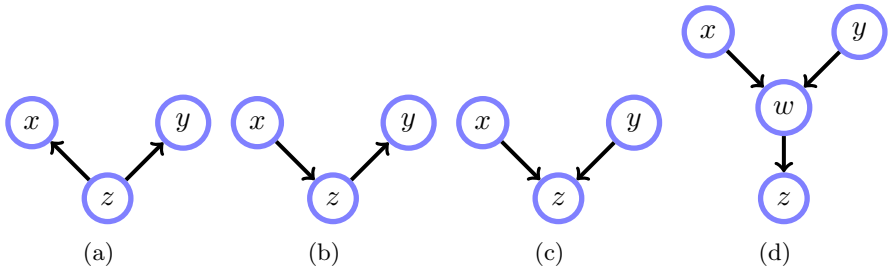


Figure 3.7: In graphs (a) and (b), variable z is not a collider. **(c)**: Variable z is a collider. Graphs (a) and (b) represent conditional independence $x \perp\!\!\!\perp y|\, z$. In graphs (c) and (d), x and y are 'graphically' conditionally dependent given variable z.

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

$$p(x, y|z) = p(x|z)p(y|z) \tag{3.3.15}$$

Similarly, for fig(3.7b), x and y are independent conditioned on z since

$$p(x, y|z) \propto p(z|x)p(x)p(y|z) \tag{3.3.16}$$

which is a function of x multiplied by a function of y. In fig(3.7c), 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 a collider — the arrows of its neighbours are pointing towards it. What about fig(3.7d)? In (d), when we condition on z, x and y will be graphically dependent, since

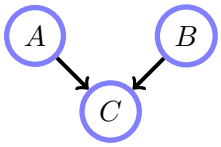
$$p(x, y|z) = \frac{p(x, y, z)}{p(z)} = \frac{1}{p(z)} \sum_w p(z|w)p(w|x, y)p(x)p(y) \neq p(x|z)p(y|z) \tag{3.3.17}$$

The above inequality holds due to the term $p(w|x, y)$ – only in special cases such as $p(w|x, y) = \text{const.}$ would x and y be independent. 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(3.8a), 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).

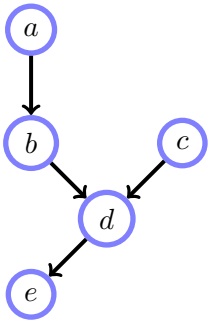
Consider the BN: $A \to B \leftarrow C$. Here A and C are (unconditionally) independent. However, conditioning of B makes them 'graphically' dependent. Intuitively, whilst we believe the root causes are independent, given the value of the observation, this tells us something about the state of *both* 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).

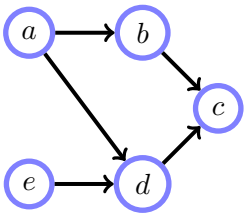


$$p(A, B, C) = p(C|A, B)p(A)p(B) \tag{3.3.18}$$

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

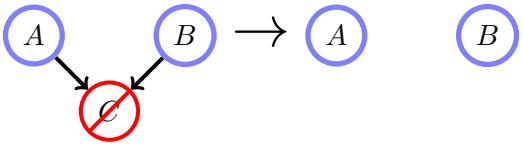


(a): The variable d is a collider along the path $a-b-d-c$, but not along the path $a-b-d-e$. Is $a \perp\!\!\!\perp e \,|\, b$? a and e are *not* d-connected since there are no colliders on the only path between a and e, and since there is a non-collider b which is in the conditioning set. Hence a and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

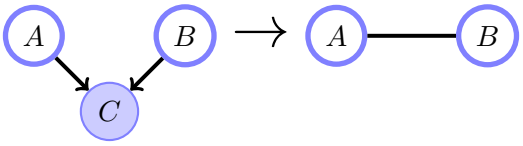


(b): The variable d is a collider along the path $a-d-e$, but not along the path $a-b-c-d-e$. Is $a \perp\!\!\!\perp e|\, c$? There are two paths between a and e, namely $a-d-e$ and $a-b-c-d-e$. The path $a-d-e$ is not blocked since although d is a collider on this path and d is not in the conditioning 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.

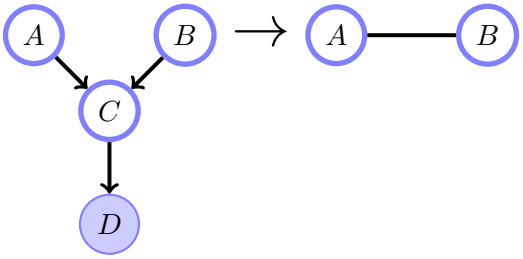
Figure 3.8: Collider examples for d-separation and d-connection.



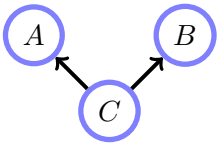
Marginalising over C makes A and B independent. A and B are (unconditionally) independent : $p(A,B) = p(A)p(B)$. In the absence of any information about the effect C, we retain this belief.



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.

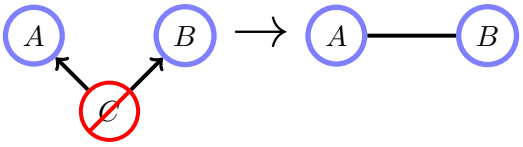


Conditioning on D, a descendent of a collider C, makes A and B (graphically) dependent — in general $p(A,B|D) \neq p(A|D)p(B|D)$.

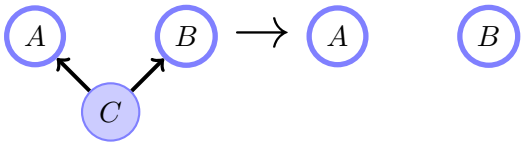


$$p(A,B,C) = p(A|C)p(B|C)p(C) \tag{3.3.19}$$

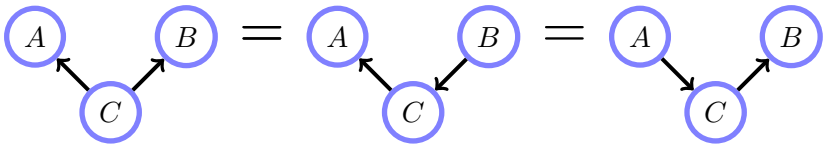
Here there is a 'cause' C and independent 'effects' A and B.



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.



Conditioning on C makes A and B independent: $p(A, B|C) = p(A|C)p(B|C)$. If you know the 'cause' C, you know everything about how each effect occurs, independent of the other effect. This is also true for reversing the arrow from A to C – in this case A would 'cause' C and then C 'cause' B. Conditioning on C blocks the ability of A to influence B.



These graphs all express the same conditional independence assumptions.

3.3.3 Graphical path manipulations for independence

Intuitively, we now have all the tools we need to understand when x is independent of y conditioned on z. Examining the rules in definition(3.3), we need to look at each path between x and y. Colouring x as red and y as green and the conditioning node z as yellow, we need to examine each path between x and y and adjust the edges, following the intuitive rules in fig(3.9).

3.3.4 d-Separation

The above description is intuitive. A more formal treatment that is amenable to computational implementation is straightforward to obtain from these intuitions. First we define the DAG concepts of d-separation and d-connection that are central to determining conditional independence in any BN with structure given by the DAG[305].

Definition 3.4 (d-connection, d-separation)**.** If G is a directed graph in which $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ are disjoint sets of vertices, then $\mathcal{X}$ and $\mathcal{Y}$ are d-connected by $\mathcal{Z}$ in G if and only if there exists an undirected path U between some vertex in $\mathcal{X}$ and some vertex in $\mathcal{Y}$ such that for every collider C on U, either C or a descendent of C is in $\mathcal{Z}$, and no non-collider on U is in $\mathcal{Z}$.

$\mathcal{X}$ and $\mathcal{Y}$ are d-separated by $\mathcal{Z}$ in G if and only if they are not d-connected by $\mathcal{Z}$ in G.

One may also phrase this as follows. For every variable $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, check every path U between x and y. A path U is said to be *blocked* if there is a node w on U such that either

1. w is a collider and neither w nor any of its descendants is in $\mathcal{Z}$, or
2. w is not a collider on U and w is in $\mathcal{Z}$.

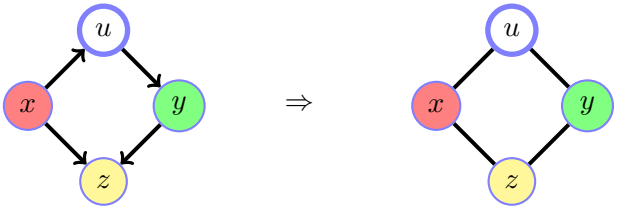
If all such paths are blocked then $\mathcal{X}$ and $\mathcal{Y}$ are d-separated by $\mathcal{Z}$. If the variable sets $\mathcal{X}$ and $\mathcal{Y}$ are d-separated by $\mathcal{Z}$, they are independent conditional on $\mathcal{Z}$ in all probability distributions such a graph can represent.

Remark 3.5 (Bayes Ball)**.** The Bayes Ball algorithm[259] 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 $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}$. $\mathcal{Y}$ is called the set of irrelevant nodes for $\mathcal{X}$ given $\mathcal{Z}$.

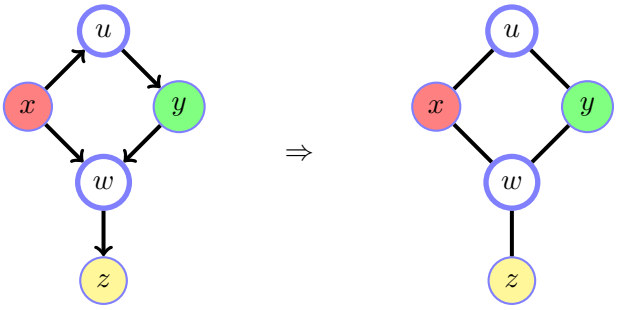
3.3.5 Graphical and distributional in/dependence

We have shown

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

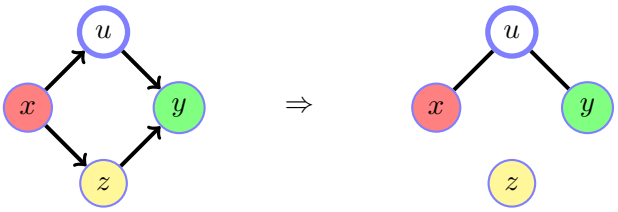


If z is a collider (bottom path) keep undirected links between the neighbours of the collider.

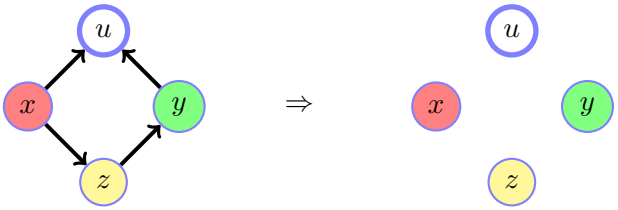


If z is a descendant of a collider, this could induce dependence so we retain the links (making them undirected)

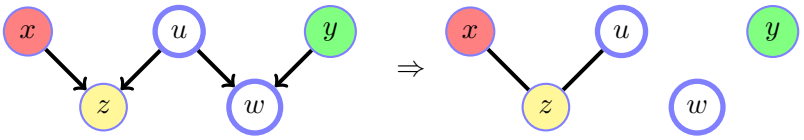
If there is a collider not in the conditioning set (upper path) we cut the links to the collider variable. In this case the upper path between x and y is 'blocked'.



If there is a non-collider which is in the conditioning set (bottom path), we cut the link between the neighbours of this non-collider which cannot induce dependence between x and y. The bottom path is 'blocked'.



In this case, neither path contributes to dependence and hence $x \perp\!\!\!\perp y | z$. Both paths are 'blocked'.



Whilst z is a collider in the conditioning set, w is a collider that is not in the conditioning set. This means that there is no path between x and y, and hence x and y are independent given z.

Figure 3.9: Graphical manipulations to determine independence $x \perp\!\!\!\perp y \mid z$. After these manipulations, if there is no undirected path between x and y, then x and y are independent, conditioned on z. Note that the graphical rules here differ from those in definition(3.3) which considered the effect on the graph having eliminated a variable (via conditioning or marginalisation). Here we consider rules for determining independence based on a graphical representation in which the variables remain in the graph.

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 $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}$. Note that the list $\mathcal{L}_P$ could contain more statements than those obtained from the graph. For example for the belief network graph

$$p(a, b, c) = p(c|a, b)p(a)p(b) \tag{3.3.20}$$

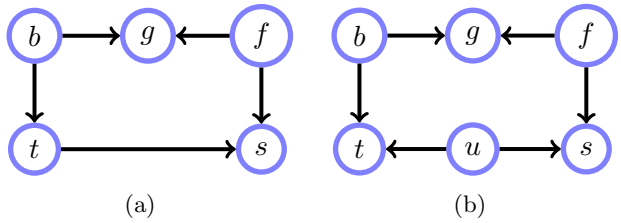


Figure 3.10: **(a)**: t and f are d-connected by g. **(b)**: b and f are d-separated by u.

which is representable by the DAG $a \to c \leftarrow b$, 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 $\mathrm{dom}(a) = \mathrm{dom}(b) = \mathrm{dom}(c) = \{0, 1\}$

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

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

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

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 $a \perp\!\!\!\not\perp b | c$ 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 $\mathcal{Z} \not\Rightarrow \mathcal{X} \perp\!\!\!\not\perp \mathcal{Y} | \mathcal{Z}$ 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.

Example 3.3. Consider the graph in fig(3.10a).

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(3.10b)? 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.

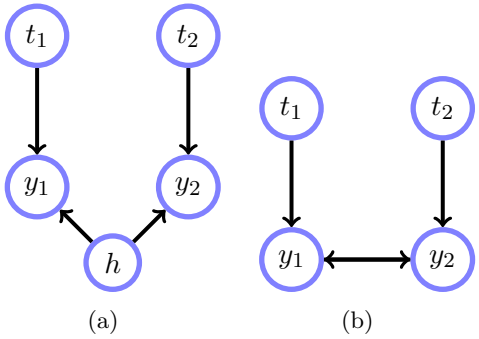


Figure 3.11: **(a)**: Two treatments t_1, t_2 and corresponding outcomes y_1, y_2. The health of a patient is represented by h. This DAG embodies the conditional independence statements $t_1 \perp\!\!\!\perp t_2, y_2 \,|\, \emptyset$, $t_2 \perp\!\!\!\perp t_1, y_1 \,|\, \emptyset$, namely that the treatments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

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 $A \to C \leftarrow B$, from which the set of conditional independence statements is $A \perp\!\!\!\perp B | \emptyset$. For another belief network with edges $A \to C \leftarrow B$ and $A \to B$, 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 same set of independence assumptions (they are *Markov equivalent*) if and only if they have the same skeleton and the same set of immoralities [79].

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). Note that whether unmarried parents having children should be considered immoral is beyond the scope of this book! ++

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(3.11a), (from [250]). 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(3.11a). 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

$$p(t_1, t_2, y_1, y_2) = p(t_1)p(t_2)\sum_h p(y_1|t_1, h)p(y_2|t_2, h)p(h) \tag{3.3.23}$$

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 @@ $y_1 \perp\!\!\!\perp t_2 | y_2$, 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 $y_2 \perp\!\!\!\perp t_1 | y_1$ 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[282]. 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(3.11a), one could depict the marginal distribution using a bi-directional edge, fig(3.11b). For a discussion of extensions of BNs using bi-directional edges see [250].

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 [238] and [79] 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 $p(rain, grasswet) = p(grasswet|rain)p(rain)$. **(d)**: We could equally have written $p(rain|grasswet)p(grasswet)$, although this appears to be causally non-sensical.

Males	Recovered	Not Recovered	Rec. Rate
Given Drug	18	12	60%
Not Given Drug	7	3	70%

Females	Recovered	Not Recovered	Rec. Rate
Given Drug	2	8	20%
Not Given Drug	9	21	30%

Combined	Recovered	Not Recovered	Rec. Rate
Given Drug	20	20	50%
Not Given Drug	16	24	40%

Table 3.1: Table for Simpson's Paradox (from [238])

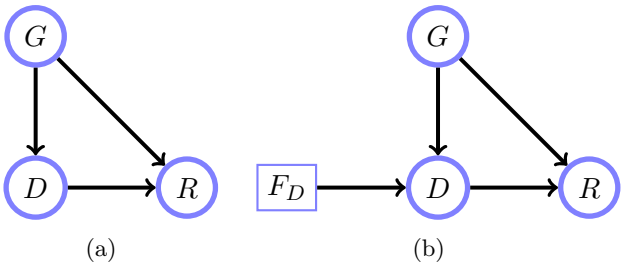


Figure 3.13: **(a)**: A DAG for the relation between Gender (G), Drug (D) and Recovery (R), see table(3.1). **(b)**: Influence diagram. No decision variable is required for G since G has no parents.

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[238] 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(3.13a),

$$p(G, D, R) = p(R|G, D)p(D|G)p(G) \tag{3.4.1}$$

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

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

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 $||$:

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

(One can also consider here G as being interventional – in this case it doesn't matter since the fact that the variable G has no parents means that for any distribution conditional on G, the prior factor $p(G)$ will not be present). Using equation (3.4.3), for the males given the drug 60% recover, versus 70% recovery when not given the drug. For the females given the drug 20% recover, versus 30% recovery when not given the drug.

Similarly,

$$p(R||D) \equiv \tilde{p}(R|D) = \frac{\sum_G p(R|G, D)p(G)}{\sum_{R,G} p(R|G, D)p(G)} = \sum_G p(R|G, D)p(G) \tag{3.4.4}$$

Using the post intervention distribution, equation (3.4.4), we have

$$p(\text{recovery}|\text{drug}) = 0.6 \times 0.5 + 0.2 \times 0.5 = 0.4 \tag{3.4.5}$$
$$p(\text{recovery}|\text{no drug}) = 0.7 \times 0.5 + 0.3 \times 0.5 = 0.5 \tag{3.4.6}$$

Hence we infer that the drug is overall not helpful, as we intuitively expect, and is consistent with the results from both subpopulations.

Summarising the above argument, $p(G, D, R) = p(R|G, D)p(G)p(D)$ means that we choose either a Male or Female patient and give them the drug or not independent of their gender, hence the absence of the term $p(D|G)$ from the joint distribution. If we were to do a purely observation calculation $p(R|D)$, the conditioning on D induces a dependency on gender (through the term $p(D|G)$ and hence on the recovery. This setup would correspond to the doctor determining first whether or not to give the drug, and then selecting a gender at random according to $p(G|D)$, which is not how we would expect a 'drug trial' to proceed. Doing a random trial in which the drug is simply given (or not) at random, according to $p(D)$ would break the dependency on the gender and correspond to a sensible interventional experiment. One way to think about such models is to consider how to draw a sample from the joint distribution of the random variables – in most cases this should clarify the role of causality in the experiment.

In contrast to the interventional calculation, the observational calculation makes no conditional independence assumptions. This means that, for example, the term $p(D|G)$ plays a role in the calculation (the reader might wish to verify that the result given in the combined data in table(3.1) is equivalent to inferring with the full distribution equation (3.4.1)).

3.4.2 The do-calculus

In making causal inferences we've seen above that we must adjust the model to reflect any causal experimental conditions. In setting any variable into a particular state we need to surgically remove all parental links of that variable. Pearl calls this the *do operator*, and contrasts an observational ('see') inference $p(x|y)$ with a causal ('make' or 'do') inference $p(x|do(y))$.

Definition 3.6 (Pearl's Do Operator)**.**

Let all the variables $\mathcal{X} = \mathcal{X}_C \cup \mathcal{X}_{\bar{C}}$ be written in terms of the intervention variables $\mathcal{X}_C$ and the non-intervention variables $\mathcal{X}_{\bar{C}}$. For a Belief Network $p(\mathcal{X}) = \prod_i p(X_i|\text{pa}(X_i))$, inferring the effect of setting variables $X_{c_1}, \ldots, X_{c_K}$, $c_k \in \mathcal{C}$, in states $\mathsf{x}_{c_1}, \ldots, \mathsf{x}_{c_K}$, is equivalent to standard evidential inference in the *post intervention distribution*:

$$p(\mathcal{X}_{\bar{C}}|do(X_{c_1} = \mathsf{x}_{c_1}), \ldots, do(X_{c_K} = \mathsf{x}_{c_K})) = \prod_{j \in \bar{C}} p\left(X_j|\text{pa}\left(X_j\right)\right) \tag{3.4.7}$$

where any parental variable included in the intervention set is set to its intervention state. An alternative notation is $p(\mathcal{X}_{\bar{C}}||\mathsf{x}_{c_1}, \ldots, \mathsf{x}_{c_K})$.

In words, for those variables for which we causally intervene and set in a particular state, the corresponding terms $p(X_{c_i}|\text{pa}\left(X_{c_i}\right))$ are removed from the original Belief Network. Graphically, the effect is to consider each intervention variable, cut the connections to its parents and set the intervention variable to its intervention state. For variables which are evidential but non-causal, the corresponding factors are not removed from the distribution. The interpretation is that the post intervention distribution corresponds to an experiment in which the causal variables are first set and non-causal variables are subsequently observed.

For a Belief Network to have a causal interpretation, it means that the ancestral order of the variables must correspond to the temporal order. This means therefore that if we start with the variables that have no parents, these must come first in time, with their children coming later, *etc.* Ancestral sampling from a causal Belief Network therefore corresponds to the temporal evolution of a physical experiment.

3.4.3 Influence diagrams and the do-calculus

Another way to represent intervention is to modify the basic BN by appending a parental decision variable F_X to any variable X on which an intervention can be made, giving rise to a so-called influence diagram[79]. For example[2], for the Simpson's paradox example, we may use, fig(3.13b),

$$\tilde{p}(D, G, R, F_D) = p(D|F_D, G)p(G)p(R|G, D)p(F_D) \tag{3.4.8}$$

where

$$p(D|F_D = \emptyset, G) \equiv p(D|\text{pa}\,(D)), \quad p(D|F_D = \mathsf{d}, G) = 1 \text{ for } D = \mathsf{d} \text{ and } 0 \text{ otherwise}$$

Hence, if the decision variable F_D is set to the empty state, the variable D is determined by the standard observational term $p(D|\text{pa}\,(D))$. If the decision variable is set to a state of D, then the variable puts all its probability in that single state of $D = \mathsf{d}$. This has the effect of replacing the conditional probability term by a unit factor and any instances of D set to the variable in its interventional state[3]. A potential advantage of this influence diagram approach over the do-calculus is that conditional independence statements can be derived using standard techniques for the augmented BN. Additionally, for learning, standard techniques apply in which the decision variables are set to the condition under which each data sample was collected (a causal or non-causal sample).

Remark 3.6 (Learning the edge directions)**.** In the absence of data from causal experiments, one should be justifiably sceptical about learning 'causal' networks. Nevertheless, one might prefer a certain direction of a link based on assumptions of the 'simplicity' of the CPTs. This preference may come from a physical intuition that whilst root causes may be uncertain, the relationship from cause to effect is relatively simple. @@ In this sense a measure of the complexity of a CPT is required, such as entropy. Similarly, a useful heuristic ++ is that root causes are independent and, amongst equally likely belief networks, the one with the smallest number of edges will correspond to the 'causal' one. Such heuristics can be numerically encoded and the edge directions learned. See also exercise(12.6). @@

3.5 Summary

- We can reason with certain or uncertain evidence using repeated application of Bayes' rule.
- A belief network represents a factorisation of a distribution into conditional probabilities of variables dependent on parental variables.
- Belief networks correspond to directed acyclic graphs.
- Variables are conditionally independent $x \perp\!\!\!\perp y|\, z$ if $p(x, y|z) = p(x|z)p(y|z)$; the absence of a link in a belief network corresponds to a conditional independence statement.
- If in the graph representing the belief network, two variables are independent, then they are independent in any distribution consistent with the belief network structure.
- Belief networks are natural for representing 'causal' influences.
- Causal questions must be addressed by an appropriate causal model.

[2]Here the influence diagram is a distribution over variables including decision variables, in contrast to the application of IDs in chapter(7).

[3]More general cases can be considered in which the variables are placed in a distribution of states [79].

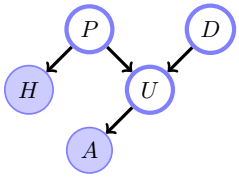


Figure 3.14: Party animal. Here all variables are binary. P = Been to Party, H = Got a Headache, D = Demotivated at work, U = Underperform at work, A =Boss Angry. Shaded variables are observed in the true state.

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

$$p(\mathcal{X}, \mathcal{Y}|\mathcal{Z}) \approx p(\mathcal{X}|\mathcal{Z})p(\mathcal{Y}|\mathcal{Z}) \tag{3.6.1}$$

even though $\mathcal{X} \not\perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}$. 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

Exercise 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:*

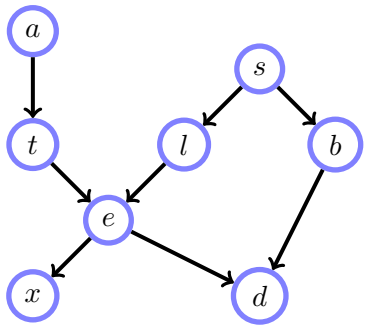
$p(U = \mathsf{tr}\|P = \mathsf{tr}, D = \mathsf{tr}) = 0.999$	$p(U = \mathsf{tr}\|P = \mathsf{fa}, D = \mathsf{tr}) = 0.9$	$p(H = \mathsf{tr}\|P = \mathsf{tr}) = 0.9$
$p(U = \mathsf{tr}\|P = \mathsf{tr}, D = \mathsf{fa}) = 0.9$	$p(U = \mathsf{tr}\|P = \mathsf{fa}, D = \mathsf{fa}) = 0.01$	$p(H = \mathsf{tr}\|P = \mathsf{fa}) = 0.2$
$p(A = \mathsf{tr}\|U = \mathsf{tr}) = 0.95$	$p(A = \mathsf{tr}\|U = \mathsf{fa}) = 0.5$	$p(P = \mathsf{tr}) = 0.2, p(D = \mathsf{tr}) = 0.4$

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

Exercise 3.3. *The Chest Clinic network[185] 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. $visit\ to\ Asia \perp\!\!\!\perp smoking | lung\ cancer, shortness\ of\ breath$

[4] The code for graphical conditional independence is given in chapter(4).



x = Positive X-ray
d = Dyspnea (Shortness of breath)
e = Either Tuberculosis or Lung Cancer
t = Tuberculosis
l = Lung Cancer
b = Bronchitis
a = Visited Asia
s = Smoker

Figure 3.15: Belief network structure for the Chest Clinic example.

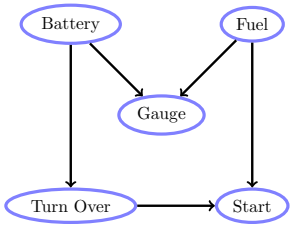


Figure 3.16: Belief network of car starting, see exercise(3.6).

Exercise 3.4. *Consider the Chest Clinic belief network in fig(3.15)[185]. Calculate by hand the values for* $p(d)$, $p(d|s = \mathsf{tr})$, $p(d|s = \mathsf{fa})$. *The table values are:*

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

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

Exercise 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* $p(d = \mathsf{tr}|b = \mathsf{fa})$ *in a non-causal interpretation, and what does this mean?*

Exercise 3.6 ([141])**.** *The network in fig(3.16) concerns the probability of a car starting, with*

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

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

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

$$p(A, S, C) = p(C|A, S)p(A)p(S) \tag{3.7.1}$$

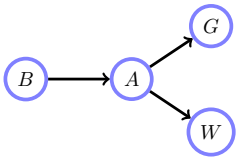
1. *Is* $A \perp\!\!\!\perp S|\emptyset$?

2. *Is* $A \perp\!\!\!\perp S | C$*?*

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 asbestos exposure?*

Exercise 3.8.

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



All variables are binary with states $\{\mathsf{tr}, \mathsf{fa}\}$*. The table entries are*

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

1. *Compute 'by hand'* (i.e. *show your working) :*

   (a) $p(B=\mathsf{tr}|W=\mathsf{tr})$

   (b) $p(B=\mathsf{tr}|W=\mathsf{tr}, G=\mathsf{fa})$

2. *Consider the same situation as above, except that now the evidence is uncertain. Mrs Gibbon thinks that the state is* $G=\mathsf{fa}$ *with probability 0.9. Similarly, Dr Watson believes in the state* $W=\mathsf{fa}$ *with value 0.7. Compute 'by hand' the posteriors under these uncertain (soft) evidences:*

   (a) $p(B=\mathsf{tr}|\tilde{W})$

   (b) $p(B=\mathsf{tr}|\tilde{W}, \tilde{G})$

Exercise 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* $A \perp\!\!\!\perp G | \emptyset$*.*

1. *Write down the belief network for the above situation.*

2. *Explain how to compute* $p(\mathsf{recover}|\mathsf{drug})$*.*

3. *Explain how to compute* $p(\mathsf{recover}|do(\mathsf{drug}), \mathsf{young})$*.*

Exercise 3.10. *Implement the Wet Grass scenario in section(3.1.1) using the* BRMLtoolbox*.*

Exercise 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.*

Exercise 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.*

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

$$\mathbf{A} = \begin{pmatrix} 0&0&1&1&0&1&0&0&0 \\ 0&0&1&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&1 \\ 0&0&1&0&0&0&1&0&0 \\ 0&0&0&1&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} 0&0&1&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&1 \\ 0&1&1&0&0&0&1&0&0 \\ 1&0&0&1&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \end{pmatrix} \tag{3.7.3}$$

Exercise 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 2 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*

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

Exercise 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*

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

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?*

Exercise 3.16. *There are 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.*

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

$$p(a,b,c) = p(c|b)p(b|a)p(a) \tag{3.7.5}$$

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

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

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

2. *Consider*

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

   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

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

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

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

3. *Show that if*

$$\mathbf{M}\mathbf{N}^T = \mathbf{m}_0\mathbf{n}_0^T \tag{3.7.10}$$

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

4. *Writing*

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

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

$$\mathbf{M}\mathbf{N}^T = \mathbf{m}_1\mathbf{n}_1^T + \mathbf{m}_2\mathbf{n}_2^T + \mathbf{m}_3\mathbf{n}_3^T \tag{3.7.12}$$

5. *Show that by setting*

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

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

$$\mathbf{m}_0 \equiv \mathbf{m}_1 + \gamma\mathbf{m}_3, \qquad \mathbf{n}_0 \equiv \mathbf{n}_1 + \lambda\mathbf{n}_2 \tag{3.7.14}$$

@@ 6. *Hence construct example tables (different from those specified in part 1 above)* $p(a)$*,* $p(b|a)$*,* $p(c|b)$ *for which* $a \perp\!\!\!\perp c$*. Verify your examples explicitly using* BRMLtoolbox.

Exercise 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* $A \perp\!\!\!\perp B$.

Exercise 3.19. *Assume that the day of the week that females are born on,* x*, is independent of the day of the week,* y*, on which males are born. Assume, however, that the old rhyme is true and that personality is dependent on the day of the week you're born on. If* a *represents the female personality type and* b *the male personality type, then* $a \perp\!\!\!\perp x$ *and* $b \perp\!\!\!\perp y$*, but* $a \perp\!\!\!\perp b$*. Whether or not a male and a female are married,* m*, depends strongly on their personality types,* $m \perp\!\!\!\perp \{a, b\}$*, but is independent of* x *and* y *if we know* a *and* b*. Draw a belief network that can represent this setting. What can we say about the (graphical) dependency between the days of the week that John and Jane are born on, given that they are not married?*

++ **Exercise 3.20.** *A survey of households in which the husband and wife each own their own car is made. The survey also states whether each household income (inc) is high or low. There are 4 car types,* $dom(h) = dom(w) = \{1, 2, 3, 4\}$*, the first two being 'cheap' and the last two being 'expensive'. The survey finds that, given their household income, the types of cars owned by a husband and wife are independent:*

$$\textit{wife's car type} \perp\!\!\!\perp \textit{husband's car type} \,|\, \textit{family income}$$

Specifically, $p(inc = \mathsf{low}) = 0.8$ *and*

$$p(w|inc = \mathsf{low}) = \begin{pmatrix} 0.7 \\ 0.3 \\ 0 \\ 0 \end{pmatrix}, \quad p(w|inc = \mathsf{high}) = \begin{pmatrix} 0.2 \\ 0.1 \\ 0.4 \\ 0.3 \end{pmatrix}$$

$$p(h|inc = \mathsf{low}) = \begin{pmatrix} 0.2 \\ 0.8 \\ 0 \\ 0 \end{pmatrix}, \quad p(h|inc = \mathsf{high}) = \begin{pmatrix} 0 \\ 0 \\ 0.3 \\ 0.7 \end{pmatrix}$$

Use BRMLtoolbox *to find the marginal* $p(w, h)$ *and show that whilst* $h \perp\!\!\!\perp w \,|\, inc$*, it is not the case that* $h \perp\!\!\!\perp w$.

Exercise 3.21. BinGame *is a two-player game with only a 'win' or 'lose' outcome. There are three players, A, B, C that have been competing in a tournament, with the following outcomes:* ++

- *A beat B 2 times.*
- *B beat C 2 times.*
- *A beat C 2 times.*
- *C beat A 1 time.*
- *Player D enters the tournament and loses twice to player C.*

We assume that each player has a skill level from $1, \ldots, 10$ *and that, given the skill levels of two players,* s_A*,* s_B*, the probability that player A beats player B is* $1/(1+\exp(s_B - s_A))$*. Assume that a priori the skill levels of the players are independent and that we place equal probability to each of the 10 skills levels. Additionally, the outcome of all games, given the skill levels are independent. Using the above tournament information :*

1. *Draw a belief network that represents the independence assumptions above.*
2. *Are the skill levels of the players a posteriori (*i.e. *given the game outcomes) independent?*
3. *Calculate the probability that player D will beat player A in a game of* BinGame.
4. *Calculate the expected skill level of each of the 4 players.*

Exercise 3.22. *Before watching a video for free on their website, the Daily Fail forces you to watch a video advert. There are in fact 10 adverts available and the Daily Fail randomly selects three advert links to present to each viewer. After clicking on one of the three advert links, the advert plays, after which the desired news video can be watched. The data from the Daily Fail from 20 website visitors is as follows:* ++

$$(1,2,3)(2,5,3)(4,7,10)(6,3,4)(6,8,5)(9,3,7)(10,2,4)(7,1,2)(8,7,2)(8,3,6)$$
$$(8,6,4)(4,3,9)(5,4,1)(9,5,1)(6,7,8)(4,9,7)(10,8,6)(5,4,3)(6,3,2)(1,4,2)$$

where (a_i, a_j, a_k) *means that advert* i *was clicked on in preference to adverts* j *and* k*.*

Assume that each advert has an implicit 'interest' value $s \in \{1, \ldots, 5\}$ *which affects the probability that viewers will click on the video. If* s_A*,* s_B*,* s_C *are the interest levels of the three randomly chosen adverts A, B, C, then the probability that the viewer clicks on advert A is assumed to be proportional to* $\exp(s_A - \max(s_B, s_C))$*. Assuming that the interest levels are a priori independent and uniform and that each viewer clicks on adverts independently of the others (given the advert interest levels), calculate the expected interest levels in each advert.*

Exercise 3.23. *The game RockPaperScissors* `https://en.wikipedia.org/wiki/Rock-paper-scissors` ++
is played by Player1 and Player2. The sequences of moves $x^1_{1:T}, x^2_{1:T}$ *(each move* x^i_t *being one of rock =* r*, paper =* p*, scissors =* s*) by both players is*

```
player1 = [r p r p r s p r s p p r r r r p r s r p p s r]
player2 = [s p r s p s p s r p s r p p r r s p r s s p r]
```

Assume that player1 plays according to a first order Markov chain, defined by

$$p(x^1_{2:T}|x^2_{1:T-1}) = \prod_{t=2}^{T} p(x^1_t|x^1_{t-1}, x^2_{t-1}) \tag{3.7.15}$$

where $x^1_t \in \{\mathsf{r}, \mathsf{p}, \mathsf{s}\}$ *and* $x^2_t \in \{\mathsf{r}, \mathsf{p}, \mathsf{s}\}$ *are the moves at time* t *by player 1 and player 2 respectively.*

1. *Calculate the probability given in equation (3.7.15) for the data above.*
2. *How much more likely is it that player 1 is playing according to this strategy, compared to playing random moves?*

3. *Calculate the probability that player 1 plays each of rock, paper, scissors at the next time step $t = T+1$.*

4. *What would be the best move for player 2 to make at time $t = T + 1$?*

where $T = 23$.

++ **Exercise 3.24.** *Consider the belief network*

$$p_1(x_1, y_1, x_2, y_2, x_3, y_3) \equiv p(x_1|y_1)p(y_1)p(x_2|x_1, y_2)p(y_2)p(x_3|x_2, y_3)p(y_3) \tag{3.7.16}$$

1. *Show that p_1 can be written as*

$$p_2(x_1, y_1, x_2, y_2, x_3, y_3) \equiv p(x_3, y_3)p(x_2, y_2|x_3, y_3)p(x_1, y_1|x_2, y_2) \tag{3.7.17}$$

2. *Show that this can be further simplified to*

$$p_3(x_1, y_1, x_2, y_2, x_3, y_3) = p(y_1|x_1)p(x_1|x_2, y_2)p(y_2|x_2)p(x_2|x_3, y_3)p(x_3, y_3) \tag{3.7.18}$$

3. *Draw the belief networks for p_1 and p_3 and show that the number of edges in p_3 is larger than in p_1 (even though they represent the same distribution). This can be considered a form of causal heuristic. Representation p_1 is the causal one in which root causes are independent. In representation p_3, the number of edges is greater than in p_1. A heuristic for finding the 'right' causal graph is to select the graph (amongst all those that are equally likely) that has the least number of edges.*

++ **Exercise 3.25.** *Consider a 'physical' system on a collection of variables $\mathbf{x}_t = x_{1,t}, \ldots, x_{4,t}$ where $x_{i,t} \in \{0, 1\}$ is state of spacial variable $i \in \{1, 2, 3, 4\}$ at time t. According to the laws of physics, only 'nearest' spatial and temporal variables are relevant for the dynamics. The dynamics is therefore encoded as the 'causal' distribution*

$$\begin{aligned} p(\mathbf{x}_{1:T}) &= p(x_{1,1})p(x_{2,1})p(x_{3,1})p(x_{4,1}) \\ &\times \prod_{t=2}^{T} p(x_{1,t}|x_{1,t-1}, x_{2,t-1})p(x_{2,t}|x_{1,t-1}, x_{2,t-1}, x_{3,t-1})p(x_{3,t}|x_{2,t-1}, x_{3,t-1}, x_{4,t-1})p(x_{4,t}|x_{3,t-1}, x_{4,t-1}) \end{aligned} \tag{3.7.19}$$

1. *Draw a belief network for $p(\mathbf{x}_{1:T})$.*

2. *Show that $p(\mathbf{x}_{1:T})$ can also be written in the 'time reversed' form*

$$p(\mathbf{x}_T) \prod_{t=1}^{T-1} p(\mathbf{x}_t|\mathbf{x}_{t+1}) \tag{3.7.20}$$

and draw a belief network on the collection of variables $\{x_{1,t}, x_{2,t}, x_{3,t}, x_{4,t}\}$, $t = 1, \ldots, T$ for this alternative representation of $p(\mathbf{x}_{1:T})$.

3. *Show that, in general, the number of edges in the 'causal' representation is $10 \times (T - 1)$, so that in the limit $T \to \infty$, the number of edges per timestep is 10. Show that in the time reversed form, the average number of edges per timestep approaches 15 as $T \to \infty$.*

++ **Exercise 3.26.** *Students who take the Machine Learning Masters degree have to take the project module and then select 8 additional modules from the list: Graphical Models, Research Methods, Applied Machine Learning, Natural Language Modelling, Bioinformatics, Information Retrieval, Machine Vision, Advanced Topics, Affective Computing, Computational Modelling. Each student has, a priori, certain skills in Maths, Programming, Organisation, Biology and Writing. These skills are a priori independent and the probability that a student is skilled in Maths is 0.95, Programming 0.8, Organisation 0.7, Biology 0.3, Writing 0.95.*

The probability of passing a module depends on certain skills. The probability of passing some modules depends on two skills, whereas others depend on only one. For example, the probability of passing Graphical

Models is 0.99 if the student is skilled in Maths and Programming, 0.75 is she is skilled in only one of these, and 0.01 if she is skilled in neither. The probability of passing Information Retrieval is 0.995 if the student is skilled in Programming and 0.1 otherwise. Other two-skill modules are Research Methods (depends on Programming and Organisation), Bioinformatics (Biology and Writing), Applied Machine Learning (Maths and Programming), Natural Language Modelling (Maths and Programming) and finally the Project (Writing and Organisation). The remaining modules depend on only one skill: Information Retrieval (depends on Programming), Machine Vision (Maths), Advances Topics (Maths), Affective Computing (Writing) and Computational Modelling (Maths). Assume that the probability of passing the modules are all independent, given the skills of the student. Additionally, assume that each two-skill module has the same probability values as for Graphical Models and each one-skill module has the same probability values as for Information Retrieval.

Provided that the student passes the project and each of the 8 modules she selects, the student will pass the Masters.

1. *If we have no additional information about a student's skills, which modules (in addition to the project) should she take in order to maximise her chance of passing the Masters? State the probability of passing the Masters with this set of modules.*

2. *If we know the student is skilled in Maths and Organisation, which would be the best set of modules to take and the associated probability of passing the Masters?*

3. *If we know the student is skilled in Biology and Writing, which would be the best set of modules to take and the associated probability of passing the Masters?*

CHAPTER 4

Graphical Models

In chapter(3) we saw how belief networks are used to represent statements about independence of variables in a probabilistic model. Belief networks are simply one way to unite probability and graphical representation. Many others exist, all under the general heading of 'graphical models'. Each has specific strengths and weaknesses. Broadly, graphical models fall into two classes: those useful for modelling, such as belief networks, and those useful for inference. This chapter will survey the most popular models from each class.

4.1 Graphical Models

Graphical Models (GMs) are depictions of independence/dependence relationships for distributions. Each class of GM is a particular union of graph and probability constructs and details the form of independence assumptions represented. GMs are useful since they provide a framework for studying a wide class of probabilistic models and associated algorithms. In particular they help to clarify modelling assumptions and provide a unified framework under which inference algorithms in different communities can be related.

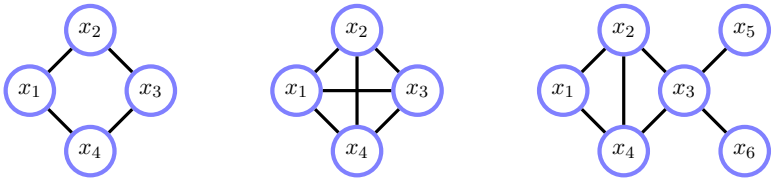
It needs to be emphasised that all forms of GM have a limited ability to graphically express conditional (in)dependence statements[282]. As we've seen, belief networks are useful for modelling ancestral conditional independence. In this chapter we'll introduce other types of GM that are more suited to representing different assumptions. Here we'll focus on Markov networks, chain graphs (which marry Belief and Markov networks) and factor graphs. There are many more inhabitants of the zoo of graphical models, see [74, 315].

The general viewpoint we adopt is to describe the problem environment using a probabilistic model, after which reasoning corresponds to performing probabilistic inference. This is therefore a two part process :

Modelling After identifying all potentially relevant variables of a problem environment, our task is to describe how these variables can interact. This is achieved using structural assumptions as to the form of the joint probability distribution of all the variables, typically corresponding to assumptions of independence of variables. Each class of graphical model corresponds to a factorisation property of the joint distribution.

Inference Once the basic assumptions as to how variables interact with each other is formed (*i.e.* the probabilistic model is constructed) all questions of interest are answered by performing inference on the distribution. This can be a computationally non-trivial step so that coupling GMs with accurate inference algorithms is central to successful graphical modelling.

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



(a) (b) (c)

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)**: $\phi(x_1, x_2, x_4)\phi(x_2, x_3, x_4)\phi(x_3, x_5)\phi(x_3, x_6)/Z_c$.

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

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

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

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

Definition 4.1 (Potential)**.** A potential $\phi(x)$ is a non-negative function of the variable x, $\phi(x) \geq 0$. A joint potential $\phi(x_1, \ldots, x_n)$ 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 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 $\mathcal{X} = \{x_1, \ldots, x_n\}$ a Markov network is defined as a product of potentials on subsets of the variables $\mathcal{X}_c \subseteq \mathcal{X}$:

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

The constant Z ensures the distribution is normalised. Graphically this is represented by an undirected graph G with $\mathcal{X}_c, c = 1, \ldots, C$ 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.

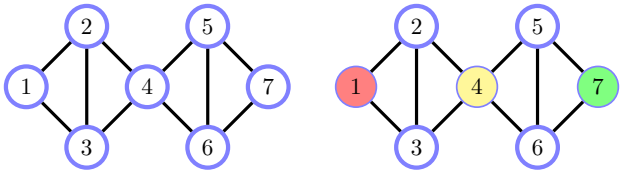
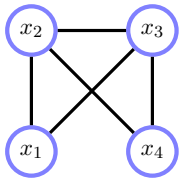


Figure 4.2: **(a)**: $\phi(1,2,3)\phi(2,3,4)\phi(4,5,6)\phi(5,6,7)$. **(b)**: By the global Markov property, since every path from 1 to 7 passes through 4, then $1 \perp\!\!\!\perp 7 | 4$.

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 $p(x_1, x_2, x_3, x_4) = \phi(x_1, x_2)\phi(x_1, x_3)\phi(x_2, x_3)\phi(x_2, x_4)\phi(x_3, x_4)/Z$.

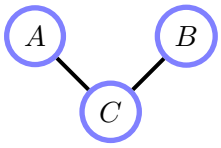


Example 4.1 (*Boltzmann machine*)**.** A Boltzmann machine is a MN on binary variables $\text{dom}(x_i) = \{0, 1\}$ of the form

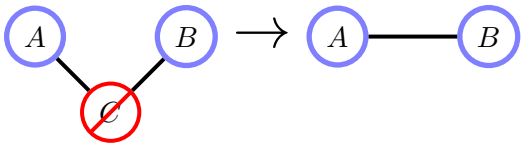
$$p(\mathbf{x}) = \frac{1}{Z(\mathbf{w}, b)} e^{\sum_{i<j} w_{ij} x_i x_j + \sum_i b_i x_i} \tag{4.2.4}$$

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 Boltzmann machine 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.

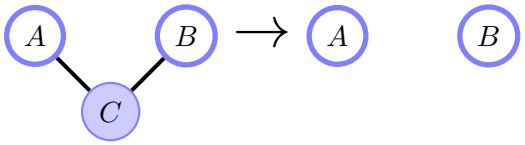
Definition 4.4 (Properties of Markov Networks)**.**



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



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 [183] for detailed proofs. Consider the MN in fig(4.2a) in which we use the shorthand $p(1) \equiv p(x_1)$, $\phi(1, 2, 3) \equiv \phi(x_1, x_2, x_3)$ *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.

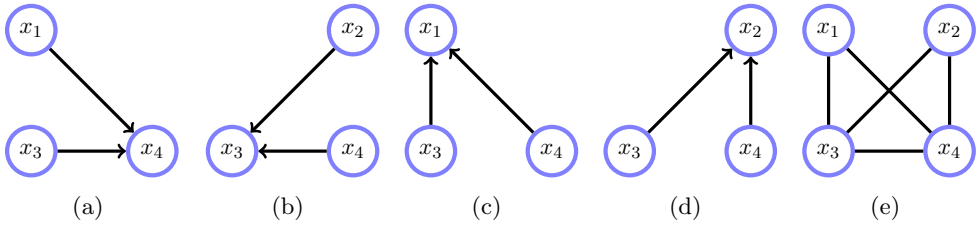


Figure 4.3: **(a-d)**: Local conditional distributions. Note that no distribution is implied for the parents of each variable. That is, in (a) we are given the conditional $p(x_4|x_1, x_3)$ – one should not read from the graph that we imply x_1 and x_3 are marginally independent. **(e)**: The Markov network consistent with the local distributions. If the local distributions are positive, by the Hammersley-Clifford theorem, the only joint distribution that can be consistent with the local distributions must be a Gibbs distribution with structure given by (e).

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 $\mathcal{A} \perp\!\!\!\perp \mathcal{B}|\mathcal{S}$.

As an example of the global Markov property, consider fig(4.2a) for which @@

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

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

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

This implies that $p(1,7|4) = p(1|4)p(7|4)$. This can be inferred since all paths from 1 to 7 pass through 4. @@

Procedure 4.1 (An algorithm for independence)**.** The separation property implies a simple algorithm for deciding $\mathcal{A} \perp\!\!\!\perp \mathcal{B}|\,\mathcal{S}$. 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 $\mathcal{A} \perp\!\!\!\perp \mathcal{B}|\,\mathcal{S}$ is true – see also section(4.2.4).

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

$$p(x|\mathcal{X}\backslash x) = p(x|\mathrm{ne}\,(x)). \tag{4.2.9}$$

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

$$x \perp\!\!\!\perp y|\,\mathcal{X}\backslash\{x, y\}\,. \tag{4.2.10}$$

4.2.2 Markov random fields

A 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 $i \in \{1,\ldots,n\}$ indexes the distributions and $\text{ne}(x_i)$ are the neighbours of variable x_i, namely that subset of the variables $x_1,\ldots,x_n$ 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

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

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

$$p(x_1|x_2,x_3) = p(x_1|x_2) \tag{4.2.12}$$

Hence

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

where the ϕ are potentials.

More generally, for any decomposable graph G, see definition(6.8), 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(4.2a), we can start with the variable x_1 and the corresponding local Markov statement $x_1 \perp\!\!\!\perp x_4, x_5, x_6, x_7 | x_2, x_3$ to write

$$p(x_1,\ldots,x_7) = p(x_1|x_2,x_3)p(x_2,x_3,x_4,x_5,x_6,x_7) \tag{4.2.14}$$

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:

$$p(x_1,x_2,x_3|x_4,x_5,x_6,x_7) = p(x_1,x_2,x_3|x_4) \tag{4.2.15}$$

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

$$p(x_2,x_3,x_4,x_5,x_6,x_7) = p(x_2,x_3|x_4,x_5,x_6,x_7)p(x_4,x_5,x_6,x_7) = p(x_2,x_3|x_4)p(x_4,x_5,x_6,x_7) \tag{4.2.16}$$

and

$$p(x_1,\ldots,x_7) = p(x_1|x_2,x_3)p(x_2,x_3|x_4)p(x_4,x_5,x_6,x_7) \tag{4.2.17}$$

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

$$p(x_1,\ldots,x_7) = p(x_1|x_2,x_3)p(x_2,x_3|x_4)p(x_4|x_5,x_6)p(x_5,x_6|x_7)p(x_7) \tag{4.2.18}$$

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 [183, 37, 220]. 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(4.1a). The theorem states that for positive potentials ϕ, the Markov conditions implied by the graph mean that the distribution must be of the form

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

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

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

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

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

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

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

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

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

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

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

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

The first factor above is independent of x_3, as required by the Markov condition. However, for $\epsilon \neq 0$, the second term varies as a function of x_3. The reason for this is that one can always find a function $\psi(x_1, x_3)$ for which

$$\psi(x_1, x_3) \neq \frac{\sum_{x_1} \phi_{12}(x_1, x_2)\phi_{41}(x_4, x_1)\psi(x_1, x_3)}{\sum_{x_1} \phi_{12}(x_1, x_2)\phi_{41}(x_4, x_1)} \tag{4.2.26}$$

since the term $\psi(x_1, x_3)$ on the left is functionally dependent on x_1 whereas the term on the right is not a function of x_1. Hence, the only way we can ensure the Markov condition holds is if $\epsilon = 0$, namely that there is no connection between x_1 and x_3.

One can generalise this argument to show that if the graph of potentials in the distribution contains a link which is not present in G, then there is some distribution for which a corresponding Markov condition cannot hold. Informally, therefore, $G \Rightarrow F$. The converse $F \Rightarrow G$ is trivial.

The Hammersley-Clifford theorem also helps resolve questions as to when a set of positive local conditional distributions $p(x_i|\text{pa}(x_i))$ could ever form a consistent joint distribution $p(x_1, \ldots, x_n)$. Each local conditional distribution $p(x_i|\text{pa}(x_i))$ corresponds to a factor on the set of variables $\{x_i, \text{pa}(x_i)\}$, so we must include such a term in the joint distribution. The MN can form a joint distribution consistent with the local conditional distributions if and only if $p(x_1, \ldots, x_n)$ factorises according to

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \exp\left(-\sum_c V_c(\mathcal{X}_c)\right) \tag{4.2.27}$$

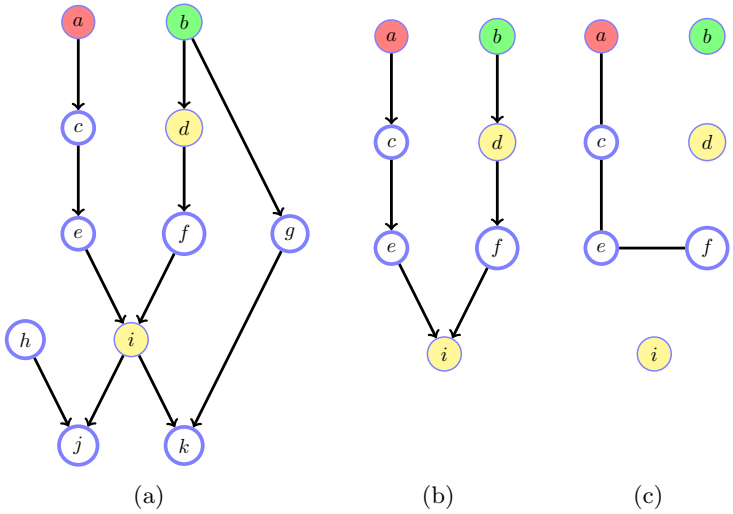


Figure 4.4: **(a)**: Belief network for which we are interested in checking conditional independence $a \perp\!\!\!\perp b | \{d, i\}$. **(b)**: Ancestral graph. **(c)**: Ancestral, moralised and separated graph for $a \perp\!\!\!\perp b \,|\, \{d, i\}$. There is no path from a red to green node so a and b are independent given d, i.

where the sum is over all cliques and $V_c(\mathcal{X}_c)$ is a real function defined over the variables in the clique indexed by c. Equation (4.2.27) is equivalent to $\prod_c \phi(\mathcal{X}_c)$, namely a MN on positive clique potentials. The graph over which the cliques are defined is an undirected graph constructed by taking each local conditional distribution $p(x_i|\text{pa}\,(x_i))$ and drawing a clique on $\{x_i, \text{pa}\,(x_i)\}$. This is then repeated over all the local conditional distributions, see fig(4.3). Note that the HC theorem does not mean that, given a set of conditional distributions, we can always form a consistent joint distribution from them – rather it states what the functional form of a joint distribution has to be for the conditionals to be consistent with the joint, see exercise(4.8).

4.2.4 Conditional independence using Markov networks

For $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$ each being collections of variables, in section(3.3.4) we discussed an algorithm to determine if $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}$ for belief networks. An alternative and more general method (since it handles directed and undirected graphs) uses the procedure below (see [79, 184]). See fig(4.4) for an example.

Procedure 4.2 (Ascertaining independence in Markov and belief networks)**.** For Markov Networks only the final separation criterion needs to be applied:

Ancestral Graph Identify the ancestors $\mathcal{A}$ of the nodes $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$. Retain the nodes $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ but remove all other nodes which are not in $\mathcal{A}$, together with any edges in or out of such nodes.

Moralisation Add a link between any two remaining nodes which have a common child, but are not already connected by an arrow. Then remove remaining arrowheads.

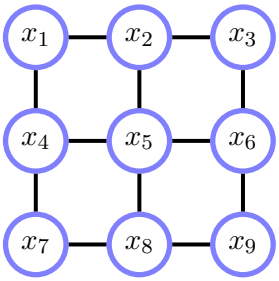
Separation Remove links neighbouring $\mathcal{Z}$. In the undirected graph so constructed, look for a path which joins a node in $\mathcal{X}$ to one in $\mathcal{Y}$. If there is no such path deduce that $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}$.

Note that the ancestral step in procedure(4.2) for belief networks is intuitive since, given a set of nodes $\mathcal{X}$ and their ancestors $\mathcal{A}$, the remaining nodes $\mathcal{D}$ form a contribution to the distribution of the form $p(\mathcal{D}|\mathcal{X}, \mathcal{A})p(\mathcal{X}, \mathcal{A})$, so that summing over $\mathcal{D}$ simply has the effect of removing these variables from the DAG.

4.2.5 Lattice Models

Undirected models have a long history in different branches of science, especially statistical mechanics on lattices and more recently as models in visual processing in which the models encourage neighbouring variables to be in the same states[37, 38, 117].

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



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

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

The Ising model

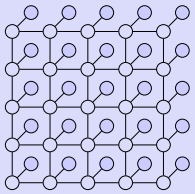
A set of potentials for equation (4.2.28) that encourages neighbouring variables to have the same state is

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

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[42]. 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 $x_i \in \{-1, +1\}$, $i = 1, \ldots, D$. 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

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

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

$$\underset{\mathcal{X}}{\text{argmax}}\, p(\mathcal{X}|\mathcal{Y}) = \underset{\mathcal{X}}{\text{argmax}}\, p(\mathcal{X}, \mathcal{Y}) \tag{4.2.31}$$

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

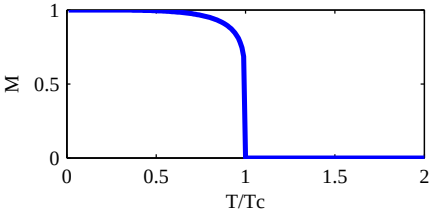


Figure 4.5: Onsager magnetisation. As the temperature T decreases towards the critical temperature T_c a phase transition occurs in which a large fraction of the variables become aligned in the same state.

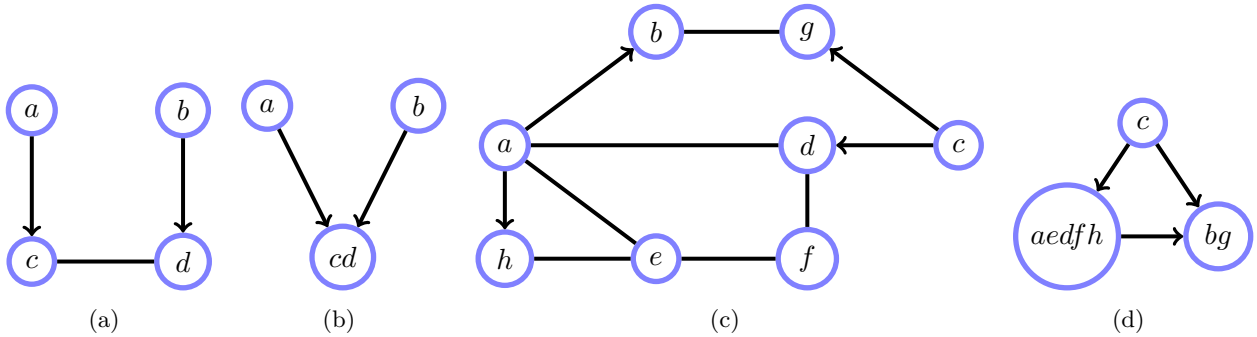
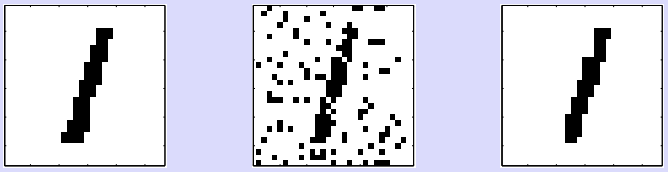


Figure 4.6: Chain graphs. The chain components are identified by deleting the directed edges and identifying the remaining connected components. **(a)**: Chain components are (a),(b),(c, d), which can be written as a BN on the cluster variables in (b). **(c)**: Chain components are $(a, e, d, f, h),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



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 $\beta = -\frac{1}{2}\sigma^{-1}(p_{corrupt})$. 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 $\alpha = 10$, $p_{corrupt} = 0.15$.

4.3 Chain Graphical Models

Chain Graphs (CGs) contain both directed and undirected links. To develop the intuition, consider fig(4.6a). 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

$$p(a, b, c, d) = p(a)p(b)p(c, d|a, b) \tag{4.3.1}$$

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

$$p(c, d|a, b) = \phi(c, d)p(c|a)p(d|b) \tag{4.3.2}$$

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

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

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

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 and their associated variables $\mathcal{X}_\tau$. Then @@

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

and

$$p\left(\mathcal{X}_\tau | \text{pa}\left(\mathcal{X}_\tau\right)\right) \propto \prod_{d \in \mathcal{D}_\tau} p(x_d | \text{pa}\left(x_d\right)) \prod_{c \in \mathcal{C}_\tau} \phi\left(\mathcal{X}_c\right) \tag{4.3.5}$$ @@

where $\mathcal{C}_\tau$ denotes the union of the cliques in component τ with ϕ being the associated functions defined on each clique; $\mathcal{D}_\tau$ is the set of variables in component τ that correspond to directed terms $p(x_d|\text{pa}\left(x_d\right))$. 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. CGs can be useful since they are more expressive of CI statements than either belief networks or Markov networks alone. The reader is referred to [183] and [107] for further details.

Example 4.3 (Chain graphs are more expressive than Belief or Markov networks)**.** Consider the chain graph in fig(4.7a), which has chain component decomposition

$$p(a, b, c, d, e, f) = p(a)p(b)p(c, d, e, f|a, b) \tag{4.3.6}$$

where

$$p(c, d, e, f|a, b) = p(c|a)\phi(c, e)\phi(e, f)\phi(d, f)p(d|b)\phi(a, b) \tag{4.3.7}$$ @@

with the normalisation requirement

$$\phi(a, b) \equiv \left(\sum_{c,d,e,f} p(c|a)\phi(c, e)\phi(e, f)\phi(d, f)p(d|b) \right)^{-1} \tag{4.3.8}$$ @@

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

$$\phi(c, e)\phi(e, f)\phi(d, f) \underbrace{\sum_{a,b} \phi(a, b)p(a)p(b)p(c|a)p(d|b)}_{\phi(c,d)} \tag{4.3.9}$$ @@

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(4.7a) could express that a and b are independent (unconditionally), *i.e.* $p(a, b) = p(a)p(b)$.

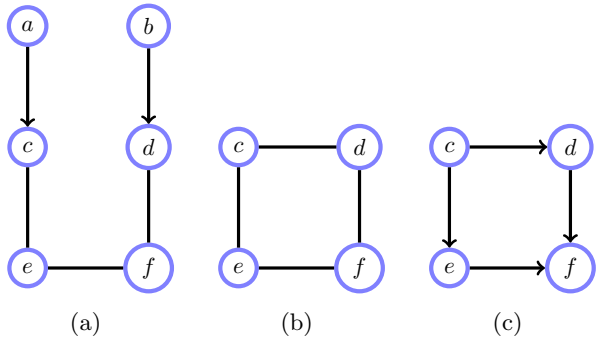


Figure 4.7: The CG (a) expresses $a \perp\!\!\!\perp b | \emptyset$ and $d \perp\!\!\!\perp e | (c, f)$. No directed graph could express both these conditions since the marginal distribution $p(c, d, e, f)$ is an undirected four cycle, (b). Any DAG on a 4 cycle must contain a collider, as in (c) and 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.

4.4 Factor Graphs

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

Definition 4.10 (*Factor Graph*)**.** Given a function

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

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.

When used to represent a distribution

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

++ a normalisation constant $Z = \sum_{\mathcal{X}} \prod_i \psi_i(\mathcal{X}_i)$ is assumed. Here $\mathcal{X}$ represents all variables in the distribution.

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

$$p(a, b, c) = \phi(a, b)\phi(a, c)\phi(b, c) \tag{4.4.3}$$

Represented as a MN, this must have a single clique, as given in fig(4.8c). However, fig(4.8c) 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(4.8b) 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(4.8a). 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(4.8d) 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(4.8e). 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 [104] for details.

[1]Formally a FG is an alternative graphical depiction of a hypergraph[87] 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.

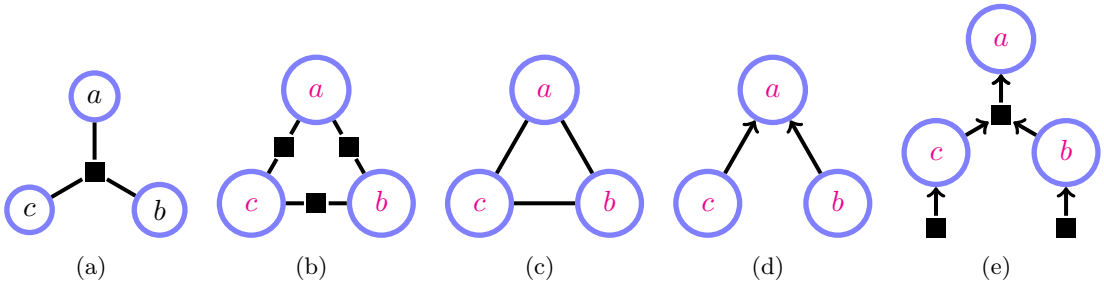


Figure 4.8: **(a)**: $\phi(a, b, c)$. **(b)**: $\phi(a, b)\phi(b, c)\phi(c, a)$. **(c)**: $\phi(a, b, c)$. Both (a) and (b) have the same undirected graphical model, (c). **(d)**: (a) is an undirected FG of (d). **(e)**: Directed FG of the BN in (d). A directed factor represents a term $p(\text{children}|\text{parents})$. 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).

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[104]. To determine whether two variables are independent given a set of conditioned variables, consider all paths connecting the two variables. If all paths are blocked, the variables are conditionally independent. A path is blocked if one or more of the following conditions is satisfied:

- One of the variables in the path is in the conditioning set.
- One of the variables or factors in the path has two incoming edges that are part of the path (variable or factor collider), and neither the variable or factor nor any of its descendants are in the conditioning set.

4.5 Expressiveness of Graphical Models

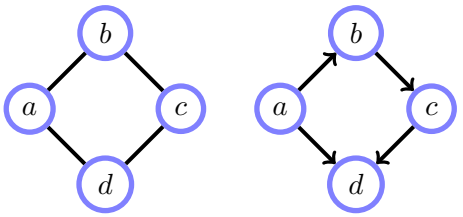
It is clear that directed distributions can be represented as undirected distributions since one can associate each (normalised) factor of the joint distribution with a potential. For example, the distribution $p(a|b)p(b|c)p(c)$ can be factored as $\phi(a, b)\phi(b, c)$, where $\phi(a, b) = p(a|b)$ and $\phi(b, c) = p(b|c)p(c)$, with $Z = 1$. Hence every belief network can be represented as some MN by simple identification of the factors in the distributions. However, in general, the associated undirected graph (which corresponds to the moralised directed graph) will contain additional links and independence information can be lost. For example, the MN of $p(c|a, b)p(a)p(b)$ is a single clique $\phi(a, b, c)$ from which one cannot graphically infer that $a \perp\!\!\!\perp b$.

The converse question is whether every undirected model can be represented by a BN with a readily derived link structure. Consider the example in fig(4.9). In this case, there is no directed model with the same link structure that can express the (in)dependencies in the undirected graph. Naturally, every probability distribution can be represented by some BN though it may not necessarily have a simple structure and be a 'fully connected' cascade style graph. In this sense the DAG cannot always graphically represent the independence properties that hold for the undirected distribution.

Definition 4.11 (Independence Maps)**.** A graph is an *independence map* (I-map) of a given distribution P if every conditional independence statement that one can derive from the graph G is true in the distribution P. That is

$$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}_G \Rightarrow \mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}_P \tag{4.5.1}$$

for all disjoint sets $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$.



(a) (b)

Figure 4.9: **(a)**: An undirected model for which we wish to find a directed equivalent. **(b)**: Every DAG with the same structure as the undirected model must have a situation where two arrows will point to a node, such as node d (otherwise one would have a cyclic graph). Summing over the states of variable d will leave a DAG on the variables a, b, c with no link between a and c. This cannot represent the undirected model since when one marginalises over d this adds a link between a and c.

Similarly, a graph is a *dependence map* (D-map) of a given distribution P if every conditional independence statement that one can derive from P is true in the graph G. That is

$$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}_G \Leftarrow \mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}_P \tag{4.5.2}$$

for all disjoint sets $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$.

A graph G which is both an I-map and a D-map for P is called a *perfect map* and

$$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}_G \Leftrightarrow \mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}_P \tag{4.5.3}$$

for all disjoint sets $\mathcal{X}$,$\mathcal{Y}$,$\mathcal{Z}$. In this case, the set of all conditional independence and dependence statements expressible in the graph G are consistent with P and *vice versa*.

Note that by contraposition, a dependence map is equivalent to

$$\mathcal{X} \top\!\!\!\top \mathcal{Y} | \mathcal{Z}_G \Rightarrow \mathcal{X} \top\!\!\!\top \mathcal{Y} | \mathcal{Z}_P \tag{4.5.4}$$

meaning that if $\mathcal{X}$ and $\mathcal{Y}$ are graphically dependent given $\mathcal{Z}$, then they are dependent in the distribution.

One way to think about this is to take a distribution P and write out a list $\mathcal{L}_P$ of all the independence statements. For a graph G, one writes a list of all the possible independence statements $\mathcal{L}_G$. Then:

$$\begin{array}{ll} \mathcal{L}_P \subseteq \mathcal{L}_G & \text{Dependence Map (D-map)} \\ \mathcal{L}_P \supseteq \mathcal{L}_G & \text{Independence Map (I-map)} \\ \mathcal{L}_P = \mathcal{L}_G & \text{Perfect Map} \end{array} \tag{4.5.5}$$

In the above we assume the statement l is contained in $\mathcal{L}$ if it is consistent with (can be derived from) the independence statements in $\mathcal{L}$.

One can also discuss whether or not a distribution class has an associated map. That is whether or not all numerical instances of distributions consistent with the specified form obey the constraints required for the map. To do so we take any numerical instance of a distribution P_i consistent with a given class P and write out a list $\mathcal{L}_{P_i}$ of all the independence statements. One then takes the intersection $\mathcal{L}_P = \cap_i \mathcal{L}_{P_i}$ of all the lists from all possible distribution instances. This list is then used in equation (4.5.5) to determine if there is an associated map. For example the distribution class

$$p(x, y, z) = p(z|x, y)p(x)p(y) \tag{4.5.6}$$

has a directed perfect map $x \to z \leftarrow y$. However, the undirected graph for the class equation (4.5.6) is fully connected so that $\mathcal{L}_G$ is empty. For any distribution consistent with equation (4.5.6) x and y are independent, a statement which is not contained in $\mathcal{L}_G$ – hence there is no undirected D-map and hence no perfect undirected map for the class represented in equation (4.5.6).

Example 4.4. Consider the distribution (class) defined on variables t_1, t_2, y_1, y_2[250]:

$$p(t_1, t_2, y_1, y_2) = p(t_1)p(t_2) \sum_h p(y_1|t_1, h)p(y_2|t_2, h)p(h) \tag{4.5.7}$$

In this case the list of all independence statements (for all distribution instances consistent with p) is

$$\mathcal{L}_P = \{y_1 \perp\!\!\!\perp t_2 | t_1,\ \ y_2 \perp\!\!\!\perp t_1 | t_2,\ \ t_2 \perp\!\!\!\perp t_1\} \tag{4.5.8}$$ @@

Consider the graph of the BN

$$p(y_2|y_1,t_1,t_2)p(y_1|t_1)p(t_1)p(t_2) \tag{4.5.9}$$ @@

For this we have

$$\mathcal{L}_G = \{y_1 \perp\!\!\!\perp t_2 | t_1,\ \ t_2 \perp\!\!\!\perp t_1\} \tag{4.5.10}$$ @@

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 a 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

$$p(x,y,z) = \phi(x,y)\phi(y,z)/Z_1 \tag{4.5.11}$$

with $\phi(x,y) = \mathsf{const}$. 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

$$\phi(x,y) = \frac{x}{y}, \qquad \phi(y,z) = yz \tag{4.5.12}$$

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,

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

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 $x \not\perp\!\!\!\perp y$ and $y \not\perp\!\!\!\perp z$, 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 [48] and [281].

4.7 Code

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

4.8 Exercises

Exercise 4.1. *1. Consider the pairwise Markov network,*

$$p(x) = \phi(x_1, x_2)\phi(x_2, x_3)\phi(x_3, x_4)\phi(x_4, x_1) \tag{4.8.1}$$

Express in terms of ϕ the following:

$$p(x_1|x_2, x_4), \qquad p(x_2|x_1, x_3), \qquad p(x_3|x_2, x_4), \qquad p(x_4|x_1, x_3) \tag{4.8.2}$$

2. For a set of local distributions defined as

$$p_1(x_1|x_2, x_4), \qquad p_2(x_2|x_1, x_3), \qquad p_3(x_3|x_2, x_4), \qquad p_4(x_4|x_1, x_3) \tag{4.8.3}$$

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

Exercise 4.2. *Consider the Markov network*

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

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.

Exercise 4.3. *Show that for the Boltzmann machine defined on binary variables x_i with*

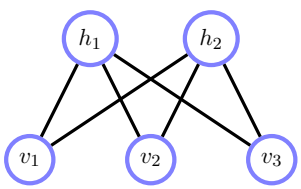
$$p(\mathbf{x}) = \frac{1}{Z(\mathbf{W}, \mathbf{b})} \exp\left(\mathbf{x}^\mathsf{T}\mathbf{W}\mathbf{x} + \mathbf{x}^\mathsf{T}\mathbf{b}\right) \tag{4.8.5}$$

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

Exercise 4.4.

The restricted Boltzmann machine *(or* Harmonium*[270]) is a constrained Boltzmann machine on a bipartite graph, consisting of a layer of visible variables $\mathbf{v} = (v_1, \ldots, v_V)$ and hidden variables $\mathbf{h} = (h_1, \ldots, h_H)$:*

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



All variables are binary taking states $0, 1$.

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

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

   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* $p(\mathbf{h}) = \prod_i p(h_i)$*?* @@

4. *Can the partition function* $Z(\mathbf{W}, \mathbf{a}, \mathbf{b})$ *be computed efficiently for the RBM?*

Exercise 4.5. *You are given that*

$$x \perp\!\!\!\perp y | (z, u), \qquad u \perp\!\!\!\perp z | \emptyset \tag{4.8.8}$$

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?*

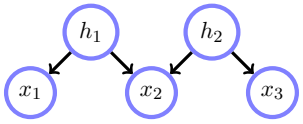
Exercise 4.6. *The undirected graph*  *represents a Markov network with nodes* x_1, x_2, x_3, x_4, x_5*, counting clockwise around the pentagon with potentials* $\phi(x_i, x_j)$*. Show that the joint distribution can be written as*

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

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

Exercise 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* $p(x_1, x_2, x_3)$*?*

Exercise 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.*

Exercise 4.9. *Research lab A states its findings about a set of variables* $x_1, \ldots, x_n$ *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?*

Exercise 4.10.
Consider the distribution

$$p(x, y, w, z) = p(z|w)p(w|x, y)p(x)p(y) \tag{4.8.10}$$

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.*

Exercise 4.11. *Consider the distribution*

$$p(t_1, t_2, y_1, y_2, h) = p(y_1|y_2, t_1, t_2, h)p(y_2|t_2, h)p(t_1)p(t_2)p(h) \tag{4.8.11}$$

1. *Draw a belief network for this distribution.*

2. *Does the distribution*

$$p(t_1, t_2, y_1, y_2) = \sum_h p(y_1|y_2, t_1, t_2, h)p(y_2|t_2, h)p(t_1)p(t_2)p(h) \tag{4.8.12}$$

have a perfect map belief network?

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

Exercise 4.12. *Consider the distribution*

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

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 $a \perp\!\!\!\perp c | \emptyset$.*

Exercise 4.13. *Show how for any singly-connected Markov network, one may construct a Markov equivalent belief network.*

Exercise 4.14. *Consider a pairwise binary Markov network defined on variables $s_i \in \{0, 1\}$, $i = 1, \ldots, N$, with $p(s) = \prod_{ij \in \mathcal{E}} \phi_{ij}(s_i, s_j)$, 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.*

++ **Exercise 4.15.** *Our interest here is to show that it is not always possible to find a joint distribution p that would give rise to a set of specified consistent distributions $\{q\}$ on subsets of the variables.*

1. *We first wish to construct a set of distributions $q_{12}(x_1, x_2)$, $q_{13}(x_1, x_3)$, $q_{23}(x_2, x_3)$ on binary variables $x_i \in \{0, 1\}$ that are 'marginally' consistent; that is*

$$\sum_{x_2} q_{12}(x_1, x_2) = \sum_{x_3} q_{13}(x_1, x_3)$$
$$\sum_{x_1} q_{12}(x_1, x_2) = \sum_{x_3} q_{23}(x_2, x_3)$$
$$\sum_{x_1} q_{13}(x_1, x_3) = \sum_{x_2} q_{23}(x_2, x_3)$$

with all the q being distributions (non negative and summing to 1). By writing $q_{12}(x_1 = 0, x_2 = 0) = y_1$, $q_{12}(x_1 = 0, x_2 = 1) = y_2$, etc. , show that the above equations can be represented as a linear system

$$\mathbf{My} = \mathbf{c}, \qquad 0 \leq y_i \leq 1$$

where $\mathbf{M}$ is a suitably defined 9×12 matrix and $\mathbf{c}$ is a suitably defined 0 dimensional vector. Hence by solving this system using linear programming (or otherwise), find a set of marginally consistent distributions q.

2. *Given a set of marginally consistent q, our interest is to see if we can find a joint distribution p which would give rise to these marginals q. That is*

$$\sum_{x_1} p(x_1, x_2, x_3) = q_{23}(x_2, x_3), \quad \sum_{x_2} p(x_1, x_2, x_3) = q_{13}(x_1, x_3), \quad \sum_{x_3} p(x_1, x_2, x_3) = q_{12}(x_1, x_2)$$

Show that, by writing the 8 states of p as $z_1, \ldots, z_8$, the above can be expressed as the linear system

$$\mathbf{Az} = \mathbf{y}, \qquad 0 \leq z_i \leq 1, \sum_i z_i = 1$$

where $\mathbf{A}$ is a suitably defined 12×8 matrix. For a marginally consistent $\mathbf{y}$ show numerically that it is not always possible to find a joint distribution $\mathbf{z}$ that would give rise to these marginals.

The set of marginals consistent with a distribution p is called the marginal polytope of p. The question we are therefore asking here is whether a given marginal set q is in the marginal polytope of p. Whilst this is straightforward to solve (using linear-programming for example), for an n-variable system the size of the linear system will be exponential in n, thus resulting in a computationally hard problem.

This issue is important for at least two reasons: (i) Imagine that different research labs are asked to examine different aspects of a problem, each lab summarising their results as a distribution on a subset of the variables. The question is whether there exists a single joint distribution that is consistent with all these marginals. (ii) In certain deterministic approximation schemes (see section(28.7.2)) the objective depends only on a set of marginals and the question is how to characterise the marginal polytope. As this question intimates, this is generally very difficult. However, as we will see in chapter(6), as least for singly-connected structures, representing a distribution in terms of locally consistent marginals is straightforward.

++

Exercise 4.16. *The question concerns cleaning up the binary image given in the file* `xnoisy.mat`. *The objective function to maximise is*

$$\sum_{i,j} W_{i,j} \mathbb{I}\left[x_i = x_j\right] + \sum_i b_i x_i, \qquad x_i \in \{0, 1\}$$

where $W_{i,j} = 10$ for neighbouring pixels (up, down, left, right) in the image and $W_{i,j} = 0$ otherwise. Using y to represent the noisy image, $b_i = 2y_i - 1$. Plot the final cleaned up image x and give the maximum objective function found for this cleaned up image.

CHAPTER 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 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 $p(x_1, \ldots, x_n)$, *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 $x_1 = \mathsf{tr}$, a marginal inference calculation is

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

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

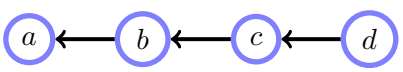


Figure 5.1: A Markov chain is of the form $p(x_T)\prod_{t=1}^{T-1} p(x_t|x_{t+1})$ for some assignment of the variables to labels x_t. Variable Elimination can be carried out in time linear in the number of variables in the chain.

discussed in more depth in section(23.1))

$$p(a,b,c,d) = p(a|b)p(b|c)p(c|d)p(d) \tag{5.1.2}$$

as given in fig(5.1), for which our task is to calculate the marginal $p(a)$. For simplicity, we assume that each of the variables has domain $\{0,1\}$. Then

$$p(a=0) = \sum_{b\in\{0,1\},c\in\{0,1\},d\in\{0,1\}} p(a=0,b,c,d) = \sum_{b\in\{0,1\},c\in\{0,1\},d\in\{0,1\}} p(a=0|b)p(b|c)p(c|d)p(d) \tag{5.1.3}$$

We could carry out this computation by simply summing each of the probabilities for the $2\times 2\times 2 = 8$ states of the variables b, c and d. This would therefore require 7 addition-of-two-numbers calls.

A more efficient approach is to push the summation over d as far to the right as possible:

$$p(a=0) = \sum_{b\in\{0,1\},c\in\{0,1\}} p(a=0|b)p(b|c) \underbrace{\sum_{d\in\{0,1\}} p(c|d)p(d)}_{\gamma_d(c)} \tag{5.1.4}$$

where $\gamma_d(c)$ is a (two state) potential. Defining $\gamma_d(c)$ requires two addition-of-two-numbers calls, one call for each state of c. Similarly, we can distribute the summation over c as far to the right as possible:

$$p(a=0) = \sum_{b\in\{0,1\}} p(a=0|b) \underbrace{\sum_{c\in\{0,1\}} p(b|c)\gamma_d(c)}_{\gamma_c(b)} \tag{5.1.5}$$

Then, finally,

$$p(a=0) = \sum_{b\in\{0,1\}} p(a=0|b)\gamma_c(b) \tag{5.1.6}$$

By distributing the summations we have made $3\times 2 - 1 = 5$ addition-of-two-numbers calls, compared to @@ $2^3 - 1 = 7$ from the naive approach. Whilst this saving may not appear much, the important point is that the number of computations for a chain of length $T+1$ would be linear, $2T$, as opposed to exponential, $2^T - 1$ for the naive approach.

This procedure is called *variable elimination* since each time we sum over the states of a variable we eliminate it from the distribution. We can always perform variable elimination in a chain efficiently since there is a natural way to distribute the summations, working inwards from the edges. Note that in the above case, the potentials are in fact always distributions – we are just recursively computing the marginal distribution of the right leaf of the chain.

One can view the elimination of a variable as passing a *message* (information) to a neighbouring node on the graph. We can calculate a univariate-marginal of any tree (singly connected graph) by starting at a leaf of the tree, eliminating the variable there, and then working inwards, nibbling off each time a leaf of the remaining tree. Provided we perform elimination from the leaves inwards, then the structure of the remaining graph is simply a subtree of the original tree, albeit with the conditional probability table entries modified. This is guaranteed to enable us to calculate any marginal $p(x_i)$ using a number of summations which scales linearly with the number of variables in the tree.

Finding conditional marginals for a chain

Consider the following inference problem, fig(5.1) : Given

$$p(a,b,c,d) = p(a|b)p(b|c)p(c|d)p(d), \tag{5.1.7}$$

find $p(d|\mathsf{a})$. This can be computed using

$$p(d|\mathsf{a}) \propto \sum_{b,c} p(\mathsf{a},b,c,d) = \sum_{b,c} p(\mathsf{a}|b)p(b|c)p(c|d)p(d) = \sum_c \underbrace{\sum_b p(\mathsf{a}|b)p(b|c)}_{\gamma_b(c)} p(c|d)p(d) \equiv \gamma_c(d) \tag{5.1.8}$$

The missing proportionality constant is found by repeating the computation for all states of variable d. Since we know that $p(d|\mathsf{a}) = k\gamma_c(d)$, where $\gamma_c(d)$ is the unnormalised result of the summation, we can use the fact that $\sum_d p(d|\mathsf{a}) = 1$ to infer that $k = 1/\sum_d \gamma_c(d)$.

In this example, the potential $\gamma_b(c)$ is not a distribution in c, nor is $\gamma_c(d)$. In general, one may view variable elimination as the passing of messages in the form of potentials from nodes to their neighbours. For belief networks variable elimination passes messages that are distributions when following the direction of the edge, and non-normalised potentials when passing messages against the direction of the edge.

Remark 5.1 (Variable elimination in trees as matrix multiplication)**.** Variable elimination is related to the associativity of matrix multiplication. For equation (5.1.2) above, we can define matrices

$$\begin{aligned}&[\mathbf{M}_{ab}]_{i,j} = p(a = i|b = j), \quad [\mathbf{M}_{bc}]_{i,j} = p(b = i|c = j),\\ &[\mathbf{M}_{cd}]_{i,j} = p(c = i|d = j), \quad [\mathbf{M}_d]_i = p(d = i), \quad [\mathbf{M}_a]_i = p(a = i)\end{aligned} \tag{5.1.9}$$

Then the marginal $\mathbf{M}_a$ can be written

$$\mathbf{M}_a = \mathbf{M}_{ab}\mathbf{M}_{bc}\mathbf{M}_{cd}\mathbf{M}_d = \mathbf{M}_{ab}(\mathbf{M}_{bc}(\mathbf{M}_{cd}\mathbf{M}_d)) \tag{5.1.10}$$

since matrix multiplication is associative. This matrix formulation of calculating marginals is called the *transfer matrix* method, and is particularly popular in the physics literature[27].

Example 5.1 (Where will the fly be?)**.**

You live in a house with three rooms, labelled $1, 2, 3$. There is a door between rooms 1 and 2 and another between rooms 2 and 3. One cannot directly pass between rooms 1 and 3 in one time-step. An annoying fly is buzzing from one room to another and there is some smelly cheese in room 1 which seems to attract the fly more. Using x_t to indicate which room the fly is in at time t, with $\mathrm{dom}(x_t) = \{1, 2, 3\}$, the movement of the fly can be described by a transition

$$p(x_{t+1} = i|x_t = j) = M_{ij} \tag{5.1.11}$$

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

$$\mathbf{M} = \begin{pmatrix} 0.7 & 0.5 & 0 \\ 0.3 & 0.3 & 0.5 \\ 0 & 0.2 & 0.5 \end{pmatrix} \tag{5.1.12}$$

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

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

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

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

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

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

where $\mathbf{v}$ is a vector with components $(1,0,0)^\mathsf{T}$, reflecting the evidence that at time $t=1$ the fly is in room 1. Computing this we have (to 4 decimal places of accuracy)

$$\mathbf{M}^4\mathbf{v} = \begin{pmatrix} 0.5746 \\ 0.3180 \\ 0.1074 \end{pmatrix} \tag{5.1.16}$$

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 time-steps. That is, we are interested in the large t behaviour of

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

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

$$\mathbf{p} = \mathbf{M}\mathbf{p} \tag{5.1.18}$$

In other words, $\mathbf{p}$ is the eigenvector of $\mathbf{M}$ with eigenvalue 1[135]. 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 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

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

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

$$p(a,b,c) = \sum_d p(a,b,c,d) = \sum_d f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d) = f_1(a,b)\, f_2(b,c) \underbrace{\sum_d f_3(c,d)\, f_4(d)}_{\mu_{d\to c}(c)} \tag{5.1.20}$$

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

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

Hence

$$\mu_{c\to b}(b) = \sum_c f_2(b,c)\, \mu_{d\to c}(c) \tag{5.1.22}$$

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 $\mu_{c\to b}(b)$ can be interpreted as carrying

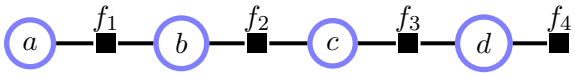


Figure 5.2: For singly connected structures without branches, simple messages from one variable to its neighbour may be defined to form an efficient marginal inference scheme.

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,

$$p(a|b)p(b|c,d)p(c)p(d)p(e|d) \tag{5.1.23}$$

has the factor graph depicted in fig(5.3)

$$f_1(a,b)\, f_2(b,c,d)\, f_3(c)\, f_4(d,e)\, f_5(d) \tag{5.1.24}$$

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

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

where $\mu_{f_2\to b}(b)$ 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

$$\mu_{f_2\to b}(b) = \sum_{c,d} f_2(b,c,d) \underbrace{f_3(c)}_{\mu_{c\to f_2}(c)} \underbrace{f_5(d) \sum_e f_4(d,e)}_{\mu_{d\to f_2}(d)} \tag{5.1.26}$$

Similarly, we can interpret

$$\mu_{d\to f_2}(d) = \underbrace{f_5(d)}_{\mu_{f_5\to d}(d)} \underbrace{\sum_e f_4(d,e)}_{\mu_{f_4\to d}(d)} \tag{5.1.27}$$

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

$$p(a) = \underbrace{\sum_b f_1(a,b)\, \mu_{f_2\to b}(b)}_{\mu_{f_1\to a}(a)} \tag{5.1.28}$$

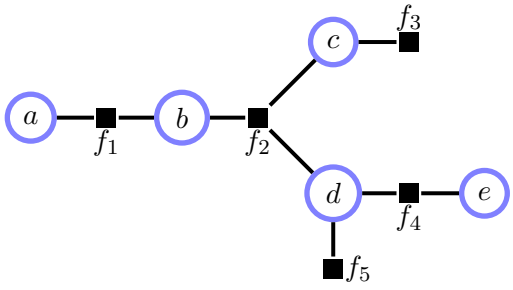


Figure 5.3: For a branching singly connected graph, it is useful to define messages from both factors to variables, and variables to factors.

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

$$\mu_{f_1\to a}(a) = \sum_b f_1(a,b) \underbrace{\mu_{f_2\to b}(b)}_{\mu_{b\to f_1}(b)} \tag{5.1.29}$$

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

$$p(b) = \underbrace{\sum_a f_1(a,b)}_{\mu_{f_1\to b}(b)} \mu_{f_2\to b}(b) \tag{5.1.30}$$

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

$$\mu_{f_2\to c}(c) = \sum_{b,d} f_2(b,c,d)\, \mu_{b\to f_2}(b)\, \mu_{d\to f_2}(d) \tag{5.1.31}$$

where $\mu_{b\to f_2}(b) \equiv \mu_{f_1\to b}(b)$. 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.

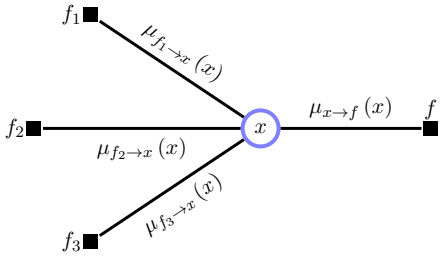
Procedure 5.1 (Sum-Product messages on Factor Graphs)**.**

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

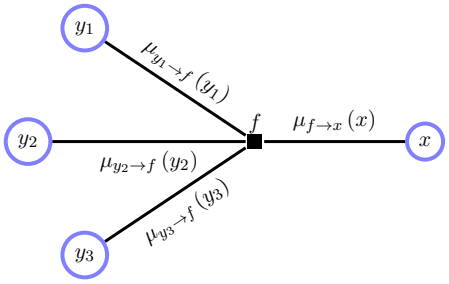
$$\mu_{x\to f}(x) = \prod_{g\in\{\mathrm{ne}(x)\setminus f\}} \mu_{g\to x}(x)$$



Factor to Variable message

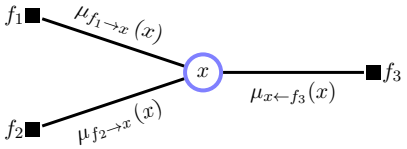
$$\mu_{f\to x}(x) = \sum_{\mathcal{X}_f\backslash x} \phi_f(\mathcal{X}_f) \prod_{y\in\{\text{ne}(f)\backslash x\}} \mu_{y\to f}(y)$$

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



Marginal

$$p(x) \propto \prod_{f\in\text{ne}(x)} \mu_{f\to x}(x)$$



For marginal inference, the important information is the relative size of the message states so that we may renormalise messages as we wish. Since the marginal will be proportional to the incoming messages for that variable, the normalisation constant is trivially obtained using the fact that the marginal must sum to 1. However, if we wish to also compute any normalisation constant using these messages, we cannot normalise the messages since this global information will then be lost.

5.1.3 Dealing with Evidence

For a distribution which splits into evidential and non-evidential variables, $\mathcal{X} = \mathcal{X}_e \cup \mathcal{X}_n$, the marginal of a non-evidential variable $p(x_i, \mathcal{X}_e)$ is given by summing over all the variables in $\mathcal{X}_n$ (except for x_i) with $\mathcal{X}_e$ set into their evidential states. There are two ways to reconcile this with the factor graph formalism. Either we can simply say that by setting the variables $\mathcal{X}_e$, we define a new factor graph on $\mathcal{X}_n$, and then pass messages on this new factor graph. Alternatively, we can define the potentials which contain the variables $\mathcal{X}_e$ by multiplying each potential that contains an evidential variable by a delta function (an indicator) that is zero unless the variable x_e is in the specified evidential state. When we then perform a factor to variable message, the sum of this modified potential over any of the evidential variable states will be zero except for that state which corresponds to the evidential setting. Another way to view this is that the sum in the factor to variable message is only over the non-evidential variables, with any evidential variables in the potential set into their evidential states.

5.1.4 Computing the marginal likelihood

For a distribution defined as products over potentials $\phi_f(\mathcal{X}_f)$

$$p(\mathcal{X}) = \frac{1}{Z}\prod_f \phi_f(\mathcal{X}_f) \tag{5.1.32}$$

the normalisation is given by

$$Z = \sum_{\mathcal{X}} \prod_f \phi_f(\mathcal{X}_f) \tag{5.1.33}$$

To compute this summation efficiently we take the product of all incoming messages to an arbitrarily chosen variable x and then sum over the states of that variable:

$$Z = \sum_x \prod_{f\in\text{ne}(x)} \mu_{f\to x}(x) \tag{5.1.34}$$

If the factor graph is derived from setting a subset of variables of a BN in evidential states then the summation over all non-evidential variables will yield the marginal on the visible (evidential) variables. For example

$$p(\mathsf{b},\mathsf{d}) = \sum_{a,c} p(a|\mathsf{b})p(\mathsf{b}|c)p(c|\mathsf{d})p(\mathsf{d}). \tag{5.1.35}$$

This can be interpreted as requiring the sum over a product of suitably defined factors. Hence one can readily find the marginal likelihood of the evidential variables for singly connected BNs.

Log messages

For the above method to work, the absolute (not relative) values of the messages are required, which prohibits renormalisation at each stage of the message passing procedure. However, without normalisation the numerical value of messages can become very small, particularly for large graphs, and numerical precision issues can occur. A remedy in this situation is to work with log messages,

$$\lambda = \log \mu \tag{5.1.36}$$

For this, the variable to factor messages

$$\mu_{x\to f}(x) = \prod_{g\in\{\mathrm{ne}(x)\setminus f\}} \mu_{g\to x}(x) \tag{5.1.37}$$

become simply

$$\lambda_{x\to f}(x) = \sum_{g\in\{\mathrm{ne}(x)\setminus f\}} \lambda_{g\to x}(x) \tag{5.1.38}$$

More care is required for the factors to variable messages, which are defined by

$$\mu_{f\to x}(x) = \sum_{\mathcal{X}_f\setminus x} \phi_f(\mathcal{X}_f) \prod_{y\in\{\mathrm{ne}(f)\setminus x\}} \mu_{y\to f}(y) \tag{5.1.39}$$

Naively, one may write

$$\lambda_{f\to x}(x) = \log\left(\sum_{\mathcal{X}_f\setminus x} \phi_f(\mathcal{X}_f)\exp\left(\sum_{y\in\{\mathrm{ne}(f)\setminus x\}} \lambda_{y\to f}(y)\right)\right) \tag{5.1.40}$$

However, the exponentiation of the log messages will cause potential numerical precision problems. A solution to this numerical difficulty is obtained by finding the largest value of the incoming log messages,

$$\lambda^*_{y\to f} = \max_{y\in\{\mathrm{ne}(f)\setminus x\}} \lambda_{y\to f}(y) \tag{5.1.41}$$

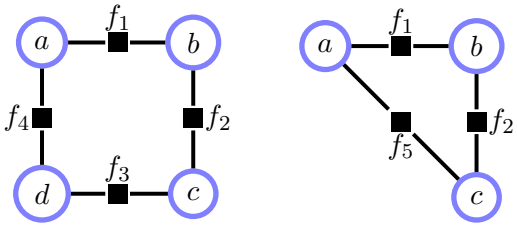
Then

$$\lambda_{f\to x}(x) = \lambda^*_{y\to f} + \log\left(\sum_{\mathcal{X}_f\setminus x} \phi_f(\mathcal{X}_f)\exp\left(\sum_{y\in\{\mathrm{ne}(f)\setminus x\}} \lambda_{y\to f}(y) - \lambda^*_{y\to f}\right)\right) \tag{5.1.42}$$

By construction the terms $\exp\left(\sum_{y\in\{\mathrm{ne}(f)\setminus x\}} \lambda_{y\to f}(y) - \lambda^*_{y\to f}\right)$ will be ≤ 1, with at least one term being equal to 1. This ensures that the dominant numerical contributions to the summation are computed accurately.

Log marginals are readily found using

$$\log p(x) = \sum_{f\in\mathrm{ne}(x)} \lambda_{f\to x}(x) \tag{5.1.43}$$



(a) (b)

Figure 5.4: **(a)**: Factor graph with a loop. **(b)**: Eliminating the variable d adds an edge between a and c, demonstrating that, in general, one cannot perform marginal inference in loopy graphs by simply passing messages along existing edges in the original graph.

5.1.5 The problem with loops

Loops cause a problem with variable elimination (or message passing) techniques since once a variable is eliminated the structure of the 'amputated' graph in general changes. For example, consider the FG

$$p(a,b,c,d) = f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(a,d) \tag{5.1.44}$$

depicted in fig(5.4a). The marginal $p(a,b,c)$ is given by

$$p(a,b,c) = f_1(a,b)\, f_2(b,c) \underbrace{\sum_d f_3(c,d)\, f_4(a,d)}_{f_5(a,c)} \tag{5.1.45}$$

which adds a link ac in the amputated graph, see fig(5.4b). This means that one cannot account for information from variable d by simply updating potentials on links in the original graph – one needs to account for the fact that the structure of the graph changes. The junction tree algorithm, chapter(6) deals with this by combining variables to make a new singly connected graph for which the graph structure remains singly connected under variable elimination.

5.2 Other Forms of Inference

5.2.1 Max-Product

A common interest is the most likely state of distribution. That is

$$\underset{x_1,x_2,\ldots,x_n}{\text{argmax}}\; p(x_1,x_2,\ldots,x_n) \tag{5.2.1}$$

To compute this efficiently for trees we exploit any factorisation structure of the distribution, analogous to the sum-product algorithm. That is, we aim to distribute the maximization so that only local computations are required. To develop the algorithm, consider a function which can be represented as an undirected chain,

$$f(x_1,x_2,x_3,x_4) = \phi(x_1,x_2)\phi(x_2,x_3)\phi(x_3,x_4) \tag{5.2.2}$$

for which we wish to find the joint state $x_1^*, x_2^*, x_3^*, x_4^*$ which maximises f. Firstly, we calculate the maximum *value* of f. Since potentials are non-negative, we may write

$$\max_{\mathbf{x}} f(\mathbf{x}) = \max_{x_1,x_2,x_3,x_4} \phi(x_1,x_2)\phi(x_2,x_3)\phi(x_3,x_4) = \max_{x_1,x_2,x_3} \phi(x_1,x_2)\phi(x_2,x_3) \underbrace{\max_{x_4} \phi(x_3,x_4)}_{\gamma_4(x_3)}$$

$$= \max_{x_1,x_2} \phi(x_1,x_2) \underbrace{\max_{x_3} \phi(x_2,x_3)\gamma_4(x_3)}_{\gamma_3(x_2)} = \max_{x_1,x_2} \phi(x_1,x_2)\gamma_3(x_2) = \max_{x_1} \underbrace{\max_{x_2} \phi(x_1,x_2)\gamma_3(x_2)}_{\gamma_2(x_1)}$$

The final equation corresponds to solving a single variable optimisation and determines both the optimal value of the function f and also the optimal state $x_1^* = \underset{x_1}{\text{argmax}}\, \gamma_2(x_1)$. Given x_1^*, the optimal x_2 is given by $x_2^* = \underset{x_2}{\text{argmax}}\, \phi(x_1^*,x_2)\gamma_3(x_2)$, and similarly $x_3^* = \underset{x_3}{\text{argmax}}\, \phi(x_2^*,x_3)\gamma_4(x_3)$, $x_4^* = \underset{x_4}{\text{argmax}}\, \phi(x_3^*,x_4)$. This procedure is called *backtracking*. Note that we could have equally started at the other end of the chain by

defining messages γ that pass information from x_i to x_{i+1}. The chain structure of the function ensures that the maximal value (and its state) can be computed in time which scales *linearly* with the number of factors in the function. There is no requirement here that the function f corresponds to a probability distribution (though the factors must be non-negative).

Example 5.2. Consider a distribution defined over binary variables:

$$p(a, b, c) \equiv p(a|b)p(b|c)p(c) \tag{5.2.3}$$

with

$$p(a = \mathsf{tr}|b = \mathsf{tr}) = 0.3, \quad p(a = \mathsf{tr}|b = \mathsf{fa}) = 0.2, \quad p(b = \mathsf{tr}|c = \mathsf{tr}) = 0.75$$
$$p(b = \mathsf{tr}|c = \mathsf{fa}) = 0.1, \quad p(c = \mathsf{tr}) = 0.4$$

What is the most likely joint configuration, $\underset{a,b,c}{\operatorname{argmax}}\, p(a, b, c)$?

Naively, we could evaluate $p(a, b, c)$ over all the 8 joint states of a, b, c and select that states with highest probability. An alternative message passing approach is to define

$$\gamma_c(b) \equiv \max_c p(b|c)p(c) \tag{5.2.4}$$

For the state $b = \mathsf{tr}$,

$$p(b = \mathsf{tr}|c = \mathsf{tr})p(c = \mathsf{tr}) = 0.75 \times 0.4, \qquad p(b = \mathsf{tr}|c = \mathsf{fa})p(c = \mathsf{fa}) = 0.1 \times 0.6 \tag{5.2.5}$$

Hence, $\gamma_c(b = \mathsf{tr}) = 0.75 \times 0.4 = 0.3$. Similarly, for $b = \mathsf{fa}$,

$$p(b = \mathsf{fa}|c = \mathsf{tr})p(c = \mathsf{tr}) = 0.25 \times 0.4 \qquad p(b = \mathsf{fa}|c = \mathsf{fa})p(c = \mathsf{fa}) = 0.9 \times 0.6 \tag{5.2.6}$$

Hence, $\gamma_c(b = \mathsf{fa}) = 0.9 \times 0.6 = 0.54$.

We now consider

$$\gamma_b(a) \equiv \max_b p(a|b)\gamma_c(b) \tag{5.2.7}$$

For $a = \mathsf{tr}$, the state $b = \mathsf{tr}$ has value

$$p(a = \mathsf{tr}|b = \mathsf{tr})\gamma_c(b = \mathsf{tr}) = 0.3 \times 0.3 = 0.09 \tag{5.2.8}$$

and state $b = \mathsf{fa}$ has value

$$p(a = \mathsf{tr}|b = \mathsf{fa})\gamma_c(b = \mathsf{fa}) = 0.2 \times 0.54 = 0.108 \tag{5.2.9}$$

Hence $\gamma_b(a = \mathsf{tr}) = 0.108$. Similarly, for $a = \mathsf{fa}$, the state $b = \mathsf{tr}$ has value

$$p(a = \mathsf{fa}|b = \mathsf{tr})\gamma_c(b = \mathsf{tr}) = 0.7 \times 0.3 = 0.21 \tag{5.2.10}$$

and state $b = \mathsf{fa}$ has value

$$p(a = \mathsf{fa}|b = \mathsf{fa})\gamma_c(b = \mathsf{fa}) = 0.8 \times 0.54 = 0.432 \tag{5.2.11}$$

giving $\gamma_b(a = \mathsf{fa}) = 0.432$. Now we can compute the optimal state

$$a^* = \underset{a}{\operatorname{argmax}}\, \gamma_b(a) = \mathsf{fa} \tag{5.2.12}$$

Given this optimal state, we can backtrack, giving

$$b^* = \underset{b}{\operatorname{argmax}}\, p(a = \mathsf{fa}|b)\gamma_c(b) = \mathsf{fa}, \quad c^* = \underset{c}{\operatorname{argmax}}\, p(b = \mathsf{fa}|c)p(c) = \mathsf{fa} \tag{5.2.13}$$

Note that in the backtracking process, we already have all the information required from the computation of the messages γ.

If we want to find the most likely state for a variable in the centre of the chain we can therefore pass messages from one end to the other followed by backtracking. This is the approach for example taken by the Viterbi algorithm for HMMs, section(23.2). Alternatively, we can send messages (carrying the result of maximisations) simultaneously from both ends of the chain and then read off the maximal state of the variable from the state which maximises the product of incoming messages. The first is a sequential procedure since we must have passed messages along the chain before we can backtrack. The second is a parallel procedure in which messages can be sent concurrently. The latter approach can be represented using a factor graph as described below.

Using a factor graph

One can also use the factor graph to compute the joint most probable state. Provided that a full schedule of message passing has occurred, the product of messages into a variable equals the maximum value of the joint function with respect to all other variables. One can then simply read off the most probable state by maximising this local potential.

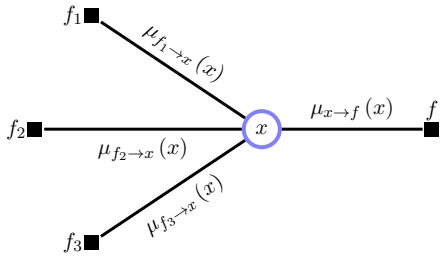
Procedure 5.2 (Max-Product messages on Factor Graphs)**.**

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 maximisation over the variables efficiently.

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

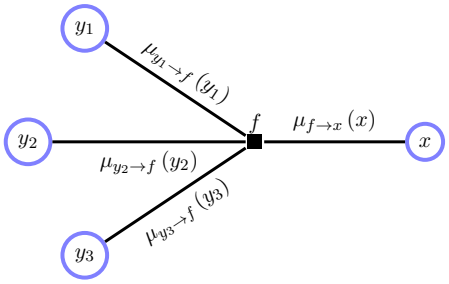
Variable to Factor message

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



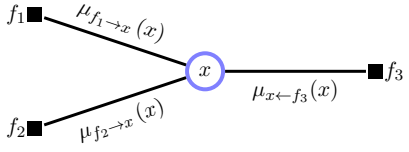
Factor to Variable message

$$\mu_{f\to x}(x) = \max_{\mathcal{X}_f\setminus x} \phi_f(\mathcal{X}_f) \prod_{y\in\{\mathrm{ne}(f)\setminus x\}} \mu_{y\to f}(y)$$



Maximal State

$$x^* = \operatorname*{argmax}_x \prod_{f\in\mathrm{ne}(x)} \mu_{f\to x}(x)$$



This algorithm is also called *belief revision*.

5.2.2 Finding the N most probable states

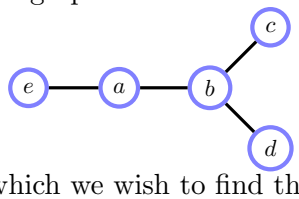
It is often of interest to calculate not just the most likely joint state, but the N most probable states, particularly in cases where the optimal state is only slightly more probable than other states. This is an interesting problem in itself and can be tackled with a variety of methods. A general technique is given by Nilson[227] which is based on the junction tree formalism, chapter(6), and the construction of candidate

lists, see for example [73].

For singly connected structures, several approaches have been developed [228, 320, 286, 273]. For the hidden Markov model, section(23.2), a simple algorithm is the N-Viterbi approach which stores the N-most probable messages at each stage of the propagation. For more general singly connected graphs one can extend the max-product algorithm to an N-max-product algorithm by retaining at each stage the N most probable messages, see below.

N-max-product

The algorithm for *N-max-product* is a minor modification of the standard max-product algorithm. Computationally, a straightforward way to accomplish this is to introduce an additional variable for each message that is used to index the most likely messages. We will first develop this for message passing on an undirected graph. Consider the distribution



$$p(a,b,c,d,e) = \phi(e,a)\phi(a,b)\phi(b,c)\phi(b,d) \tag{5.2.14}$$

for which we wish to find the two most probable values. Using the notation

$$\max_x{}^{i} f(x) \tag{5.2.15}$$

for the i^{th} highest value of $f(x)$, the maximisation over d can be expressed using the message

$$\gamma_d(b,1) = \max_d{}^{1} \phi(b,d), \qquad \gamma_d(b,2) = \max_d{}^{2} \phi(b,d) \tag{5.2.16}$$

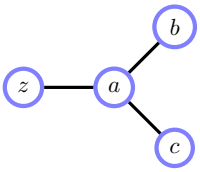
Defining messages similarly, the 2 most likely values of $p(a,b,c,d,e)$ can be computed using

$$\max_{a,b,c,d,e}{}^{1:2} \phi(e,a)\phi(a,b)\phi(b,c)\phi(b,d) = \max_{e,m_a}{}^{1:2} \max_{a,m_b}{}^{1:2} \phi(e,a) \underbrace{\max_{b,m_c,m_d}{}^{1:2} \phi(a,b) \underbrace{\max_c{}^{1:2} \phi(b,c)}_{\gamma_c(b,m_c)} \underbrace{\max_d{}^{1:2} \phi(b,d)}_{\gamma_d(b,m_d)}}_{\gamma_b(a,m_b)}}_{\gamma_a(e,m_a)} \tag{5.2.17}$$

where m_a, m_b, m_c and m_d index the two highest values. At the final stage we now have a table with $\dim(e)\times 2$ entries, from which we compute the highest two joint states of e, m_a. Given these first most likely joint states, e^*, m_a^* one then backtracks to find the most likely state of a, m_b using $\arg\max^{1:2}_{a,m_b} \phi(e^*,a)\gamma_b(a,m_b)$. One continues backtracking, then finding the most likely state of b, m_c, m_d and finally c and d. One may then restart the backtracking using the second most likely state of e, m_a and continue to find the most likely states to have given rise to this, leading to the second most likely joint state.

The translation of this to the factor graph formalism is straightforward and contained in `maxNprodFG.m`. Essentially the only modification required is to define extended messages which contain the N-most likely messages computed at each stage. A variable to factor message consists of the product of extended messages. For a factor to variable message, all extended messages from the neighbours are multiplied together into a large table. The N-most probable messages are retained, defining a new extended message. The N-most probable states for each variable can then be read off by finding the variable state that maximises the product of incoming extended messages.

Branches are the bottleneck in the above computation. Consider a term as part of a larger system with



$$\phi(z,a)\phi(a,b)\phi(a,c) \tag{5.2.18}$$

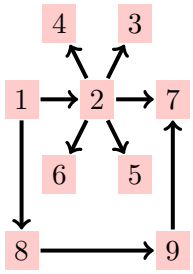


Figure 5.5: State transition diagram (weights not shown). The shortest (unweighted) path from state 1 to state 7 is $1-2-7$. Considered as a Markov chain (random walk), the most probable path from state 1 to state 7 is $1-8-9-7$. The latter path is longer but more probable since for the path $1-2-7$, the probability of exiting from state 2 into state 7 is $1/5$ (assuming each transition is equally likely). Note that in this example, states 3,4,5,6,7 are 'absorbing' – one can enter, but not exit these states. This would correspond to adding self-loops on nodes 3,4,5,6,7; these are not drawn here to improve clarity. See `demoMostProbablePath.m`

We would like to pass a message along the branch from b to a and then from c to a and then from a to the rest of the graph. To compute the most likely value we can find this from

$$\max_a \phi(z,a) \left\{ \max_b \phi(a,b) \right\} \left\{ \max_c \phi(a,c) \right\} \tag{5.2.19}$$

which represents an efficient approach since the maximisation can be carried out over each branch separately. However, to find the second most likely value, we cannot write

$$\max_a{}^{2} \phi(z,a) \left\{ \max_b \phi(a,b) \right\} \left\{ \max_c \phi(a,c) \right\} \tag{5.2.20}$$

For a fixed z, this would erroneously always force a to be in a different state than the state corresponding to the most likely value. Similarly, we cannot assume that the second most likely state corresponds to finding the second most likely state of each factor. Unlike the single most likely state case, therefore, we cannot distribute the maximisations over each branch but have to search over all branch contributions concurrently. This therefore corresponds to an exponentially complex search for the N-highest joint branch states. Whilst non branching links are non-problematic, the degree D of a variable's node (in either the FG or undirected representation) contributes an additional exponential term N^D to the computational complexity.

5.2.3 Most probable path and shortest path

What is the most likely path from state a to state b for an N state Markov chain? Note that this is not necessarily the same as the shortest path, as explained in fig(5.5). If we consider a path of length T, this has probability

$$p(s_2|s_1=\mathsf{a})p(s_3|s_2)\ldots p(s_T=\mathsf{b}|s_{T-1}) \tag{5.2.21}$$

Finding the most probable path can then be readily solved using the max-product (or max-sum algorithm for the log-transitions) on a simple serial factor graph. To deal with the issue that we don't know the optimal T, one approach is to redefine the probability transitions such that the desired state b is an *absorbing state* of the chain (that is, one can enter this state but not leave it). With this redefinition, the most probable joint state will correspond to the most probable state on the product of $N-1$ transitions. This approach is demonstrated in `demoMostProbablePath.m`, along with the more direct approaches described below.

An alternative, cleaner approach is as follows: for the Markov chain we can dispense with variable-to-factor and factor-to-variable messages and use only variable-to-variable messages. If we want to find the most likely set of states $\mathsf{a}, s_2, \ldots, s_{T-1}, \mathsf{b}$ to get us there, then this can be computed by defining the maximal path probability $E(\mathsf{a} \to \mathsf{b}, T)$ to get from a to b in T-timesteps:

$$E(\mathsf{a}\to\mathsf{b},T) = \max_{s_2,\ldots,s_{T-1}} p(s_2|s_1=\mathsf{a})p(s_3|s_2)p(s_4|s_3)\ldots p(s_T=\mathsf{b}|s_{T-1}) \tag{5.2.22}$$

$$= \max_{s_3,\ldots,s_{T-1}} \underbrace{\max_{s_2} p(s_2|s_1=\mathsf{a})p(s_3|s_2)}_{\gamma_{2\to3}(s_3)} p(s_4|s_3)\ldots p(s_T=\mathsf{b}|s_{T-1}) \tag{5.2.23}$$

To compute this efficiently we define messages

$$\gamma_{t\to t+1}(s_{t+1}) = \max_{s_t} \gamma_{t-1\to t}(s_t)\, p(s_{t+1}|s_t), \quad t \geq 2, \qquad \gamma_{1\to 2}(s_2) = p(s_2|s_1=\mathsf{a}) \tag{5.2.24}$$

until the point

$$E(\mathsf{a} \to \mathsf{b}, T) = \max_{s_{T-1}} \gamma_{T-2 \to T-1}(s_{T-1})\, p(s_T = \mathsf{b}|s_{T-1}) = \gamma_{T-1 \to T}(s_T = \mathsf{b}) \tag{5.2.25}$$

We can now proceed to find the maximal path probability for timestep $T+1$. Since the messages up to time $T-1$ will be the same as before, we need only compute one additional message, $\gamma_{T-1 \to T}(s_T)$, from which

$$E(\mathsf{a} \to \mathsf{b}, T+1) = \max_{s_T} \gamma_{T-1 \to T}(s_T)\, p(s_{T+1} = \mathsf{b}|s_T) = \gamma_{T \to T+1}(s_{T+1} = \mathsf{b}) \tag{5.2.26}$$

We can proceed in this manner until we reach $E(\mathsf{a} \to \mathsf{b}, N)$ where N is the number of nodes in the graph. We don't need to go beyond this number of steps since those that do must necessarily contain non-simple paths. (A *simple path* is one that does not include the same state more than once.) The optimal time t^* is then given by which of $E(\mathsf{a} \to \mathsf{b}, 2), \ldots, E(\mathsf{a} \to \mathsf{b}, N)$ is maximal. Given t^* one can begin to backtrack[1]. Since

$$E(\mathsf{a} \to \mathsf{b}, t^*) = \max_{s_{t^*-1}} \gamma_{t^*-2 \to t^*-1}(s_{t^*-1})\, p(s_{t^*} = \mathsf{b}|s_{t^*-1}) \tag{5.2.27}$$

we know the optimal state

$$\mathsf{s}^*_{t^*-1} = \operatorname*{argmax}_{s_{t^*-1}} \gamma_{t^*-2 \to t^*-1}(s_{t^*-1})\, p(s_{t^*} = \mathsf{b}|s_{t^*-1}) \tag{5.2.28}$$

We can then continue to backtrack:

$$\mathsf{s}^*_{t^*-2} = \operatorname*{argmax}_{\mathsf{s}_{t^*-2}} \gamma_{t^*-3 \to t^*-2}(s_{t^*-2})\, p(\mathsf{s}^*_{t^*-1}|s_{t^*-2}) \tag{5.2.29}$$

and so on. See `mostprobablepath.m`.

- In the above derivation we do not use any properties of probability, except that p must be non-negative (otherwise sign changes can flip a whole sequence 'probability' and the local' message recursion no longer applies). One can consider the algorithm as finding the optimal 'product' path from a to b.
- It is straightforward to modify the algorithm to solve the (single-source, single-sink) *shortest weighted path* problem. One way to do this is to replace the Markov transition probabilities with $\exp(-u(s_t|s_{t-1}))$, where $u(s_t|s_{t-1})$ is the edge weight and is infinite if there is no edge from s_{t-1} to s_t. This approach is taken in `shortestpath.m` which is able to deal with either positive or negative edge weights. This method is therefore more general than the well-known Dijkstra's algorithm [122] which requires weights to be positive. If a negative edge cycle exists, the code returns the shortest weighted length N path, where N is the number of nodes in the graph. See `demoShortestPath.m`.
- The above algorithm is efficient for the single-source, single-sink scenario, since the messages contain only N states, meaning that the overall storage is $O(N^2)$.
- As it stands, the algorithm is numerically impractical since the messages are recursively multiplied by values usually less than 1 (at least for the case of probabilities). One will therefore quickly run into numerical underflow (or possibly overflow in the case of non-probabilities) with this method.

To fix the final point above, it is best to work by defining the logarithm of E. Since this is a monotonic transformation, the most probable path defined through $\log E$ is the same as that obtained from E. In this case

$$L(\mathsf{a} \to \mathsf{b}, T) = \max_{s_2,\ldots,s_{T-1}} \log\left[p(s_2|s_1 = \mathsf{a})p(s_3|s_2)p(s_4|s_3)\ldots p(s_T = b|s_{T-1})\right] \tag{5.2.30}$$

$$= \max_{s_2,\ldots,s_{T-1}} \left[\log p(s_2|s_1 = \mathsf{a}) + \sum_{t=3}^{T-1} \log p(s_t|s_{t-1}) + \log p(s_T = \mathsf{b}|s_{T-1})\right] \tag{5.2.31}$$

@@

[1] An alternative to finding t^* is to define self-transitions with probability 1, and then use a fixed time $T = N$. Once the desired state b is reached, the self-transition then preserves the chain in state b for the remaining timesteps. This procedure is used in `mostprobablepathmult.m`

We can therefore define new messages

$$\lambda_{t\to t+1}(s_{t+1}) = \max_{s_t}\left[\lambda_{t-1\to t}(s_t) + \log p(s_{t+1}|s_t)\right] \tag{5.2.32}$$

One then proceeds as before by finding the most probable t^* defined on L, and backtracks.

Remark 5.2. A possible confusion is that optimal paths can be efficiently found 'when the graph is loopy'. Note that the graph in fig(5.5) is a state-transition diagram, not a graphical model. The graphical model corresponding to this simple Markov chain is the Belief Network $\prod_t p(s_t|s_{t-1})$, a linear serial structure. Hence the underlying graphical model is a simple chain, which explains why computation is efficient.

Most probable path (multiple-source, multiple-sink)

If we need the most probable path between all states a and b, one could re-run the above single-source-single-sink algorithm for all a and b. A computationally more efficient approach is to observe that one can define a message for each starting state a:

$$\gamma_{t\to t+1}(s_{t+1}|\mathsf{a}) = \max_{s_t}\gamma_{t-1\to t}(s_t|\mathsf{a})\,p(s_{t+1}|s_t) \tag{5.2.33}$$

and continue until we find the maximal path probability matrix for getting from any state a to any state b in T timesteps:

$$E(\mathsf{a}\to\mathsf{b}, T) = \max_{s_{T-1}}\gamma_{T-2\to T-1}(s_{T-1}|\mathsf{a})\,p(s_T = \mathsf{b}|s_{T-1}) \tag{5.2.34}$$

Since we know the message $\gamma_{T-2\to T-1}(s_{T-1}|\mathsf{a})$ for all states a, we can readily compute the most probable path from all starting states a to all states b after T steps. This requires passing an $N\times N$ matrix message γ. We can then proceed to the next timestep $T+1$. Since the messages up to time $T-1$ will be the same as before, we need only compute one additional message, $\gamma_{T-1\to T}(s_T)$, from which

$$E(\mathsf{a}\to\mathsf{b}, T+1) = \max_{s_T}\gamma_{T-1\to T}(s_T|\mathsf{a})\,p(s_{T+1} = \mathsf{b}|s_T) \tag{5.2.35}$$

In this way one can then efficiently compute the optimal path probabilities for all starting states a and end states b after t timesteps. To find the optimal corresponding path, backtracking proceeds as before, see `mostprobablepathmult.m` and `demoMostProbablePathMult.m`. One can also use the same algorithm to solve the multiple-source, multiple-sink shortest weighted path problem using exponentiated negative edge weights, as before. This is a variant of the Floyd-Warshall-Roy algorithm[122].

5.2.4 Mixed inference

An often encountered situation is to infer the most likely state of a joint marginal, possibly given some evidence. For example, given a distribution $p(x_1,\ldots,x_n)$, find

$$\operatorname*{argmax}_{x_1,x_2,\ldots,x_m} p(x_1,x_2,\ldots,x_m) = \operatorname*{argmax}_{x_1,x_2,\ldots,x_m}\sum_{x_{m+1},\ldots,x_n} p(x_1,\ldots,x_n) \tag{5.2.36}$$

In general, even for tree structured $p(x_1,\ldots,x_n)$, the optimal marginal state cannot be computed efficiently. One way to see this is that due to the summation the resulting joint marginal does not have a structured factored form as products of simpler functions of the marginal variables. Finding the most probable joint marginal then requires a search over all the joint marginal states – a task exponential in m. An approximate solution is provided by the EM algorithm (see section(11.2) and exercise(5.7)).

5.3 Inference in Multiply Connected Graphs

We briefly discuss here some relatively straightforward approaches to dealing with multiply connected graphs that are conceptually straightforward, or build on the repeated use of singly connected structures. We discuss a more general algorithm in chapter(6).

Algorithm 5.1 Compute marginal $p(x_1|\text{evidence})$ from distribution $p(x) = \prod_f \phi_f(\{x\}_f)$. Assumes non-evidential variables are ordered $x_1, \ldots, x_n$.

procedure BUCKET ELIMINATION($p(x) = \prod_f \phi_f(\{x\}_f)$.)
  Initialize all bucket potentials to unity. ▷ Fill buckets
  while There are potentials left in the distribution **do**
    For each potential ϕ_f, find its highest variable x_j (according to the ordering).
    Multiply ϕ_f with the potential in bucket j and remove ϕ_f the distribution.
  end while
  for $i =$ bucket n to 1 **do** ▷ Empty buckets
    For bucket i sum over the states of variable x_i and call this potential γ_i
    Identify the highest variable x_h of potential γ_i
    Multiply the existing potential in bucket h by γ_i
  end for
  The marginal $p(x_1|\text{evidence})$ is proportional to γ_1.
  return $p(x_1|\text{evidence})$ ▷ The conditional marginal.
end procedure

5.3.1 Bucket elimination

We consider here a general conditional marginal variable elimination method that works for any distribution (including multiply connected graphs). Bucket elimination is presented in algorithm(5.1) and can be considered a way to organise the distributed summation[84]. The algorithm is perhaps best explained by a simple example, as given below.

Example 5.3 (Bucket Elimination)**.** Consider the problem of calculating the marginal $p(f)$ of

$$p(a,b,c,d,e,f,g) = p(f|d)p(g|d,e)p(c|a)p(d|a,b)p(a)p(b)p(e), \tag{5.3.1}$$

see fig(2.1a). Whilst this is singly-connected, this serves to explain the general procedure.

$$p(f) = \sum_{a,b,c,d,e,g} p(a,b,c,d,e,f,g) = \sum_{a,b,c,d,e,g} p(f|d)p(g|d,e)p(c|a)p(d|a,b)p(a)p(b)p(e) \tag{5.3.2}$$

We can distribute the summation over the various terms as follows: e, b and c are end nodes, so that we can sum over their values:

$$p(f) = \sum_{d,a,g} p(f|d)p(a) \left(\sum_b p(d|a,b)p(b) \right) \left(\sum_c p(c|a) \right) \left(\sum_e p(g|d,e)p(e) \right) \tag{5.3.3}$$

For convenience, let's write the terms in the brackets as $\sum_b p(d|a,b)p(b) \equiv \gamma_b(a,d)$, $\sum_e p(g|d,e)p(e) \equiv \gamma_e(d,g)$. The term $\sum_c p(c|a)$ is equal to unity, and we therefore eliminate this node directly. Rearranging terms, we can write

$$p(f) = \sum_{d,a,g} p(f|d)p(a)\gamma_b(a,d)\,\gamma_e(d,g) \tag{5.3.4}$$

If we think of this graphically, the effect of summing over b, c, e is effectively to remove or 'eliminate' those variables. We can now carry on summing over a and g since these are end points of the new graph:

$$p(f) = \sum_d p(f|d) \left(\sum_a p(a)\gamma_b(a,d) \right) \left(\sum_g \gamma_e(d,g) \right) \tag{5.3.5}$$

Again, this defines new potentials $\gamma_a(d)$, $\gamma_g(d)$, so that the final answer can be found from

$$p(f) = \sum_d p(f|d)\gamma_a(d)\,\gamma_g(d) \tag{5.3.6}$$

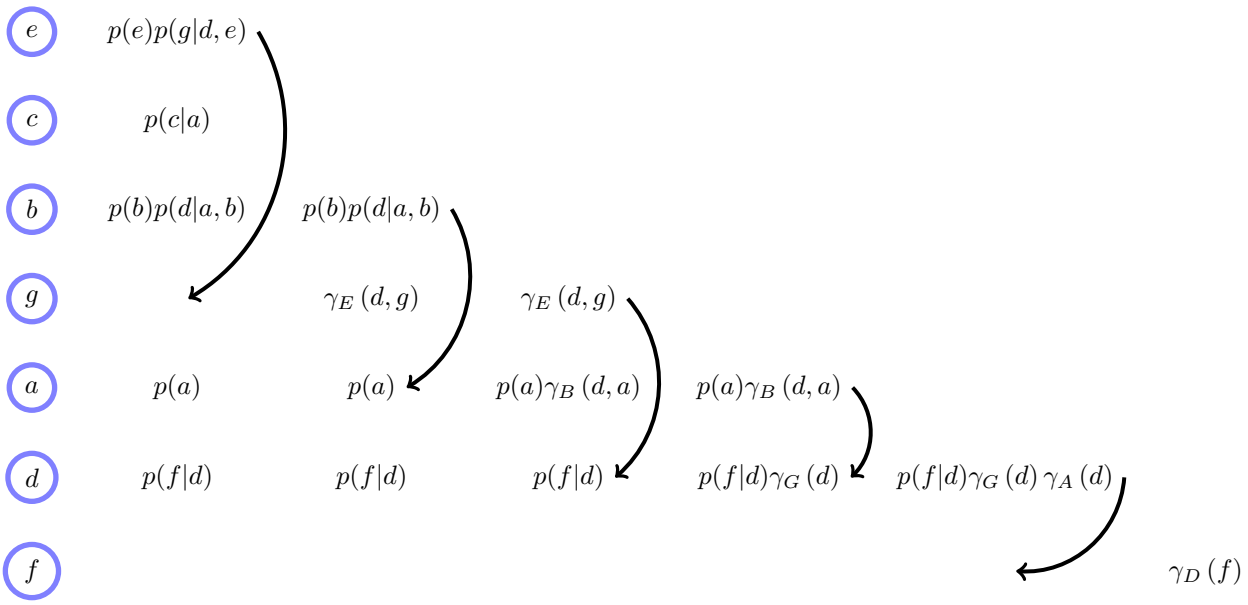


Figure 5.6: The bucket elimination algorithm applied to the graph fig(2.1). At each stage, at least one node is eliminated from the graph. The second stage of eliminating c is trivial since $\sum_c p(c|a) = 1$ and has therefore been skipped over since this bucket does not send any message.

We illustrate this in fig(5.6). Initially, we define an ordering of the variables, beginning with the one that we wish to find the marginal for – a suitable ordering is therefore, f, d, a, g, b, c, e. Then starting with the highest bucket e (according to our ordering f, d, a, g, b, c, e), we put all the potentials that mention e in the e bucket. Continuing with the next highest bucket, c, we put all the remaining potentials that mention c in this c bucket, etc. The result of this initialisation procedure is that terms (conditional distributions) in the DAG are distributed over the buckets, as shown in the left most column of fig(5.6). Eliminating then the highest bucket e, we pass a message to node g. Immediately, we can also eliminate bucket c since this sums to unity. In the next column, we have now two fewer buckets, and we eliminate the highest remaining bucket, this time b, passing a message to bucket a, and so on.

There are some important observations we can make about bucket elimination:

@@

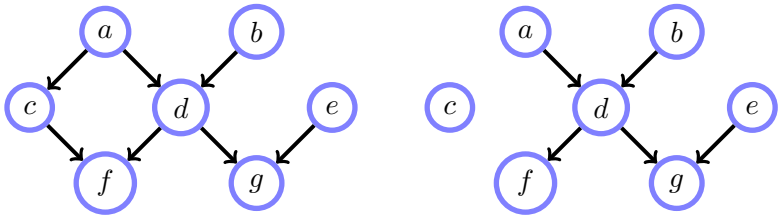
1. To compute say $p(x_1|\text{evidence})$ we need to re-order the variables (so that the required marginal variable is labelled x_1) and repeat bucket elimination. Hence each query (calculation of a marginal in this case) requires re-running the algorithm. It would be more efficient to reuse messages, rather than recalculating them each time.

2. In general, bucket elimination constructs multi-variable messages γ from bucket to bucket. The storage requirements of a multi-variable message are exponential in the number of variables of the message.

3. For trees we can always choose a variable ordering to render the computational complexity to be linear in the number of variables. Such an ordering is called perfect, definition(6.9), and indeed it can be shown that a perfect ordering can always easily be found for singly connected graphs (see [93]). However, orderings exist for which bucket elimination will be extremely inefficient.

5.3.2 Loop-cut conditioning

For multiply connected distributions we run into some difficulty with the message passing routines such as the sum-product algorithm which are designed to work on singly connected graphs only. One way to solve the difficulties of multiply connected (loopy) graphs is to identify nodes that, when removed, would reveal a singly connected subgraph[237]. Consider the example of fig(5.7). Imagine that we wish to calculate a



(a) (b)

Figure 5.7: A multiply connected graph (a) reduced to a singly connected graph (b) by conditioning on the variable c.

marginal, say $p(d)$. Then

$$p(d) = \sum_c \sum_{a,b,e,f,g} \underbrace{p(c|a)p(a)}_{p^*(a)} p(d|a,b)p(b) \underbrace{p(f|c,d)}_{p^*(f|d)} p(g|d,e) \tag{5.3.7}$$

where the p^* potentials are not necessarily distributions. For each state of c, the product of the new potentials on the variables a, b, e, f, g is singly connected, so that standard singly connected message passing can be used to perform inference. We will need to perform inference for each state of variable c, each state defining a new singly connected graph (with the same structure) but with modified potentials.

More generally, we can define a set of variables $\mathcal{C}$, called the *loop cut set* and run singly connected inference for each joint state of the cut-set variables $\mathcal{C}$. This can also be used for finding the most likely state of a multiply connected joint distribution as well. Hence, for a computational price exponential in the loop-cut size, we can calculate the marginals (or the most likely state) for a multiply connected distribution. However, determining a small cut set is in general difficult, and there is no guarantee that this will anyway be small for a given graph. Whilst this method is able to handle loops in a general manner, it is not particularly elegant since the concept of messages now only applies conditioned on the cut set variables, and how to re-use messages for inference of additional quantities of interest becomes unclear. We will discuss an alternative method for handling multiply connected distributions in chapter(6).

5.4 Message Passing for Continuous Distributions

For parametric continuous distributions $p(x|\theta_x)$, message passing corresponds to passing parameters θ of the distributions. For the sum-product algorithm, this requires that the operations of multiplication and integration over the variables are closed with respect to the family of distributions. This is the case, for example, for the Gaussian distribution – the marginal (integral) of a Gaussian is another Gaussian, and the product of two Gaussians is a Gaussian, see section(8.4). This means that we can then implement the sum-product algorithm based on passing mean and covariance parameters. To implement this requires some tedious algebra to compute the appropriate message parameter updates. At this stage, the complexities from performing such calculations are a potential distraction, though the interested reader may refer to `demoSumprodGaussMoment.m`, `demoSumprodGaussCanon.m` and `demoSumprodGaussCanonLDS.m` and also chapter(24) for examples of message passing with Gaussians. For more general exponential family distributions, message passing is essentially straightforward, though again the specifics of the updates may be tedious to work out. In cases where the operations of marginalisation and products are not closed within the family, the distributions need to be projected back to the chosen message family. Expectation propagation, section(28.8), is relevant in this case.

5.5 Summary

- For a tree-structured factor graph, non-mixed inference is essentially linear in the number of nodes in the graph (provided the variables are discrete or the inference operations form a tractable closed family).
- Computation on trees can be achieved using local 'message passing' algorithms, analogous to dynamic programming.

- The sum-product and max-product algorithm are particularly useful for computing marginal and most likely inferences respectively.
- Message-passing also holds for continuous variables based on passing messages that update the parameters of the distribution.
- Shortest-path problems can be solved using such message-passing approaches.
- Inference in non-trees (multiply connected distributions) is more complex since there is a fill-in effect when variables are eliminated that adds additional links to the graph.
- Inference in multiply connected graphs can be achieved using techniques such as cut-set conditioning which, by conditioning on a subset of variables, reveal a singly connected structure. However, this is generally inefficient since the messages cannot be readily re-used.

A take-home message from this chapter is that (non-mixed) inference in singly connected structures is usually computationally tractable. Notable exceptions are when the message passing operations are not closed within the message family, or representing messages explicitly requires an exponential amount of space. This happens for example when the distribution can contain both discrete and continuous variables, such as the Switching Linear Dynamical system, which we discuss in chapter(25).

Broadly speaking, inference in multiply connected structures is more complex and may be intractable. However, we do not want to give the impression that this is always the case. Notable exceptions are: finding the most likely state in an attractive pairwise MN, section(28.9); finding the most likely state and marginals in a binary planar MN with pure interactions, see for example [128, 261]. For N variables in the graph, a naive use of general purpose routines such as the junction tree algorithm for these inferences would result in an $O\left(2^N\right)$ computation, whereas clever algorithms are able to return the exact results in $O\left(N^3\right)$ operations. Of interest is *bond propagation*[192] which is an intuitive node elimination method to perform marginal inference in pure-interaction Ising models.

5.6 Code

The code below implements message passing on a tree structured factor graph. The FG is stored as an adjacency matrix with the message between FG node i and FG node j given in $A_{i,j}$.
`FactorGraph.m`: Return a factor graph adjacency matrix and message numbers
`sumprodFG.m`: Sum-Product algorithm on a factor graph

In general it is recommended to work in log-space in the Max-Product case, particularly for large graphs since the product of messages can become very small. The code provided does not work in log space and as such may not work on large graphs; writing this using log-messages is straightforward but leads to less readable code. An implementation based on log-messages is left as an exercise for the interested reader.
`maxprodFG.m`: Max-Product algorithm on a factor graph
`maxNprodFG.m`: N-Max-Product algorithm on a factor graph

5.6.1 Factor graph examples

For the distribution from fig(5.3), the following code finds the marginals and most likely joint states. The number of states of each variable is chosen at random.
`demoSumprod.m`: Test the Sum-Product algorithm
`demoMaxprod.m`: Test the Max-Product algorithm
`demoMaxNprod.m`: Test the Max-N-Product algorithm

5.6.2 Most probable and shortest path

`mostprobablepath.m`: Most probable path
`demoMostProbablePath.m`: Most probable versus shortest path demo

`demoShortestPath.m`: The shortest path demo works for both positive and negative edge weights. If negative weight cycles exist, the code finds the best length N shortest path.
`mostprobablepathmult.m`: Most probable path – multi-source, multi-sink
`demoMostProbablePathMult.m`: Demo of most probable path – multi-source, multi-sink

5.6.3 Bucket elimination

The efficiency of Bucket Elimination depends critically on the elimination sequence chosen. In the demonstration below we find the marginal of a variable in the Chest Clinic exercise using a randomly chosen elimination order. The desired marginal variable is specified as the last to be eliminated. For comparison we use an elimination sequence based on decimating a triangulated graph of the model, as discussed in section(6.5.1), again under the constraint that the last variable to be 'decimated' is the marginal variable of interest. For this smarter choice of elimination sequence, the complexity of computing this single marginal is roughly the same as that for the Junction Tree algorithm, using the same triangulation.
`bucketelim.m`: Bucket Elimination
`demoBucketElim.m`: Demo Bucket Elimination

5.6.4 Message passing on Gaussians

The following code hints at how message passing may be implemented for continuous distributions. The reader is referred to the BRMLtoolbox for further details and also section(8.4) for the algebraic manipulations required to perform marginalisation and products of Gaussians. The same principle holds for any family of distributions which is closed under products and marginalisation, and the reader may wish to implement specific families following the method outlined for Gaussians.
`demoSumprodGaussMoment.m`: Sum-product message passing based on Gaussian Moment parameterisation

5.7 Exercises

Exercise 5.1. *Given a pairwise singly connected Markov network of the form*

$$p(x) = \frac{1}{Z} \prod_{i \sim j} \phi(x_i, x_j) \tag{5.7.1}$$

explain how to efficiently compute the normalisation factor (also called the partition function) Z as a function of the potentials ϕ.

Exercise 5.2. *Consider a pairwise Markov network defined on binary variables:*

$$p(x) = \phi(x_1, x_{100}) \prod_{i=1}^{99} \phi(x_i, x_{i+1}) \tag{5.7.2}$$

Is it possible to compute $\underset{x_1,\ldots,x_{100}}{\operatorname{argmax}} p(x)$ *efficiently?*

Exercise 5.3. *You are employed by a web start up company that designs virtual environments, in which players can move between rooms. The rooms which are accessible from another in one time step is given by the* 100×100 *matrix* $\mathbf{M}$*, stored in* `virtualworlds.mat`*, where* $M_{ij} = 1$ *means that there is a door between rooms* i *and* j *(*$M_{ij} = M_{ji}$*).* $M_{ij} = 0$ *means that there is no door between rooms* i *and* j*.* $M_{ii} = 1$ *meaning that in one time step, one can stay in the same room. You can visualise this matrix by typing* `imagesc(M)`*.*

1. *Write a list of rooms which cannot be reached from room 2 after 10 time steps.*
2. *The manager complains that takes at least 13 time steps to get from room 1 to room 100. Is this true?*
3. *Find the most likely path (sequence of rooms) to get from room 1 to room 100.*
4. *If a single player were to jump randomly from one room to another (or stay in the same room), with no preference between rooms, what is the probability at time* $t \gg 1$ *the player will be in room 1? Assume that effectively an infinite amount of time has passed and the player began in room 1 at* $t = 1$*.*

5. *If two players are jumping randomly between rooms (or staying in the same room), explain how to compute the probability that, after an infinite amount of time, at least one of them will be in room 1? Assume that both players begin in room 1.*

Exercise 5.4. *Consider the hidden Markov model:*

$$p(v_1, \ldots, v_T, h_1, \ldots, h_T) = p(h_1)p(v_1|h_1)\prod_{t=2}^{T} p(v_t|h_t)p(h_t|h_{t-1}) \tag{5.7.3}$$

in which $dom(h_t) = \{1, \ldots, H\}$ *and* $dom(v_t) = \{1, \ldots, V\}$ *for all* $t = 1, \ldots, T$.

1. *Draw a belief network representation of the above distribution.*
2. *Draw a factor graph representation of the above distribution.*
3. *Use the factor graph to derive a Sum-Product algorithm to compute marginals* $p(h_t|v_1, \ldots, v_T)$. *Explain the sequence order of messages passed on your factor graph.*
4. *Explain how to compute* $p(h_t, h_{t+1}|v_1, \ldots, v_T)$.
5. *Show that the belief network for* $p(h_1, \ldots, h_T)$ *is a simple linear chain, whilst* $p(v_1, \ldots, v_T)$ *is a fully connected cascade belief network.*

Exercise 5.5. *For a singly connected Markov Network,* $p(x) = p(x_1, \ldots, x_n)$, *the computation of a marginal* $p(x_i)$ *can be carried out efficiently. Similarly, the most likely joint state* $x^* = \arg\max_{x_1,\ldots,x_n} p(x)$ *can be computed efficiently. Explain when the most likely joint state of a marginal can be computed efficiently,* i.e. *under what circumstances could one efficiently (in* $O(m)$ *time) compute* $\operatorname{argmax}_{x_1,x_2,\ldots,x_m} p(x_1, \ldots, x_m)$ *for* $m < n$?

Exercise 5.6. *Consider the internet with webpages labelled* $1, \ldots, N$. *If webpage* j *has a link to webpage* i, *then we place an element of the matrix* $L_{ij} = 1$, *otherwise* $L_{ij} = 0$. *By considering a random jump from webpage* j *to webpage* i *to be given by the transition probability*

$$M_{ij} = \frac{L_{ij}}{\sum_i L_{ij}} \tag{5.7.4}$$

what is the probability that after an infinite amount of random surfing, one ends up on webpage i? *How could you relate this to the potential 'relevance' of a webpage in terms of a search engine?*

Exercise 5.7. *A special time-homogeneous hidden Markov model is given by*

$$p(x_1, \ldots, x_T, y_1, \ldots, y_T, h_1, \ldots, h_T) = p(x_1|h_1)p(y_1|h_1)p(h_1)\prod_{t=2}^{T} p(h_t|h_{t-1})p(x_t|h_t)p(y_t|h_t) \tag{5.7.5}$$

The variable x_t *has 4 states,* $dom(x_t) = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$ *(numerically labelled as states 1,2,3,4). The variable* y_t *has 4 states,* $dom(y_t) = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$. *The hidden or latent variable* h_t *has 5 states,* $dom(h_t) = \{1, \ldots, 5\}$. *The HMM models the following (fictitious) process:*

In humans, Z-factor proteins are a sequence on states of the variables $x_1, x_2, \ldots, x_T$. *In bananas Z-factor proteins are also present, but represented by a different sequence* $y_1, y_2, \ldots, y_T$. *Given a sequence* $x_1, \ldots, x_T$ *from a human, the task is to find the corresponding sequence* $y_1, \ldots, y_T$ *in the banana by first finding the most likely joint latent sequence, and then the most likely banana sequence given this optimal latent sequence. That is, we require*

$$\operatorname*{argmax}_{y_1,\ldots,y_T} p(y_1, \ldots, y_T|h_1^*, \ldots, h_T^*) \tag{5.7.6}$$

where

$$h_1^*, \ldots, h_T^* = \operatorname*{argmax}_{h_1,\ldots,h_T} p(h_1, \ldots, h_T|x_1, \ldots, x_T) \tag{5.7.7}$$

The file `banana.mat` *contains the emission distributions* `pxgh` *(*$p(x|h)$*),* `pygh` *(*$p(y|h)$*) and transition* `phtghtm` *(*$p(h_t|h_{t-1})$*). The initial hidden distribution is given in* `ph1` *(*$p(h_1)$*). The observed* x *sequence is given in* `x`.

1. *Explain mathematically and in detail how to compute the optimal y-sequence, using the two-stage procedure as stated above.*

2. *Write a* MATLAB *routine that computes and displays the optimal y-sequence, given the observed x-sequence. Your routine must make use of the Factor Graph formalism.*

3. *Explain whether or not it is computationally tractable to compute*

$$\underset{y_1,\ldots,y_T}{\text{argmax}}\ p(y_1,\ldots,y_T|x_1,\ldots,x_T) \tag{5.7.8}$$

4. *Bonus question: By considering $y_1,\ldots,y_T$ as parameters, explain how the EM algorithm, section(11.2), may be used to find* $\underset{y_1,\ldots,y_T}{\text{argmax}}\ p(y_1,\ldots,y_T|x_1,\ldots,x_T)$. *Implement this approach with a suitable initialisation for the optimal parameters $y_1,\ldots,y_T$.*

Exercise 5.8. *There are a set of firstnames:* `david`, `anton`, `fred`, `jim`, `barry` *which are numbered 1 (*`david`*) to 5 (*`barry`*) and a set of surnames:* `barber`, `ilsung`, `fox`, `chain`, `fitzwilliam`, `quinceadams`, `grafvonunterhosen`, *which are numbered 1 (*`barber`*) to 7 (*`grafvonunterhosen`*). A string is generated by first randomly sampling a character from* `a` *to* `z`. *Then we either continue this random character sampling with probability 0.8, or we start to generate a firstname. A firstname is chosen uniformly at random. After generating a firstname, we generate a character at random. We continue to generate another letter at random with probability 0.8, or start to generate a surname (chosen uniformly from the set of surnames). We continue until the last letter of the surname is generated. The process then goes back to start (unless we are at the end timepoint $T = 10000$). For example, we might generate then* `dtyjimdfilsungffdavidmjfox...`. *The catch is that the process of character generation is very noisy. The character we want to generate is only generated correctly with probability 0.3, with probability 0.7 that another character (uniformly at random) will be generated. Given the 10000 character sequence in the file* `noisystring.mat`, *you must decode the noisystring to find the most likely intended 'clean' sequence. Once you have this sequence, you will be able to find a set of (firstname,surname) pairs as you traverse along the clean sequence. Construct a matrix $m(i,j)$ with the counts of the number of occurrences of the pair $(firstname(i), surname(j))$ in your clean sequence and display this matrix.* ++

Exercise 5.9. *A security company is employed to keep watch on the behaviour of people moving through a train station. Using video cameras they are able to track the x, y position of 500 people. The matrix in* `drunkproblemX.mat` *contains the position of the 500 people through time, with a 1 in the matrix representing that a person is occupying that position. This matrix is generated using the program drunkmover.m which you may wish to examine. If a person moves out of the grid, they are moved back into a random position in the grid, as described in* `drunkmover.m`. *All but one of the people move only to a single neighbouring point on the x-y grid in one time step. However, person 1 is a fast and dangerous drunk that we want to track. The drunk can move from (x, y) to $(x \pm 2, y \pm 2)$ at the next time step.* ++

Devise a method to track where you think the dangerous drunk is and give a list of the (x, y) coordinates of the single most likely path the drunk took through the station.

Exercise 5.10. *The file* `BearBulldata` *contains the price of an asset through $T = 200$ timepoints. The price takes values from 1 to 100. If the market is in a 'bear' state, the price changes from time $t-1$ to t with probability transition matrix $pbear(price(t), price(t-1))$. If the market is in the 'bull' state, the price changes from time $t-1$ to t with transition matrix $pbull(price(t), price(t-1))$. If the market is in a 'bear' state it will remain so with probability 0.8. If the market is in a bull state it will remain so with probability 0.7. You may assume that at timestep 1, the market is uniformly in either a bear or bull state and also that the price distribution at timestep 1 is uniform. Use this model to compute the probability of the price at time $T+1$ given all the observed prices from time 1 to T; that is $p(price(T+1)|price(1:T))$. Using this probability, compute the expected gain $price(T+1) - price(T)$ in the price of the asset, and also the standard deviation in this price gain $price(T+1) - price(T)$.* ++

Exercise 5.11. *You find yourself adrift in outer space after a space station disaster. Your rocket suit has only very simple controls that allow you to accelerate at time t an amount $a_i(t) \in \{-1, 0, 1\}$ in each of three* ++

dimensions of space $i \in \{1,2,3\}$ independently. At each time t, you can therefore command your spacesuit to provide an acceleration vector

$$\mathbf{a}_t = \begin{pmatrix} a_1(t) \\ a_2(t) \\ a_3(t) \end{pmatrix}, \qquad a_i(t) \in \{-1, 0, +1\} \tag{5.7.9}$$

According to discrete time Newton's laws, this changes the velocity $\mathbf{v}_t$ and position $\mathbf{x}_t$ according to the Markovian updates

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \delta \mathbf{a}_t \tag{5.7.10}$$
$$\mathbf{x}_{t+1} = \mathbf{x}_t + \delta \mathbf{v}_t \tag{5.7.11}$$

where $\delta = 0.1$ is a given value corresponding to a unit of real time. Each time an acceleration $+1$ or -1 is applied a unit amount of fuel is used. Your task is to get from the origin $\mathbf{x}_1 = (0,0,0)^T$ at time 1 to rendezvous with the rescue station at $\mathbf{x}_{102} = (4.71, -6.97, 8.59)^T$ at time 102 using the minimum amount of fuel (it doesn't matter what speed you have, as long as you rendezvous with the rescue station). At time $t = 1$, you are stationary, $\mathbf{v}_1 = (0,0,0)^T$. The total amount of fuel used is $\sum_{t=1}^{100} \sum_{i=1}^{3} |a_i(t)|$.

1. *For each of the three spatial dimensions $i \in \{1,2,3\}$, write down a single equation that relates $a_i(t)$, $t \in \{1,2,\ldots,100\}$ to the rescue station at $\mathbf{x}_{102}$.*

2. *What is the minimum amount of fuel that can be used to rendezvous with the rescue station at time 102?*

3. *Assuming that there is a sequence $\mathbf{a}_1, \ldots, \mathbf{a}_{100}$ that will obtain a rendezvous with the rescue station at time 102, explain if you believe there is an efficient way to calculate the minimum amount of fuel. If there is an efficient algorithm, state it. Otherwise explain why no efficient algorithm is available.*

++ **Exercise 5.12.** *An algorithm exists (for example Dijkstra's algorithm) that finds the minimum weighted path between two specified nodes on a graph; the algorithm however only generally works if all weights are non-negative. You have a minimum weighted path problem but with weights u_{ij} that can be negative. A friend suggests that you can still use Dijkstra's algorithm by making a new set of edge weights that are all non-negative, $w_{ij} = u_{ij} - u^*$ where u^* is the minimum value of all the u_{ij} weights. Explain why this will not generally give the correct minimum weight path.*

++ **Exercise 5.13.** *It's the year 2515 and the tax collector Simo Hurtta needs to go on a trip starting from planet 1 and ending on planet 1725. Interplanetary travel is only possible between certain planets, with the cost of going from planet i to planet j equal to the Euclidian distance between the planets $\sqrt{(\mathbf{x}^i - \mathbf{x}^j)^2}$ minus the tax that he will collect on planet j. Given the set of planet positions* `x`*, interplanetary travel possibilities (the matrix $A_{ij} = 1$ if it is possible to go from planet i to planet j and collectable taxes for each planet (*`t`*) (see* `SimoHurtta.mat`*), what is the minimum cost of travelling from planet 1 to planet 1725? (Assume that Hurrta does not collect taxes from planet 1.)*

CHAPTER 6

The Junction Tree Algorithm

When the distribution is multiply-connected it would be useful to have a generic inference approach that is efficient in its reuse of messages. In this chapter we discuss an important structure, the junction tree, that by clustering variables enables one to perform message-passing efficiently (although the structure on which the message-passing occurs may consist of intractably large clusters). The most important thing is the junction tree itself, based on which different message-passing procedures can be considered. The junction tree helps forge links with the computational complexity of inference in fields from computer science to statistics and physics.

6.1 Clustering Variables

In chapter(5) we discussed efficient inference for singly-connected graphs, for which variable elimination and message passing schemes are appropriate. In the multiply connected case, however, one cannot in general perform inference by passing messages only along existing links in the graph. The idea behind the junction tree algorithm (JTA) is to form a new representation of the graph in which variables are clustered together, resulting in a singly-connected graph in the cluster variables (albeit on a different graph). The main focus of the development will be on marginal inference, though similar techniques apply to different inferences, such as finding the most probable state of the distribution.

At this stage it is important to point out that the JTA is not a magic method to deal with intractabilities resulting from multiply connected graphs; it is simply a way to perform correct inference on a multiply connected graph by transforming to a singly connected structure. Carrying out the inference on the resulting junction tree may still be computationally intractable. For example, the junction tree representation of a general two-dimensional Ising model is a single supernode containing all the variables. Inference in this case is exponentially complex in the number of variables. Nevertheless, even in cases where implementing the JTA may be intractable, the JTA provides useful insight into the representation of distributions that can form the basis for approximate inference. In this sense the JTA is key to understanding issues related to representations and complexity of inference and is central to the development of efficient inference algorithms.

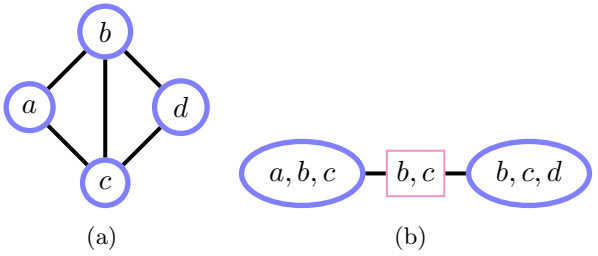
6.1.1 Reparameterisation

Consider the chain

$$p(a, b, c, d) = p(a|b)p(b|c)p(c|d)p(d) \tag{6.1.1}$$

From the definition of conditional probability, we can reexpress this as

$$p(a, b, c, d) = \frac{p(a, b)}{p(b)} \frac{p(b, c)}{p(c)} \frac{p(c, d)}{p(d)} p(d) = \frac{p(a, b)p(b, c)p(c, d)}{p(b)p(c)} \tag{6.1.2}$$



(a) (b)

Figure 6.1: (a) Markov network $\phi(a,b,c)\phi(b,c,d)$. (b) Clique graph representation of (a).

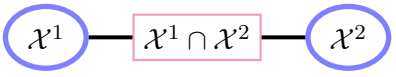
A useful insight is that the distribution can therefore be written as a product of marginal distributions, divided by a product of the intersection of the marginal distributions: Looking at the numerator $p(a,b)p(b,c)p(c,d)$ this cannot be a distribution over a,b,c,d since we are overcounting b and c, where this overcounting of b arises from the overlap between the sets $\{a,b\}$ and $\{b,c\}$, which have b as their intersection. Similarly, the overcounting of c arises from the overlap between the sets $\{b,c\}$ and $\{c,d\}$. Intuitively, we need to correct for this overcounting by dividing by the distribution on the intersections. Given the transformed representation as a product of marginals divided by a product of their intersection (the right equation in (6.1.2)), a marginal such as $p(a,b)$ can be read off directly from the factors in the new expression. The aim of the junction tree algorithm is to form just such a representation of the distribution which contains the marginals explicitly. We want to do this in a way that works for belief and Markov networks, and also deals with the multiply connected case. In order to do so, an appropriate way to parameterise the distribution is in terms of a clique graph, as described in the next section.

6.2 Clique Graphs

Definition 6.1 (Clique Graph)**.** A clique graph consists of a set of potentials, $\phi_1(\mathcal{X}^1),\ldots,\phi_n(\mathcal{X}^n)$ each defined on a set of variables $\mathcal{X}^i$. For neighbouring cliques on the graph, defined on sets of variables $\mathcal{X}^i$ and $\mathcal{X}^j$, the intersection $\mathcal{X}^s = \mathcal{X}^i \cap \mathcal{X}^j$ is called the *separator* and has a corresponding potential $\phi_s(\mathcal{X}^s)$. A clique graph represents the function

$$\frac{\prod_c \phi_c(\mathcal{X}^c)}{\prod_s \phi_s(\mathcal{X}^s)} \tag{6.2.1}$$

For notational simplicity we will usually drop the clique potential index c. Graphically clique potentials are represented by circles/ovals, and separator potentials by rectangles.



The graph on the left represents $\phi(\mathcal{X}^1)\phi(\mathcal{X}^2)/\phi(\mathcal{X}^1 \cap \mathcal{X}^2)$.

Clique graphs translate Markov networks into structures convenient for carrying out inference. Consider the Markov network in fig(6.1a)

$$p(a,b,c,d) = \frac{\phi(a,b,c)\phi(b,c,d)}{Z} \tag{6.2.2}$$

An equivalent representation is given by the clique graph in fig(6.1b), defined as the product of the numerator clique potentials, divided by the product of the separator potentials. In this case the separator potential may be set to the normalisation constant Z. By summing we have

$$Zp(a,b,c) = \phi(a,b,c)\sum_d \phi(b,c,d), \qquad Zp(b,c,d) = \phi(b,c,d)\sum_a \phi(a,b,c) \tag{6.2.3}$$

Multiplying the two expressions, we have

$$Z^2 p(a,b,c)p(b,c,d) = \left(\phi(a,b,c)\sum_d \phi(b,c,d)\right)\left(\phi(b,c,d)\sum_a \phi(a,b,c)\right) = Z^2 p(a,b,c,d)\sum_{a,d} p(a,b,c,d)$$

(6.2.4)

In other words

$$p(a,b,c,d) = \frac{p(a,b,c)p(b,c,d)}{p(c,b)} \tag{6.2.5}$$

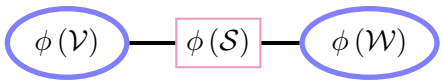
The important observation is that the distribution can be written in terms of its marginals on the variables in the original cliques and that, as a clique graph, it has the same structure as before. All that has changed is that the original clique potentials have been replaced by the marginals of the distribution and the separator by the marginal defined on the separator variables $\phi(a,b,c) \to p(a,b,c)$, $\phi(b,c,d) \to p(b,c,d)$, $Z \to p(c,b)$. The usefulness of this representation is that if we are interested in the marginal $p(a,b,c)$, this can be read off from the transformed clique potential. To make use of this representation, we require a systematic way of transforming the clique graph potentials so that at the end of the transformation the new potentials contain the marginals of the distribution.

Remark 6.1. Note that, whilst visually similar, a Factor Graph and a Clique Graph are different representations. In a clique graph the nodes contain sets of variables, which may share variables with other nodes.

6.2.1 Absorption

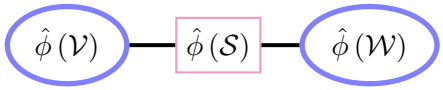
Consider neighbouring cliques $\mathcal{V}$ and $\mathcal{W}$, sharing the variables $\mathcal{S}$ in common. In this case, the distribution on the variables $\mathcal{X} = \mathcal{V} \cup \mathcal{W}$ is

$$p(\mathcal{X}) = \frac{\phi(\mathcal{V})\phi(\mathcal{W})}{\phi(\mathcal{S})} \tag{6.2.6}$$



and our aim is to find a new representation

$$p(\mathcal{X}) = \frac{\hat{\phi}(\mathcal{V})\hat{\phi}(\mathcal{W})}{\hat{\phi}(\mathcal{S})} \tag{6.2.7}$$



in which the potentials are given by

$$\hat{\phi}(\mathcal{V}) = p(\mathcal{V}), \quad \hat{\phi}(\mathcal{W}) = p(\mathcal{W}), \quad \hat{\phi}(\mathcal{S}) = p(\mathcal{S}) \tag{6.2.8}$$

@@ In this example, we can explicitly work out the new potentials as function of the old potentials by computing the marginals as follows:

$$p(\mathcal{W}) = \sum_{\mathcal{V}\setminus\mathcal{S}} p(\mathcal{X}) = \sum_{\mathcal{V}\setminus\mathcal{S}} \frac{\phi(\mathcal{V})\phi(\mathcal{W})}{\phi(\mathcal{S})} = \phi(\mathcal{W}) \frac{\sum_{\mathcal{V}\setminus\mathcal{S}} \phi(\mathcal{V})}{\phi(\mathcal{S})} \tag{6.2.9}$$

and

$$p(\mathcal{V}) = \sum_{\mathcal{W}\setminus\mathcal{S}} p(\mathcal{X}) = \sum_{\mathcal{W}\setminus\mathcal{S}} \frac{\phi(\mathcal{V})\phi(\mathcal{W})}{\phi(\mathcal{S})} = \phi(\mathcal{V}) \frac{\sum_{\mathcal{W}\setminus\mathcal{S}} \phi(\mathcal{W})}{\phi(\mathcal{S})} \tag{6.2.10}$$

There is a symmetry present in the two equations above – they are the same under interchanging $\mathcal{V}$ and $\mathcal{W}$. One way to describe these equations is through 'absorption'. We say that the cluster $\mathcal{W}$ 'absorbs' information from cluster $\mathcal{V}$ by the following updating procedure. First we define a new separator

$$\phi^*(\mathcal{S}) = \sum_{\mathcal{V}\setminus\mathcal{S}} \phi(\mathcal{V}) \tag{6.2.11}$$

and refine the $\mathcal{W}$ potential using

$$\phi^*(\mathcal{W}) = \phi(\mathcal{W}) \frac{\phi^*(\mathcal{S})}{\phi(\mathcal{S})} \tag{6.2.12}$$

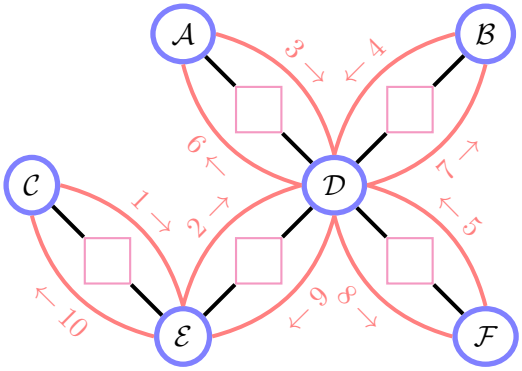


Figure 6.2: An example absorption schedule on a clique tree. Many valid schedules exist under the constraint that messages can only be passed to a neighbour when all other messages have been received.

The advantage of this interpretation is that the new representation is still a valid clique graph representation of the distribution since

$$\frac{\phi(\mathcal{V})\phi^*(\mathcal{W})}{\phi^*(\mathcal{S})} = \frac{\phi(\mathcal{V})\phi(\mathcal{W})\frac{\phi^*(\mathcal{S})}{\phi(\mathcal{S})}}{\phi^*(\mathcal{S})} = \frac{\phi(\mathcal{V})\phi(\mathcal{W})}{\phi(\mathcal{S})} = p(\mathcal{X}) \tag{6.2.13}$$

For this simple two clique graph, we see that after $\mathcal{W}$ absorbs information from $\mathcal{V}$ then $\phi^*(\mathcal{W}) = p(\mathcal{W})$, which can be verified by comparing the right of equation (6.2.9) with equation (6.2.12). With these new updated potentials, we now go back along the graph, from $\mathcal{W}$ to $\mathcal{V}$. After $\mathcal{V}$ absorbs information from $\mathcal{W}$ then $\phi^*(\mathcal{V})$ contains the marginal $p(\mathcal{V})$. After the separator $\mathcal{S}$ has participated in absorption along both directions, then the separator potential will contain $p(\mathcal{S})$ (this is not the case after only a single absorption). To see this, consider absorbing from $\mathcal{W}$ to $\mathcal{V}$ using the updated potentials $\phi^*(\mathcal{W})$ and $\phi^*(\mathcal{S})$

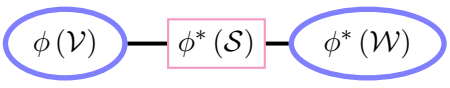
$$\phi^{**}(\mathcal{S}) = \sum_{\mathcal{W}\setminus\mathcal{S}} \phi^*(\mathcal{W}) = \sum_{\mathcal{W}\setminus\mathcal{S}} \frac{\phi(\mathcal{W})\phi^*(\mathcal{S})}{\phi(\mathcal{S})} = \sum_{\{\mathcal{W}\cup\mathcal{V}\}\setminus\mathcal{S}} \frac{\phi(\mathcal{W})\phi(\mathcal{V})}{\phi(\mathcal{S})} = p(\mathcal{S}) \tag{6.2.14}$$

Continuing, we have the new potential $\phi^*(\mathcal{V})$ given by

$$\phi^*(\mathcal{V}) = \frac{\phi(\mathcal{V})\phi^{**}(\mathcal{S})}{\phi^*(\mathcal{S})} = \frac{\phi(\mathcal{V})\sum_{\mathcal{W}\setminus\mathcal{S}}\phi(\mathcal{W})\phi^*(\mathcal{S})/\phi(\mathcal{S})}{\phi^*(\mathcal{S})} = \frac{\sum_{\mathcal{W}\setminus\mathcal{S}}\phi(\mathcal{V})\phi(\mathcal{W})}{\phi(\mathcal{S})} = p(\mathcal{V}) \tag{6.2.15}$$

Hence, in terms of equation (6.2.7), the new representation is $\hat{\phi}(\mathcal{V}) = \phi^*(\mathcal{V})$, $\hat{\phi}(\mathcal{S}) = \phi^{**}(\mathcal{S})$, $\hat{\phi}(\mathcal{W}) = \phi^*(\mathcal{W})$.

Definition 6.2 (Absorption)**.**



Let $\mathcal{V}$ and $\mathcal{W}$ be neighbours in a clique graph, let $\mathcal{S}$ be their separator, and let $\phi(\mathcal{V})$, $\phi(\mathcal{W})$ and $\phi(\mathcal{S})$ be their potentials. Absorption from $\mathcal{V}$ to $\mathcal{W}$ through $\mathcal{S}$ replaces the tables $\phi(\mathcal{S})$ and $\phi(\mathcal{W})$ with

$$\phi^*(\mathcal{S}) = \sum_{\mathcal{V}\setminus\mathcal{S}} \phi(\mathcal{V}) \qquad \phi^*(\mathcal{W}) = \phi(\mathcal{W})\frac{\phi^*(\mathcal{S})}{\phi(\mathcal{S})} \tag{6.2.16}$$

We say that clique $\mathcal{W}$ absorbs information from clique $\mathcal{V}$. The potentials are written on the clique graph (left) to highlight the potential updating.

6.2.2 Absorption schedule on clique trees

Having defined the local message propagation approach, we need to define an update ordering for absorption. In general, a node $\mathcal{V}$ can send exactly one message to a neighbour $\mathcal{W}$, and it may only be sent when $\mathcal{V}$ has received a message from each of its other neighbours. We continue this sequence of absorptions until a message has been passed in both directions along every link. See, for example, fig(6.2). Note that there are many valid message passing schemes in this case.

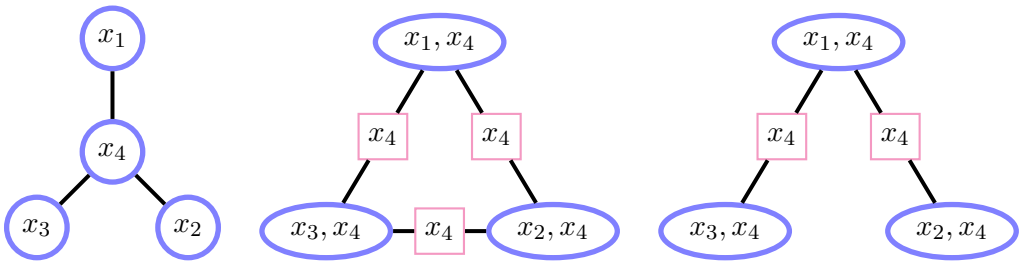


Figure 6.3: **(a)**: Singly connected Markov network. **(b)**: Clique graph. **(c)**: Clique tree.

Definition 6.3 (Absorption Schedule)**.** A clique can send a message to a neighbour, provided it has already received messages from all other neighbours.

6.3 Junction Trees

There are a few stages we need to go through in order to transform a distribution into an appropriate structure for inference. Initially we explain how to do this for singly connected structures before moving onto the multiply connected case.

Consider the singly connected Markov network, fig(6.3a)

$$p(x_1, x_2, x_3, x_4) = \phi(x_1, x_4)\phi(x_2, x_4)\phi(x_3, x_4) \tag{6.3.1}$$

The clique graph of this singly connected Markov network is multiply connected, fig(6.3b), where the separator potentials are all set to unity. Nevertheless, let's try to reexpress the Markov network in terms of marginals. First we have the relations

$$p(x_1, x_4) = \sum_{x_2, x_3} p(x_1, x_2, x_3, x_4) = \phi(x_1, x_4) \sum_{x_2} \phi(x_2, x_4) \sum_{x_3} \phi(x_3, x_4) \tag{6.3.2}$$

$$p(x_2, x_4) = \sum_{x_1, x_3} p(x_1, x_2, x_3, x_4) = \phi(x_2, x_4) \sum_{x_1} \phi(x_1, x_4) \sum_{x_3} \phi(x_3, x_4) \tag{6.3.3}$$

$$p(x_3, x_4) = \sum_{x_1, x_2} p(x_1, x_2, x_3, x_4) = \phi(x_3, x_4) \sum_{x_1} \phi(x_1, x_4) \sum_{x_2} \phi(x_2, x_4) \tag{6.3.4}$$

Taking the product of the three marginals, we have

$$p(x_1, x_4)p(x_2, x_4)p(x_3, x_4) = \phi(x_1, x_4)\phi(x_2, x_4)\phi(x_3, x_4) \underbrace{\left(\sum_{x_1} \phi(x_1, x_4) \sum_{x_2} \phi(x_2, x_4) \sum_{x_3} \phi(x_3, x_4) \right)^2}_{p(x_4)^2} \tag{6.3.5}$$

This means that the Markov network can be expressed in terms of marginals as

$$p(x_1, x_2, x_3, x_4) = \frac{p(x_1, x_4)p(x_2, x_4)p(x_3, x_4)}{p(x_4)p(x_4)} \tag{6.3.6}$$

Hence a valid clique graph is also given by the representation fig(6.3c). Indeed, if a variable (here x_4) occurs on every separator in a clique graph loop, one can remove that variable from an arbitrarily chosen separator in the loop. If this leaves an empty separator, we may then simply remove it. This shows that in such cases we can transform the clique graph into a clique tree (*i.e.* a singly connected clique graph). Provided that the original Markov network is singly connected, one can always form a clique tree in this manner.

6.3.1 The running intersection property

Sticking with the above example, consider the clique tree in fig(6.3)

$$\frac{\phi(x_3, x_4)\phi(x_1, x_4)\phi(x_2, x_4)}{\phi_1(x_4)\phi_2(x_4)} \tag{6.3.7}$$

as a representation of the distribution (6.3.1) where we set $\phi_1(x_4) = \phi_2(x_4) = 1$ to make this match. Now perform absorption on this clique tree:

We absorb $(x_3, x_4) \rightsquigarrow (x_1, x_4)$. The new separator is

$$\phi_1^*(x_4) = \sum_{x_3} \phi(x_3, x_4) \tag{6.3.8}$$

and the new potential is

$$\phi^*(x_1, x_4) = \phi(x_1, x_4)\frac{\phi_1^*(x_4)}{\phi_1(x_4)} = \phi(x_1, x_4)\phi_1^*(x_4) \tag{6.3.9}$$

Now $(x_1, x_4) \rightsquigarrow (x_2, x_4)$. The new separator is

$$\phi_2^*(x_4) = \sum_{x_1} \phi^*(x_1, x_4) \tag{6.3.10}$$

and the new potential is

$$\phi^*(x_2, x_4) = \phi(x_2, x_4)\frac{\phi_2^*(x_4)}{\phi_2(x_4)} = \phi(x_2, x_4)\phi_2^*(x_4) \tag{6.3.11}$$

Since we've 'hit the buffers' in terms of message passing, the potential $\phi(x_2, x_4)$ cannot be updated further. Let's examine more carefully the value of this new potential,

$$\phi^*(x_2, x_4) = \phi(x_2, x_4)\phi_2^*(x_4) = \phi(x_2, x_4)\sum_{x_1} \phi^*(x_1, x_4) \tag{6.3.12}$$

$$= \phi(x_2, x_4)\sum_{x_1} \phi(x_1, x_4)\sum_{x_3} \phi(x_3, x_4) = \sum_{x_1, x_3} p(x_1, x_2, x_3, x_4) = p(x_2, x_4) \tag{6.3.13}$$

Hence the new potential $\phi^*(x_2, x_4)$ contains the marginal $p(x_2, x_4)$.

To complete a full round of message passing we need to have passed messages in a valid schedule along both directions of each separator. To do so, we continue as follows:

We absorb $(x_2, x_4) \rightsquigarrow (x_1, x_4)$. The new separator is

$$\phi_2^{**}(x_4) = \sum_{x_2} \phi^*(x_2, x_4) \tag{6.3.14}$$

and

$$\phi^{**}(x_1, x_4) = \phi^*(x_1, x_4)\frac{\phi_2^{**}(x_4)}{\phi_2^*(x_4)} \tag{6.3.15}$$

Note that $\phi_2^{**}(x_4) = \sum_{x_2} \phi^*(x_2, x_4) = \sum_{x_2} p(x_2, x_4) = p(x_4)$ so that now, after absorbing through both directions, the separator contains the marginal $p(x_4)$. The reader may show that $\phi^{**}(x_1, x_4) = p(x_1, x_4)$.

Finally, we absorb $(x_1, x_4) \rightsquigarrow (x_3, x_4)$. The new separator is

$$\phi_1^{**}(x_4) = \sum_{x_1} \phi^{**}(x_1, x_4) = p(x_4) \tag{6.3.16}$$

and

$$\phi^*(x_3, x_4) = \phi(x_3, x_4)\frac{\phi_1^{**}(x_4)}{\phi_1^*(x_4)} = p(x_3, x_4) \tag{6.3.17}$$

Hence, after a full round of message passing, the new potentials all contain the correct marginals.

The new representation is *consistent* in the sense that for any (not necessarily neighbouring) cliques $\mathcal{V}$ and $\mathcal{W}$ with intersection $\mathcal{I}$, and corresponding potentials $\phi(\mathcal{V})$ and $\phi(\mathcal{W})$,

$$\sum_{\mathcal{V}\setminus\mathcal{I}} \phi(\mathcal{V}) = \sum_{\mathcal{W}\setminus\mathcal{I}} \phi(\mathcal{W}) \tag{6.3.18}$$

Note that bidirectional absorption following a valid schedule guarantees local consistency for neighbouring cliques, as in the example above, provided that we started with a clique tree which is a correct representation of the distribution. To ensure global consistency, if a variable occurs in two cliques it must be present in all cliques on any path connecting these cliques. An extreme example would be if we removed the link between @@ cliques (x_3, x_4) and (x_1, x_4). In this case this is still a clique tree (forest); however global consistency could not be guaranteed since the information required to make clique (x_3, x_4) consistent with the rest of the graph cannot reach this clique.

Formally, the requirement for the propagation of local to global consistency is that the clique tree is a ++ junction tree, as defined below. See also exercise(6.12).

Definition 6.4 (Junction Tree)**.** A clique tree is a junction tree if, for each pair of nodes, $\mathcal{V}$ and $\mathcal{W}$, all nodes on the path between $\mathcal{V}$ and $\mathcal{W}$ contain the intersection $\mathcal{V} \cap \mathcal{W}$. This is also called the *running intersection property*.

From this definition local consistency will be passed on to any neighbours and the distribution will be globally consistent. Proofs for these results are contained in [162].

Example 6.1 (A consistent junction tree)**.** To gain some intuition about the meaning of consistency, consider the junction tree in fig(6.4d). After a full round of message passing on this tree, each link is consistent, and the product of the potentials divided by the product of the separator potentials is just the original distribution itself. Imagine that we are interested in calculating the marginal for the node abc. That requires summing over all the other variables, $defgh$. If we consider summing over h then, because the link is consistent,

$$\sum_h \phi^*(e, h) = \phi^*(e) \tag{6.3.19}$$

so that the ratio $\sum_h \frac{\phi^*(e,h)}{\phi^*(e)}$ is unity, and the effect of summing over node h is that the link between eh and dce can be removed, along with the separator. The same happens for the link between node eg and dce, and also for cf to abc. The only nodes remaining are now dce and abc and their separator c, which have so far been unaffected by the summations. We still need to sum out over d and e. Again, because the link is consistent,

$$\sum_{de} \phi^*(d, c, e) = \phi^*(c) \tag{6.3.20}$$

so that the ratio $\sum_{de} \frac{\phi^*(d,c,e)}{\phi^*(c)} = 1$. The result of the summation of all variables not in abc therefore produces unity for the cliques and their separators, and the summed potential representation reduces simply to the potential $\phi^*(a, b, c)$ which is the marginal $p(a, b, c)$. It is clear that a similar effect will happen for other nodes. We can then obtain the marginals for individual variables by simple brute force summation over the other variables in that potential, for example $p(f) = \sum_c \phi^*(c, f)$.

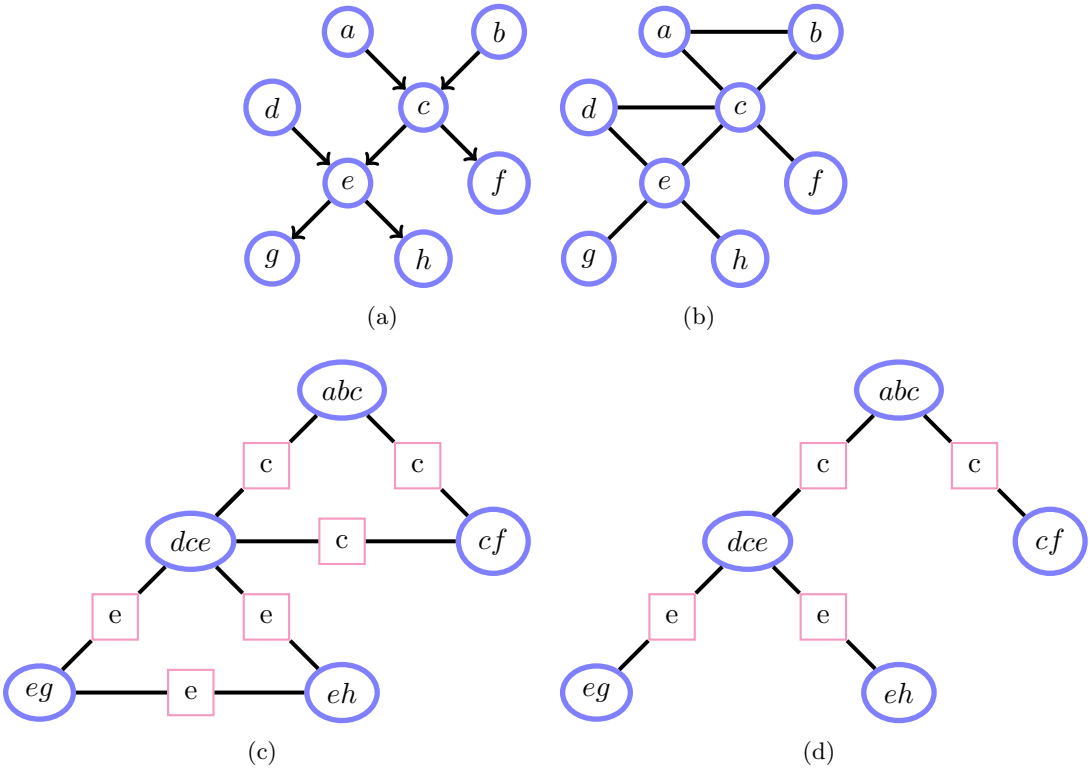


Figure 6.4: **(a)**: Belief Network. **(b)**: Moralised version of (a). **(c)**: Clique graph of (b). **(d)**: A junction tree. This satisfies the running intersection property that for any two nodes which contain a variable in common, any clique on the path linking the two nodes also contains that variable.

6.4 Constructing a Junction Tree for Singly-Connected Distributions

6.4.1 Moralisation

For belief networks an initial step is required, which is not required in the case of undirected graphs.

Definition 6.5 (Moralisation)**.** For each variable x add an undirected link between all parents of x and replace the directed link to x from its parents by undirected links. This creates a 'moralised' Markov @@ network.

6.4.2 Forming the clique graph

The clique graph is formed by identifying the cliques in the Markov network and adding a link between cliques that have a non-empty intersection. Add a separator between the intersecting cliques.

6.4.3 Forming a junction tree from a clique graph

For a singly connected distribution, any maximal weight spanning tree of a clique graph is a junction tree.

Definition 6.6 (Junction Tree)**.** A junction tree is obtained by finding a maximal weight spanning tree of the clique graph. The weight of the tree is defined as the sum of all the separator weights of the tree, where the separator weight is the number of variables in the separator.

If the clique graph contains loops, then all separators on the loop contain the same variable. By continuing to remove loop links until a tree is revealed, we obtain a junction tree.

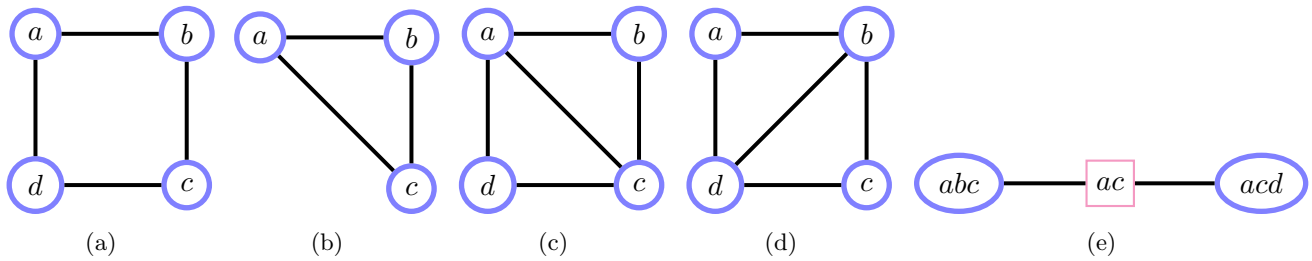


Figure 6.5: **(a)**: An undirected graph with a loop. **(b)**: Eliminating node d adds a link between a and c in the subgraph. **(c)**: The induced representation for the graph in (a). **(d)**: Equivalent induced representation. **(e)**: Junction tree for (a).

Example 6.2 (Forming a Junction Tree)**.** Consider the Belief Network in fig(6.4a). The moralisation procedure gives fig(6.4b). Identifying the cliques in this graph, and linking them together gives the clique graph in fig(6.4c). There are several possible junction trees one could obtain from this clique graph, and one is given in fig(6.4d).

6.4.4 Assigning potentials to cliques

Definition 6.7 (Clique Potential Assignment)**.** Given a junction tree and a function defined as the product of a set of potentials $\phi\left(\mathcal{X}^1\right), \ldots, \phi\left(\mathcal{X}^n\right)$, a valid clique potential assignment places potentials in JT cliques whose variables can contain them such that the product of the JT clique potentials, divided by the JT separator potentials, is equal to the function.

A simple way to achieve this assignment is to list all the potentials and order the JT cliques arbitrarily. Then, for each potential, search through the JT cliques until the first is encountered for which the potential variables are a subset of the JT clique variables. Subsequently the potential on each JT clique is taken as the product of all clique potentials assigned to the JT clique. Lastly, we assign all JT separators to unity. This approach is taken in `jtassignpot.m`. Note that in some instances it can be that a junction tree clique is assigned to unity.

Example 6.3. For the belief network of fig(6.4a), we wish to assign its potentials to the junction tree fig(6.4d). In this case the assignment is unique and is given by

$$\begin{array}{lcl} \phi(abc) & = & p(a)p(b)p(c|a,b) \\ \phi(dce) & = & p(d)p(e|d,c) \\ \phi(cf) & = & p(f|c) \\ \phi(eg) & = & p(g|e) \\ \phi(eh) & = & p(h|e) \end{array} \tag{6.4.1}$$

All separator potentials are initialised to unity.

6.5 Junction Trees for Multiply-Connected Distributions

When the distribution contains loops, the construction outlined in section(6.4) does not result in a junction tree. The reason is that, due to the loops, variable elimination changes the structure of the remaining graph.

To see this, consider the following distribution,

$$p(a,b,c,d) = \phi(a,b)\phi(b,c)\phi(c,d)\phi(d,a) \tag{6.5.1}$$

as shown in fig(6.5a). Let's first try to make a clique graph. We have a choice about which variable first to marginalise over. Let's choose d:

$$p(a,b,c) = \phi(a,b)\phi(b,c)\sum_d \phi(c,d)\phi(d,a) \tag{6.5.2}$$

The remaining subgraph therefore has an extra connection between a and c, see fig(6.5b). We can express the joint in terms of the marginals using

$$p(a,b,c,d) = \frac{p(a,b,c)}{\sum_d \phi(c,d)\phi(d,a)}\phi(c,d)\phi(d,a) \tag{6.5.3}$$

To continue the transformation into marginal form, let's try to replace the numerator terms with probabilities. We can do this by considering

$$p(a,c,d) = \phi(c,d)\phi(d,a)\sum_b \phi(a,b)\phi(b,c) \tag{6.5.4}$$

Plugging this into the above equation, we have

$$p(a,b,c,d) = \frac{p(a,b,c)p(a,c,d)}{\sum_d \phi(c,d)\phi(d,a)\sum_b \phi(a,b)\phi(b,c)} \tag{6.5.5}$$

We recognise that the denominator is simply $p(a,c)$, hence

$$p(a,b,c,d) = \frac{p(a,b,c)p(a,c,d)}{p(a,c)}. \tag{6.5.6}$$

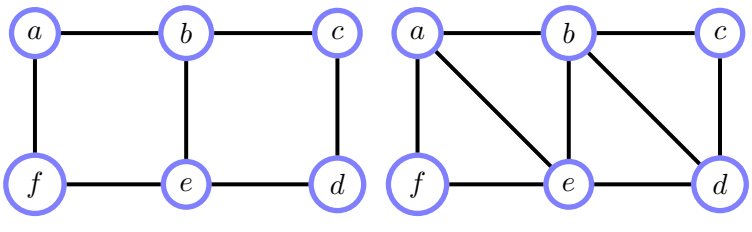
This means that a valid clique graph for the distribution fig(6.5a) must contain cliques larger than those in the original distribution. To form a JT based on products of cliques divided by products of separators, we could start from the *induced representation* fig(6.5c). Alternatively, we could have marginalised over variables a and c, and ended up with the equivalent representation fig(6.5d).

Generally, the result from variable elimination and re-representation in terms of the induced graph is that a link is added between any two variables on a loop (of length 4 or more) which does not have a chord. This is called *triangulation*. A Markov network on a triangulated graph can always be written in terms of the product of marginals divided by the product of separators. Armed with this new induced representation, we can form a junction tree.

Example 6.4. A slightly more complex loopy distribution is depicted in fig(6.6a),

$$p(a,b,c,d,e,f) = \phi(a,b)\phi(b,c)\phi(c,d)\phi(d,e)\phi(e,f)\phi(a,f)\phi(b,e) \tag{6.5.7}$$

There are different induced representations depending on which variables we decide to eliminate. The reader may convince herself that one such induced representation is given by fig(6.6b).



(a) (b)

Figure 6.6: **(a)**: Loopy 'ladder' Markov network. **(b)**: Induced representation.

Definition 6.8 (Triangulated (Decomposable) Graph)**.** An undirected graph is triangulated if every loop of length 4 or more has a chord. An equivalent term is that the graph is *decomposable* or *chordal*. From this definition, one may show that an undirected graph is triangulated if and only if its clique graph has a junction tree.

6.5.1 Triangulation algorithms

When a variable is eliminated from a graph, links are added between all the neighbours of the eliminated variable. A triangulation algorithm is one that produces a graph for which there exists a variable elimination order that introduces no extra links in the graph.

For discrete variables the complexity of inference scales exponentially with clique sizes in the triangulated graph since absorption requires computing tables on the cliques. It is therefore of some interest to find a triangulated graph with small clique sizes. However, finding the triangulated graph with the smallest maximal clique is a computationally hard problem for a general graph, and heuristics are unavoidable. Below we describe two simple algorithms that are generically reasonable, although there may be cases where an alternative algorithm may be considerably more efficient[57, 29, 207].

Remark 6.2 ('Triangles')**.** Note that a triangulated graph is not one in which 'squares in the original graph have triangles within them in the triangulated graph'. Whilst this is the case for fig(6.6b), this is not true for fig(6.10d). The term triangulation refers to the fact that *every* 'square' (*i.e.* loop of length 4) must have a 'triangle', with edges added until this criterion is satisfied. See also fig(6.7).

Greedy variable elimination

An intuitive way to think of triangulation is to first start with *simplicial nodes*, namely those which, when eliminated do not introduce any extra links in the remaining graph. Next consider a non-simplicial node of the remaining graph that has the minimal number of neighbours. Then add a link between all neighbours of this node and then eliminate this node from the graph. Continue until all nodes have been eliminated. (This procedure corresponds to Rose-Tarjan Elimination[251] with a particular node elimination choice). By labelling the nodes eliminated in sequence, we obtain a perfect ordering (see below). In the case that (discrete) variables have different numbers of states, a more refined version is to choose the non-simplicial node i which, when eliminated, leaves the smallest clique table size (the product of the size of all the state dimensions of the neighbours of node i). See fig(6.8) for an example.

Procedure 6.1 (Variable Elimination)**.** In Variable Elimination, one simply picks any non-deleted node x in the graph, and then adds links to all the neighbours of x. Node x is then deleted. One repeats this until all nodes have been deleted[251].

Definition 6.9 (Perfect Elimination Order)**.** Let the n variables in a Markov network be ordered from 1 to n. The ordering is perfect if, for each node i, the neighbours of i that are later in the ordering, and i itself, form a (maximal) clique. This means that when we eliminate the variables in sequence from 1 to n, no additional links are induced in the remaining marginal graph. A graph which admits a perfect elimination order is decomposable, and vice versa.

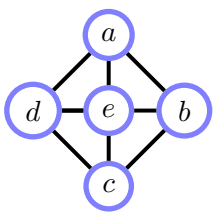


Figure 6.7: This graph is not triangulated, despite its 'triangular' appearance. The loop $a - b - c - d - a$ does not have a chord.

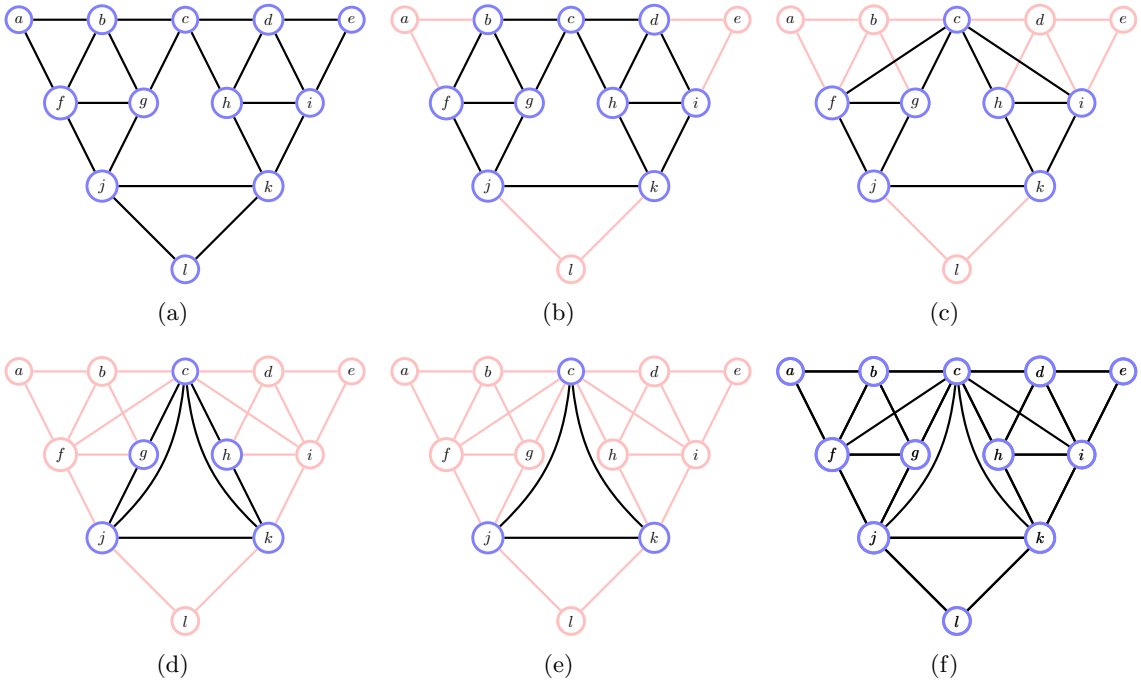


Figure 6.8: **(a)**: Markov network for which we seek a triangulation via greedy variable elimination. We first eliminate the simplicial nodes a, e, l. **(b)**: We then eliminate variables b, d since these only add a single extra link to the induced graph. **(c)**: There are no simplicial nodes at this stage, and we choose to eliminate f and i, each elimination adding only a single link. **(d)**: We eliminate g and h since these are simplicial. **(e)**: The remaining variables $\{c, j, k\}$ may be eliminated in any order. **(f)**: Final triangulation. The variable elimination (partial) order is $\{a, e, l\}, \{b, d\}, \{f, i\}, \{g, h\}, \{c, j, k\}$ where the brackets indicate that the order in which the variables inside the bracket are eliminated is irrelevant. Compared with the triangulation produced by the max-cardinality checking approach in fig(6.10d), this triangulation is more parsimonious.

Whilst this variable elimination guarantees a triangulated graph, its efficiency depends heavily on the sequence of nodes chosen to be eliminated. Several heuristics for this have been proposed, including the one below, which corresponds to choosing x to be the node with the minimal number of neighbours.

Maximum cardinality checking

Algorithm(6.1) terminates with success if the graph is triangulated. Not only is this a sufficient condition for a graph to be triangulated, but it is also necessary [288]. It processes each node and the time to process a node is quadratic in the number of adjacent nodes. This triangulation checking algorithm also suggests a triangulation construction algorithm – we simply add a link between the two neighbours that caused the algorithm to FAIL, and then restart the algorithm. The algorithm is restarted from the beginning, not just continued from the current node. This is important since the new link may change the connectivity between previously labelled nodes. See fig(6.10) for an example[1].

6.6 The Junction Tree Algorithm

We now have all the steps required for inference in multiply connected graphs, given in the procedure below.

Procedure 6.2 (Junction tree algorithm)**.**

[1]This example is due to David Page `www.cs.wisc.edu/~dpage/cs731`

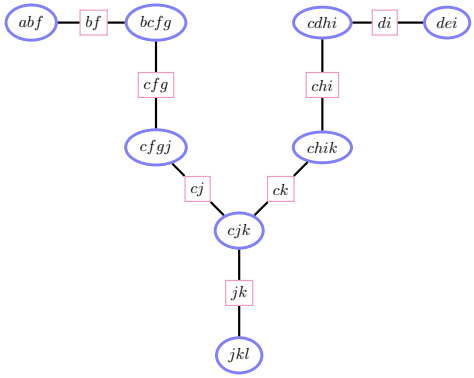


Figure 6.9: Junction tree formed from the triangulation fig(6.8f). One may verify that this satisfies the running intersection property.

Algorithm 6.1 A check if a graph is decomposable (triangulated). The graph is triangulated if, after cycling through all the n nodes in the graph, the FAIL criterion is not encountered.

1: Choose any node in the graph and label it 1.
2: **for** $i = 2$ to n **do**
3: Choose the node with the most labeled neighbours and label it i.
4: If any two labeled neighbours of i are not adjacent to each other, FAIL.
5: **end for**

Where there is more than one node with the most labeled neighbours, the tie may be broken arbitrarily.

Moralisation Marry the parents. This is required only for directed distributions. Note that all the parents of a variable are married together – a common error is to marry only the 'neighbouring' parents.

Triangulation Ensure that every loop of length 4 or more has a chord.

Junction Tree Form a junction tree from cliques of the triangulated graph, removing any unnecessary links in a loop on the cluster graph. Algorithmically, this can be achieved by finding a tree with maximal spanning weight with weight w_{ij} given by the number of variables in the separator between cliques i and j. Alternatively, given a clique elimination order (with the lowest cliques eliminated first), one may connect each clique i to the single neighbouring clique $j > i$ with greatest edge weight w_{ij}.

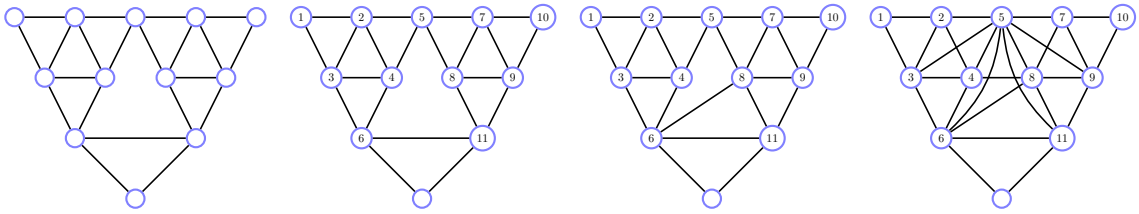
Potential Assignment Assign potentials to junction tree cliques and set the separator potentials to unity.

Message Propagation Carry out absorption until updates have been passed along both directions of every link on the JT. The clique marginals can then be read off from the JT.

An example is given in fig(6.11).

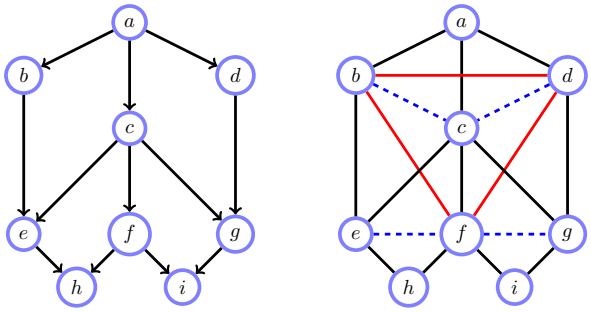
6.6.1 Remarks on the JTA

- The algorithm provides an upper bound on the computation required to calculate marginals in the graph. There may exist more efficient algorithms in particular cases, although generally it is believed that there cannot be much more efficient approaches than the JTA since every other approach must perform a triangulation[161, 188]. One particular special case is that of marginal inference for a binary variable Markov network on a two-dimensional lattice containing only pure quadratic interactions. In this case the complexity of computing a marginal inference is $O\left(n^3\right)$ where n is the number of variables in the distribution. This is in contrast to the pessimistic exponential complexity suggested by the JTA.
- One might think that the only class of distributions for which essentially a linear time algorithm is available are singly connected distributions. However, there are decomposable graphs for which the cliques have limited size meaning that inference is tractable. For example an extended version of the 'ladder' in fig(6.6a) has a simple induced decomposable representation fig(6.6b), for which marginal inference would be linear in the number of rungs in the ladder. Effectively these structures are *hyper trees* in which the complexity is then related to the *tree width* of the graph[87].



(a) (b) (c) (d)

Figure 6.10: Starting with the Markov network in (a), the maximum cardinality check algorithm proceeds until (b), where an additional link is required, see (c). One continues until the fully triangulated graph (d) is found.



(a) (b)

Figure 6.11: **(a)**: Original loopy Belief Network. **(b)**: The moralisation links (dashed) are between nodes e and f and between nodes f and g. The other additional links come from triangulation. The clique size of the resulting clique tree (not shown) is four.

- Ideally, we would like to find a triangulated graph which has minimal clique size. However, it can be shown to be a computationally hard problem to find the most efficient triangulation. In practice, most general purpose triangulation algorithms are chosen to provide reasonable, but clearly not always optimal, generic performance.
- Numerical over/under flow issues can occur under repeated multiplication of potentials. If we only care about marginals we can avoid numerical difficulties by normalising potentials at each step; these missing normalisation constants can always be found under the normalisation constraint. If required one can always store the values of these local renormalisations, should, for example, the global normalisation constant of a distribution be required, see section(6.6.2).
- After clamping variables in evidential states, running the JTA returns the joint distribution on the non-evidential variables $\mathcal{X}_c$ in a clique with all the evidential variables clamped in their evidential states, $p(\mathcal{X}_c, \mathsf{evidence})$. From this conditionals are straightforward to calculate.
- Representing the marginal distribution of a set of variables $\mathcal{X}$ which are not contained within a single clique is in general computationally difficult. Whilst the probability of any state of $p(\mathcal{X})$ may be computed efficiently, there are in general an exponential number of such states. A classical example in this regard is the HMM, section(23.2) which has a singly connected joint distribution $p(\mathcal{V}, \mathcal{H})$. However the marginal distribution $p(\mathcal{V})$ is fully connected. This means that for example whilst the entropy of $p(\mathcal{V}, \mathcal{H})$ is straightforward to compute, the entropy of the marginal $p(\mathcal{V})$ is intractable.

6.6.2 Computing the normalisation constant of a distribution

For a Markov network

$$p(\mathcal{X}) = \frac{1}{Z} \prod_i \phi(\mathcal{X}_i) \tag{6.6.1}$$

how can we find Z efficiently? If we used the JTA on the unnormalised distribution $\prod_i \phi(\mathcal{X}_i)$, we would have the equivalent representation:

$$p(\mathcal{X}) = \frac{1}{Z} \frac{\prod_c \phi(\mathcal{X}_c)}{\prod_s \phi(\mathcal{X}_s)} \tag{6.6.2}$$

where s and c are the separator and clique indices. Since the distribution must normalise, we can obtain Z from

$$Z = \sum_{\mathcal{X}} \frac{\prod_c \phi(\mathcal{X}_c)}{\prod_s \phi(\mathcal{X}_s)} \tag{6.6.3}$$

For a consistent JT, summing first over the variables of a simplicial JT clique (not including the separator variables), the marginal clique will cancel with the corresponding separator to give a unity term so that the clique and separator can be removed. This forms a new JT for which we then eliminate another simplicial clique. Continuing in this manner we will be left with a single numerator potential so that

$$Z = \sum_{\mathcal{X}_c} \phi(\mathcal{X}_c) \tag{6.6.4}$$

This is true for any clique c, so it makes sense to choose one with a small number of states so that the resulting raw summation is efficient. Hence in order to compute the normalisation constant of a distribution one runs the JT algorithm on an unnormalised distribution and the global normalisation is then given by the local normalisation of any clique. Note that if the graph is disconnected (there are isolated cliques), the normalisation is the product of the connected component normalisation constants.

6.6.3 The marginal likelihood

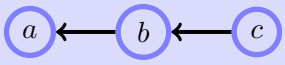
Our interest here is the computation of $p(\mathcal{V})$ where $\mathcal{V} \subset \mathcal{X}$ is a subset of the full variable set $\mathcal{X}$. Naively, one could carry out this computation by summing over all the non-evidential variables (hidden variables $\mathcal{H} = \mathcal{X} \backslash \mathcal{V}$) explicitly. In cases where this is computationally impractical an alternative is to use

$$p(\mathcal{H}|\mathcal{V}) = \frac{p(\mathcal{V}, \mathcal{H})}{p(\mathcal{V})} \tag{6.6.5}$$

One can view this as a product of clique potentials divided by the normalisation $p(\mathcal{V})$, for which the general method of section(6.6.2) may be directly applied. See `demoJTree.m`.

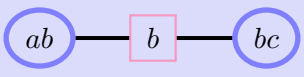
6.6.4 Some small JTA examples

Example 6.5 (A simple example of the JTA)**.**
Consider running the JTA on the simple graph



$$p(a, b, c) = p(a|b)p(b|c)p(c) \tag{6.6.6}$$

The moralisation and triangulation steps are trivial, and the JTA is given immediately by the figure on the right. A valid assignment is



$$\phi(a, b) = p(a|b),\ \phi(b) = 1,\ \phi(b, c) = p(b|c)p(c) \tag{6.6.7}$$

To find a marginal $p(b)$ we first run the JTA:

- Absorbing from ab through b, the new separator is $\phi^*(b) = \sum_a \phi(a, b) = \sum_a p(a|b) = 1$.
- The new potential on (b, c) is given by

$$\phi^*(b, c) = \frac{\phi(b, c)\phi^*(b)}{\phi(b)} = \frac{p(b|c)p(c) \times 1}{1} \tag{6.6.8}$$

- Absorbing from bc through b, the new separator is

$$\phi^{**}(b) = \sum_c \phi^*(b, c) = \sum_c p(b|c)p(c) \tag{6.6.9}$$

- The new potential on (a, b) is given by

$$\phi^*(a, b) = \frac{\phi(a, b)\phi^{**}(b)}{\phi^*(b)} = \frac{p(a|b)\sum_c p(b|c)p(c)}{1} \tag{6.6.10}$$

This is therefore indeed equal to the marginal since $\sum_c p(a, b, c) = p(a, b)$.
The new separator $\phi^{**}(b)$ contains the marginal $p(b)$ since

$$\phi^{**}(b) = \sum_c p(b|c)p(c) = \sum_c p(b, c) = p(b) \tag{6.6.11}$$

Example 6.6 (Finding a conditional marginal)**.** Continuing with the distribution in example(6.5), we consider how to compute $p(b|a = 1, c = 1)$. First we clamp the evidential variables in their states. Then we claim that the effect of running the JTA is to produce on a set of clique variables $\mathcal{X}$ the marginals on the cliques $p(\mathcal{X}, \mathcal{V})$. We demonstrate this below:

- In general, the new separator is given by $\phi^*(b) = \sum_a \phi(a, b) = \sum_a p(a|b) = 1$. However, since a is clamped in state $a = 1$, then the summation is not carried out over a, and we have instead $\phi^*(b) = p(a = 1|b)$.
- The new potential on the (b, c) clique is given by

$$\phi^*(b, c) = \frac{\phi(b, c)\phi^*(b)}{\phi(b)} = \frac{p(b|c = 1)p(c = 1)p(a = 1|b)}{1} \tag{6.6.12}$$

- The new separator is normally given by

$$\phi^{**}(b) = \sum_c \phi^*(b, c) \tag{6.6.13}$$ @@

However, since c is clamped in state 1, we have instead

$$\phi^{**}(b) = p(b|c = 1)p(c = 1)p(a = 1|b) \tag{6.6.14}$$

- The new potential on (a, b) is given by

$$\phi^*(a, b) = \frac{\phi(a, b)\phi^{**}(b)}{\phi^*(b)} = \frac{p(a = 1|b)p(b|c = 1)p(c = 1)p(a = 1|b)}{p(a = 1|b)} = p(a = 1|b)p(b|c = 1)p(c = 1) \tag{6.6.15}$$

which is the joint distribution $p(a = 1, b, c = 1)$. @@

The effect of clamping a set of variables $\mathcal{V}$ in their evidential states and running the JTA is that, for a clique i which contains the set of non-evidential variables $\mathcal{H}^i$, the consistent potential from the JTA contains the marginal $p(\mathcal{H}^i, \mathcal{V})$. Finding a conditional marginal is then straightforward by ensuring normalisation.

Example 6.7 (finding the likelihood $p(a = 1, c = 1)$)**.** One may also use the JTA to compute the marginal likelihood for variables not in the same clique since the effect of clamping the variables in their evidential states and running the JTA produces the joint marginals, such as $\phi^*(a, b) = p(a = 1, b, c = 1)$. Then calculating the likelihood is easy since we just sum out over the non-evidential variables of any converged potential : $p(a = 1, c = 1) = \sum_b \phi^*(a, b) = \sum_b p(a = 1, b, c = 1)$.

6.6.5 Shafer-Shenoy propagation

Consider the Markov network in fig(6.12a) for which a junction tree is given in fig(6.12b). We use the obvious notation shortcut of writing the variable indices alone. In the absorption procedure, we essentially store the result of message passing in the potentials and separators. An alternative message passing scheme for the junction tree can be derived as follows: Consider computing the marginal on the variables $2, 3, 4$, which involves summing over variables $1, 5, 6$:

$$p(2,3,4) = \sum_{1,5,6} \phi(1,2,5)\phi(1,3,6)\phi(1,2,3)\phi(2,3,4) \tag{6.6.16}$$

$$= \underbrace{\sum_{1} \underbrace{\sum_{5} \phi(1,2,5)}_{\lambda_{125\to 123}} \underbrace{\sum_{6} \phi(1,3,6)}_{\lambda_{136\to 123}} \phi(1,2,3)}_{\lambda_{123\to 234}} \phi(2,3,4) \tag{6.6.17}$$

In general, for a clique i with potential $\phi(\mathcal{V}_i)$, and neighbouring clique j with potential $\phi(\mathcal{V}_j)$, provided we have received messages from the other neighbours of i, we can send a message

$$\lambda_{i\to j} = \sum_{\mathcal{V}_i \setminus \mathcal{V}_j} \phi(\mathcal{V}_i) \prod_{k\neq j} \lambda_{k\to i} \tag{6.6.18}$$

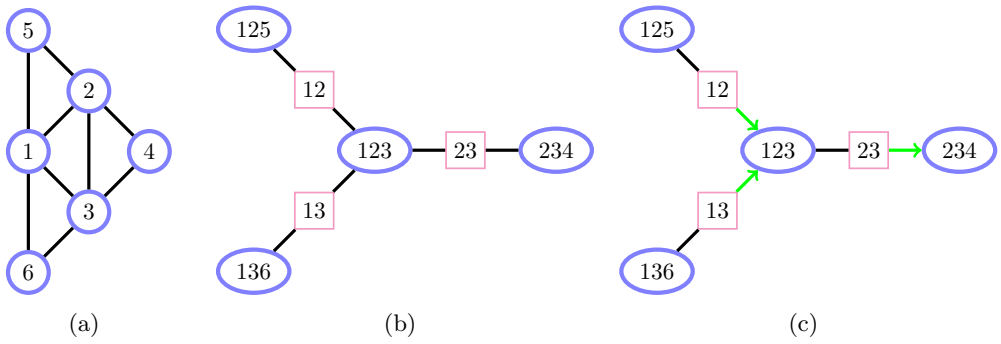
Once a full round of message passing has been completed, the marginal of any clique is given by the product of incoming messages.

This message passing scheme is called Shafer-Shenoy propagation and has the property that no division of potentials is required, unlike absorption. On the other hand, to compute a message we need to take the product of all incoming messages; in absorption this is not required since the effect of the message passing is stored in the clique potentials. The separators are not required in the Shafer-Shenoy approach and we use them here only to indicate which variables the messages depend on. Both absorption and Shafer-Shenoy propagation are valid message passing schemes on the junction tree and the relative efficacy of the approaches depends on the topology of the junction tree[188].

6.7 Finding the Most Likely State

It is often of interest to compute the most likely joint state of a distribution:

$$\underset{x_1,\ldots,x_n}{\text{argmax}}\; p(x_1,\ldots,x_n) \tag{6.7.1}$$



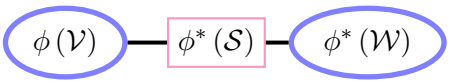
(a) (b) (c)

Figure 6.12: **(a)**: Markov network. **(b)**: Junction tree. Under absorption, once we have absorbed from 125 to 123 and 136 to 123, the result of these absorptions is stored in the new potential on 123. After this we can absorb from 123 to 234 by operating on the 123 potential and then sending this information to 234. **(c)**: In Shafer-Shenoy updating, we send a message from a clique to a neighbouring clique based on the product of all incoming messages.

Since the development of the JTA is based around a variable elimination procedure and the max operator distributes over the distribution as well, eliminating a variable by maximising over that variable will have the same effect on the graph structure as summation did. This means that a junction tree is again an appropriate structure on which to perform max operations. Once a JT has been constructed, one then uses the Max Absorption procedure (see below), to perform maximisation over the variables. After a full round of absorption has been carried out, the cliques contain the distribution on the variables of the clique with all remaining variables set to their optimal states. The optimal local states can then be found by explicit optimisation of each clique potential separately.

Note that this procedure holds also for non-distributions – in this sense this is an example of a more general dynamic programming procedure applied in a case where the underlying graph is multiply connected. This demonstrates how to efficiently compute the optimum of a multiply connected function defined as the product on potentials.

Definition 6.10 (Max Absorption).



Let $\mathcal{V}$ and $\mathcal{W}$ be neighbours in a clique graph, let $\mathcal{S}$ be their separator, and let $\phi(\mathcal{V})$, $\phi(\mathcal{W})$ and $\phi(\mathcal{S})$ be their potentials. Absorption replaces the tables $\phi(\mathcal{S})$ and $\phi(\mathcal{W})$ with

$$\phi^*(\mathcal{S}) = \max_{\mathcal{V}\backslash\mathcal{S}} \phi(\mathcal{V}) \qquad \phi^*(\mathcal{W}) = \phi(\mathcal{W}) \frac{\phi^*(\mathcal{S})}{\phi(\mathcal{S})}$$

Once messages have been passed in both directions over all separators, according to a valid schedule, the most-likely joint state can be read off from maximising the state of the clique potentials. This is implemented in `absorb.m` and `absorption.m` where a flag is used to switch between either sum or max absorption.

6.8 Reabsorption : Converting a Junction Tree to a Directed Network

It is sometimes useful to be able to convert a consistent JT (in which a full round of message passing has occurred) back to a BN of a desired form. For example, if one wishes to draw samples from a Markov network, this can be achieved by ancestral sampling on an equivalent directed structure, see section(27.2.2). @@

Definition 6.11 (Reabsorption).

Let $\mathcal{V}$ and $\mathcal{W}$ be neighbouring cliques in a directed consistent JT in which each clique in the tree has at most one parent. Furthermore, let $\mathcal{S}$ be their separator, and $\phi(\mathcal{V})$, $\phi(\mathcal{W})$ and $\phi(\mathcal{S})$ be the potentials. Reabsorption into $\mathcal{W}$ removes the separator and forms a (set) conditional distribution @@

$$p(\mathcal{W}\backslash\mathcal{S}|\mathcal{V}) = \frac{\phi(\mathcal{W})}{\phi(\mathcal{S})} \tag{6.8.1}$$

We say that clique $\mathcal{W}$ reabsorbs the separator $\mathcal{S}$.

Revisiting the example from fig(6.4), we have the JT given in fig(6.13a). To find a valid directed representation we first orient the JT edges consistently away from a chosen root node (see `singleparenttree.m`), thereby forming a directed JT which has the property that each clique has at most one parent clique. Consider fig(6.13a) which represents

$$p(a,b,c,d,e,f,g,h) = \frac{p(e,g)p(d,c,e)p(a,b,c)p(c,f)p(e,h)}{p(e)p(c)p(c)p(e)} \tag{6.8.2}$$

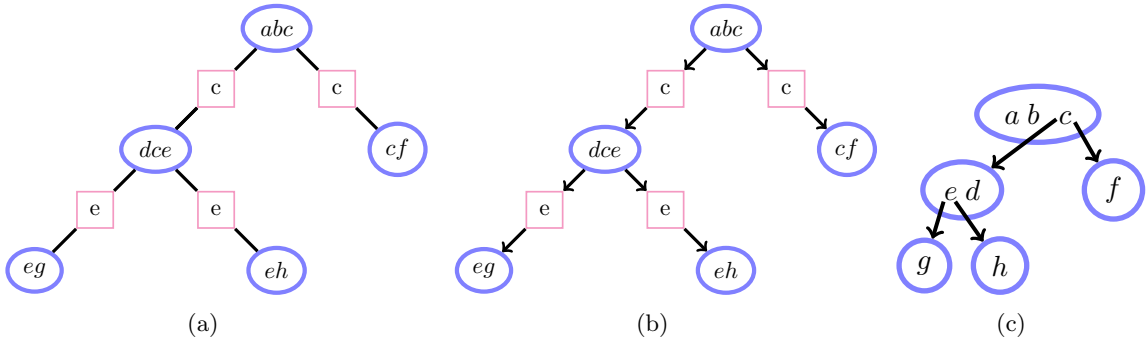


Figure 6.13: **(a)**: Junction tree. **(b)**: Directed junction tree in which all edges are consistently oriented away from the clique (abc). **(c)**: A set chain formed from the junction tree by reabsorbing each separator into its child clique.

We now have many choices as to which clique re-absorbs a separator. One such choice would give

@@ $$p(a,b,c,d,e,f,g,h) = p(g|e)p(d,e|c)p(a,b,c)p(f|c)p(h|e) \tag{6.8.3}$$

This can be represented using a so-called *set chain*[185] in fig(6.13c) (set chains generalise Belief Networks to a product of clusters of variables conditioned on parents). By writing each of the set conditional probabilities as local conditional BNs, one may also form a BN. For example, one such would be given from the decomposition

$$p(c|a,b)p(b|a)p(a)p(g|e)p(f|c)p(h|e)p(d|e,c)p(e|c) \tag{6.8.4}$$

6.9 The Need For Approximations

The JTA provides an upper bound on the complexity of (marginal/max) inference and attempts to exploit the structure of the graph to reduce computations. However, in a great deal of interesting applications the use of the JTA algorithm would result in clique-sizes in the triangulated graph that are prohibitively large. A classical situation in which this can arise are disease-symptom networks. For example, for the graph in fig(6.14), the triangulated graph of the diseases is fully connected, meaning that no simplification can occur in general. This situation is common in such bipartite networks, even when the children only have a small number of parents. Intuitively, as one eliminates each parent, links are added between other parents, mediated via the common children. Unless the graph is highly regular, analogous to a form of hidden Markov model, this fill-in effect rapidly results in large cliques and intractable computations.

Dealing with large cliques in the triangulated graph is an active research topic and we'll discuss strategies for approximate inference in chapter(28).

6.9.1 Bounded width junction trees

In some applications we may be at liberty to choose the structure of the Markov network. For example, if we wish to fit a Markov network to data, we may wish to use as complex a Markov network as we can computationally afford. In such cases we desire that the clique sizes of the resulting triangulated Markov network are smaller than a specified 'tree width' (considering the corresponding junction tree as a hypertree). This results in a 'thin' junction tree. A simple way to do this is to start with a graph and include a randomly chosen edge provided that the size of all cliques in the resulting triangulated graph is below a specified maximal width. See `demoThinJT.m` and `makeThinJT.m` which assumes an initial graph G and a graph of candidate edges C, iteratively expanding G until a maximal tree width limit is reached. See also [11] for a discussion on learning an appropriate Markov structure based on data.

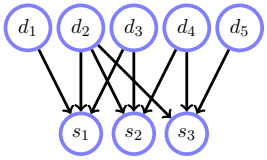


Figure 6.14: 5 diseases giving rise to 3 symptoms. The triangulated graph contains a 5 clique of all the diseases.

6.10 Summary

- The junction tree is a structure on clusters of variables such that, under inference operations such as marginalisation, the junction-tree structure remains invariant. This resolves the fill-in issue when using message-passing on a multiply connected graph.
- The key stages are moralisation, triangulation, potential assignment, and message-passing.
- There are different propagation algorithms, including absorption and Shafer-Shenoy. These are both valid message-passing algorithms on the junction tree and differ in their efficiency depending on the branch-structure of the junction tree.
- The junction tree algorithm does not make a difficult inference problem necessarily any easier. It is simply a way to organise the computations required to correctly carry out message-passing. The computational complexity is dominated by the clique-size and there is no guarantee that one can find cliques with small sizes in general.
- The junction tree algorithm is clever, but not clairvoyant. It provides only an upper bound on the computational complexity of inference. It may be that there are problems which possess additional structure, not immediately apparent, that can be exploited to reduce the computational complexity of inference much below that suggested by the junction-tree approach.

6.11 Code

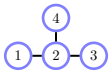
`absorb.m`: Absorption update $\mathcal{V} \to \mathcal{S} \to \mathcal{W}$
`absorption.m`: Full absorption schedule over tree
`jtree.m`: Form a junction tree
`triangulate.m`: Triangulation based on simple node elimination

6.11.1 Utility routines

Knowing if an undirected graph is a tree, and returning a valid elimination sequence is useful. A connected graph is a tree if the number of edges plus 1 is equal to the number of nodes. However, for a possibly disconnected graph this is not the case. The code `istree.m` deals with the possibly disconnected case, returning a valid elimination sequence if the graph is singly connected. The routine is based on the observation that any singly connected graph must always possess a simplicial node which can be eliminated to reveal a smaller singly connected graph.

`istree.m`: If graph is singly connected return 1 and elimination sequence
`elimtri.m`: Node elimination on a triangulated graph, with given end node
`demoJTree.m`: Chest clinic demo

6.12 Exercises

Exercise 6.1. *Show that the Markov network*  *is not perfect elimination ordered and give a perfect elimination labelling for this graph.*

Exercise 6.2. *Consider the following distribution:*

$$p(x_1, x_2, x_3, x_4) = \phi(x_1, x_2)\phi(x_2, x_3)\phi(x_3, x_4) \tag{6.12.1}$$

1. *Draw a clique graph that represents this distribution and indicate the separators on the graph.*
2. *Write down an alternative formula for the distribution $p(x_1, x_2, x_3, x_4)$ in terms of the marginal probabilities $p(x_1, x_2)$, $p(x_2, x_3)$, $p(x_3, x_4)$, $p(x_2)$, $p(x_3)$*

Exercise 6.3. *Consider the distribution*

$$p(x_1, x_2, x_3, x_4) = \phi(x_1, x_2)\phi(x_2, x_3)\phi(x_3, x_4)\phi(x_4, x_1) \tag{6.12.2}$$

1. *Write down a junction tree for the above distribution.*
2. *Carry out the absorption procedure and demonstrate that this gives the correct result for the marginal $p(x_1)$.*

Exercise 6.4. *Consider the distribution*

$$p(a, b, c, d, e, f, g, h, i) = p(a)p(b|a)p(c|a)p(d|a)p(e|b)p(f|c)p(g|d)p(h|e, f)p(i|f, g) \tag{6.12.3}$$

1. *Draw the belief network for this distribution.*
2. *Draw the moralised graph.*
3. *Draw the triangulated graph. Your triangulated graph should contain cliques of the smallest size possible.*
4. *Draw a junction tree for the above graph and verify that it satisfies the running intersection property.*
5. *Describe a suitable initialisation of clique potentials.*
6. *Describe the absorption procedure and write down an appropriate message updating schedule.*

Exercise 6.5. *This question concerns the distribution*

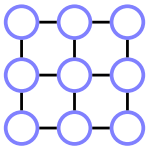
$$p(a, b, c, d, e, f) = p(a)p(b|a)p(c|b)p(d|c)p(e|d)p(f|a, e) \tag{6.12.4}$$

1. *Draw the Belief Network for this distribution.*
2. *Draw the moralised graph.*
3. *Draw the triangulated graph. Your triangulated graph should contain cliques of the smallest size possible.*
4. *Draw a junction tree for the above graph and verify that it satisfies the running intersection property.*
5. *Describe a suitable initialisation of clique potentials.*
6. *Describe the Absorption procedure and an appropriate message updating schedule.*
7. *Show that the distribution can be expressed in the form*

$$p(a|f)p(b|a, c)p(c|a, d)p(d|a, e)p(e|a, f)p(f) \tag{6.12.5}$$

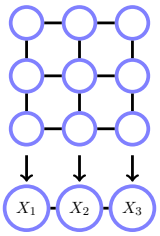
Exercise 6.6.

For the undirected graph on the square lattice as shown, draw a triangulated graph with the smallest clique sizes possible.



Exercise 6.7.

Consider a binary variable Markov Random Field $p(x) = Z^{-1}\prod_{i>j}\phi(x_i, x_j)$*, defined on the* $n \times n$ *lattice with* $\phi(x_i, x_j) = e^{\mathbb{I}[x_i=x_j]}$ *for* i *a neighbour of* j *on the lattice and* $i > j$*. A naive way to perform inference is to first stack all the variables in the* t^{th} *column and call this cluster variable* X_t*, as shown. The resulting graph is then singly connected. What is the complexity of computing the normalisation constant based on this cluster representation? Compute* $\log Z$ *for* $n = 10$*.*



Exercise 6.8. *Given a consistent junction tree on which a full round of message passing has occurred, explain how to form a belief network from the junction tree.*

Exercise 6.9. *The file* `diseaseNet.mat` *contains the potentials for a disease bi-partite belief network, with 20 diseases* $d_1, \ldots, d_{20}$ *and 40 symptoms,* $s_1, \ldots, s_{40}$*. The disease variables are numbered from 1 to 20 and the symptoms from 21 to 60. Each disease and symptom is a binary variable, and each symptom connects to 3 parent diseases.*

1. *Using the* BRMLtoolbox*, construct a junction tree for this distribution and use it to compute all the marginals of the symptoms,* $p(s_i = 1)$*.*

2. *Explain how to compute the marginals* $p(s_i = 1)$ *in a way more efficient than using the junction tree formalism. By implementing this method, compare it with the results from the junction tree algorithm.*

3. *Symptoms 1 to 5 are present (state 1), symptoms 6 to 10 not present (state 2), and the rest not known. Compute the marginal* $p(d_i = 1|s_{1:10})$ *for all diseases.*

Exercise 6.10. *Consider the distribution*

$$p(y|x_1, \ldots, x_T)p(x_1)\prod_{t=2}^{T} p(x_t|x_{t-1})$$

where all variables are binary.

1. *Draw a junction tree for this distribution and explain the computational complexity of computing* $p(x_T)$*, as suggested by the junction tree algorithm.*

2. *By using an approach different from the plain JTA above, explain how* $p(x_T)$ *can be computed in time that scales linearly with* T*.*

Exercise 6.11. *Analogous to* `jtpot=absorption(jtpot,jtsep,infostruct)`*, write a routine* `[jtpot jtmess]=ShaferShenoy(jtpot,infostruct)` *that returns the clique marginals and messages for a junction tree under Shafer-Shenoy updating. Modify* `demoJTree.m` *to additionally output your results for marginals and conditional marginals alongside those obtained using absorption.*

Exercise 6.12 (Clique elimination)**.** *Since a junction tree is indeed a tree, it must have an extremal ('edge') clique* $\mathcal{X}_1$ *that connects to only one other clique through their separator* $\mathcal{S}_1$*. More generally, we must be able to label the cliques in an elimination order,* $1, 2, \ldots, n$ *such that by eliminating clique* i *we only have cliques left with label greater than* i*. For a separator connected to two cliques* j *and* k*, we then number this separator the lower of the two values,* $\min(j, k)$*. Using this elimination ordering of the cliques we can write the JT in the form (using* $\phi(\mathcal{X})$ *to represent clique potentials over clique variables* $\mathcal{X}$ *and* $\psi(\mathcal{S})$ *to represent separator potentials over separator variables* $\mathcal{S}$*)*

$$p(\mathcal{X}) = \frac{\phi_1(\mathcal{X}_1)}{\psi_1(\mathcal{S}_1)}\frac{\prod_{c\geq 2}\phi_c(\mathcal{X}_c)}{\prod_{s\geq 2}\psi_s(\mathcal{S}_s)} \tag{6.12.6}$$

1. *Now eliminate clique* 1 *by summing over all variables in clique* 1 *that are not in the separator* $\mathcal{S}_1$. *Show that this gives a new JT on cliques* $c \geq 2$ *and separators* $s \geq 2$ *of the form*

$$\phi'_d(\mathcal{X}_d)\frac{\prod_{c\geq 2, c\neq d}\phi_c(\mathcal{X}_c)}{\prod_{s\geq 2}\psi_s(\mathcal{S}_s)} \tag{6.12.7}$$

where d *is the clique neighbouring clique* 1 *and*

$$\phi'_d(\mathcal{X}_d) = \frac{\phi_d(\mathcal{X}_d)\sum_{\mathcal{X}_1\setminus\mathcal{S}_1}\phi_1(\mathcal{X}_1)}{\psi_1(\mathcal{S}_1)} \tag{6.12.8}$$

Furthermore, by considering that

$$p(\mathcal{X}) = p(\mathcal{X}_1\setminus\mathcal{S}_1|\mathcal{S}_1)p(\mathcal{X}\setminus\{\mathcal{X}_1\setminus\mathcal{S}_1\})$$

show that by updating

$$\psi'_1(\mathcal{S}_1) = \sum_{\mathcal{X}_1\setminus\mathcal{S}_1}\phi_1(\mathcal{X}_1)$$

the updated JT on $p(\mathcal{X})$ *still represents the original distribution and relate this to the absorption procedure.*

2. *Show that by continuing in this manner, eliminating cliques one by one, the last clique must contain the marginal* $p(\mathcal{X}_n)$.

3. *Explain how by now reversing the elimination schedule, and eliminating cliques one by one, updating their potentials in a similar manner to Equation(6.12.8), the updated cliques* $\phi_j(\mathcal{X}_j)$ *will contain the marginals* $p(\mathcal{X}_j)$.

4. *Hence explain why, after updating cliques according to the forward and reversed elimination schedules, the cliques must be globally consistent.*

CHAPTER 7

Making Decisions

So far we've considered modelling and inference of distributions. In cases where we need to make decisions under uncertainty, we need to additionally express how useful making the right decision is. In this chapter we are particularly interested in the case when a sequence of decisions need to be taken. The corresponding sequential decision theory problems can be solved using either a general decision tree approach or by exploiting structure in the problem based on extending the belief network framework and the corresponding inference routines. The framework is related to problems in control theory and reinforcement learning.

7.1 Expected Utility

This chapter concerns situations in which decisions need to be taken under uncertainty. Consider the following scenario: you are asked if you wish to take a bet on the outcome of tossing a fair coin. If you bet and win, you gain £100. If you bet and lose, you lose £200. If you don't bet, the cost to you is zero. We can set this up using a two state variable x, with $\mathrm{dom}(x) = \{\mathsf{win}, \mathsf{lose}\}$, a decision variable d with $\mathrm{dom}(d) = \{\mathsf{bet}, \mathsf{no\ bet}\}$ and utilities as follows:

$$U(\mathsf{win}, \mathsf{bet}) = 100, \quad U(\mathsf{lose}, \mathsf{bet}) = -200, \quad U(\mathsf{win}, \mathsf{no\ bet}) = 0, \quad U(\mathsf{lose}, \mathsf{no\ bet}) = 0 \tag{7.1.1}$$

Since we don't know the state of x, in order to make a decision about whether or not to bet, arguably the best we can do is work out our expected winnings/losses under the situations of betting and not betting[258]. If we bet, we would expect to gain

$$U(\mathsf{bet}) = p(\mathsf{win}) \times U(\mathsf{win}, \mathsf{bet}) + p(\mathsf{lose}) \times U(\mathsf{lose}, \mathsf{bet}) = 0.5 \times 100 - 0.5 \times 200 = -50$$

If we don't bet, the expected gain is zero, $U(\mathsf{no\ bet}) = 0$. Based on taking the decision which maximises expected utility, we would therefore be advised not to bet.

Definition 7.1 (Subjective Expected Utility)**.** The utility of a decision is

$$U(d) = \langle U(d, x) \rangle_{p(x)} \tag{7.1.2}$$

where $p(x)$ is the distribution of the outcome x and d represents the decision.

7.1.1 Utility of money

You are a wealthy individual, with £1,000,000 in your bank account. You are asked if you would like to participate in a fair coin tossing bet in which, if you win, your bank account will become £1,000,000,000.

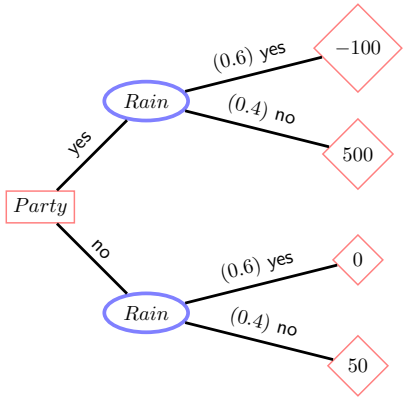


Figure 7.1: A decision tree containing chance nodes (denoted with ovals), decision nodes (denoted with rectangles) and utility nodes (denoted with diamonds). Note that a decision tree is not a graphical representation of a belief network with additional nodes. Rather, a decision tree is an explicit enumeration of the possible choices that can be made, beginning with the leftmost decision node, with probabilities on the links out of chance nodes.

However, if you lose, your bank account will contain only £1000. Assuming the coin is fair, should you take the bet? If we take the bet our expected bank balance would be

$$U(\text{bet}) = 0.5 \times 1,000,000,000 + 0.5 \times 1000 = 500,000,500.00 \tag{7.1.3}$$

If we don't bet, our bank balance will remain at £1,000,000. Based on expected utility, we are therefore advised to take the bet. (Note that if one considers instead the amount one will win or lose, one may show that the difference in expected utility between betting and not betting is the same, exercise(7.7)).

Whilst the above makes mathematical sense, few people who are millionaires are likely to be willing to risk losing almost everything in order to become a billionaire. This means that the subjective utility of money is not simply the quantity of money. In order to better reflect the situation, the utility of money would need to be a non-linear function of money, growing slowly for large quantities of money and decreasing rapidly for small quantities of money, exercise(7.2).

7.2 Decision Trees

Decision trees (DTs) are a way to graphically organise a sequential decision process. A decision tree contains decision nodes, each with branches for each of the alternative decisions. Chance nodes (random variables) also appear in the tree, with the utility of each branch computed at the leaf of each branch. The expected utility of any decision can then be computed on the basis of the weighted summation of all branches from the decision to all leaves from that branch.

Example 7.1 (Party)**.** Consider the decision problem as to whether or not to go ahead with a fund-raising garden party. If we go ahead with the party and it subsequently rains, then we will lose money (since very few people will show up); on the other hand, if we don't go ahead with the party and it doesn't rain we're free to go and do something else fun. To characterise this numerically, we use:

$$p(Rain = \text{rain}) = 0.6,\ p(Rain = \text{no rain}) = 0.4 \tag{7.2.1}$$

The utility is defined as

$$U(\text{party}, \text{rain}) = -100,\quad U(\text{party}, \text{no rain}) = 500,\quad U(\text{no party}, \text{rain}) = 0,\quad U(\text{no party}, \text{no rain}) = 50 \tag{7.2.2}$$

We represent this situation in fig(7.1). The question is, should we go ahead with the party? Since we don't know what will actually happen to the weather, we compute the expected utility of each decision:

$$U(\text{party}) = \sum_{Rain} U(\text{party}, Rain)p(Rain) = -100 \times 0.6 + 500 \times 0.4 = 140 \tag{7.2.3}$$

$$U(\text{no party}) = \sum_{Rain} U(\text{no party}, Rain)p(Rain) = 0 \times 0.6 + 50 \times 0.4 = 20 \tag{7.2.4}$$

Based on expected utility, we are therefore advised to go ahead with the party. The maximal expected utility is given by (see `demoDecParty.m`)

$$\max_{Party} \sum_{Rain} p(Rain)U(Party, Rain) = 140 \tag{7.2.5}$$

Example 7.2 (Party-Friend)**.** An extension of the Party problem is that if we decide not to go ahead with the party, we have the opportunity to visit a friend. However, we're not sure if this friend will be in. The question is should we still go ahead with the party?

We need to quantify all the uncertainties and utilities. If we go ahead with the party, the utilities are as before:

$$U_{party}(\text{party}, \text{rain}) = -100, \quad U_{party}(\text{party}, \text{no rain}) = 500 \tag{7.2.6}$$

with

$$p(Rain = \text{rain}) = 0.6, \quad p(Rain = \text{no rain}) = 0.4 \tag{7.2.7}$$

If we decide not to go ahead with the party, we will consider going to visit a friend. In making the decision not to go ahead with the party we have utilities

$$U_{party}(\text{no party}, \text{rain}) = 0, \quad U_{party}(\text{no party}, \text{no rain}) = 50 \tag{7.2.8}$$

The probability that the friend is in depends on the weather according to

$$p(Friend = \text{in}|\text{rain}) = 0.8, \quad p(Friend = \text{in}|\text{no rain}) = 0.1, \tag{7.2.9}$$

The other probabilities are determined by normalisation. We additionally have

$$U_{visit}(\text{friend in}, \text{visit}) = 200, \quad U_{visit}(\text{friend out}, \text{visit}) = -100 \tag{7.2.10}$$

with the remaining utilities zero. The two sets of utilities add up so that the overall utility of any decision sequence is $U_{party} + U_{visit}$. The decision tree for the Party-Friend problem is shown is fig(7.2). For each decision sequence the utility of that sequence is given at the corresponding leaf of the DT. Note that the leaves contain the total utility $U_{party} + U_{visit}$. Solving the DT corresponds to finding for each decision node the maximal expected utility possible (by optimising over future decisions). At any point in the tree choosing that action which leads to the child with highest expected utility will lead to the optimal strategy. Using this, we find that the optimal expected utility has value 140 and is given by going ahead with the party, see `demoDecPartyFriend.m`.

Mathematically, we can express the optimal expected utility for the Party-Friend example by summing over un-revealed variables and optimising over future decisions:

$$\max_{Party} \sum_{Rain} p(Rain) \max_{Visit} \sum_{Friend} p(Friend|Rain) \left[U_{party}(Party, Rain) + U_{visit}(Visit, Friend)\mathbb{I}\left[Party = \text{no}\right]\right] \tag{7.2.11}$$

where the term $\mathbb{I}\left[Party = \text{no}\right]$ has the effect of curtailing the DT if the party goes ahead. To answer the question as to whether or not to go ahead with the party, we take that state of $Party$ that corresponds to the maximal expected utility above. The way to read equation (7.2.11) is to start from the last decision that needs to be taken, in this case $Visit$. When we are at the $Visit$ stage we assume that we will have previously made a decision about $Party$ and also will have observed whether or not it is raining. However,

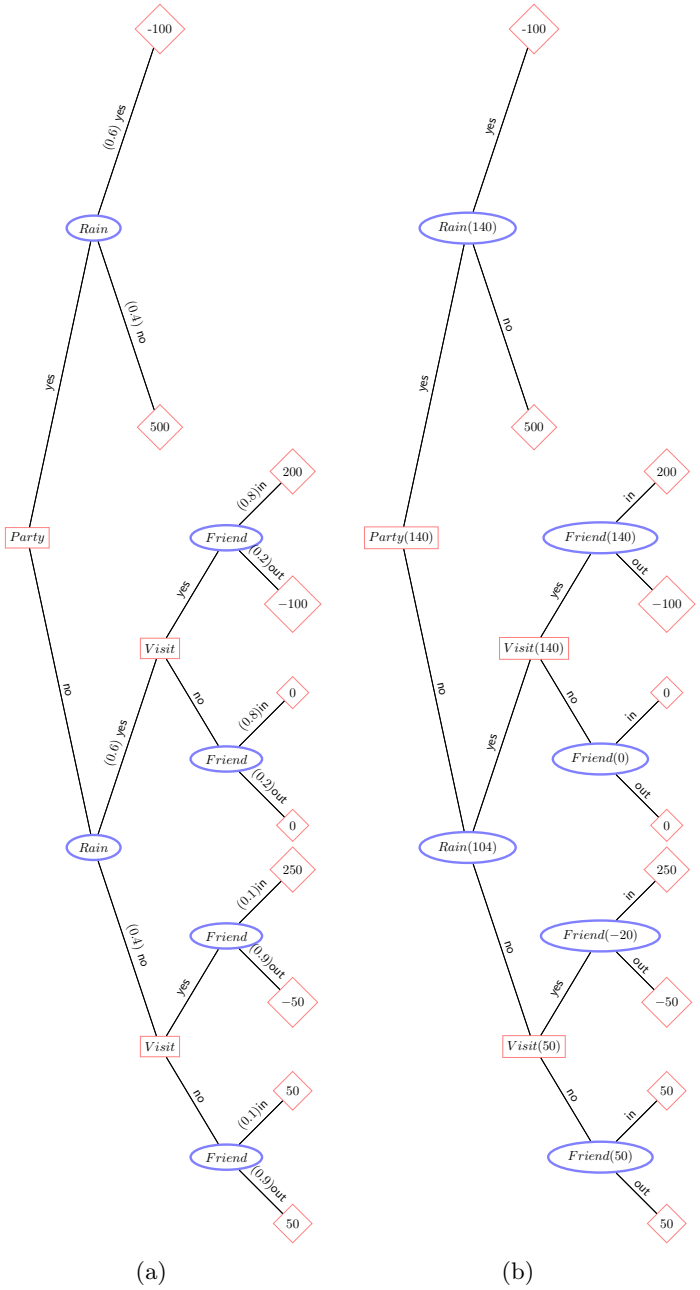


Figure 7.2: Solving a Decision Tree. **(a)**: Decision Tree for the Party-Friend problem, example(7.2). **(b)**: Solving the DT corresponds to making the decision with the highest expected future utility. This can be achieved by starting at the leaves (utilities). For a chance parent node x, the utility of the parent is the expected utility of that variable. For example, at the top of the DT we have the *Rain* variable with the children -100 (probability 0.6) and 500 (probability 0.4). Hence the expected utility of the *Rain* node is $-100 \times 0.6 + 500 \times 0.4 = 140$. For a decision node, the value of the node is the optimum of its child values. One recurses thus backwards from the leaves to the root. For example, the value of the *Rain* chance node in the lower branch is given by $140 \times 0.6 + 50 \times 0.4 = 104$. The optimal decision sequence is then given at each decision node by finding which child node has the maximal value. Hence the overall best decision is to decide to go ahead with the party. If we decided not to do so, and it does not rain, then the best decision we could take would be to not visit the friend (which has an expected utility of 50). A more compact description of this problem is given by the influence diagram, fig(7.4). See also `demoDecPartyFriend.m`.

we don't know whether or not our friend will be in, so we compute the expected utility by averaging over this unknown. We then take the optimal decision by maximising over $Visit$. Subsequently we move to the next-to-last decision, assuming that what we will do in the future is optimal. Since in the future we will have taken a decision under the uncertain *Friend* variable, the current decision can then be taken under uncertainty about *Rain* and maximising this expected optimal utility over *Party*. Note that the sequence of maximisations and summations matters – changing the order will in general result in a different problem with a different expected utility[1].

For the Party-Friend example the DT is asymmetric since if we decide to go ahead with the party we will not visit the friend, curtailing the further decisions present in the lower half of the tree. Whilst the DT approach is flexible and can handle decision problems with arbitrary structure, a drawback is that the same nodes are often repeated throughout the decision tree. For a longer sequence of decisions, the number of branches in the tree can grow exponentially with the number of decisions, making this representation impractical.

[1] If one only had a sequence of summations, the order of the summations is irrelevant – likewise for the case of all maximisations. However, summation and maximisation operators do not in general commute.

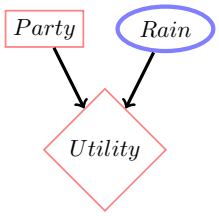


Figure 7.3: An influence diagram which contains random variables (denoted with ovals/circles) Decision nodes (denoted with rectangles) and Utility nodes (denoted with diamonds). Contrasted with fig(7.1) this is a more compact representation of the structure of the problem. The diagram represents the expression $p(rain)u(party, rain)$. In addition the diagram denotes an ordering of the variables with $party \prec rain$ (according to the convention given by equation (7.3.1)).

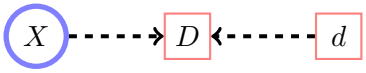
7.3 Extending Bayesian Networks for Decisions

An *influence diagram* is a Bayesian Network with additional Decision nodes and Utility nodes [150, 162, 176]. The decision nodes have no associated distribution and the utility nodes are deterministic functions of their parents. The utility and decision nodes can be either continuous or discrete; for simplicity, in the examples here the decisions will be discrete.

A benefit of decision trees is that they are general and explicitly encode the utilities and probabilities associated with each decision and event. In addition, we can readily solve small decision problems using decision trees. However, when the sequence of decisions increases, the number of leaves in the decision tree grows and representing the tree can become an exponentially complex problem. In such cases it can be useful to use an Influence Diagram (ID). An ID states which information is required in order to make each decision, and the order in which these decisions are to be made. The details of the probabilities and utilities are not specified in the ID, and this can enable a more compact description of the decision problem.

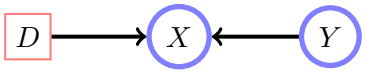
7.3.1 Syntax of influence diagrams

Information Links An *information link* from a random variable into a decision node



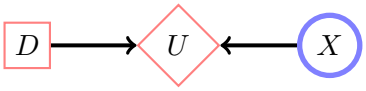
indicates that the state of the variable X will be known before decision D is taken. Information links from another decision node d in to D similarly indicate that decision d is known before decision D is taken. We use a dashed link to denote that decision D is not functionally related to its parents.

Random Variables Random variables may depend on the states of parental random variables (as in belief networks), but also decision node states:



As decisions are taken, the states of some random variables will be revealed. To emphasise this we typically shade a node to denote that its state will be revealed during the sequential decision process.

Utilities A utility node is a deterministic function of its parents. The parents can be either random variables or decision nodes.



In the party example, the BN trivially consists of a single node, and the Influence Diagram is given in fig(7.3). The more complex Party-Friend problem is depicted in fig(7.4). The ID generally provides a more compact representation of the structure of problem than a DT, although details about the specific probabilities and utilities are not present in the ID.

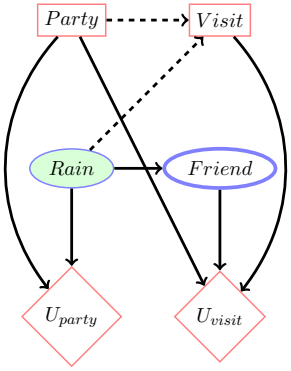


Figure 7.4: An influence diagram for the Party-Friend problem, example(7.2). The partial ordering is $Party^* \prec Rain \prec Visit^* \prec Friend$. The dashed-link from party to visit is not strictly necessary but retained in order to satisfy the convention that there is a directed path connecting all decision nodes.

Partial ordering

An ID defines a *partial ordering* of the nodes. We begin by writing those variables $\mathcal{X}_0$ whose states are known (evidential variables) before the first decision D_1. We then find that set of variables $\mathcal{X}_1$ whose states are revealed before the second decision D_2. Subsequently the set of variables $\mathcal{X}_t$ is revealed before decision D_{t+1}. The remaining fully-unobserved variables are placed at the end of the ordering:

$$\mathcal{X}_0 \prec D_1 \prec \mathcal{X}_1 \prec D_2, \ldots, \prec \mathcal{X}_{n-1} \prec D_n \prec \mathcal{X}_n \tag{7.3.1}$$

with $\mathcal{X}_k$ being the variables revealed between decision D_k and D_{k+1}. The term 'partial' refers to the fact that there is no order implied amongst the variables within the set $\mathcal{X}_n$. For notational clarity, at points below we will indicate decision variables with * to reinforce that we maximise over these variables, and sum over the non-starred variables. Where the sets are empty we omit writing them. In fig(7.5a) we consider @@ an oil exploration situation in which there is first a decision to be taken as to whether or not to carry out a seismic test, with an associated utility (cost) U_1. The result of this test is represented by the variable *Seismic*, and depends on whether there is oil present. Based on this seismic result, a subsequent decision is to be taken as to whether or not to drill for oil, which has an associated utility U_2. The ordering is $Test^* \prec Seismic \prec Drill^* \prec Oil$.

The optimal first decision D_1 is determined by computing

$$U(D_1|\mathcal{X}_0) \equiv \sum_{\mathcal{X}_1} \max_{D_2} \ldots \sum_{\mathcal{X}_{n-1}} \max_{D_n} \sum_{\mathcal{X}_n} \prod_{i\in\mathcal{I}} p\left(x_i|\mathrm{pa}\left(x_i\right)\right) \sum_{j\in\mathcal{J}} U_j\left(\mathrm{pa}\left(u_j\right)\right) \tag{7.3.2}$$

for each state of the decision D_1, given $\mathcal{X}_0$. In equation (7.3.2) above $\mathcal{I}$ denotes the set of indices for the random variables, and $\mathcal{J}$ the indices for the utility nodes. For each state of the conditioning variables, the optimal decision D_1 is found using

$$\operatorname*{argmax}_{D_1} U(D_1|\mathcal{X}_0) \tag{7.3.3}$$

Remark 7.1 (Reading off the partial ordering)**.** Sometimes it can be tricky to read the partial ordering from the ID. A method is to identify the first decision D_1 and then any variables $\mathcal{X}_0$ that need to be observed to make that decision. Then identify the next decision D_2 and the variables $\mathcal{X}_1$ that are revealed after decision D_1 is taken and before decision D_2 is taken, *etc.* This gives the partial ordering $\mathcal{X}_0 \prec D_1 \prec \mathcal{X}_1 \prec D_2, \ldots$. Place any unrevealed variables at the end of the ordering.

Implicit and explicit information links

The information links are a potential source of confusion. An information link specifies explicitly which quantities are known before that decision is taken[2]. We also implicitly assume the *no forgetting assumption* that all past decisions and revealed variables are available at the current decision (the revealed variables

[2]Some authors prefer to write all information links where possible, and others prefer to leave them implicit. Here we largely take the implicit approach. For the purposes of computation, all that is required is a partial ordering; one can therefore view this as 'basic' and the information links as superficial (see [73]).

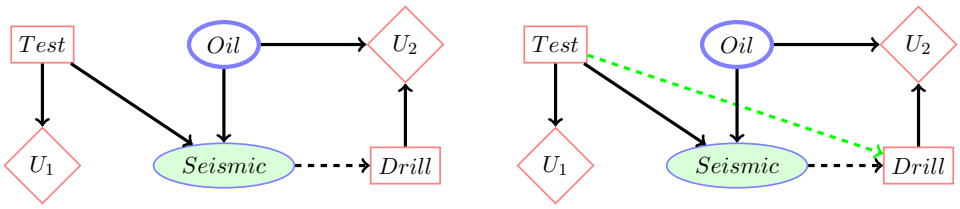


Figure 7.5: **(a)**: The partial ordering is $Test^* \prec Seismic \prec Drill^* \prec Oil$. The explicit information links from $Test$ to $Seismic$ and from $Seismic$ to $Drill$ are both fundamental in the sense that removing either results in a different partial ordering. The shaded node emphasises that the state of this variable will be revealed during the sequential decision process. Conversely, the non-shaded node will never be observed. **(b)**: Based on the ID in (a), there is an implicit link from $Test$ to $Drill$ since the decision about $Test$ is taken before $Seismic$ is revealed.

@@ are necessarily the parents of all future decision nodes). If we were to include all such information links, IDs would get potentially rather messy. In fig(7.5), both explicit and implicit information links are demonstrated. We call an information link *fundamental* if its removal would alter the partial ordering.

Causal consistency

@@ For an Influence Diagram to be consistent a current decision cannot affect the past. This means that any random variable descendants of a decision D in the ID must come later in the partial ordering.

Asymmetry

IDs are most convenient when the corresponding DT is symmetric. However, some forms of asymmetry are relatively straightforward to deal with in the ID framework. For our Party-Friend example, the DT is asymmetric. However, this is easily dealt with in the ID by using a link from $Party$ to U_{visit} which removes the contribution from U_{visit} when the party goes ahead.

More complex issues arise when the set of variables that can be observed depends on the decision sequence taken. In this case the DT is asymmetric. In general, Influence Diagrams are not well suited to modelling such asymmetries, although some effects can be mediated either by careful use of additional variables, or extending the ID notation. See [73] and [162] for further details of these issues and possible resolutions.

Example 7.3 (Should I do a PhD?)**.** Consider a decision whether or not to do PhD as part of our education (E). Taking a PhD incurs costs, U_C both in terms of fees, but also in terms of lost income. However, if we have a PhD, we are more likely to win a Nobel Prize (P), which would certainly be likely to boost our Income (I), subsequently benefitting our finances (U_B). This setup is depicted in fig(7.6a). The ordering is (excluding empty sets)

$$E^* \prec \{I, P\} \tag{7.3.4}$$

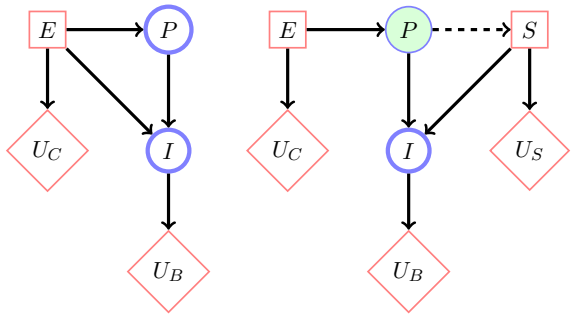
and

$$\mathrm{dom}(E) = (\mathsf{do\ PhD}, \mathsf{no\ PhD}), \quad \mathrm{dom}(I) = (\mathsf{low}, \mathsf{average}, \mathsf{high}), \quad \mathrm{dom}(P) = (\mathsf{prize}, \mathsf{no\ prize}) \tag{7.3.5}$$

The probabilities are

$$p(\mathsf{win\ Nobel\ prize}|\mathsf{no\ PhD}) = 0.0000001 \qquad p(\mathsf{win\ Nobel\ prize}|\mathsf{do\ PhD}) = 0.001 \tag{7.3.6}$$

$$\begin{array}{lll} p(\mathsf{low}|\mathsf{do\ PhD}, \mathsf{no\ prize}) = 0.1 & p(\mathsf{average}|\mathsf{do\ PhD}, \mathsf{no\ prize}) = 0.5 & p(\mathsf{high}|\mathsf{do\ PhD}, \mathsf{no\ prize}) = 0.4 \\ p(\mathsf{low}|\mathsf{no\ PhD}, \mathsf{no\ prize}) = 0.2 & p(\mathsf{average}|\mathsf{no\ PhD}, \mathsf{no\ prize}) = 0.6 & p(\mathsf{high}|\mathsf{no\ PhD}, \mathsf{no\ prize}) = 0.2 \\ p(\mathsf{low}|\mathsf{do\ PhD}, \mathsf{prize}) = 0.01 & p(\mathsf{average}|\mathsf{do\ PhD}, \mathsf{prize}) = 0.04 & p(\mathsf{high}|\mathsf{do\ PhD}, \mathsf{prize}) = 0.95 \\ p(\mathsf{low}|\mathsf{no\ PhD}, \mathsf{prize}) = 0.01 & p(\mathsf{average}|\mathsf{no\ PhD}, \mathsf{prize}) = 0.04 & p(\mathsf{high}|\mathsf{no\ PhD}, \mathsf{prize}) = 0.95 \end{array} \tag{7.3.7}$$



(a) (b)

Figure 7.6: **(a)**: Education E incurs some cost, but also gives a chance to win a prestigious science prize. Both of these affect our likely Incomes, with corresponding long term financial benefits. **(b)**: The start-up scenario.

The utilities are

$$U_C(\text{do PhD}) = -50000, \quad U_C(\text{no PhD}) = 0, \tag{7.3.8}$$

$$U_B(\text{low}) = 100000, \quad U_B(\text{average}) = 200000, \quad U_B(\text{high}) = 500000 \tag{7.3.9}$$

The expected utility of Education is

$$U(E) = \sum_{I,P} p(I|E,P)p(P|E)\left[U_C(E) + U_B(I)\right] \tag{7.3.10}$$

so that $U(\text{do phd}) = 260174.000$, whilst not taking a PhD is $U(\text{no phd}) = 240000.0244$, making it on average beneficial to do a PhD. See `demoDecPhD.m`.

Example 7.4 (PhDs and Start-up companies)**.** Influence Diagrams are particularly useful when a sequence of decisions is taken. For example, in fig(7.6b) we model a new situation (now dropping the link from E to I simply to reduce the table size below) in which someone has first decided whether or not to take a PhD. Ten years later in their career they decide whether or not to make a start-up company. This decision is based on whether or not they won the Nobel Prize. The start-up decision is modelled by S with $\text{dom}(S) = (\text{tr}, \text{fa})$. If we make a start-up, this will cost some money in terms of investment. However, the potential benefit in terms of our income could be high. ++

We model this with (the other required table entries being taken from example(7.3)):

$$\begin{array}{lll} p(\text{low}|\text{start up}, \text{no prize}) = 0.1 & p(\text{average}|\text{start up}, \text{no prize}) = 0.5 & p(\text{high}|\text{start up}, \text{no prize}) = 0.4 \\ p(\text{low}|\text{no start up}, \text{no prize}) = 0.2 & p(\text{average}|\text{no start up}, \text{no prize}) = 0.6 & p(\text{high}|\text{no start up}, \text{no prize}) = 0.2 \\ p(\text{low}|\text{start up}, \text{prize}) = 0.005 & p(\text{average}|\text{start up}, \text{prize}) = 0.005 & p(\text{high}|\text{start up}, \text{prize}) = 0.99 \\ p(\text{low}|\text{no start up}, \text{prize}) = 0.05 & p(\text{average}|\text{no start up}, \text{prize}) = 0.15 & p(\text{high}|\text{no start up}, \text{prize}) = 0.8 \end{array} \tag{7.3.11}$$

and

$$U_S(\text{start up}) = -200000, \qquad U_S(\text{no start up}) = 0 \tag{7.3.12}$$

Our interest is to advise whether or not it is desirable (in terms of expected utility) to take a PhD, now bearing in mind that later one may or may not win the Nobel Prize, and may or may not make a start-up company.

The ordering is (eliding empty sets)

$$E^* \prec P \prec S^* \prec I \tag{7.3.13}$$

The expected optimal utility for any state of E is

$$U(E) = \sum_P \max_S \sum_I p(I|S,P)p(P|E)\left[U_S(S) + U_C(E) + U_B(I)\right] \tag{7.3.14}$$

where we assume that the optimal decisions are taken in the future. Computing the above, we find

$$U(\text{do PhD}) = 190195.00, \qquad U(\text{no PhD}) = 240000.02 \tag{7.3.15}$$

Hence, we are better off not doing a PhD. See `demoDecPhd.m`.

7.4 Solving Influence Diagrams

Solving an influence diagram means computing the optimal decision or sequence of decisions. The direct variable elimination approach is to take equation (7.3.2) and perform the required sequence of summations and maximisations explicitly. Due to the causal consistency requirement the future cannot influence the past. To help matters with notation, we order the variables and decisions such that we may write the belief network of the influence diagram as

$$p(x_{1:T}, d_{1:T}) = \prod_{t=1}^{T} p(x_t|x_{1:t-1}, d_{1:t}) \tag{7.4.1}$$

@@ For a general utility $u(x_{1:T}, d_{1:T})$, solving the ID then corresponds to carrying out the operations

$$\max_{d_1} \sum_{x_1} \ldots \max_{d_T} \sum_{x_T} \prod_{t=1}^{T} p(x_t|x_{1:t-1}, d_{1:t}) u(x_{1:T}, d_{1:T}) \tag{7.4.2}$$

Let's look at eliminating first x_T and then d_T. Our aim is to write a new ID on the reduced variables $x_{1:T-1}, d_{1:T-1}$. Since x_T and d_T only appear in the final factor of the belief network, we can write

$$\max_{d_1} \sum_{x_1} \ldots \max_{d_{T-1}} \sum_{x_{T-1}} \prod_{t=1}^{T-1} p(x_t|x_{1:t-1}, d_{1:t}) \max_{d_T} \sum_{x_T} p(x_T|x_{1:T-1}, d_{1:T}) u(x_{1:T}, d_{1:T}) \tag{7.4.3}$$

which is then a new ID

$$\max_{d_1} \sum_{x_1} \ldots \max_{d_{T-1}} \sum_{x_{T-1}} \prod_{t=1}^{T-1} p(x_t|x_{1:t-1}, d_{1:t}) \tilde{u}(x_{1:T-1}, d_{1:T-1}) \tag{7.4.4}$$

with modified potential

$$\tilde{u}(x_{1:T-1}, d_{1:T-1}) \equiv \max_{d_T} \sum_{x_T} p(x_T|x_{1:T-1}, d_{1:T}) u(x_{1:T}, d_{1:T}) \tag{7.4.5}$$

This however doesn't exploit the fact that the utilities will typically have structure. Without loss of generality, we may also write the utility as one that is independent of x_T, d_T and one that depends on x_T, d_T:

$$u(x_{1:T}, d_{1:T}) = u_a(x_{1:T-1}, d_{1:T-1}) + u_b(x_{1:T}, d_{1:T}) \tag{7.4.6}$$

Then eliminating x_T, d_T updates the utility to

$$\tilde{u}(x_{1:T-1}, d_{1:T-1}) = u_a(x_{1:T-1}, d_{1:T-1}) + \max_{d_T} \sum_{x_T} p(x_T|x_{1:T-1}, d_{1:T}) u_b(x_{1:T}, d_{1:T}) \tag{7.4.7}$$

7.4.1 Messages on an ID

For an ID with two sets of variables $\mathcal{X}_1, \mathcal{X}_2$ and associated decision sets $\mathcal{D}_1$ and $\mathcal{D}_2$, we can write the belief network as

$$p(\mathcal{X}|\mathcal{D}) = p(\mathcal{X}_2|\mathcal{X}_1, \mathcal{D}_1, \mathcal{D}_2)p(\mathcal{X}_1|\mathcal{D}_1) \tag{7.4.8}$$

where $\mathcal{D}_1 \prec \mathcal{X}_1 \prec \mathcal{D}_2 \prec \mathcal{X}_2$, and corresponding utilities

$$u(\mathcal{X}, \mathcal{D}) = u(\mathcal{X}_1, \mathcal{D}) + u(\mathcal{X}_1, \mathcal{X}_2, \mathcal{D}) \tag{7.4.9}$$

The optimal utility is given by

$$u^{opt} = \max_{\mathcal{D}_1} \sum_{\mathcal{X}_1} \max_{\mathcal{D}_2} \sum_{\mathcal{X}_2} p(\mathcal{X}|\mathcal{D})u(\mathcal{X}, \mathcal{D}) \tag{7.4.10}$$

After eliminating $\mathcal{X}_2$ and $\mathcal{D}_2$, we obtain the ID

$$p(\mathcal{X}_1|\mathcal{D}_1)\left(u(\mathcal{X}_1, \mathcal{D}) + \max_{\mathcal{D}_2} \sum_{\mathcal{X}_2} p(\mathcal{X}_2|\mathcal{X}_1, \mathcal{D}_1, \mathcal{D}_2)u(\mathcal{X}_1, \mathcal{X}_2, \mathcal{D})\right) \tag{7.4.11}$$ @@

which we can express in terms of the original distribution $p(\mathcal{X}|\mathcal{D})$ as

$$\left(\sum_{(\mathcal{X},\mathcal{D})_2}^{*} p(\mathcal{X}|\mathcal{D})\right)\left(u(\mathcal{X}_1, \mathcal{D}) + \frac{1}{\sum^{*}_{(\mathcal{X},\mathcal{D})_2} p(\mathcal{X}|\mathcal{D})} \sum_{(\mathcal{X},\mathcal{D})_2}^{*} p(\mathcal{X}|\mathcal{D})u(\mathcal{X}_1, \mathcal{X}_2, \mathcal{D})\right) \tag{7.4.12}$$

where $\sum^{*}_{\mathcal{Y}}$ refers to summing first over the chance variables in $\mathcal{Y}$ and then maximising over the decision variables in $\mathcal{Y}$. These updates then define an ID on a reduced set of variables and can be viewed as messages. The potential usefulness of equation (7.4.12) is that it may be applied to IDs that are causally consistent (future decisions cannot affect the past) but which are not expressed directly in a causal form.

7.4.2 Using a junction tree

In complex IDs computational efficiency in carrying out the series of summations and maximisations may be an issue and one therefore seeks to exploit structure in the ID. It is intuitive that some form of junction tree style algorithm is applicable. The treatment here is inspired by [160]; a related approach which deals with more general chain graphs is given in [73]. We can first represent an ID using decision potentials which consist of two parts, as defined below.

Definition 7.2 (Decision Potential)**.** A *decision potential* on a clique C contains two potentials: a *probability potential* ρ_C and a *utility potential* μ_C. The joint potentials for the junction tree are defined as

$$\rho = \prod_{C \in \mathcal{C}} \rho_C, \qquad \mu = \sum_{C \in \mathcal{C}} \mu_C \tag{7.4.13}$$

with the junction tree representing the term $\rho\mu$.

In this case there are constraints on the triangulation, imposed by the (inverse of the) partial ordering which restricts the variables elimination sequence. This results in a so-called *strong Junction Tree*. The sequence of steps required to construct a JT for an ID is given by the following procedure: @@

Procedure 7.1 (Making a strong junction tree)**.**

Remove Information Edges Parental links of decision nodes are removed.

Moralization Marry all parents of the remaining nodes.

Remove Utility Nodes Remove the utility nodes and their parental links.

Strong Triangulation Form a triangulation based on an elimination order which obeys the partial ordering of the variables.

Strong Junction Tree From the strongly triangulated graph, form a junction tree and orient the edges towards the strong root (the clique that appears last in the elimination sequence).

The cliques are then ordered according to the sequence in which they are eliminated. The separator probability cliques are initialised to the identity, with the separator utilities initialised to zero. The probability cliques are then initialised by placing conditional probability factors into the lowest available clique (that is the probability factors are placed in the cliques closest to the leaves of the tree, furthest from the root) that can contain them, and similarly for the utilities. Remaining probability cliques are set to the identity and utility cliques to zero.

Example 7.5 (Junction Tree)**.** An example of a junction tree for an ID is given in fig(7.7a). The moralisation and triangulation links are given in fig(7.7b). The orientation of the edges follows the partial ordering with the leaf cliques being the first to disappear under the sequence of summations and maximisations.

A by-product of the above steps is that the cliques describe the fundamental dependencies on previous decisions and observations. In fig(7.7a), for example, the information link from f to D_2 is not present in the moralised-triangulated graph fig(7.7b), nor in the associated cliques of fig(7.7c). This is because once e is revealed, the utility U_4 is independent of f, giving rise to the two-branch structure in fig(7.7b). Nevertheless, the information link from f to D_2 is fundamental since it specifies that f will be revealed – removing this link would therefore change the partial ordering.

Absorption

By analogy with the definition of messages in section(7.4.1), for two neighbouring cliques C_1 and C_2, where C_1 is closer to the strong root of the JT (the last clique defined through the elimination order), we define

$$\rho_S = \sum_{C_2 \setminus S}^{*} \rho_{C_2}, \qquad \mu_S = \sum_{C_2 \setminus S}^{*} \rho_{C_2} \mu_{C_2} \tag{7.4.14}$$

$$\rho_{C_1}^{new} = \rho_{C_1} \rho_S, \qquad \mu_{C_1}^{new} = \mu_{C_1} + \frac{\mu_S}{\rho_S} \tag{7.4.15}$$

In the above $\sum_C^*$ is a 'generalised marginalisation' operation – it sums over those elements of clique C which are random variables and maximises over the decision variables in the clique. The order of this sequence of sums and maximisations follows the partial ordering defined by $\prec$.

Absorption is then carried out from the leaves inwards to the root of the strong JT. The optimal setting of a decision D_1 can then be computed from the root clique. Subsequently backtracking may be applied to infer the optimal decision trajectory. The optimal decision for D can be obtained by working with the clique containing D which is closest to the strong root and setting any previously taken decisions and revealed observations into their evidential states. See `demoDecAsia.m` for an example.

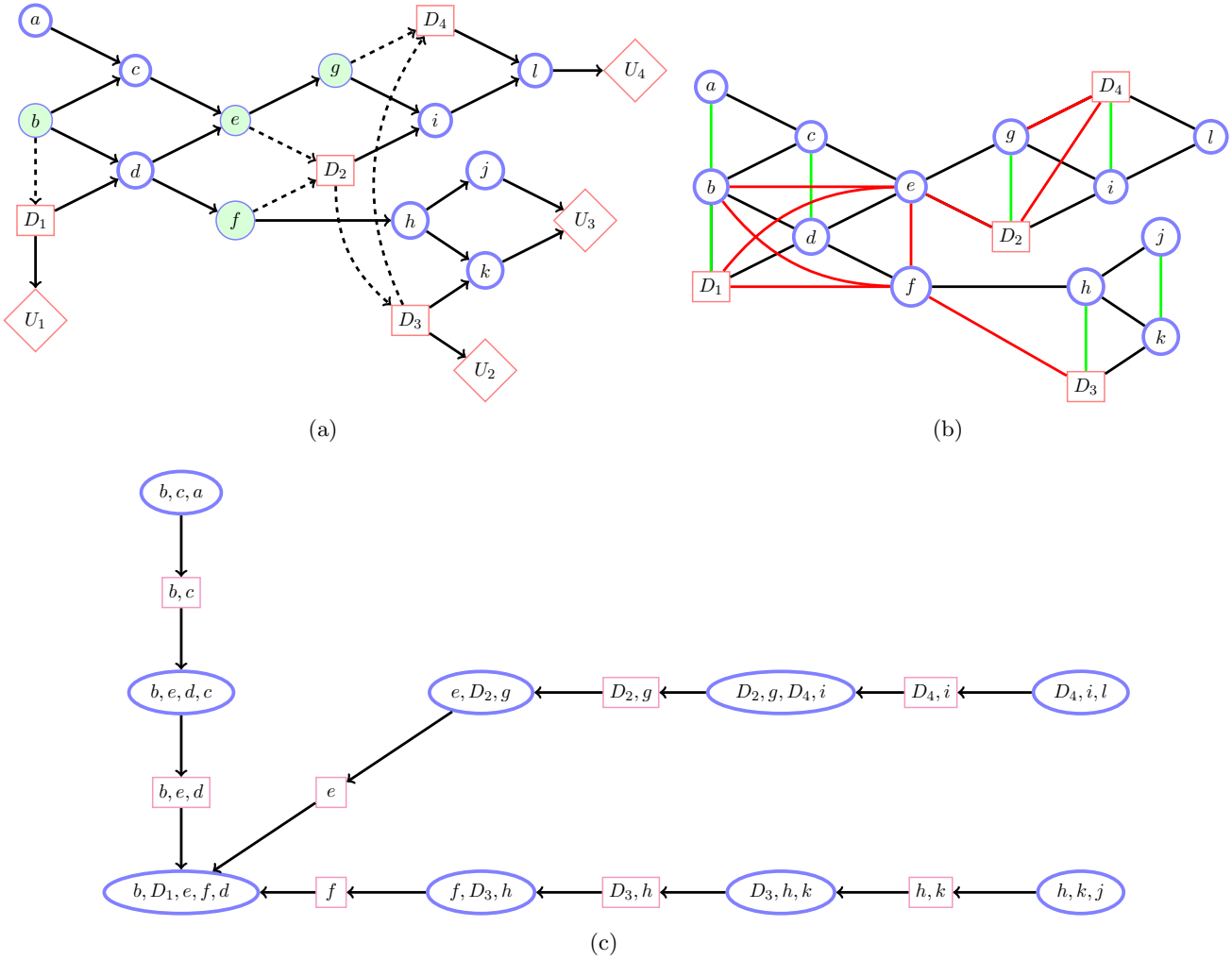
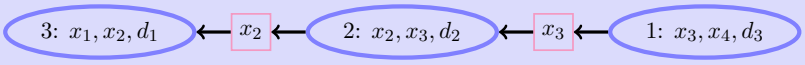


Figure 7.7: **(a)**: Influence Diagram, adapted from [160]. Causal consistency is satisfied since there is a directed path linking all decisions in sequence. The partial ordering is $b \prec D_1 \prec (e, f) \prec D_2 \prec (\cdot) \prec D_3 \prec g \prec D_4 \prec (a, c, d, h, i, j, k, l)$. **(b)**: Moralised and strongly triangulated graph. Moralisation links are in green, strong triangulation links are in red. **(c)**: Strong Junction Tree. Absorption passes information from the leaves of the tree towards the root.

Example 7.6 (Absorption on a chain)**.** For the ID of fig(7.8), the moralisation and triangulation steps are trivial and give the JT:



where the cliques are indexed according the elimination order. The probability and utility cliques are initialised to

$$\begin{aligned} \rho_3(x_1, x_2, d_1) &= p(x_2|x_1, d_1) \quad \mu_3(x_1, x_2, d_1) = 0 \\ \rho_2(x_2, x_3, d_2) &= p(x_3|x_2, d_2) \quad \mu_2(x_2, x_3, d_2) = u(x_2) \\ \rho_1(x_3, x_4, d_3) &= p(x_4|x_3, d_3) \quad \mu_1(x_3, x_4, d_3) = u(x_3) + u(x_4) \end{aligned} \tag{7.4.16}$$

with the separator cliques initialised to

$$\begin{aligned} \rho_{1-2}(x_3) &= 1 \quad \mu_{1-2}(x_3) = 0 \\ \rho_{2-3}(x_2) &= 1 \quad \mu_{2-3}(x_2) = 0 \end{aligned} \tag{7.4.17}$$

Updating the separator we have the new probability potential

$$\rho_{1-2}(x_3)^* = \max_{d_3} \sum_{x_4} \rho_1(x_3, x_4, d_3) = 1 \tag{7.4.18}$$

and utility potential

$$\mu_{1-2}(x_3)^* = \max_{d_3} \sum_{x_4} \rho_1(x_3, x_4, d_3)\, \mu_1(x_3, x_4, d_3) = \max_{d_3} \sum_{x_4} p(x_4|x_3, d_3)\,(u(x_3) + u(x_4)) \tag{7.4.19}$$

$$= \max_{d_3} \left(u(x_3) + \sum_{x_4} p(x_4|x_3, d_3) u(x_4) \right) \tag{7.4.20}$$

At the next step we update the probability potential

@@

$$\rho_2(x_2, x_3, d_2)^* = \rho_2(x_2, x_3, d_2)\, \rho_{1-2}(x_3)^* = p(x_3|x_2, d_2) \tag{7.4.21}$$

and utility potential

$$\mu_2(x_2, x_3, d_2)^* = \mu_2(x_2, x_3, d_2) + \frac{\mu_{1-2}(x_3)^*}{\rho_{1-2}(x_3)} = u(x_2) + \max_{d_3} \left(u(x_3) + \sum_{x_4} p(x_4|x_3, d_3) u(x_4) \right) \tag{7.4.22}$$

The next separator decision potential is

$$\rho_{2-3}(x_2)^* = \max_{d_2} \sum_{x_3} \rho_2(x_2, x_3, d_2)^* = 1 \tag{7.4.23}$$

$$\mu_{2-3}(x_2)^* = \max_{d_2} \sum_{x_3} \rho_2(x_2, x_3, d_2)\, \mu_2(x_2, x_3, d_2)^* \tag{7.4.24}$$

$$= \max_{d_2} \sum_{x_3} p(x_3|x_2, d_2) \left(u(x_2) + \max_{d_3} \left(u(x_3) + \sum_{x_4} p(x_4|x_3, d_3) u(x_4) \right) \right) \tag{7.4.25}$$

Finally we end up with the root decision potential

$$\rho_3(x_1, x_2, d_1)^* = \rho_3(x_1, x_2, d_1)\, \rho_{2-3}(x_2)^* = p(x_2|x_1, d_1) \tag{7.4.26}$$

and

$$\mu_3(x_1, x_2, d_1)^* = \mu_3(x_2, x_1, d_1) + \frac{\mu_{2-3}(x_2)^*}{\rho_{2-3}(x_2)^*} \tag{7.4.27}$$

$$= \max_{d_2} \sum_{x_3} p(x_3|x_2, d_2) \left(u(x_2) + \max_{d_3} \left(u(x_3) + \sum_{x_4} p(x_4|x_3, d_3) u(x_4) \right) \right) \tag{7.4.28}$$

From the final decision potential we have the expression

$$\rho_3(x_1, x_2, d_1)^*\, \mu_3(x_1, x_2, d_1)^* \tag{7.4.29}$$

which is equivalent to that which would be obtained by simply distributing the summations and maximisations over the original ID. At least for this special case, we therefore have verified that the JT approach yields the correct root clique potentials.

7.5 Markov Decision Processes

Consider a Markov chain with transition probabilities $p(x_{t+1} = i|x_t = j)$. At each time t we consider an action (decision), which affects the state at time $t + 1$. We describe this by

$$p(x_{t+1} = i|x_t = j, d_t = k) \tag{7.5.1}$$

@@ Associated with each state x_t is a utility $u(x_t)$.This model is depicted in fig(7.8). More generally one could consider utilities that depend on transitions and decisions, $u(x_{t+1} = i, x_t = j, d_t = k)$ and also time

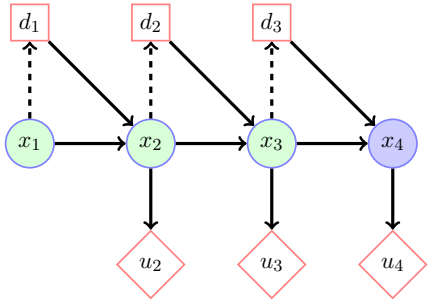


Figure 7.8: Markov Decision Process. These can be used to model planning problems of the form 'how do I get to where I want to be incurring the lowest total cost?'. They are readily solvable using a message passing algorithm.

dependent versions of all of these, $p_t(x_{t+1} = i|x_t = j, d_t = k)$, $u_t(x_{t+1} = i, x_t = j, d_t = k)$. We'll stick with the time-independent (stationary) case here since the generalisations are conceptually straightforward at the expense of notational complexity. Markov Decision Processes (MDPs) can be used to solve planning tasks such as how to get to a desired goal state as quickly as possible.

For positive utilities, the total utility of any state-decision path $x_{1:T}, d_{1:T}$ is defined as (assuming we know the initial state x_1)

$$U(x_{1:T}) \equiv \sum_{t=2}^{T} u(x_t) \tag{7.5.2}$$

and the probability with which this happens is given by

$$p(x_{2:T}|x_1, d_{1:T-1}) = \prod_{t=1}^{T-1} p(x_{t+1}|x_t, d_t) \tag{7.5.3}$$

At time $t = 1$ we want to make that decision d_1 that will lead to maximal expected total utility

$$U(d_1|x_1) \equiv \sum_{x_2} \max_{d_2} \sum_{x_3} \max_{d_3} \sum_{x_4} \ldots \max_{d_{T-1}} \sum_{x_T} p(x_{2:T}|x_1, d_{1:T-1}) U(x_{1:T}) \tag{7.5.4}$$

Our task is to compute $U(d_1|x_1)$ for each state of d_1 and then choose that state with maximal expected total utility. To carry out the summations and maximisations efficiently, we could use the junction tree approach, as described in the previous section. However, in this case, the ID is sufficiently simple that a direct message passing approach can be used to compute the expected utility.

7.5.1 Maximising expected utility by message passing

Consider the time-dependent decisions (non-stationary policy) MDP

$$\prod_{t=1}^{T-1} p(x_{t+1}|x_t, d_t) \sum_{t=2}^{T} u(x_t) \tag{7.5.5}$$

For the specific example in fig(7.8) the joint model of the BN and utility is

$$p(x_4|x_3, d_3)p(x_3|x_2, d_2)p(x_2|x_1, d_1)\left(u(x_2) + u(x_3) + u(x_4)\right) \tag{7.5.6}$$

To decide on how to take the first optimal decision, we need to compute

$$U(d_1|x_1) = \sum_{x_2} \max_{d_2} \sum_{x_3} \max_{d_3} \sum_{x_4} p(x_4|x_3, d_3)p(x_3|x_2, d_2)p(x_2|x_1, d_1)\left(u(x_2) + u(x_3) + u(x_4)\right) \tag{7.5.7}$$

Since only $u(x_4)$ depends on x_4 explicitly, we can write

$$U(d_1|x_1) = \sum_{x_2} \max_{d_2} \sum_{x_3} p(x_3|x_2, d_2)p(x_2|x_1, d_1)\left(u(x_2) + u(x_3) + \max_{d_3} \sum_{x_4} p(x_4|x_3, d_3)u(x_4)\right) \tag{7.5.8}$$

Defining a message and corresponding *value*

$$u_{3\leftarrow 4}(x_3) \equiv \max_{d_3} \sum_{x_4} p(x_4|x_3, d_3)u(x_4), \qquad v(x_3) \equiv u(x_3) + u_{3\leftarrow 4}(x_3) \tag{7.5.9}$$

we can write

$$U(d_1|x_1) = \sum_{x_2} \max_{d_2} \sum_{x_3} p(x_3|x_2, d_2)p(x_2|x_1, d_1) \left(u(x_2) + v(x_3)\right) \tag{7.5.10}$$

In a similar manner, only the last term depends on x_3 and hence

$$U(d_1|x_1) = \sum_{x_2} p(x_2|x_1, d_1) \left(u(x_2) + \max_{d_2} \sum_{x_3} p(x_3|x_2, d_2)v(x_3) \right) \tag{7.5.11}$$

Defining similarly the value

$$v(x_2) \equiv u(x_2) + \max_{d_2} \sum_{x_3} p(x_3|x_2, d_2)v(x_3) \tag{7.5.12}$$

then

$$U(d_1|x_1) = \sum_{x_2} p(x_2|x_1, d_1)v(x_2) \tag{7.5.13}$$

Given $U(d_1|x_1)$ above, we can then find the optimal decision d_1 by

$$d_1^*(x_1) = \operatorname*{argmax}_{d_1} U(d_1|x_1) \tag{7.5.14}$$

7.5.2 Bellman's equation

In a Markov Decision Process, as above, we can define utility messages recursively as

$$u_{t-1\leftarrow t}(x_{t-1}) \equiv \max_{d_{t-1}} \sum_{x_t} p(x_t|x_{t-1}, d_{t-1}) \left[u(x_t) + u_{t\leftarrow t+1}(x_t)\right] \tag{7.5.15}$$

It is more common to define the value of being in state x_t as

$$v_t(x_t) \equiv u(x_t) + u_{t\leftarrow t+1}(x_t), \qquad v_T(x_T) = u(x_T) \tag{7.5.16}$$

and write then the equivalent recursion

$$v_{t-1}(x_{t-1}) = u(x_{t-1}) + \max_{d_{t-1}} \sum_{x_t} p(x_t|x_{t-1}, d_{t-1})v_t(x_t) \tag{7.5.17}$$

The optimal decision d_t^* is then given by

$$d_t^*(x_t) = \operatorname*{argmax}_{d_t} \sum_{x_{t+1}} p(x_{t+1}|x_t, d_t)v_{t+1}(x_{t+1}) \tag{7.5.18}$$

Equation(7.5.17) is called Bellman's equation[31][3].

[3]The continuous-time analog has a long history in physics and is called the Hamilton-Jacobi equation.

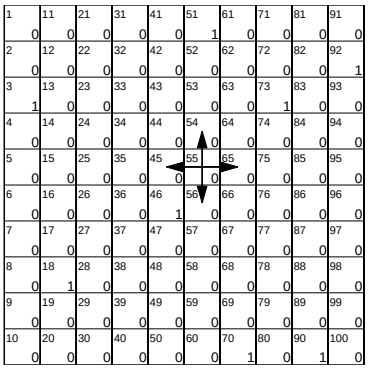


Figure 7.9: States defined on a two dimensional grid. In each square the top left value is the state number, and the bottom right is the utility of being in that state. An 'agent' can move from a state to a neighbouring state, as indicated. The task is to solve this problem such that for any position (state) one knows how to move optimally to maximise the expected utility. This means that we need to move towards the goal states (states with non-zero utility). See `demoMDP`.

7.6 Temporally Unbounded MDPs

In the previous discussion about MDPs we assumed a given end time, T, from which one can propagate messages back from the end of the chain. The infinite T case would appear to be ill-defined since the sum of utilities

$$u(x_1) + u(x_2) + \ldots + u(x_T) \tag{7.6.1}$$

will in general be unbounded. There is a simple way to avoid this difficulty. If we let $u^* = \max_s u(s)$ be the largest value of the utility and consider the sum of modified utilities for a chosen *discount factor* $0 < \gamma < 1$

$$\sum_{t=1}^{T} \gamma^t u(x_t) \leq u^* \sum_{t=1}^{T} \gamma^t = \gamma u^* \frac{1-\gamma^T}{1-\gamma} \tag{7.6.2}$$

where we used the result for a geometric series. In the limit $T \to \infty$ this means that the summed modified utility $\gamma^t u(x_t)$ is finite. The only modification required to our previous discussion is to include a factor γ in the message definition. Assuming that we are at convergence, we define a value $v(x_t = \mathsf{s})$ dependent only on the state s, and not the time. This means we replace the time-dependent Bellman's value recursion equation (7.5.17) with the time-independent equation

$$v(\mathsf{s}) \equiv u(\mathsf{s}) + \gamma \max_{\mathsf{d}} \sum_{\mathsf{s}'} p(x_t = \mathsf{s}' | x_{t-1} = \mathsf{s}, d_{t-1} = \mathsf{d}) v(\mathsf{s}') \tag{7.6.3}$$

We then need to solve equation (7.6.3) for the value $v(\mathsf{s})$ for all states s. The optimal decision *policy* when one is in state $x_t = \mathsf{s}$ is then given by

$$\mathsf{d}^*(\mathsf{s}) = \operatorname*{argmax}_{\mathsf{d}} \sum_{\mathsf{s}'} p(x_{t+1} = \mathsf{s}' | x_t = \mathsf{s}, d_t = \mathsf{d}) v(\mathsf{s}') \tag{7.6.4}$$

For a deterministic transition p (*i.e.* for each decision d, only one state s' is available), this means that the best decision is the one that takes us to the accessible state with highest value.

Equation(7.6.3) seems straightforward to solve. However, the max operation means that the equations are non-linear in the value v and no closed form solution is available. Two popular techniques for solving equation (7.6.3), are Value and Policy iteration, which we describe below. When the number of states S is very large, approximate solutions are required. Sampling and state-dimension reduction techniques are described in [62].

7.6.1 Value iteration

A naive procedure is to iterate equation (7.6.3) until convergence, assuming some initial guess for the values (say uniform). One can show that this value iteration procedure is guaranteed to converge to a unique optimum[36]. The convergence rate depends on γ — the smaller γ is, the faster is the convergence. An example of value iteration is given in fig(7.10).

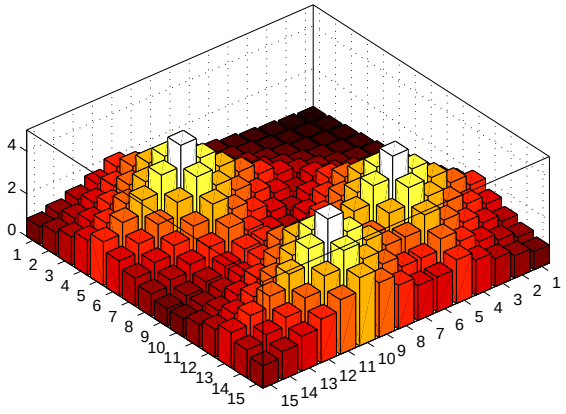


Figure 7.10: Value Iteration on a set of 225 states, corresponding to a 15×15 two dimensional grid. Deterministic transitions are allowed to neighbours on the grid, $\{\mathsf{stay}, \mathsf{left}, \mathsf{right}, \mathsf{up}, \mathsf{down}\}$. There are three goal states, each with utility 1 – all other states have utility 0. Plotted is the value $v(\mathsf{s})$ for $\gamma = 0.9$ after 30 updates of Value Iteration, where the states index a point on the $x - y$ grid. The optimal decision for any state on the grid is to go to the neighbouring state with highest value. See `demoMDP`.

7.6.2 Policy iteration

In policy iteration we first assume we know the optimal decision $\mathsf{d}^*(\mathsf{s})$ for any state s. We may use this in equation (7.6.3) to give

$$v(\mathsf{s}) = u(\mathsf{s}) + \gamma \sum_{\mathsf{s}'} p(x_t = \mathsf{s}' | x_{t-1} = \mathsf{s}, \mathsf{d}^*(\mathsf{s}))v(\mathsf{s}') \tag{7.6.5}$$

The maximisation over d has disappeared since we have assumed we already know the optimal decision for each state s. For fixed $\mathsf{d}^*(\mathsf{s})$, equation (7.6.5) is now linear in the value. Defining the value $\mathbf{v}$ and utility $\mathbf{u}$ vectors and transition matrix $\mathbf{P}$,

$$[\mathbf{v}]_{\mathsf{s}} = v(\mathsf{s}), \qquad [\mathbf{u}]_{\mathsf{s}} = u(\mathsf{s}), \qquad [\mathbf{P}]_{\mathsf{s}',\mathsf{s}} = p(\mathsf{s}'|\mathsf{s}, \mathsf{d}^*(\mathsf{s})) \tag{7.6.6}$$

in matrix notation, equation (7.6.5) becomes

$$\mathbf{v} = \mathbf{u} + \gamma \mathbf{P}^{\mathsf{T}} \mathbf{v} \Leftrightarrow \left(\mathbf{I} - \gamma \mathbf{P}^{\mathsf{T}}\right) \mathbf{v} = \mathbf{u} \Leftrightarrow \mathbf{v} = \left(\mathbf{I} - \gamma \mathbf{P}^{\mathsf{T}}\right)^{-1} \mathbf{u} \tag{7.6.7}$$

These linear equations are readily solved with Gaussian Elimination. Using this, the optimal policy is recomputed using equation (7.6.4). The two steps of solving for the value, and recomputing the policy are iterated until convergence. The procedure may be initialised by guessing an initial $\mathsf{d}^*(\mathsf{s})$, then solve the linear equations (7.6.5) for the value, or alternatively guessing the initial values and solving for the initial policy.

@@ **Example 7.7** (A grid-world MDP)**.** We define a set of states on an $N \times N$ grid with corresponding utilities @@ for each state, as given for example in fig(7.9) for $N = 10$. The agent is able to deterministically move to a neighbouring grid state at each time step. After initialising the value of each grid state to unity, the @@ converged value for each state is obtained. An example on a 15×15 grid is given in fig(7.10) in which the utilities are zero everywhere except for 3 states. The optimal policy is then given by moving to the neighbouring grid state with highest value.

7.6.3 A curse of dimensionality

Consider the following Tower of Hanoi problem. There are 4 pegs a, b, c, d and 10 disks numbered from 1 to 10. You may move a single disk from one peg to another – however, you are not allowed to put a bigger numbered disk on top of a smaller numbered disk. Starting with all disks on peg a, how can you move them all to peg d in the minimal number of moves?

This would appear to be a straightforward Markov decision process in which the transitions are allowed disk moves. If we use x to represent the state of the disks on the 4 pegs, naively this has $4^{10} = 1048576$ states

(some are equivalent up to permutation of the pegs, which reduces this by a factor of 2). This large number of states renders this naive approach computationally problematic.

Many interesting real-world problems suffer from this large number of states issue so that a naive approach to find the best decision is computationally infeasible. Finding efficient exact and also approximate state representations is a key aspect to solving large scale MDPs, see for example [209].

7.7 Variational Inference and Planning

For the finite-horizon stationary policy MDP, learning the optimal policy can be addressed by a variety of methods. Two popular approaches are policy gradients and EM style procedures – see for example [110] and section(11.2).
For many MDPs of interest the optimal policy is deterministic[284], so that methods which explicitly seek for deterministic policies are of interest. For this reason and to remain close to our discussion on policy and value iteration, which involved deterministic policies, we focus on this case in our brief discussion here, referring the reader to other texts [82, 299, 108, 109] for the details on the non-deterministic case. For a time-independent deterministic policy $d(\mathsf{s})$ that maps a state s to a decision d (which we write as π for short), we have the expected utility

$$U(\pi) = \sum_{t=1}^{T} \sum_{x_t} u_t(x_t) \sum_{x_{1:t-1}} \prod_{\tau=1}^{t} p(x_\tau | x_{\tau-1}, d(x_{\tau-1})) \tag{7.7.1}$$

with the convention $p(x_1|x_0, d(x_0)) = p(x_1)$. Viewed as a factor graph, this is simply a set of chains, so that for any policy π, the expected utility can be computed easily. In principle one could then attempt to optimise U with respect to the policy directly. An alternative is to use an EM style procedure[108]. To do this we define a (trans-dimensional) distribution

$$\hat{p}(x_{1:t}, t) = \frac{u_t(x_t)}{Z(\pi)} \prod_{\tau=1}^{t} p(x_\tau | x_{\tau-1}, d(x_{\tau-1})) \tag{7.7.2}$$

The normalisation constant $Z(\pi)$ of this distribution is

$$\sum_{t=1}^{T} \sum_{x_{1:t}} u_t(x_t) \prod_{\tau=1}^{t} p(x_\tau | x_{\tau-1}, d(x_{\tau-1})) = \sum_{t=1}^{T} \sum_{x_{1:t}} u_t(x_t) \prod_{\tau=1}^{t} p(x_\tau | x_{\tau-1}, d(x_{\tau-1})) = U(\pi) \tag{7.7.3}$$

If we now define a variational distribution $q(x_{1:t}, t)$, and consider

$$\mathrm{KL}(q(x_{1:t}, t) | \hat{p}(x_{1:t}, t)) \geq 0 \tag{7.7.4}$$

this gives the lower bound

$$\log U(\pi) \geq -H(q(x_{1:t}, t)) + \left\langle \log u_t(x_t) \prod_{\tau=1}^{t} p(x_\tau | x_{\tau-1}, d(x_{\tau-1})) \right\rangle_{q(x_{1:t}, t)} \tag{7.7.5}$$

where $H(q(x_{1:t}, t))$ is the entropy of the distribution $q(x_{1:t}, t)$. In terms of an EM algorithm, the M-step requires the dependency on π alone, which is

$$E(\pi) = \sum_{t=1}^{T} \sum_{\tau=1}^{t} \langle \log p(x_\tau | x_{\tau-1}, d(x_{\tau-1})) \rangle_{q(x_\tau, x_{\tau-1}, t)} \tag{7.7.6}$$

$$= \sum_{t=1}^{T} \sum_{\tau=1}^{t} q(x_\tau = \mathsf{s}', x_{\tau-1} = \mathsf{s}, t) \log p(x_\tau = \mathsf{s}' | x_{\tau-1} = \mathsf{s}, d(x_{\tau-1}) = \mathsf{d}) \tag{7.7.7}$$

For each given state s we now attempt to find the optimal decision d, which corresponds to maximising

$$\hat{E}(\mathsf{d}|\mathsf{s}) = \sum_{\mathsf{s}'} \left\{ \sum_{t=1}^{T} \sum_{\tau=1}^{t} q(x_\tau = \mathsf{s}', x_{\tau-1} = \mathsf{s}, t) \right\} \log p(\mathsf{s}' | \mathsf{s}, \mathsf{d}) \tag{7.7.8}$$

Defining

$$q(\mathsf{s}'|\mathsf{s}) \propto \sum_{t=1}^{T} \sum_{\tau=1}^{t} q(x_\tau = \mathsf{s}', x_{\tau-1} = \mathsf{s}, t) \tag{7.7.9}$$

we see that for given s, up to a constant, $\hat{E}(\mathsf{d}|\mathsf{s})$ is the Kullback-Leibler divergence between $q(\mathsf{s}'|\mathsf{s})$ and $p(\mathsf{s}'|\mathsf{s},\mathsf{d})$ so that the optimal decision d is given by the index of the distribution $p(s'|\mathsf{s},\mathsf{d})$ most closely aligned with $q(s'|\mathsf{s})$:

$$\mathsf{d}^*(\mathsf{s}) = \operatorname*{argmin}_{\mathsf{d}} \mathrm{KL}\big(q(s'|\mathsf{s})|p(s'|\mathsf{s},\mathsf{d})\big) \tag{7.7.10}$$

The E-step concerns the computation of the marginal distributions required in the M-step. The optimal q distribution is proportional to $\hat{p}$ evaluated at the previous decision function d:

$$q(x_{1:t}, t) \propto u_t(x_t) \prod_{\tau=1}^{t} p(x_\tau|x_{\tau-1}, d(x_{\tau-1})) \tag{7.7.11}$$

For a constant discount factor γ at each time-step and an otherwise time-independent utility[4]

$$u_t(x_t) = \gamma^t u(x_t) \tag{7.7.12}$$

using this

$$q(x_{1:t}, t) \propto \gamma^t u(x_t) \prod_{\tau=1}^{t} p(x_\tau|x_{\tau-1}, d(x_{\tau-1})) \tag{7.7.13}$$

For each t this is a simple Markov chain for which the pairwise transition marginals required for the M-step, equation (7.7.9) are straightforward. This requires inference in a series of Markov models of different lengths. This can be done efficiently using a single forward and backward pass [299, 110].

EM and related methods follow closely the spirit of inference in graphical models, but can exhibit disappointingly slow convergence. More recently, an alternative method using Lagrange Duality shows very promising performance and the reader is referred to [111] for details. Note that this EM algorithm formally fails in the case of a deterministic environment (the transition $p(x_t|x_{t-1}, d_{t-1})$ is deterministic) – see exercise(7.8) for an explanation and exercise(7.9) for a possible resolution.

Remark 7.2 (Solving an MDP – easy or hard?)**.** The discussion in section(7.5.1) highlights that solving a linear-chain influence diagram (finding the optimal decision at each timestep) is straightforward, and can be achieved using a simple message passing algorithm, scaling linearly with the length of the chain. In contrast, finding the optimal time-independent policy π typically is much more complex – hence the reason for many different algorithms that attempt to find optimal policies in areas such as time-independent control, reinforcement learning and games. Mathematically, the reason for this difference is that the constraint in the time-independent case that the policy must be the same over all time steps, leads to a graphical structure that is no longer a chain, with all timepoints connected to a single π. In this case, in general, no simple linear time message passing algorithm is available to optimise the resulting objective.

7.8 Financial Matters

Utility and decision theory play a major role in finance, both in terms of setting prices based on expected future gains, but also in determining optimal investments. In the following sections we briefly outline two such basic applications.

[4] In the standard MDP framework it is more common to define $u_t(x_t) = \gamma^{t-1} u(x_t)$ so that for comparison with the standard Policy/Value routines one needs to divide the expected utility by γ.

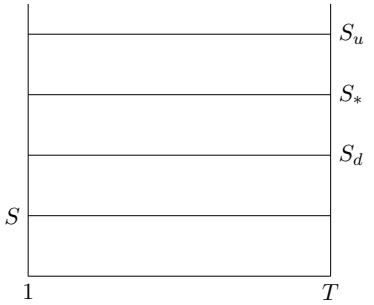


Figure 7.11: Options pricing: An asset has market value $£S$ at time $t = 1$. We assume that the asset will have market value either $£S_u$ or $£S_d$ at time T. The owner and 'options buyer' (client) agree that if the market value is greater than the 'strike price' $£S_*$ at time T, the client has the right to purchase the asset for $£S_*$. The question is: how much should the owner of the asset charge the client for the privilege of having the right to purchase the asset?

7.8.1 Options pricing and expected utility

An owner has an asset currently priced by the market at $£S$. The owner wants to give us the opportunity to purchase this asset at time T for an agreed price of $£S_*$. At time T, if the market price goes 'up' beyond the strike price to $£S_u$ we will decide to purchase the asset from the owner for $£S_*$ and sell the asset at this increased value, see fig(7.11). If, however, the price remains 'down' below the strike price at $£S_d$, we will walk away, leaving the owner with the asset. The question is, how much should the owner charge, $£C$ for this option of us being able to buy the asset for the agreed price at time T? To help answer this, we also need to know how much a risk-free investment would make over the same time period (*i.e.* how much we would get from putting money in a safe bank). We assume the interest rate R is known for the time period T, say 0.06. We call the two people the owner and the client (who may or may not buy the asset).

Two possibilities case

For simplicity, we assume the asset can take only the prices S_u or S_d at time T. We also assume we know the probabilities of these events (see below), namely ρ and $1-\rho$. Let's work out the expected utilities for both parties:

Asset goes up:

$$U(\mathsf{up},\mathsf{client}) = \underbrace{S_u - S_*}_{\text{immediate profit from selling}} - \underbrace{C}_{\text{option cost}} - \underbrace{CR}_{\text{lost interest on option cost}} \tag{7.8.1}$$

$$U(\mathsf{up},\mathsf{owner}) = \underbrace{S_* - S}_{\text{immediate loss from sale}} + \underbrace{C}_{\text{option cost}} + \underbrace{CR}_{\text{gained interest on option cost}} - \underbrace{SR}_{\text{lost interest}} \tag{7.8.2}$$

The final SR term above comes from entering into the option deal – otherwise the owner could have just sold the asset at time 1, and then put the resulting sum in the bank.

Asset goes down:

$$U(\mathsf{down},\mathsf{client}) = -\underbrace{C}_{\text{option cost}} - \underbrace{CR}_{\text{lost interest on option cost}} \tag{7.8.3}$$

The above follows since in this case the client doesn't act on his option to sell the asset.

$$U(\mathsf{down},\mathsf{owner}) = \underbrace{S_d - S}_{\text{change in asset value}} + \underbrace{C}_{\text{option cost}} + \underbrace{CR}_{\text{gained interest on option cost}} - \underbrace{SR}_{\text{lost interest}} \tag{7.8.4}$$

The expected utility for the client is

$$\begin{aligned} U(\mathsf{client}) &= \rho \times U(\mathsf{up},\mathsf{client}) + (1-\rho) \times U(\mathsf{down},\mathsf{client}) && (7.8.5)\\ &= \rho\,(S_u - S_* - C - CR) + (1-\rho)\,(-C - CR) && (7.8.6)\\ &= \rho\,(S_u - S_*) - C(1+R) && (7.8.7) \end{aligned}$$

The expected utility for the owner is

$$\begin{aligned} U(\mathsf{owner}) &= \rho \times U(\mathsf{up},\mathsf{owner}) + (1-\rho) \times U(\mathsf{down},\mathsf{owner}) && (7.8.8)\\ &= \rho\,(S_* - S + C + CR - SR) + (1-\rho)\,(S_d - S + C + CR - SR) && (7.8.9)\\ &= \rho\,(S_* - S_d) + S_d - S + C(1+R) - SR && (7.8.10) \end{aligned}$$

It seems reasonable to assume that both the client and the owner should have the same expected benefit, $U(\mathsf{client}) = U(\mathsf{owner})$. Hence

$$\rho\,(S_u - S_*) - C(1+R) = \rho\,(S_* - S_d) + S_d - S + C(1+R) - SR \tag{7.8.11}$$

Solving for C, we find

$$C = \frac{\rho\,(S_u - 2S_* + S_d) - S_d + S(1+R)}{2(1+R)} \tag{7.8.12}$$

All the quantities required to price the option are assumed known, except for ρ. One way to set this is described below.

Setting ρ

It seems reasonable to expect ρ to be set by some process which describes the true probability of the price increase. However, given a value for ρ and knowing the two possible prices S_u and S_d, the owner can compute the expected utility of just holding on to the asset. This is

$$\rho S_u + (1-\rho)S_d - S \tag{7.8.13}$$

Alternatively, the owner could sell the asset for $£S$ and put his money in the bank, collecting the interest RS at time T. In a fair market we must have that the expected reward for holding on to the asset is the same as the risk-free return on the asset:

$$\rho S_u + (1-\rho)S_d - S = RS \quad \Rightarrow \quad \rho = \frac{S(1+R) - S_d}{S_u - S_d} \tag{7.8.14}$$

Using this value to price the option in equation (7.8.12) ensures that the expected gain from offering the option and not offering the option is the same to the owner and, furthermore that if the option is available, the expected reward for both parties is the same.

7.8.2 Binomial options pricing model

If we have more than two timepoints, we can readily extend the above. It's easiest to assume that at each time t we have two price change possibilities – either the price can go up by a factor $u > 1$ or down by a factor $d < 1$. For T timesteps, we then have a set of possible values the asset can take at time T. For some of them the client will sell the asset (when $S_T > S_*$) otherwise not. We need to work out the expected gains for both the owner and client as before, assuming first that we know ρ (which is the probability of the price increasing by a factor u in one time step). For a sequence of n ups and $T-n$ downs, we have a price $S_T = Su^n d^{T-n}$. If this is greater than S_*, then the client will sell the asset and have utility

$$\mathbb{I}\left[Su^n d^{T-n} > S_*\right]\left(Su^n d^{T-n} - S_* - C(1+R)\right) \tag{7.8.15}$$

The probability of n ups and $T-n$ downs is

$$\beta(T,n,\rho) \equiv \binom{T}{n}\rho^n(1-\rho)^{T-n} \tag{7.8.16}$$

where $\binom{T}{n}$ is the binomial coefficient. Hence the total expected utility for the client on the upside is

$$U(\mathsf{client},\mathsf{up}) = \sum_{n=0}^{T}\beta(T,n,\rho)\mathbb{I}\left[Su^n d^{T-n} > S_*\right]\left(Su^n d^{T-n} - S_* - C(1+R)\right) \tag{7.8.17}$$

Similarly,

$$U(\mathsf{client},\mathsf{down}) = -C(1+R)\sum_{n=0}^{T}\beta(T,n,\rho)\mathbb{I}\left[Su^n d^{T-n} < S_*\right] \tag{7.8.18}$$

The total expected utility for the client is then

$$U(\text{client}) = U(\text{client}, \text{up}) + U(\text{client}, \text{down}) \tag{7.8.19}$$

Similarly, for the owner

$$U(\text{owner}, \text{up}) = (S_* - S(1+R) + C(1+R)) \sum_{n=0}^{T} \beta(T, n, \rho) \mathbb{I}\left[Su^n d^{T-n} > S_*\right] \tag{7.8.20}$$

and

$$U(\text{owner}, \text{down}) = \sum_{n=0}^{T} \beta(T, n, \rho) \mathbb{I}\left[Su^n d^{T-n} < S_*\right] \left(Su^n d^{T-n} - S(1+R) + C(1+R)\right) \tag{7.8.21}$$ @@

and

$$U(\text{owner}) = U(\text{owner}, \text{up}) + U(\text{owner}, \text{down}) \tag{7.8.22}$$

Setting

$$U(\text{client}) = U(\text{owner}) \tag{7.8.23}$$

results in a simple linear equation for C.

Setting ρ

To set ρ, we can use a similar logic as in the two timestep case. First we compute the expected value of the asset at time T, which is

$$\sum_{n=0}^{T} \beta(T, n, \rho) Su^n d^{T-n} \tag{7.8.24}$$

and equate the expected gain equal to the gain from a risk free investment in the asset:

$$\sum_{n=0}^{T} \beta(T, n, \rho) Su^n d^{T-n} - S = RS \quad \Rightarrow \quad \sum_{n=0}^{T} \beta(T, n, \rho) u^n d^{T-n} = R + 1 \tag{7.8.25}$$

Knowing u and d, we can solve the above for ρ, and then use this in the equation for C. We can learn u and d from past observed data (in the literature they are often related to the observed variance in the prices[75]).

The binomial options pricing approach is a relatively simplistic way to price options. The celebrated Black-Scholes method [46] is essentially a limiting case in which the number of timepoints becomes infinite[151].

7.8.3 Optimal investment

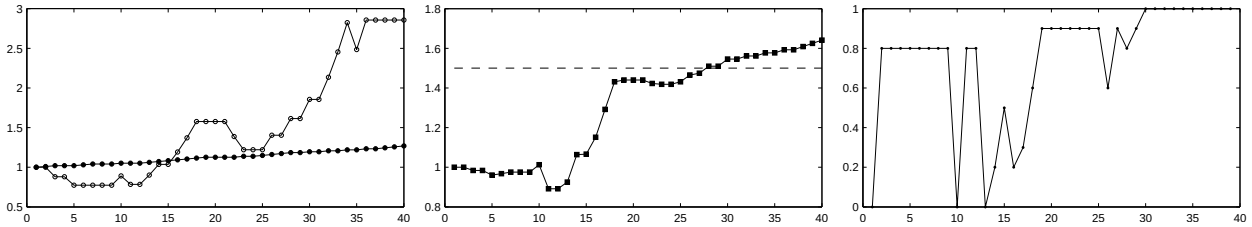
Another example of utility in finance, and which is related to Markov decision processes, is the issue of how best to invest your wealth in order to maximise some future criterion. We'll consider here a very simple setup, but one which can be readily extended to more complex scenarios. We assume that we have two assets, a and b with prices at time t given by s_t^a, s_t^b. The prices are assumed to independently follow Markovian updating:

$$s_t^a = s_{t-1}^a(1 + \epsilon_t^a) \quad \Rightarrow \quad p(s_t^a | s_{t-1}^a, \epsilon_t^a) = \delta\left(s_t^a - (1 + \epsilon_t^a) s_{t-1}^a\right) \tag{7.8.26}$$

where $\delta(\cdot)$ is the Dirac delta function. The price increments follow a Markov transition

$$p(\epsilon_t^a, \epsilon_t^b | \epsilon_{t-1}^a, \epsilon_{t-1}^b) = p(\epsilon_t^a | \epsilon_{t-1}^a) p(\epsilon_t^b | \epsilon_{t-1}^b) \tag{7.8.27}$$

Using this one can model effects such as price increments being likely to stay the same (as in bank interest) or more variable (as in the stock market).



(a) (b) (c)

Figure 7.12: **(a)**: Two assets through time – a risky asset whose value fluctuates wildly, and a stable asset that grows slowly. **(b)**: The wealth of our portfolio over time, based on investing with the hope to achieve a wealth of 1.5 at $T = 40$. **(c)**: The optimal investment decisions through time, with 1 corresponding to placing all wealth in the safe asset and 0 placing all the money on the risky asset.

We have an investment decision $0 \leq d_t \leq 1$ that says what fraction of our current wealth w_t we buy of asset a at time t. We will invest the rest of our wealth in asset b. If asset a is priced at s_t^a and we decide to use a fraction d_t of our current wealth w_t to purchase a quantity q_t^a of asset a and quantity q_t^b of asset b, these are given by

$$q_t^a = \frac{d_t w_t}{s_t^a}, \qquad q_t^b = \frac{w_t(1-d_t)}{s_t^b} \tag{7.8.28}$$

At time step $t+1$, the prices of assets a and b will have changed to s_{t+1}^a, s_{t+1}^b, so that our new wealth will be

$$w_{t+1} = q_t^a s_{t+1}^a + q_t^b s_{t+1}^b = \frac{d_t w_t s_{t+1}^a}{s_t^a} + \frac{w_t(1-d_t)s_{t+1}^b}{s_t^b} = w_t\left(d_t(1+\epsilon_{t+1}^a) + (1-d_t)(1+\epsilon_{t+1}^b)\right) \tag{7.8.29}$$

This can be expressed as a transition

$$p(w_{t+1}|w_t, \epsilon_{t+1}^a, \epsilon_{t+1}^b, d_t) = \delta\left(w_{t+1} - w_t\left(d_t(1+\epsilon_{t+1}^a) + (1-d_t)(1+\epsilon_{t+1}^b)\right)\right) \tag{7.8.30}$$

At an end time T we have a utility $u(w_T)$ that expresses our satisfaction with our wealth. Given that we start with a wealth w_1 and assume we know $\epsilon_1^a, \epsilon_1^b$, we want to find the best decision d_1 that will maximise our expected utility at time T. To do so, we also bear in mind that at any intermediate time $1 < t < T$ we can adjust the fraction d_t of wealth in asset a. The Markov chain is given by (see also section(23.1))

$$p(\epsilon_{1:T}^a, \epsilon_{1:T}^b, w_{2:T}|\epsilon_1^a, \epsilon_1^b, w_1, d_{1:T-1}) = \prod_{t=2}^{T} p(\epsilon_t^a|\epsilon_{t-1}^a)p(\epsilon_t^b|\epsilon_{t-1}^b)p(w_t|w_{t-1}, \epsilon_t^a, \epsilon_t^b, d_{t-1}) \tag{7.8.31}$$

The expected utility of a decision d_1 is

$$U(d_1|\epsilon_1^a, \epsilon_1^b, w_1) = \sum_{\epsilon_2^a, \epsilon_2^b, w_2} \ldots \max_{d_{T-2}} \sum_{\epsilon_{T-1}^a, \epsilon_{T-1}^b, w_{T-1}} \max_{d_{T-1}} \sum_{\epsilon_T^a, \epsilon_T^b, w_T} p(\epsilon_{1:T}^a, \epsilon_{1:T}^b, w_{2:T}|\epsilon_1^a, \epsilon_1^b, w_1, d_{1:T-1})u(w_T) \tag{7.8.32}$$

which, for the corresponding influence diagram, corresponds to the ordering

$$d_1 \prec \left\{\epsilon_2^a, \epsilon_2^b, w_2\right\} \prec d_2 \prec \ldots \prec \left\{\epsilon_{T-1}^a, \epsilon_{T-1}^b, w_{T-1}\right\} \prec d_{T-1} \prec \left\{\epsilon_T^a, \epsilon_T^b, w_T\right\} \tag{7.8.33}$$

To compute $U(d_1|\epsilon_1^a, \epsilon_1^b, w_1)$, we first can carry out the operations at T to give a message

$$\gamma_{T-1\leftarrow T}(\epsilon_{T-1}^a, \epsilon_{T-1}^b, w_{T-1}) \equiv \max_{d_{T-1}} \sum_{\epsilon_T^a, \epsilon_T^b, w_T} p(\epsilon_T^a|\epsilon_{T-1}^a)p(\epsilon_T^b|\epsilon_{T-1}^b)p(w_T|w_{T-1}, \epsilon_T^a, \epsilon_T^b, d_{T-1})u(w_T) \tag{7.8.34}$$

And, generally,

$$\gamma_{t-1\leftarrow t}(\epsilon_{t-1}^a, \epsilon_{t-1}^b, w_{t-1}) \equiv \max_{d_{t-1}} \sum_{\epsilon_t^a, \epsilon_t^b, w_t} p(\epsilon_t^a|\epsilon_{t-1}^a)p(\epsilon_t^b|\epsilon_{t-1}^b)p(w_t|w_{t-1}, \epsilon_t^a, \epsilon_t^b, d_{t-1})\gamma_{t\leftarrow t+1}(\epsilon_t^a, \epsilon_t^b, w_t) \tag{7.8.35}$$

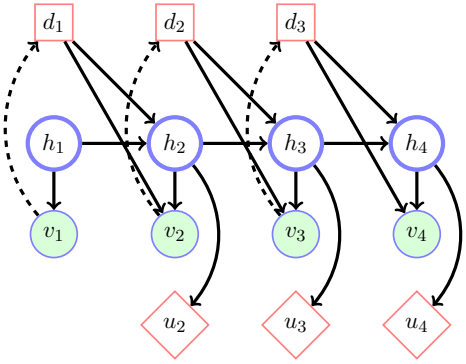


Figure 7.13: An example Partially Observable Markov Decision Process (POMDP). The 'hidden' variables h are never observed. In solving the Influence Diagram we are required to first sum over variables that are never observed; doing so will couple together all past observed variables and decisions; any decision at time t will then depend on all previous decisions. Note that the no-forgetting principle means that we do not need to explicitly write that each decision depends on all previous observations – this is implicitly assumed.

so that

$$U(d_1|\epsilon_1^a, \epsilon_1^b, w_1) = \sum_{\epsilon_2^a, \epsilon_2^b, w_2} p(\epsilon_2^a|\epsilon_1^a) p(\epsilon_2^b|\epsilon_1^b) p(w_2|w_1, \epsilon_2^a, \epsilon_2^b, d_1) \gamma_{2\leftarrow 3}(\epsilon_2^a, \epsilon_2^b, w_2) \tag{7.8.36}$$

Note that this process is not equivalent to a 'myopic' strategy which would make an investment decision to maximise the expected next-period wealth.

For a continuous wealth w_t the above messages are difficult to represent. A simple strategy, therefore, is to discretise all wealth values and also the price changes ϵ_t^a, ϵ_t^b and investment decisions d_t, see exercise(7.13). In this case, one needs to approximate the delta function $\delta(x)$ by the distribution which is zero for every state except that discrete state which is closest to the real value x.

Example 7.8 (Optimal investment)**.** We demonstrate a simple optimal portfolio investment problem in fig(7.12), in which there is a safe bank asset and a risky 'stock market' asset. We start with a unit wealth and wish to obtain a wealth of 1.5 at time $t = 40$. If we place all our money in the bank, we will not be able to reach this desired amount, so we must place some of our wealth at least in the risky asset. In the beginning the stock market does poorly, and our wealth is correspondingly poor. The stock market picks up sufficiently so that after around $t = 20$, we no longer need to take many risks and may place most of our money in the bank, confident that we will reach our investment objective.

7.9 Further Topics

7.9.1 Partially observable MDPs

In a *POMDP* there are states that are not observed. This seemingly innocuous extension of the MDP case can lead however to computational difficulties. Let's consider the situation in fig(7.13), and attempt to compute the optimal expected utility based on the sequence of summations and maximisations. The sum over the hidden variables couples all the decisions and observations, meaning that we no longer have a simple chain structure for the remaining maximisations. For a POMDP of length t, this leads to an intractable problem with complexity exponential in t. An alternative view is to recognise that all past decisions and observations $v_{1:t}, d_{1:t-1}$, can be summarised in terms of a belief in the current latent state, $p(h_t|v_{1:t}, d_{1:t-1})$. This suggests that instead of having an actual state, as in the MDP case, we need to use a *distribution* over states to represent our current knowledge. One can therefore write down an effective MDP albeit over belief distributions, as opposed to finite states. Approximate techniques are required to solve the resulting 'infinite' state MDPs, and the reader is referred to more specialised texts for a study of approximation procedures. See for example [162, 165].

7.9.2 Reinforcement learning

Reinforcement Learning deals mainly with time-independent Markov Decision Processes. The added twist is that the transition $p(\mathsf{s}'|\mathsf{s}, \mathsf{d})$ (and possibly the utility) is unknown. Initially an 'agent' begins to explore

the set of states and utilities (rewards) associated with taking decisions. The set of accessible states and their rewards populates as the agent traverses its environment. Consider for example a maze problem with a given start and goal state, though with an unknown maze structure. The task is to get from the start to the goal in the minimum number of moves on the maze. Clearly there is a balance required between curiosity and acting to maximise the expected reward. If we are too curious (don't take optimal decisions given the currently available information about the maze structure) and continue exploring the possible maze routes, this may be bad. On the other hand, if we don't explore the possible maze states, we might never realise that there is a much more optimal short-cut to follow than that based on our current knowledge. This exploration-exploitation tradeoff is central to the difficulties of RL. See [284] for an extensive discussion of reinforcement learning.

From model based to model free learning

Consider a MDP with state transitions $p(x_{t+1}|x_t, d_t)$ and policy $p(d_t|x_t)$. For simplicity we consider utilities that depend only on the state x_t. The expected utility of taking decision d_t in state x_t, $U(d_t|x_t)$, can be derived using a similar argument as in section(7.5.2), for a discount factor γ. Analogous to equation (7.5.10), we have (now including discounting)

$$U(d_t|x_t) = \sum_{x_{t+1}} p(x_{t+1}|x_t, d_t) \left(u(x_{t+1}) + \gamma \max_d \sum_{x'} p(x'|x_{t+1}, d) v(x') \right) \tag{7.9.1}$$

Using the value recursion

$$v(x_{t+1}) = u(x_{t+1}) + \gamma \max_d \sum_{x'} p(x'|x_{t+1}, d) v(x') \tag{7.9.2}$$

we can write

$$U(d_t|x_t) = \sum_{x_{t+1}} p(x_{t+1}|x_t, d_t) v(x_{t+1}) \tag{7.9.3}$$

Substituting this into equation (7.9.2) we have

$$v(x_{t+1}) = u(x_{t+1}) + \gamma \max_d U(d|x_{t+1}) \tag{7.9.4}$$

Substituting equation (7.9.4) into equation (7.9.3) we obtain

$$U(d_t|x_t) = \sum_{x_{t+1}} p(x_{t+1}|x_t, d_t) \left(u(x_{t+1}) + \gamma \max_d U(d|x_{t+1}) \right) \tag{7.9.5}$$

If we know the model $p(x_{t+1}|x_t, d_t)$ we can solve equation (7.9.5) for $U(d|x)$. Given this solution, when we are in state x, the optimal policy is to take the decision $d = \arg\max_d U(d|x)$. In the case that we do not wish to explicitly store or describe a model $p(x_{t+1}|x_t, d_t)$ we can use a sample from this transition to approximate equation (7.9.5); if we are in state x_t and take decision d_t the environment returns for us a sample x_{t+1}. This gives the one-sample estimate to equation (7.9.5):

$$\tilde{U}(d_t|x_t) = u(x_{t+1}) + \gamma \max_d \tilde{U}(d|x_{t+1}) \tag{7.9.6}$$

This gives a highly stochastic update and to ensure convergence it is preferable to use (see below)

$$\tilde{U}_{t+1}(d_t|x_t) = (1 - \alpha_t)\tilde{U}_t(d_t|x_t) + \alpha_t \left(u(x_{t+1}) + \gamma \max_d \tilde{U}_t(d|x_{t+1}) \right) \tag{7.9.7}$$

which can be written

$$\tilde{U}_{t+1}(d_t|x_t) = \tilde{U}_t(d_t|x_t) + \alpha_t \left(u(x_{t+1}) + \gamma \max_d \tilde{U}_t(d|x_{t+1}) - \tilde{U}_t(d_t|x_t) \right) \tag{7.9.8}$$

where the learning rate satisfies $0 \leq \alpha < 1$, $\sum_t \alpha_t = \infty$, $\sum_t \alpha_t^2 < \infty$. (For example $\alpha_t = 1/t$.) This gives a procedure called Q-learning for updating the approximation to U based on samples from the environment[5]. This is a simple and powerful scheme and as such is one of the most popular model-free methods in reinforcement learning. A complicating factor is that if we select a decision based on $d = \arg\max_d \tilde{U}(d|x)$ this influences the sample that will be next drawn. Nevertheless, under certain conditions (essentially all decisions are repeatedly sampled for each state), the sample estimate $\tilde{U}_t(d|x)$ converges to the exact $U(d|x)$ in the limit $t \to \infty$ [312].

Moving average estimator of the mean

In order to motivate the above Q-learning approach, we consider a way to estimate the average $\mu \equiv \sum_x xp(x)$ of a distribution $p(x)$ from a sequence of single samples $x_1, \ldots, x_t$. Based on the sequence, a naive approximation is to take a single sample

$$\tilde{\mu}_t \equiv x_t \tag{7.9.9}$$

Whilst this is an unbiased estimate, it is not consistent. As t increases, the approximation does not improve towards the correct answer μ. A better approximation is to consider the moving average. Defining

$$\hat{\mu}_t \equiv \frac{1}{t}(x_1 + \ldots + x_t) \tag{7.9.10}$$

we have

$$\hat{\mu}_{t+1} = \frac{1}{t+1}(x_1 + \ldots + x_t + x_{t+1}) \tag{7.9.11}$$

From these we can easily derive

$$\hat{\mu}_{t+1} = (1 - \alpha_t)\hat{\mu}_t + \alpha_t x_{t+1} \tag{7.9.12}$$

where $\alpha_t \equiv 1/(t+1)$. This is an unbiased estimator of μ and in the limit $t \to \infty$ converges to the exact mean μ. This is the procedure that is used in Q-learning in which a moving average is used to estimate the expected reward.

Bayesian reinforcement learning

For a given set of environment data $\mathcal{X}$ (observed transitions and utilities) one aspect of the RL problem can be considered as finding the policy that maximises expected reward, given only prior belief about the environment and observed decisions and states. If we assume we know the utility function but not the transition, we may write

$$U(\pi|\mathcal{X}) = \langle U(\pi|\theta)\rangle_{p(\theta|\mathcal{X})} \tag{7.9.13}$$

where θ represents the environment state transition,

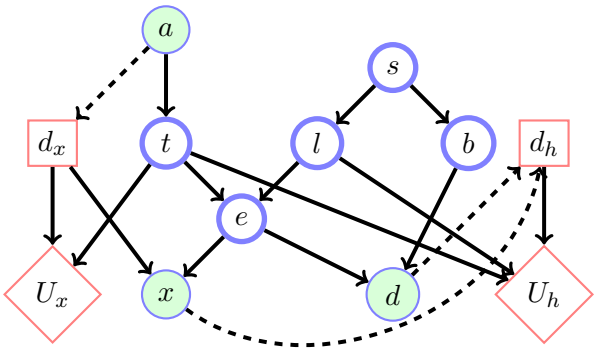
$$\theta = p(x_{t+1}|x_t, d_t) \tag{7.9.14}$$

Given a set of observed states and decisions,

$$p(\theta|\mathcal{X}) \propto p(\mathcal{X}|\theta)p(\theta) \tag{7.9.15}$$

where $p(\theta)$ is a prior on the transition. Similar techniques to the EM style training can be carried through in this case as well[82, 299, 109]. Rather than the policy being a function of the state and the environment θ, optimally one needs to consider a policy $p(d_t|x_t, b(\theta))$ as a function of the state and the belief in the environment, $b(\theta) \equiv p(\theta|\mathcal{X})$. This means that, for example, if the belief in the environment has high entropy, the agent can recognise this and explicitly carry out decisions/actions to explore the environment. A further complication in RL is that the data collected $\mathcal{X}$ depends on the policy π. If we write t for an

[5] The standard notation in the Q-learning literature uses $Q(x, d)$ in place of $U(d|x)$.



s = Smoking
x = Positive X-ray
d = Dyspnea (Shortness of breath)
e = Either Tuberculosis or Lung Cancer
t = Tuberculosis
l = Lung Cancer
b = Bronchitis
a = Visited Asia
d_h = Hospitalise?
d_x = Take X-ray?

Figure 7.14: Influence Diagram for the 'Chest Clinic' Decision example.

'episode' in which policy π_t is followed and data $\mathcal{X}_t$ collected, then the utility of the policy π given all the historical information is

$$U(\pi|\pi_{1:t}, \mathcal{X}_{1:t}) = \langle U(\pi|\theta)\rangle_{p(\theta|\mathcal{X}_{1:t},\pi_{1:t})} \tag{7.9.16}$$

Depending on the prior on the environment, and also on how long each episode is, we will have different posteriors for the environment parameters. If we then set

$$\pi_{t+1} = \operatorname*{argmax}_{\pi} U(\pi|\pi_{1:t}, \mathcal{X}_{1:t}) \tag{7.9.17}$$

this affects the data we collect at the next episode $\mathcal{X}_{t+1}$. In this way, the trajectory of policies $\pi_1, \pi_2, \ldots$ can be very different depending on the episodes and priors.

7.10 Summary

- One way to take decisions is to take that decision that maximises the expected utility of the decision.
- Sequential decision problems can be modelled using decision trees. These are powerful but unwieldy in long decision sequences.
- Influence diagrams extend belief networks to the decision arena. Efficient inference approaches carry over to this case as well, including extensions using the strong junction tree formalism.
- The sequence in which information is revealed and decisions are taken is specified in the influence diagram. The optimal utility is not invariant to the corresponding partial-ordering.
- Markov decision processes correspond to a simple chain-like influence diagram, for which inference is straightforward, and corresponds to the classical Bellman equations.
- Reinforcement learning can be considered an extension of the Markov decision framework when the model of the environment in which the agent acts needs to be learned on the basis of experience.

In this chapter we discussed planning and control as an inference problem with particular attention to discrete variables. See example(28.2) for an application of approximate inference to continuous control.

7.11 Code

7.11.1 Sum/Max under a partial order

`maxsumpot.m`: Generalised elimination operation according to a partial ordering
`sumpotID.m`: Sum/max an ID with probability and decision potentials
`demoDecParty.m`: Demo of summing/maxing an ID

7.11.2 Junction trees for influence diagrams

There is no need to specify the information links provided that a partial ordering is given. In the code `jtreeID.m` no check is made that the partial ordering is consistent with the influence diagram. In this case, the first step of the junction tree formulation in section(7.4.2) is not required. Also the moralisation and removal of utility nodes is easily dealt with by defining utility potentials and including them in the moralisation process.

The strong triangulation is found by a simple variable elimination scheme which seeks to eliminate a variable with the least number of neighbours, provided that the variable may be eliminated according to the specified partial ordering. The junction tree is constructed based only on the elimination clique sequence $\mathcal{C}_1, \ldots, \mathcal{C}_N$. obtained from the triangulation routine. The junction tree is then obtained by connecting a clique $\mathcal{C}_i$ to the first clique $j > i$ that is connected to this clique. Clique $\mathcal{C}_i$ is then eliminated from the graph. In this manner a junction tree of connected cliques is formed. We do not require the separators for the influence diagram absorption since these can be computed and discarded on the fly.

Note that the code only computes messages from the leaves to the root of the junction tree, which is sufficient for taking decisions at the root. If one desires an optimal decision at a non-root, one would need to absorb probabilities into a clique which contains the decision required. These extra forward probability absorptions are required because information about any unobserved variables can be affected by decisions and observations in the past. This extra forward probability schedule is not given in the code and left as an exercise for the interested reader.

`jtreeID.m`: Junction Tree for an Influence Diagram
`absorptionID.m`: Absorption on an Influence Diagram
`triangulatePorder.m`: Triangulation based on a partial ordering
`demoDecPhD.m`: Demo for utility of Doing PhD and Startup

7.11.3 Party-Friend example

The code below implements the Party-Friend example in the text. To deal with the asymmetry the $Visit$ utility is zero if $Party$ is in state `yes`.

`demoDecPartyFriend.m`: Demo for Party-Friend

7.11.4 Chest Clinic with Decisions

The table for the Chest Clinic Decision network, fig(7.14) is taken from exercise(3.4), see [132, 73]. There is a slight modification however to the $p(x|e)$ table. If an x-ray is taken, then information about x is available. However, if the decision is not to take an x-ray no information about x is available. This is a form of asymmetry. A straightforward approach in this case is to make d_x a parent of the x variable and set the

distribution of x to be uninformative if $d_x = \mathsf{fa}$.

$$\begin{array}{ll}
p(a = \mathsf{tr}) = 0.01 & p(s = \mathsf{tr}) = 0.5 \\
p(t = \mathsf{tr}|a = \mathsf{tr}) = 0.05 & p(t = \mathsf{tr}|a = \mathsf{fa}) = 0.01 \\
p(l = \mathsf{tr}|s = \mathsf{tr}) = 0.1 & p(l = \mathsf{tr}|s = \mathsf{fa}) = 0.01 \\
p(b = \mathsf{tr}|s = \mathsf{tr}) = 0.6 & p(b = \mathsf{tr}|s = \mathsf{fa}) = 0.3 \\
p(x = \mathsf{tr}|e = \mathsf{tr}, d_x = \mathsf{tr}) = 0.98 & p(x = \mathsf{tr}|e = \mathsf{fa}, d_x = \mathsf{tr}) = 0.05 \\
p(x = \mathsf{tr}|e = \mathsf{tr}, d_x = \mathsf{fa}) = 0.5 & p(x = \mathsf{tr}|e = \mathsf{fa}, d_x = \mathsf{fa}) = 0.5 \\
p(d = \mathsf{tr}|e = \mathsf{tr}, b = \mathsf{tr}) = 0.9 & p(d = \mathsf{tr}|e = \mathsf{tr}, b = \mathsf{fa}) = 0.3 \\
p(d = \mathsf{tr}|e = \mathsf{fa}, b = \mathsf{tr}) = 0.2 & p(d = \mathsf{tr}|e = \mathsf{fa}, b = \mathsf{fa}) = 0.1
\end{array} \tag{7.11.1}$$

The two utilities are designed to reflect the costs and benefits of taking an x-ray and hospitalising a patient:

$d_h = \mathsf{tr}$	$t = \mathsf{tr}$	$l = \mathsf{tr}$	180
$d_h = \mathsf{tr}$	$t = \mathsf{tr}$	$l = \mathsf{fa}$	120
$d_h = \mathsf{tr}$	$t = \mathsf{fa}$	$l = \mathsf{tr}$	160
$d_h = \mathsf{tr}$	$t = \mathsf{fa}$	$l = \mathsf{fa}$	15
$d_h = \mathsf{fa}$	$t = \mathsf{tr}$	$l = \mathsf{tr}$	2
$d_h = \mathsf{fa}$	$t = \mathsf{tr}$	$l = \mathsf{fa}$	4
$d_h = \mathsf{fa}$	$t = \mathsf{fa}$	$l = \mathsf{tr}$	0
$d_h = \mathsf{fa}$	$t = \mathsf{fa}$	$l = \mathsf{fa}$	40

(7.11.2)

$d_x = \mathsf{tr}$	$t = \mathsf{tr}$	0
$d_x = \mathsf{tr}$	$t = \mathsf{fa}$	1
$d_x = \mathsf{fa}$	$t = \mathsf{tr}$	10
$d_x = \mathsf{fa}$	$t = \mathsf{fa}$	10

(7.11.3)

We assume that we know whether or not the patient has been to Asia, before deciding on taking an x-ray. The partial ordering is then

$$a \prec d_x \prec \{d, x\} \prec d_h \prec \{b, e, l, s, t\} \tag{7.11.4}$$

The demo `demoDecAsia.m` produces the results:

```
utility table:
asia = yes takexray = yes  49.976202
asia = no  takexray = yes  46.989441
asia = yes takexray = no   48.433043
asia = no  takexray = no   47.460900
```

which shows that optimally one should take an x-ray only if the patient has been to Asia.
`demoDecAsia.m`: Junction Tree Influence Diagram demo

7.11.5 Markov decision processes

In `demoMDP.m` we consider a simple two dimensional grid in which an 'agent' can move to a grid square either above, below, left, right of the current square, or stay in the current square. We defined goal states (grid squares) that have high utility, with others having zero utility.
`demoMDPclean.m`: Demo of Value and policy iteration for a simple MDP
`MDPsolve.m`: MDP solver using value or policy iteration
Routines for efficient MDP variational solvers are available from the book website. There is also code for fast Lagrange duality techniques, which are beyond the scope of our discussion here.

7.12 Exercises

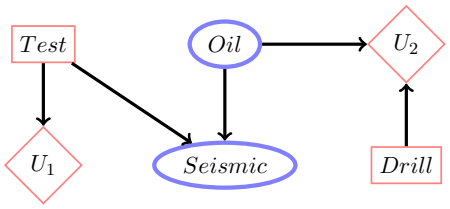
Exercise 7.1. *You play a game in which you have a probability p of winning. If you win the game you gain an amount £S and if you lose the game you lose an amount £S. Show that the expected gain from playing the game is £$(2p-1)S$.*

Exercise 7.2. *It is suggested that the utility of money is based, not on the amount, but rather how much we have relative to other people. Assume a distribution $p(i)$, $i = 1, \ldots, 10$ of incomes using a histogram with 10*

bins, each bin representing an income range. Use a histogram to roughly reflect the distribution of incomes in society, namely that most incomes are around the average with few very wealthy and few extremely poor people. Now define the utility of an income x as the chance that income x will be higher than a randomly chosen income y (under the distribution you defined) and relate this to the cumulative distribution of p. Write a program to compute this probability and plot the resulting utility as a function of income. Now repeat the coin tossing bet of section(7.1.1) so that if one wins the bet one's new income will be placed in the top histogram bin, whilst if one loses one's new income is in the lowest bin. Compare the optimal expected utility decisions under the situations in which one's original income is (i) average, and (ii) much higher than average.

Exercise 7.3.

Derive a partial ordering for the ID on the right, and explain how this ID differs from that of fig(7.5).



Exercise 7.4. *This question follows closely* `demoMDP.m`*, and represents a problem in which a pilot wishes to land an airplane. The matrix $U(x, y)$ in the file* `airplane.mat` *contains the utilities of being in position x, y and is a very crude model of a runway and taxiing area. The airspace is represented by an 18×15 grid ($Gx = 18, Gy = 15$ in the notation employed in* `demoMDP.m`*). The matrix $U(8, 4) = 2$ represents that position $(8, 4)$ is the desired parking bay of the airplane (the vertical height of the airplane is not taken in to account). The positive values in U represent runway and areas where the airplane is allowed. Zero utilities represent neutral positions. The negative values represent unfavourable positions for the airplane. By examining the matrix U you will see that the airplane should preferably not veer off the runway, and also should avoid two small villages close to the airport.*

At each timestep the plane can perform one of the following actions stay up down left right:
For stay, the airplane stays in the same x, y position.
For up, the airplane moves to the $x, y + 1$ position.
For down, the airplane moves to the $x, y - 1$ position.
For left, the airplane moves to the $x - 1, y$ position.
For right, the airplane moves to the $x + 1, y$ position.

A move that takes the airplane out of the airspace is not allowed, and the plane remains in its current position x, y. For example, if we issue the right action, then we stay at x, y if $x + 1, y$ is out of the airspace. @@

1. *The airplane begins in at point $x = 1, y = 13$. Assuming that an action deterministically results in the intended grid move, find the optimal x_t, y_t sequence for times $t = 1, \ldots,$ for the position of the aircraft.*

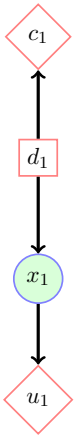
2. *The pilot tells you that there is a fault with the airplane for the right action. Provided $x + 1, y$ is in the airspace for current position x, y:* @@

   If $x, y + 1$ is out of the airspace, then we go right to $x + 1, y$ with probability 1.

   If $x, y+1$ is in the airspace, we go right to $x+1, y$ with probability 0.9 and up to $x, y+1$ with probability 0.1.

   Assuming again that the airplane begins at point $x = 1, y = 13$, return the sequence of positions x_t, y_t, $t = 1, \ldots,$ the aircraft would ideally reach. @@

Exercise 7.5.



The influence diagram depicted describes the first stage of a game. The decision variable $dom(d_1) = \{\textsf{play}, \textsf{not play}\}$*, indicates the decision to either play the first stage or not. If you decide to play, there is a cost* $c_1(\textsf{play}) = C_1$*, but no cost otherwise,* $c_1(\textsf{no play}) = 0$*. The variable* x_1 *describes if you win or lose the game,* $dom(x_1) = \{\textsf{win}, \textsf{lose}\}$*, with probabilities:*

$$p(x_1 = \textsf{win}|d_1 = \textsf{play}) = p_1, \qquad p(x_1 = \textsf{win}|d_1 = \textsf{no play}) = 0 \tag{7.12.1}$$

The utility of winning/losing is

$$u_1(x_1 = \textsf{win}) = W_1, \qquad u_1(x_1 = \textsf{lose}) = 0 \tag{7.12.2}$$

Show that the expected utility gain of playing this game is

$$U(d_1 = \textsf{play}) = p_1 W_1 - C_1 \tag{7.12.3}$$

Exercise 7.6. *Exercise(7.5) above describes the first stage of a new two-stage game. If you win the first stage* $x_1 = \textsf{win}$*, you have to make a decision* d_2 *as to whether or not play in the second stage* $dom(d_2) = \{\textsf{play}, \textsf{not play}\}$*. If you do not win the first stage, you cannot enter the second stage. If you decide to play the second stage, you win with probability* p_2*:*

$$p(x_2 = \textsf{win}|x_1 = \textsf{win}, d_2 = \textsf{play}) = p_2 \tag{7.12.4}$$

If you decide not to play the second stage there is no chance to win:

$$p(x_2 = \textsf{win}|x_1 = \textsf{win}, d_2 = \textsf{not play}) = 0 \tag{7.12.5}$$

The cost of playing the second stage is

$$c_2(d_2 = \textsf{play}) = C_2, \qquad c_2(d_2 = \textsf{no play}) = 0 \tag{7.12.6}$$

and the utility of winning/losing the second stage is

$$u_2(x_2 = \textsf{win}) = W_2, \qquad u_2(x_2 = \textsf{lose}) = 0 \tag{7.12.7}$$

1. *Draw an Influence Diagram that describes this two-stage game.*

2. *A gambler needs to decide if he should even enter the first stage of this two-stage game. Show that based on taking the optimal future decision* d_2 *the expected utility based on the first decision is:*

$$U(d_1 = \textsf{play}) = \begin{cases} p_1(p_2 W_2 - C_2) + p_1 W_1 - C_1 & \text{if } \; p_2 W_2 - C_2 \geq 0 \\ p_1 W_1 - C_1 & \text{if } \; p_2 W_2 - C_2 \leq 0 \end{cases} \tag{7.12.8}$$

Exercise 7.7. *You have* $£B$ *in your bank account. You are asked if you would like to participate in a bet in which, if you win, your bank account will become* $£W$*. However, if you lose, your bank account will contain only* $£L$*. You win the bet with probability* p_w*.*

1. *Assuming that the utility is given by the number of pounds in your bank account, write down a formula for the expected utility of taking the bet,* $U(\textsf{bet})$ *and also the expected utility of not taking the bet,* $U(\textsf{no bet})$*.*

2. *The above situation can be formulated differently. If you win the bet you gain* $£(W - B)$*. If you lose the bet you lose* $£(B - L)$*. Compute the expected amount of money you gain if you bet* $U_{gain}(\textsf{bet})$ *and if you don't bet* $U_{gain}(\textsf{no bet})$*.*

3. *Show that* $U(\textsf{bet}) - U(\textsf{no bet}) = U_{gain}(\textsf{bet}) - U_{gain}(\textsf{no bet})$*.*

Exercise 7.8. *Consider an objective*

$$F(\theta) = \sum_x U(x)p(x|\theta) \tag{7.12.9}$$

for a positive function $U(x)$ and that our task is to maximise F with respect to θ. An Expectation-Maximisation style bounding approach (see section(11.2)) can be derived by defining the auxiliary distribution

$$\tilde{p}(x|\theta) = \frac{U(x)p(x|\theta)}{F(\theta)} \tag{7.12.10}$$

so that by considering $KL(q(x)|\tilde{p}(x))$ for some variational distribution $q(x)$ we obtain the bound

$$\log F(\theta) \geq -\left\langle \log q(x) \right\rangle_{q(x)} + \left\langle \log U(x) \right\rangle_{q(x)} + \left\langle \log p(x|\theta) \right\rangle_{q(x)} \tag{7.12.11}$$

The M-step states that the optimal q distribution is given by

$$q(x) = \tilde{p}(x|\theta_{old}) \tag{7.12.12}$$

At the E-step of the algorithm the new parameters θ_{new} are given by maximising the 'energy' term

$$\theta_{new} = \operatorname*{argmax}_\theta \left\langle \log p(x|\theta) \right\rangle_{\tilde{p}(x|\theta_{old})} \tag{7.12.13}$$

Show that for a deterministic distribution

$$p(x|\theta) = \delta\left(x, f(\theta)\right) \tag{7.12.14}$$

the E-step fails, giving $\theta_{new} = \theta_{old}$.

Exercise 7.9. *Consider an objective*

$$F_\epsilon(\theta) = \sum_x U(x)p_\epsilon(x|\theta) \tag{7.12.15}$$

for a positive function $U(x)$ and

$$p_\epsilon(x|\theta) = (1-\epsilon)\delta\left(x, f(\theta)\right) + \epsilon n(x), \quad 0 \leq \epsilon \leq 1 \tag{7.12.16}$$

and an arbitrary distribution $n(x)$. Our task is to maximise F with respect to θ. As the previous exercise showed, if we attempt an EM algorithm in the limit of a deterministic model $\epsilon = 0$, then no-updating occurs and the EM algorithm fails to find θ that optimises $F_0(\theta)$.

1. *Show that*

$$F_\epsilon(\theta) = (1-\epsilon)F_0(\theta) + \epsilon \sum_x n(x)U(x) \tag{7.12.17}$$

and hence

$$F_\epsilon(\theta_{new}) - F_\epsilon(\theta_{old}) = (1-\epsilon)\left[F_0(\theta_{new}) - F_0(\theta_{old})\right] \tag{7.12.18}$$

2. *Show that if for $\epsilon > 0$ we can find a θ_{new} such that $F_\epsilon(\theta_{new}) > F_\epsilon(\theta_{old})$, then necessarily $F_0(\theta_{new}) > F_0(\theta_{old})$.*

3. *Using this result, derive an EM-style algorithm that guarantees to increase $F_\epsilon(\theta)$ (unless we are already at an optimum) for $\epsilon > 0$ and therefore guarantees to increase $F_0(\theta)$. Hint: use*

$$\tilde{p}(x|\theta) = \frac{U(x)p_\epsilon(x|\theta)}{F_\epsilon(\theta)} \tag{7.12.19}$$

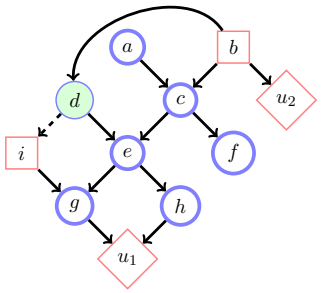
and consider $KL(q(x)|\tilde{p}(x))$ for some variational distribution $q(x)$.

Exercise 7.10. *The file* `IDjensen.mat` *contains probability and utility tables for the influence diagram of fig(7.7a). Using* BRMLtoolbox*, write a program that returns the maximal expected utility for this ID using a strong junction tree approach, and check the result by explicit summation and maximisation. Similarly, your program should output the maximal expected utility for both states of* d_1*, and check that the computation using the strong junction tree agrees with the result from explicit summation and maximisation.*

Exercise 7.11. *For a POMDP, explain the structure of the strong junction tree, and relate this to the complexity of inference in the POMDP.*

Exercise 7.12.

(i) Define a partial order for the ID depicted. (ii) Draw a (strong) junction tree for this ID.



Exercise 7.13. `exerciseInvest.m` *contains the parameters for a simple investment problem in which the prices of two assets, a and b follow Markovian updating, as in section(7.8.3). The transition matrices of these are given, as is the end time* T*, initial wealth* w_1*, and initial price movements* ϵ_1^a*,* ϵ_1^b*, wealth and investment states. Write a function of the form*

```
[d1 val]=optdec(epsilonA1,epsilonB1,desired,T,w1,pars)
```

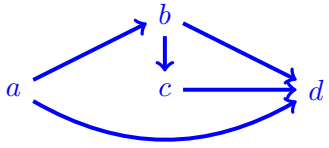
where `desired` *is the desired wealth level at time* T*. The end utility is defined by*

$$u(w_T) = \begin{cases} 10000 & w_T \geq 1.5 w_1 \\ 0 & w_T < 1.5 w_1 \end{cases} \tag{7.12.20}$$

@@ *Using your routine, compute the optimal expected utility and decision at time 1. (Round wealth values to the nearest possible discrete values given in* `exerciseInvest.m`*.) Draw also an influence diagram that describes this Markov decision problem.*

++ **Exercise 7.14.** *Tom writes two cheques, one of which has a value of twice the other. You can't see the value of either cheque and are asked to pick one. The one you select has a value of £100. You can cash this cheque, or exchange it for the other cheque. Your friend Randy thinks that it doesn't really matter what you do since you have a 50% chance of having in your hand the higher value cheque. What do you think you should do? What if you don't even look at the value of the cheque you first select?*

++ **Exercise 7.15.** *You're planning on going on a Summer trip and are contemplating the best route to take. You start at location a and want to get to location d. The graph of locations and (one-way) roads between them is shown below*



Each segment has an associated road length cost. For example, C_{ab} *is the cost of going from a to b. However, we don't know if the road from b to d is going to be open; historically, it is open with probability p. Whilst we don't know when we start out at a whether the road from b to d will be open, if we decide to go along the route* $a \to b$*, when we get to b we will know if the road* $b \to d$ *is open. If it's open, we can decide whether to go along* $b \to d$*, or the alternative* $b \to c \to d$*. If* $b \to d$ *is not open, then we must take the route* $b \to c \to d$*.*

1. *Using the above graph, the road costs* $C_{ab}, C_{bc}, C_{bd}, C_{cd}, C_{ad}$ *and probability p, explicitly show how to decide whether it is better (in terms of minimal expected total cost) to initially take the road* $a \to b$ *or go directly* $a \to d$*.*

2. *For the same graph as above, a friend suggests a way to decide which route to take. Assuming $C_{bd} < C_{bc} + C_{cd}$, and that $b \to d$ is open, it's clear that (provided you go from a to b) you would prefer then to go from b to d directly. Otherwise, if $b \to d$ is not open, you will go along $b \to c \to d$.*

   You look at the possible routes, namely $a \to d$, $a \to b \to d$, $a \to b \to c \to d$ and calculate the expected cost of each route. You should then select the route with the minimal expected cost and decide to go from a to b if the route with the minimal cost includes going from a to b. Is there any difference between this approach and the one in question part (1) above?

Exercise 7.16.

++

1. *Consider that, for any function $f(x, y)$ of two variables x and y*

$$\max_x f(x, y) \geq f(x, y)$$

   Show that

$$\sum_y \max_x f(x, y) \geq \max_x \sum_y f(x, y)$$

2. *For a finite T, a Markov Decision Process specifies a distribution over a set of states $s_{2:T}$, given actions $a_{1:T}$ as follows:*

$$p(s_{2:T}|a_{1:T-1}) = \prod_{t=2}^{T} p(s_t|s_{t-1}, a_{t-1})$$

   Once an action a_t is taken, the new state s_{t+1} will be revealed. Given the initial state s_1, the optimal probability that s_T is in state 1 is given by

$$P^*(s_1) \equiv \max_{a_1} \sum_{s_2} \max_{a_2} \ldots \sum_{s_{T-1}} \max_{a_{T-1}} p(s_{2:T-1}, s_T = 1|a_{1:T-1}, s_1)$$

   Derive an $O(T)$ algorithm that will calculate $P^*(s_1)$ and decide which action to take at timestep 1.

3. *Define*

$$P^{**}(s_1) \equiv \max_{a_{1:T-1}} p(s_T = 1|a_{1:T-1}, s_1)$$

   and show that $P^*(s_1) \geq P^{**}(s_1)$, giving also an intuitive explanation why this must be the case.

Exercise 7.17. *At timestep 1, you start out at position $s_1 = 0$. At each time step there are 3 actions you can take: down, same, up. The down action means $s_{t+1} = s_t - 1$, same means $s_{t+1} = s_t$ and up means $s_{t+1} = s_t + 1$. Your task is to hit one of two possible targets at time $t = 50$. One of the targets is at position $+25$ and the other is at position -25. If you hit a target, you receive a reward of 1, otherwise 0. Unfortunately, the noise in the environment means that when you take an action this is only carried out with probability 0.8, with probability 0.1 of each the other two possible actions being carried out instead. What's the optimal expected reward of taking the actions down, same, up at timestep 1?* ++

Part II

Learning in Probabilistic Models

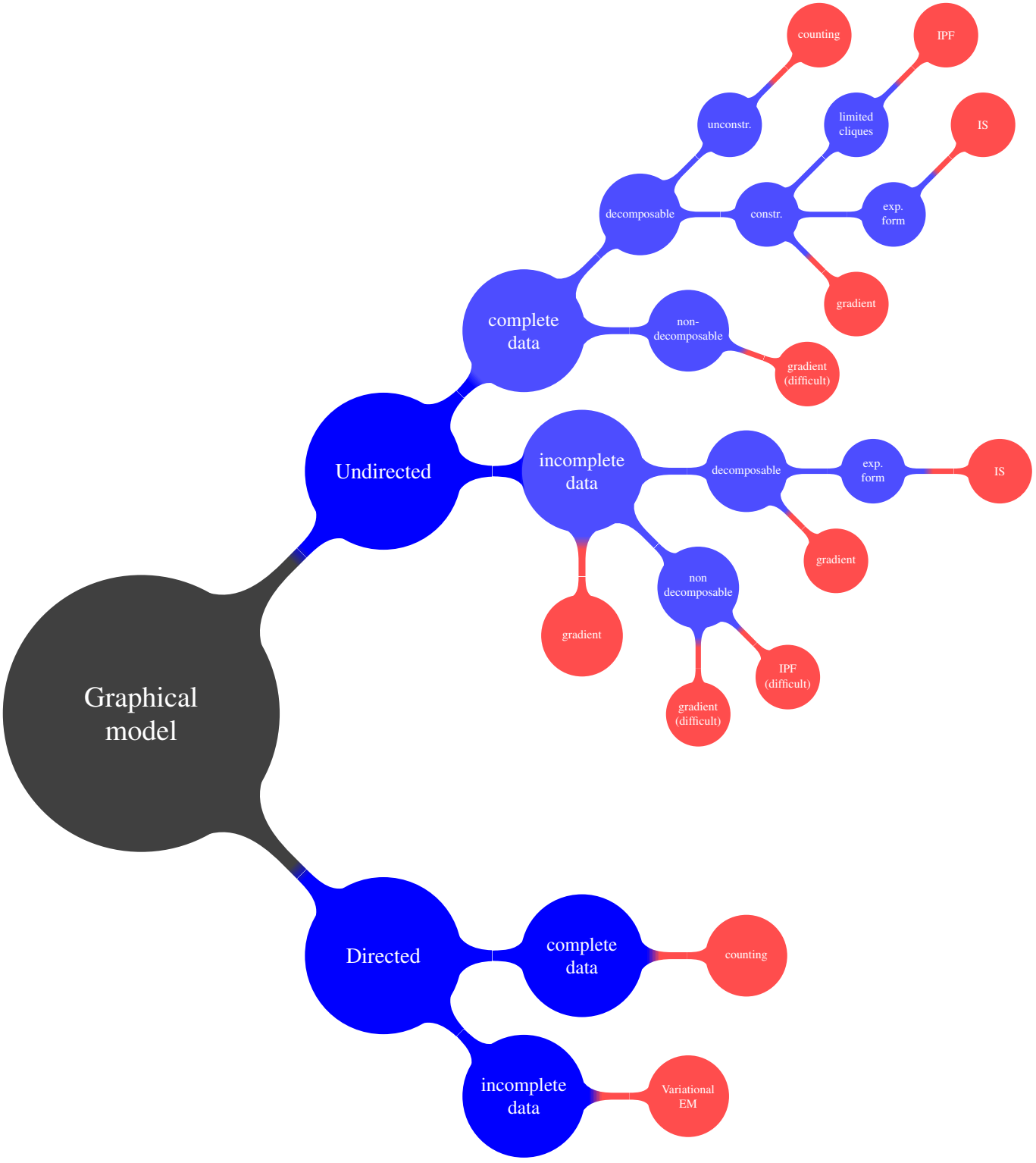
Introduction to Part II

In Part II we address how to learn a model from data. In particular we will discuss learning a model as a form of inference on an extended distribution, now taking into account the parameters of the model.

Learning a model or model parameters from data forces us to deal with uncertainty since with only limited data we can never be certain which is the 'correct' model. We also address how the structure of a model, not just its parameters, can in principle be learned.

In Part II we show how learning can be achieved under simplifying assumptions, such as maximum likelihood that set parameters by those that would most likely reproduce the observed data. We also discuss the problems that arise when, as is often the case, there is missing data.

Together with Part I, Part II prepares the basic material required to embark on understanding models in Machine Learning, having the tools required to learn models from data and subsequently query them to answer questions of interest.



Maximum likelihood algorithms for learning in graphical models. The leaf nodes denote specific algorithms described in Part II.

CHAPTER 8

Statistics for Machine Learning

In this chapter we discuss some classical distributions and their manipulations. In previous chapters we've assumed that we know the distributions and have concentrated on the inference problem. In machine learning we will typically not fully know the distributions and need to learn them from available data. This means we need familiarity with standard distributions, for which the data will later be used to set the parameters.

8.1 Representing Data

The numeric encoding of data can have a significant effect on performance and an understanding of the options for representing data is therefore of considerable importance. We briefly outline three central encodings below.

8.1.1 Categorical

For categorical (or nominal) data, the observed value belongs to one of a number of classes, with no intrinsic ordering, and can be represented simply by an integer. An example of a categorical variable would be the description of the type of job that someone does, *e.g.* healthcare, education, financial services, transport, homeworker, unemployed, engineering *etc.* which could be represented by the values $1, 2, \ldots, 7$. Another way to transform such data into numerical values would be to use 1-of-m encoding. For example, if there are four kinds of jobs: soldier, sailor, tinker, spy, we could represent a soldier as (1,0,0,0), a sailor as (0,1,0,0), a tinker as (0,0,1,0) and a spy as (0,0,0,1). In this encoding the distance between the vectors representing two different professions is constant. Note that 1-of-m encoding induces dependencies in the profession attributes since if one of the attributes is 1, the others must be zero.

8.1.2 Ordinal

An ordinal variable consists of categories with an ordering or ranking of the categories, *e.g.* cold, cool, warm, hot. In this case, to preserve the ordering, we could use say -1 for cold, 0 for cool, +1 for warm and +2 for hot. This choice is somewhat arbitrary, and one should bear in mind that results may be dependent on the numerical coding used.

8.1.3 Numerical

Numerical data takes on values that are real numbers, *e.g.* a temperature measured by a thermometer, or the salary that someone earns.

8.2 Distributions

Distributions over discrete variables, section(1.1) have been the focus of much of the book up to this point. Here we discuss also distributions over continuous variables, for which the concepts of marginalisation and conditioning carry over from the discrete case, simply on replacing summation over the discrete states with integration over the continuous domain of the variable.

Definition 8.1 (Probability Density Functions)**.** For a continuous variable x, the probability density $p(x)$ is defined such that

$$p(x) \geq 0, \qquad \int_{-\infty}^{\infty} p(x)dx = 1, \qquad p(a \leq x \leq b) = \int_{a}^{b} p(x)dx \tag{8.2.1}$$

We will also refer to continuous probability densities as distributions.

Definition 8.2 (Averages and Expectation)**.**

$$\langle f(x) \rangle_{p(x)} \tag{8.2.2}$$

denotes the average or expectation of $f(x)$ with respect to the distribution $p(x)$. A common alternative notation is

$$\mathbb{E}\left(f(x)\right) \tag{8.2.3}$$

When the context is clear, one may drop the notational dependency on $p(x)$. The notation

$$\langle f(x)|y \rangle \tag{8.2.4}$$

is shorthand for the average of $f(x)$ conditioned on knowing the state of variable y, *i.e.* the average of $f(x)$ with respect to the distribution $p(x|y)$.

An advantage of the expectation notations is that they hold whether the distribution is over continuous or discrete variables. In the discrete case

$$\langle f(x) \rangle \equiv \sum_{\mathsf{x}} f(x = \mathsf{x}) p(x = \mathsf{x}) \tag{8.2.5}$$

and for continuous variables,

$$\langle f(x) \rangle \equiv \int_{-\infty}^{\infty} f(x) p(x) dx \tag{8.2.6}$$

The reader might wonder what $\langle x \rangle$ means when x is discrete. For example, if $\mathrm{dom}(x) = \{\mathsf{apple}, \mathsf{orange}, \mathsf{pear}\}$, with associated probabilities $p(x)$ for each of the states, what does $\langle x \rangle$ refer to? Clearly, $\langle f(x) \rangle$ makes sense if $f(x = \mathsf{x})$ maps the state x to a numerical value. For example $f(x = \mathsf{apple}) = 1$, $f(x = \mathsf{orange}) = 2$, $f(x = \mathsf{pear}) = 3$ for which $\langle f(x) \rangle$ is meaningful. Unless the states of the discrete variable are associated with a numerical value, then $\langle x \rangle$ has no meaning.

Result 8.1 (Change of variables)**.** For a univariate continuous random variable x with distribution $p(x)$ the transformation $y = f(x)$, where $f(x)$ is a monotonic function, has distribution

$$p(y) = p(x) \left(\frac{df}{dx} \right)^{-1}, \quad x = f^{-1}(y) \tag{8.2.7}$$

For multivariate $\mathbf{x}$ and bijection $\mathbf{f}(\mathbf{x})$, then $\mathbf{y} = \mathbf{f}(\mathbf{x})$ has distribution

$$p(\mathbf{y}) = p(\mathbf{x} = \mathbf{f}^{-1}(\mathbf{y})) \left| \det \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right) \right|^{-1} \tag{8.2.8}$$

where the Jacobian matrix has elements

$$\left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]_{ij} = \frac{\partial f_i(\mathbf{x})}{\partial x_j} \tag{8.2.9}$$

Sometimes one needs to consider transformations between different dimensions. For example, if $\mathbf{z}$ has lower dimension than $\mathbf{x}$, then one may introduce additional variables $\mathbf{z}'$ to define a new multivariate $\mathbf{y} = (\mathbf{z}, \mathbf{z}')$ with the same dimension as $\mathbf{x}$. Then one applies the above transformation to give the distribution on the joint variables $\mathbf{y}$, from which $p(\mathbf{z})$ can be obtained by marginalisation.

Definition 8.3 (Moments)**.** The k^{th} moment of a distribution is given by the average of x^k under the distribution:

$$\left\langle x^k \right\rangle_{p(x)} \tag{8.2.10}$$

For $k = 1$, we have the mean, typically denoted by μ,

$$\mu \equiv \langle x \rangle \tag{8.2.11}$$

Definition 8.4 (Cumulative Distribution Function)**.** For a univariate distribution $p(x)$, the CDF is defined as

$$cdf(y) \equiv p(x \leq y) = \langle \mathbb{I}\left[x \leq y\right] \rangle_{p(x)} \tag{8.2.12}$$

For an unbounded domain, $cdf(-\infty) = 0$ and $cdf(\infty) = 1$.

Definition 8.5 (Moment Generating Function)**.** For a distribution $p(x)$, we define the moment generating function $g(t)$ as

$$g(t) = \left\langle e^{tx} \right\rangle_{p(x)} \tag{8.2.13}$$

The usefulness of this is that by differentiating $g(t)$, we 'generate' the moments,

$$\lim_{t \to 0} \frac{d^k}{dt^k} g(t) = \left\langle x^k \right\rangle_{p(x)} \tag{8.2.14}$$

Definition 8.6 (Mode)**.** The mode x_* of a distribution $p(x)$ is the state of x at which the distribution takes its highest value, $x_* = \arg\max_x p(x)$. A distribution could have more than one mode (be multi-modal). A widespread abuse of terminology is to refer to any isolated local maximum of $p(x)$ to be a mode.

Definition 8.7 (Variance and Correlation)**.**

$$\sigma^2 \equiv \left\langle \left(x - \langle x \rangle\right)^2 \right\rangle_{p(x)} \tag{8.2.15}$$

The variance measures the 'spread' of a distribution around the mean. The square root of the variance, σ is called the *standard deviation* and is a natural length scale suggesting how far typical values drawn from $p(x)$ might be from the mean. The notation $\mathrm{var}(x)$ is also used to emphasise for which variable the variance is computed. The reader may show that an equivalent expression is

$$\sigma^2 \equiv \left\langle x^2 \right\rangle - \langle x \rangle^2 \tag{8.2.16}$$

For a multivariate distribution the matrix with elements

$$\Sigma_{ij} = \langle (x_i - \mu_i)(x_j - \mu_j) \rangle \tag{8.2.17}$$

where $\mu_i = \langle x_i \rangle$ is called the *covariance matrix*. The diagonal entries of the covariance matrix contain the variance of each variable. An equivalent expression is

$$\Sigma_{ij} = \langle x_i x_j \rangle - \langle x_i \rangle \langle x_j \rangle \tag{8.2.18}$$

The *correlation matrix* has elements

$$\rho_{ij} = \left\langle \frac{(x_i - \mu_i)}{\sigma_i} \frac{(x_j - \mu_j)}{\sigma_j} \right\rangle \tag{8.2.19}$$

where σ_i is the deviation of variable x_i. The correlation is a normalised form of the covariance so that each element is bounded $-1 \leq \rho_{ij} \leq 1$. The reader will note a resemblance of the correlation coefficient and the scalar product, equation (A.1.3). See also exercise(8.40).

For independent variables x_i and x_j, $x_i \perp\!\!\!\perp x_j$ the covariance Σ_{ij} is zero. Similarly independent variables have zero correlation – they are 'uncorrelated'. Note however that the converse is not generally true – two variables can be uncorrelated but dependent. A special case is for when x_i and x_j are Gaussian distributed for which independence is equivalent to being uncorrelated, see exercise(8.2).

Definition 8.8 (Skewness and Kurtosis)**.** The skewness is a measure of the asymmetry of a distribution:

$$\gamma_1 \equiv \frac{\left\langle (x - \langle x \rangle)^3 \right\rangle_{p(x)}}{\sigma^3} \tag{8.2.20}$$

where σ^2 is the variance of x with respect to $p(x)$. A positive skewness means the distribution has a heavy tail to the right. Similarly, a negative skewness means the distribution has a heavy tail to the left.

The kurtosis is a measure of how peaked around the mean a distribution is:

$$\gamma_2 \equiv \frac{\left\langle (x - \langle x \rangle)^4 \right\rangle_{p(x)}}{\sigma^4} - 3 \tag{8.2.21}$$

A distribution with positive kurtosis has more mass around its mean than would a Gaussian with the same mean and variance. These are also called *super Gaussian*. Similarly a negative kurtosis (*sub Gaussian*) distribution has less mass around its mean than the corresponding Gaussian. The kurtosis is defined such that a Gaussian has zero kurtosis (which accounts for the -3 term in the definition).

Definition 8.9 (Delta function)**.** For continuous x, we define the Dirac delta function

$$\delta(x - x_0) \tag{8.2.22}$$

which is zero everywhere except at x_0, where there is a spike. $\int_{-\infty}^{\infty} \delta(x - x_0) dx = 1$ and @@

$$\int_{-\infty}^{\infty} \delta(x - x_0) f(x) dx = f(x_0) \tag{8.2.23}$$

One can view the Dirac delta function as an infinitely narrow Gaussian: $\delta(x - x_0) = \lim_{\sigma \to 0} \mathcal{N}\left(x | x_0, \sigma^2\right)$.

The Kronecker delta,

$$\delta_{x,x_0} \tag{8.2.24}$$

is similarly zero everywhere, except for $\delta_{x_0,x_0} = 1$. The Kronecker delta is equivalent to $\delta_{x,x_0} = \mathbb{I}[x = x_0]$. We use the expression $\delta(x, x_0)$ to denote either the Dirac or Kronecker delta, depending on the context.

Definition 8.10 (Empirical Distribution)**.** For a set of datapoints $x^1, \ldots, x^N$, which are states of a random variable x, the empirical distribution has probability mass distributed evenly over the datapoints, and zero elsewhere.

For a discrete variable x the empirical distribution is, see fig(8.1),

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}\left[x = x^n\right] \tag{8.2.25}$$

where N is the number of datapoints.

For a continuous distribution we have

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} \delta\left(x - x^n\right) \tag{8.2.26}$$

where $\delta(x)$ is the Dirac Delta function.

The mean of the empirical distribution is given by the sample mean of the datapoints

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} x^n \tag{8.2.27}$$

Similarly, the variance of the empirical distribution is given by the sample variance

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^{N} \left(x^n - \hat{\mu}\right)^2 \tag{8.2.28}$$

For vectors the sample mean vector has elements

$$\hat{\mu}_i = \frac{1}{N} \sum_{n=1}^{N} x_i^n \tag{8.2.29}$$

and sample covariance matrix has elements

$$\hat{\Sigma}_{ij} = \frac{1}{N} \sum_{n=1}^{N} \left(x_i^n - \hat{\mu}_i\right)\left(x_j^n - \hat{\mu}_j\right) \tag{8.2.30}$$

8.2.1 The Kullback-Leibler Divergence KL$(q|p)$

The Kullback-Leibler divergence KL$(q|p)$ measures the 'difference' between distributions q and p[72].

Definition 8.11 (KL divergence)**.** For two distributions $q(x)$ and $p(x)$

$$\mathrm{KL}(q|p) \equiv \left\langle \log q(x) - \log p(x) \right\rangle_{q(x)} \geq 0 \tag{8.2.31}$$

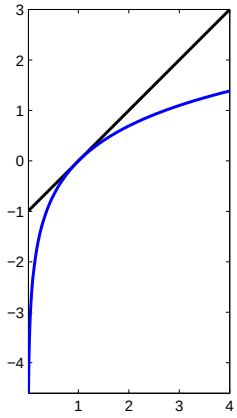
Figure 8.1: Empirical distribution over a discrete variable with 4 states. The empirical samples consist of n samples at each of states $1, 2, 4$ and $2n$ samples at state 3 where $n > 0$. On normalising this gives a distribution with values $0.2, 0.2, 0.4, 0.2$ over the 4 states.

The KL divergence is ≥ 0

The KL divergence is widely used and it is therefore important to understand why the divergence is positive.

To see this, consider the following linear bound on the function $\log(x)$

$$\log(x) \leq x - 1 \tag{8.2.32}$$



as plotted in the figure on the right. Replacing x by $p(x)/q(x)$ in the above bound

$$\frac{p(x)}{q(x)} - 1 \geq \log \frac{p(x)}{q(x)} \tag{8.2.33}$$

Since probabilities are non-negative, we can multiply both sides by $q(x)$ to obtain

$$p(x) - q(x) \geq q(x) \log p(x) - q(x) \log q(x) \tag{8.2.34}$$

We now integrate (or sum in the case of discrete variables) both sides. Using $\int p(x)dx = 1, \int q(x)dx = 1$,

$$1 - 1 \geq \langle \log p(x) - \log q(x) \rangle_{q(x)} \tag{8.2.35}$$

Rearranging gives

$$\langle \log q(x) - \log p(x) \rangle_{q(x)} \equiv \mathrm{KL}(q|p) \geq 0 \tag{8.2.36}$$

The KL divergence is zero if and only if the two distributions are exactly the same.

Definition 8.12 (α-divergence)**.** For two distributions $q(x)$ and $p(x)$ and real α the α-divergence is defined as

$$D_\alpha(p|q) \equiv \frac{1 - \left\langle \frac{p^{\alpha-1}(x)}{q^{\alpha-1}(x)} \right\rangle_{p(x)}}{\alpha(1-\alpha)} \geq 0 \tag{8.2.37}$$

The Kullback-Leibler divergence $\mathrm{KL}(p|q)$ corresponds to $D_1(p|q)$ and $\mathrm{KL}(q|p) = D_0(p|q)$, which is readily verified using L'Hôpital's rule.

8.2.2 Entropy and information

For both discrete and continuous variables, the *entropy* is defined as

$$H(p) \equiv - \langle \log p(x) \rangle_{p(x)} \tag{8.2.38}$$

For continuous variables, this is also called the *differential entropy*, see also exercise(8.34). The entropy is a measure of the uncertainty in a distribution. One way to see this is that

$$H(p) = -\mathrm{KL}(p|u) + \mathsf{const.} \tag{8.2.39}$$

where u is a uniform distribution. Since $\mathrm{KL}(p|u) \geq 0$, the less like a uniform distribution p is, the smaller will be the entropy. Or, vice versa, the more similar p is to a uniform distribution, the greater will be the entropy. Since the uniform distribution contains the least information a priori about which state $p(x)$ is in, the entropy is therefore a measure of the a priori uncertainty in the state occupancy. For a discrete distribution we can permute the state labels without changing the entropy. For a discrete distribution the entropy is positive, whereas the differential entropy can be negative.
The mutual information is a measure of dependence between (sets of) variables $\mathcal{X}$ and $\mathcal{Y}$, conditioned on variables $\mathcal{Z}$.

Definition 8.13 (Mutual Information)**.**

$$\mathrm{MI}(\mathcal{X};\mathcal{Y}|\mathcal{Z}) \equiv \langle \mathrm{KL}(p(\mathcal{X},\mathcal{Y}|\mathcal{Z})|p(\mathcal{X}|\mathcal{Z})p(\mathcal{Y}|\mathcal{Z}))\rangle_{p(\mathcal{Z})} \geq 0 \tag{8.2.40}$$

If $\mathcal{X} \perp\!\!\!\perp \mathcal{Y}|\,\mathcal{Z}$ is true, then $\mathrm{MI}(\mathcal{X};\mathcal{Y}|\mathcal{Z})$ is zero, and vice versa. When $\mathcal{Z} = \emptyset$, the average over $p(\mathcal{Z})$ is absent and one writes $\mathrm{MI}(\mathcal{X};\mathcal{Y})$.

8.3 Classical Distributions

Definition 8.14 (Bernoulli Distribution)**.** The Bernoulli distribution concerns a discrete binary variable x, with $\mathrm{dom}(x) = \{0, 1\}$. The states are not merely symbolic, but real values 0 and 1.

$$p(x = 1) = \theta \tag{8.3.1}$$

From normalisation, it follows that $p(x = 0) = 1 - \theta$. From this

$$\langle x\rangle = 0 \times p(x = 0) + 1 \times p(x = 1) = \theta \tag{8.3.2}$$

The variance is given by $\mathrm{var}(x) = \theta\,(1 - \theta)$.

Definition 8.15 (Categorical Distribution)**.** The categorical distribution generalises the Bernoulli distribution to more than two (symbolic) states. For a discrete variable x, with symbolic states $\mathrm{dom}(x) = \{1, \ldots, C\}$,

$$p(x = c) = \theta_c, \qquad \sum_c \theta_c = 1 \tag{8.3.3}$$

The Dirichlet is conjugate to the categorical distribution.

Definition 8.16 (Binomial Distribution)**.** The Binomial describes the distribution of a discrete two-state variable x, with $\mathrm{dom}(x) = \{\mathsf{1}, \mathsf{0}\}$ where the states are symbolic. The probability that in n Bernoulli Trials (independent samples), $x^1, \ldots, x^n$ there will be k 'success' states $\mathsf{1}$ observed is

$$p(y = k|\theta) = \binom{n}{k}\theta^k\,(1 - \theta)^{n-k}\,, \qquad y \equiv \sum_{i=1}^{n} \mathbb{I}\left[x^i = \mathsf{1}\right] \tag{8.3.4}$$

where $\binom{n}{k} \equiv n!/(k!(n - k)!)$ is the binomial coefficient. The mean and variance are

$$\langle y\rangle = n\theta, \qquad \mathrm{var}(y) = n\theta\,(1 - \theta) \tag{8.3.5}$$

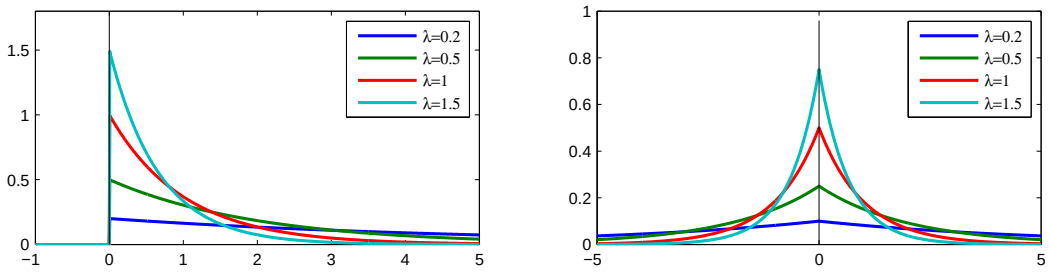
The Beta distribution is the conjugate prior for the Binomial distribution.

Definition 8.17 (Multinomial Distribution)**.** Consider a multi-state variable x, with $\mathrm{dom}(x) = \{1, \ldots, K\}$, with corresponding state probabilities $\theta_1, \ldots, \theta_K$. We then draw n samples from this distribution. The probability of observing the state 1 y_1 times, state 2 y_2 times, ..., state K y_K times in the n samples is

@@ $$p(y_1, \ldots, y_K|\theta) = \frac{n!}{y_1! \ldots y_K!}\prod_{i=1}^{K}\theta_i^{y_i} \tag{8.3.6}$$

@@ where $n = \sum_{i=1}^{K} y_i$.

$$\langle y_i\rangle = n\theta_i, \quad \mathrm{var}(y_i) = n\theta_i\,(1 - \theta_i)\,, \quad \langle y_i y_j\rangle - \langle y_i\rangle\,\langle y_j\rangle = -n\theta_i\theta_j \;\; (i \neq j) \tag{8.3.7}$$



(a) (b)

Figure 8.2: **(a)**: Exponential distribution. **(b)**: Laplace (double exponential) distribution.

The Dirichlet distribution is the conjugate prior for the multinomial distribution.

Definition 8.18 (Poisson Distribution)**.** The Poisson distribution can be used to model situations in which the expected number of events scales with the length of the interval within which the events can occur. If λ is the expected number of events per unit interval, then the distribution of the number of events x within an interval $t\lambda$ is

$$p(x = k|\lambda) = \frac{1}{k!} e^{-\lambda t} (\lambda t)^k, \qquad k = 0, 1, 2, \ldots \tag{8.3.8}$$

For a unit length interval ($t = 1$),

$$\langle x \rangle = \lambda, \quad \text{var}(x) = \lambda \tag{8.3.9}$$

The Poisson distribution can be derived as a limiting case of a Binomial distribution in which the success probability scales as $\theta = \lambda/n$, in the limit $n \to \infty$.

Definition 8.19 (Uniform distribution)**.** For a variable x, the distribution is uniform if $p(x) = \mathsf{const.}$ over the domain of the variable.

Definition 8.20 (Exponential Distribution)**.** For $x \geq 0$, see fig(8.2a),

$$p(x|\lambda) \equiv \lambda e^{-\lambda x} \tag{8.3.10}$$

One can show that for rate λ

$$\langle x \rangle = \frac{1}{\lambda}, \qquad \text{var}(x) = \frac{1}{\lambda^2} \tag{8.3.11}$$

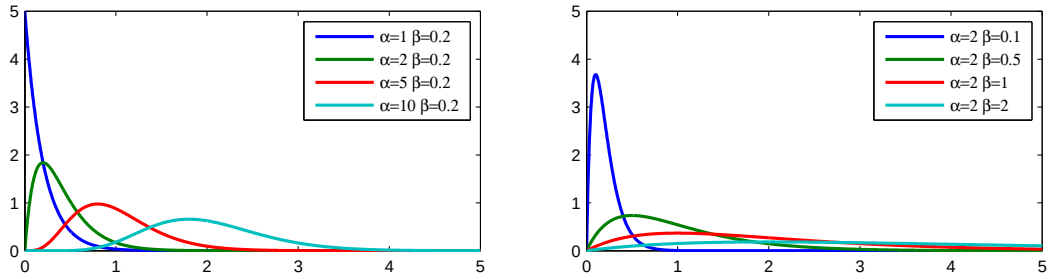
The alternative parameterisation $b = 1/\lambda$ is called the scale.

Definition 8.21 (Gamma Distribution)**.**

$$Gam\,(x|\alpha, \beta) = \frac{1}{\beta\Gamma(\alpha)} \left(\frac{x}{\beta}\right)^{\alpha-1} e^{-\frac{x}{\beta}}, \qquad x \geq 0, \alpha > 0, \beta > 0 \tag{8.3.12}$$

α is called the shape parameter, β is the scale parameter and the Gamma function is defined as

$$\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt \tag{8.3.13}$$



(a) (b)

Figure 8.3: Gamma distribution. **(a)**: varying α for fixed β. **(b)**: varying β for fixed α.

The parameters are related to the mean and variance through

$$\alpha = \left(\frac{\mu}{s}\right)^2, \qquad \beta = \frac{s^2}{\mu} \tag{8.3.14}$$

where μ is the mean of the distribution and s is the standard deviation. The mode is given by $(\alpha - 1)\beta$, for $\alpha \geq 1$, see fig(8.3).

An alternative parameterisation uses the inverse scale

$$Gam^{is}(x|\alpha, \beta) = Gam(x|\alpha, 1/\beta) \propto x^{\alpha-1} e^{-\beta x} \tag{8.3.15}$$

Definition 8.22 (Inverse Gamma distribution)**.**

$$InvGam(x|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{1}{x^{\alpha+1}} e^{-\beta/x} \tag{8.3.16}$$

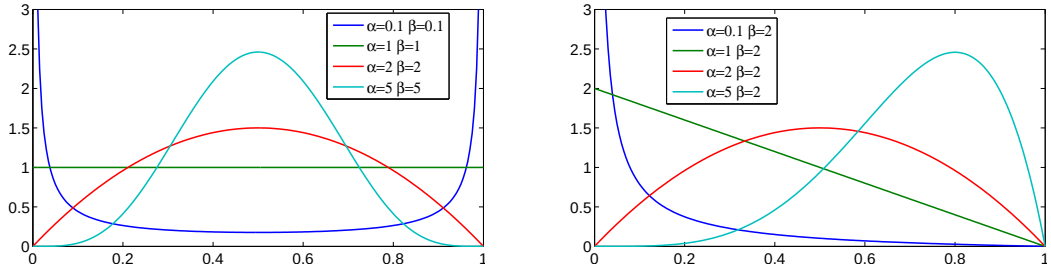
This has mean $\beta/(\alpha - 1)$ for $\alpha > 1$ and variance $\frac{\beta^2}{(\alpha-1)^2(\alpha-2)}$ for $\alpha > 2$.

Definition 8.23 (Beta Distribution)**.**

$$p(x|\alpha, \beta) = B(x|\alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1 - x)^{\beta-1}, \quad 0 \leq x \leq 1 \tag{8.3.17}$$

where the Beta function is defined as

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)} \tag{8.3.18}$$



(a) (b)

Figure 8.4: Beta distribution. The parameters α and β can also be written in terms of the mean and variance, leading to an alternative parameterisation, see exercise(8.16).

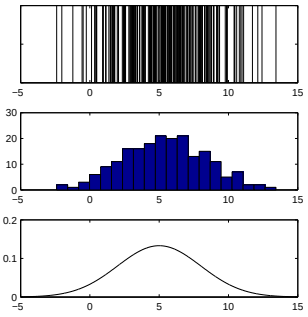


Figure 8.5: Top: 200 datapoints $x^1, \ldots, x^{200}$ drawn from a Gaussian distribution. Each vertical line denotes a datapoint at the corresponding x value on the horizontal axis. Middle: Histogram using 10 equally spaced bins of the datapoints. Bottom: Gaussian distribution $\mathcal{N}(x|\mu = 5, \sigma = 3)$ from which the datapoints were drawn. In the limit of an infinite amount of data, and limitingly small bin size, the normalised histogram tends to the Gaussian probability density function.

and $\Gamma(x)$ is the Gamma function. Note that the distribution can be flipped by interchanging x for $1 - x$, which is equivalent to interchanging α and β. See fig(8.4).

The mean and variance are given by

$$\langle x \rangle = \frac{\alpha}{\alpha + \beta}, \qquad \text{var}(x) = \frac{\alpha\beta}{(\alpha + \beta)^2 (\alpha + \beta + 1)} \tag{8.3.19}$$

Definition 8.24 (Laplace Distribution)**.**

$$p(x|\lambda) \equiv \lambda e^{-\frac{1}{b}|x-\mu|} \tag{8.3.20}$$

For scale b

$$\langle x \rangle = \mu, \qquad \text{var}(x) = 2b^2 \tag{8.3.21}$$

The Laplace distribution is also known as the Double Exponential distribution, fig(8.2b).

Definition 8.25 (Univariate Gaussian Distribution)**.**

$$p(x|\mu, \sigma^2) = \mathcal{N}\left(x|\mu, \sigma^2\right) \equiv \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \tag{8.3.22}$$

where μ is the mean of the distribution, and σ^2 the variance. This is also called the *normal distribution*.

One can show that the parameters indeed correspond to

$$\mu = \langle x \rangle_{\mathcal{N}(x|\mu,\sigma^2)}, \qquad \sigma^2 = \left\langle (x - \mu)^2 \right\rangle_{\mathcal{N}(x|\mu,\sigma^2)} \tag{8.3.23}$$

For $\mu = 0$ and $\sigma = 1$, the Gaussian is called the *standard normal distribution*. See fig(8.5) for a depiction of the univariate Gaussian and samples therefrom.

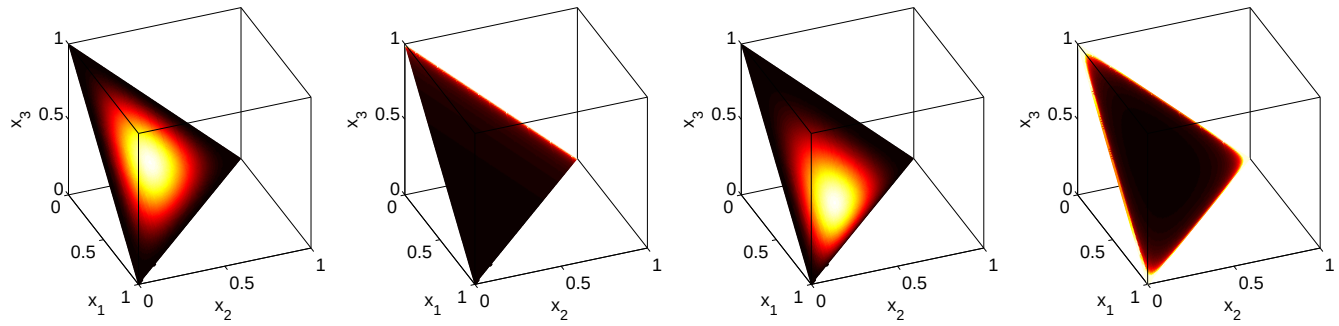
Definition 8.26 (Student's t-distribution)**.**

$$p(x|\mu, \lambda, \nu) = \textit{Student}\,(x|\mu, \lambda, \nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})} \left(\frac{\lambda}{\nu\pi}\right)^{\frac{1}{2}} \left[1 + \frac{\lambda\,(x-\mu)^2}{\nu}\right]^{-\frac{\nu+1}{2}} \tag{8.3.24}$$

where μ is the mean, ν the degrees of freedom, and λ scales the distribution. The variance is given by

$$\text{var}(x) = \frac{\nu}{\lambda\,(\nu - 2)}, \text{ for } \nu > 2 \tag{8.3.25}$$

For $\nu \to \infty$ the distribution tends to a Gaussian with mean μ and variance $1/\lambda$. As ν decreases the tails of the distribution become fatter.



(a) (b) (c) (d)

Figure 8.6: Dirichlet distribution with parameter (u_1, u_2, u_3) displayed on the simplex $x_1, x_2, x_3 \geq 0, x_1 + x_2 + x_3 = 1$. Black denotes low probability and white high probability. **(a)**: $(3, 3, 3)$ **(b)**: $(0.1, 1, 1)$. **(c)**: $(4, 3, 2)$. **(d)**: $(0.05, 0.05, 0.05)$.

The t-distribution can be derived from a *scaled mixture*

$$p(x|\mu, a, b) = \int_{\tau=0}^{\infty} \mathcal{N}\left(x|\mu, \tau^{-1}\right) Gam^{is}\left(\tau|a, b\right) d\tau \tag{8.3.26}$$

$$= \int_{\tau=0}^{\infty} \left(\frac{\tau}{2\pi}\right)^{\frac{1}{2}} e^{-\frac{\tau}{2}(x-\mu)^2} b^a e^{-b\tau} \tau^{a-1} \frac{1}{\Gamma(a)} d\tau \tag{8.3.27}$$

$$= \frac{b^a}{\Gamma(a)} \frac{\Gamma(a+\frac{1}{2})}{\sqrt{2\pi}} \frac{1}{\left(b + \frac{1}{2}(x-\mu)^2\right)^{a+\frac{1}{2}}} \tag{8.3.28}$$

This matches equation (8.3.24) on setting $\nu = 2a$ and $\lambda = a/b$.

Definition 8.27 (Dirichlet Distribution)**.** The Dirichlet distribution is a distribution on probability distributions, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_Q)$, $\alpha_i \geq 0$, $\sum_i \alpha_i = 1$:

$$p(\boldsymbol{\alpha}) = \frac{1}{Z(\mathbf{u})} \delta\left(\sum_{i=1}^{Q} \alpha_i - 1\right) \prod_{q=1}^{Q} \alpha_q^{u_q - 1} \mathbb{I}\left[\alpha_q \geq 0\right] \tag{8.3.29}$$

where

$$Z(\mathbf{u}) = \frac{\prod_{q=1}^{Q} \Gamma(u_q)}{\Gamma\left(\sum_{q=1}^{Q} u_q\right)} \tag{8.3.30}$$

It is conventional to denote the distribution as

$$\text{Dirichlet}\left(\boldsymbol{\alpha}|\mathbf{u}\right) \tag{8.3.31}$$

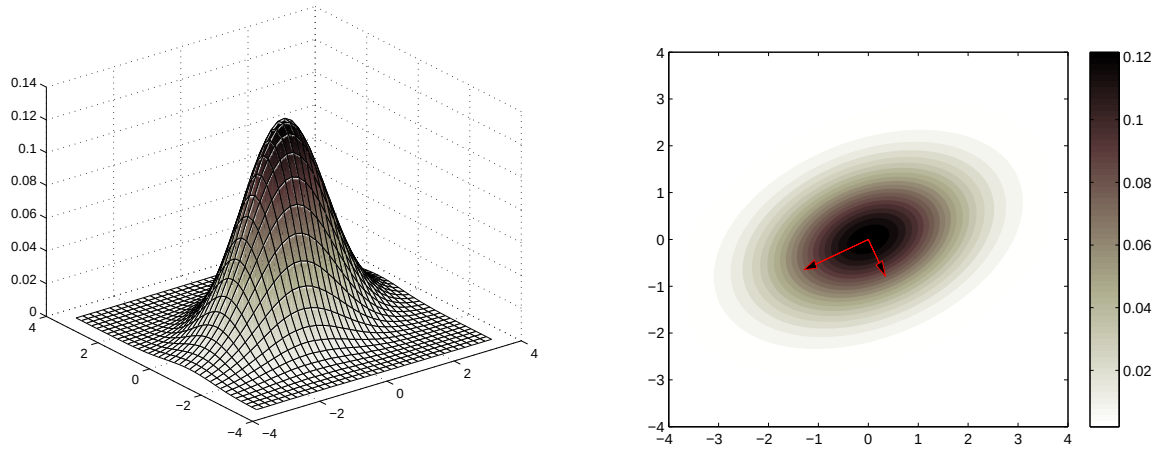
The parameter $\mathbf{u}$ controls how strongly the mass of the distribution is pushed to the corners of the simplex. Setting $u_q = 1$ for all q corresponds to a uniform distribution, fig(8.6). In the binary case $Q = 2$, this is equivalent to a Beta distribution.

The product of two Dirichlet distributions is another Dirichlet distribution

$$\text{Dirichlet}\left(\boldsymbol{\theta}|\mathbf{u}_1\right) \text{Dirichlet}\left(\boldsymbol{\theta}|\mathbf{u}_2\right) = \text{Dirichlet}\left(\boldsymbol{\theta}|\mathbf{u}_1 + \mathbf{u}_2\right) \tag{8.3.32}$$

The marginal of a Dirichlet is also Dirichlet:

$$\int_{\theta_j} \text{Dirichlet}\left(\boldsymbol{\theta}|\mathbf{u}\right) = \text{Dirichlet}\left(\boldsymbol{\theta}_{\setminus j}|\mathbf{u}_{\setminus j}\right) \tag{8.3.33}$$



(a) (b)

Figure 8.7: **(a)**: Bivariate Gaussian with mean $(0, 0)$ and covariance $[1, 0.5; 0.5, 1.75]$. Plotted on the vertical axis is the probability density value $p(x)$. **(b)**: Probability density contours for the same bivariate Gaussian. Plotted are the unit eigenvectors scaled by the square root of their eigenvalues, $\sqrt{\lambda_i}$.

The marginal of a single component θ_i is a Beta distribution:

$$p(\theta_i) = B\left(\theta_i | u_i, \sum_{j \neq i} u_j\right) \tag{8.3.34}$$

8.4 Multivariate Gaussian

The multivariate Gaussian plays a central role in data analysis and as such we discuss its properties in some detail.

Definition 8.28 (Multivariate Gaussian Distribution)**.**

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \equiv \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma})}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \tag{8.4.1}$$

where $\boldsymbol{\mu}$ is the mean vector of the distribution, and $\boldsymbol{\Sigma}$ the covariance matrix. The inverse covariance $\boldsymbol{\Sigma}^{-1}$ is called the *precision*.

One may show

$$\boldsymbol{\mu} = \langle \mathbf{x} \rangle_{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})}, \qquad \boldsymbol{\Sigma} = \left\langle (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\mathsf{T} \right\rangle_{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})} \tag{8.4.2}$$

Note that $\det(\rho\mathbf{M}) = \rho^D \det(\mathbf{M})$, where $\mathbf{M}$ is a $D \times D$ matrix, which explains the dimension independent notation in the normalisation constant of definition(8.28).

The *moment representation* uses $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ to parameterise the Gaussian. The alternative *canonical representation*

$$p(\mathbf{x}|\mathbf{b}, \mathbf{M}, c) = c e^{-\frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{M}\mathbf{x} + \mathbf{x}^\mathsf{T}\mathbf{b}} \tag{8.4.3}$$

is related to the moment representation via

$$\boldsymbol{\Sigma} = \mathbf{M}^{-1}, \qquad \boldsymbol{\mu} = \mathbf{M}^{-1}\mathbf{b}, \qquad \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma})}} = c e^{\frac{1}{2}\mathbf{b}^\mathsf{T}\mathbf{M}^{-1}\mathbf{b}} \tag{8.4.4}$$

The multivariate Gaussian is widely used and it is instructive to understand the geometric picture. This can be achieved by viewing the distribution in a different co-ordinate system. First we use the fact that every real symmetric matrix $D \times D$ has an eigen-decomposition

$$\boldsymbol{\Sigma} = \mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^{\mathsf{T}} \tag{8.4.5}$$

where $\mathbf{E}^{\mathsf{T}}\mathbf{E} = \mathbf{I}$ and $\boldsymbol{\Lambda} = \text{diag}\,(\lambda_1, \ldots, \lambda_D)$. In the case of a covariance matrix, all the eigenvalues λ_i are positive. This means that one can use the transformation

$$\mathbf{y} = \boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{E}^{\mathsf{T}}\left(\mathbf{x} - \boldsymbol{\mu}\right) \tag{8.4.6}$$

so that

$$\left(\mathbf{x} - \boldsymbol{\mu}\right)^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\left(\mathbf{x} - \boldsymbol{\mu}\right) = \left(\mathbf{x} - \boldsymbol{\mu}\right)^{\mathsf{T}}\mathbf{E}\boldsymbol{\Lambda}^{-1}\mathbf{E}^{\mathsf{T}}\left(\mathbf{x} - \boldsymbol{\mu}\right) = \mathbf{y}^{\mathsf{T}}\mathbf{y} \tag{8.4.7}$$

Under this transformation, the multivariate Gaussian reduces to a product of D univariate zero-mean unit variance Gaussians (since the Jacobian of the transformation is a constant). This means that we can view a multivariate Gaussian as a shifted, scaled and rotated version of a 'standard' (zero mean, unit covariance) Gaussian in which the centre is given by the mean, the rotation by the eigenvectors, and the scaling by the square root of the eigenvalues, as depicted in fig(8.7b). A Gaussian with covariance $\boldsymbol{\Sigma} = \rho\mathbf{I}$ for some scalar ρ is an example of an *isotropic* meaning 'same under rotation'. For any isotropic distribution, contours of equal probability are spherical around the origin.

Result 8.2 (Product of two Gaussians)**.** The product of two Gaussians is another Gaussian, with a multiplicative factor, exercise(8.35):

$$\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right)\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\right) = \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)\frac{\exp\left(-\frac{1}{2}\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right)^{\mathsf{T}}\mathbf{S}^{-1}\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right)\right)}{\sqrt{\det\left(2\pi\mathbf{S}\right)}} \tag{8.4.8}$$

where $\mathbf{S} \equiv \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2$ and the mean and covariance are given by

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}_1\mathbf{S}^{-1}\boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_2\mathbf{S}^{-1}\boldsymbol{\mu}_1 \qquad \boldsymbol{\Sigma} = \boldsymbol{\Sigma}_1\mathbf{S}^{-1}\boldsymbol{\Sigma}_2 \tag{8.4.9}$$

8.4.1 Completing the square

A useful technique in manipulating Gaussians is completing the square. For example, the expression

$$\exp\left(-\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x} + \mathbf{b}^{\mathsf{T}}\mathbf{x}\right) \tag{8.4.10}$$

can be transformed as follows. First we complete the square:

$$\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x} - \mathbf{b}^{\mathsf{T}}\mathbf{x} = \frac{1}{2}\left(\mathbf{x} - \mathbf{A}^{-1}\mathbf{b}\right)^{\mathsf{T}}\mathbf{A}\left(\mathbf{x} - \mathbf{A}^{-1}\mathbf{b}\right) - \frac{1}{2}\mathbf{b}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{b} \tag{8.4.11}$$

Hence

@@ $$\exp\left(-\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x} + \mathbf{b}^{\mathsf{T}}\mathbf{x}\right) = \mathcal{N}\left(\mathbf{x}|\mathbf{A}^{-1}\mathbf{b}, \mathbf{A}^{-1}\right)\sqrt{\det\left(2\pi\mathbf{A}^{-1}\right)}\exp\left(\frac{1}{2}\mathbf{b}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{b}\right) \tag{8.4.12}$$

From this one can derive

$$\int \exp\left(-\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x} + \mathbf{b}^{\mathsf{T}}\mathbf{x}\right)d\mathbf{x} = \sqrt{\det\left(2\pi\mathbf{A}^{-1}\right)}\exp\left(\frac{1}{2}\mathbf{b}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{b}\right) \tag{8.4.13}$$

Result 8.3 (Linear Transform of a Gaussian)**.** Let $\mathbf{y}$ be linearly related to $\mathbf{x}$ through

$$\mathbf{y} = \mathbf{M}\mathbf{x} + \boldsymbol{\eta} \tag{8.4.14}$$

where $\mathbf{x} \perp\!\!\!\perp \boldsymbol{\eta}$, $\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$. Then the marginal $p(\mathbf{y}) = \int_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$ is a Gaussian

$$p(\mathbf{y}) = \mathcal{N}\left(\mathbf{y}|\mathbf{M}\boldsymbol{\mu}_x + \boldsymbol{\mu}, \mathbf{M}\boldsymbol{\Sigma}_x\mathbf{M}^\mathsf{T} + \boldsymbol{\Sigma}\right) \tag{8.4.15}$$

Result 8.4 (Partitioned Gaussian)**.** Consider a distribution $\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ defined jointly over two vectors $\mathbf{x}$ and $\mathbf{y}$ of potentially differing dimensions,

$$\mathbf{z} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \tag{8.4.16}$$

with corresponding mean and partitioned covariance

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{pmatrix} \tag{8.4.17}$$

where $\boldsymbol{\Sigma}_{yx} \equiv \boldsymbol{\Sigma}_{xy}^\mathsf{T}$. The marginal distribution is given by

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx}) \tag{8.4.18}$$

and conditional

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(\mathbf{x}\middle|\boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\left(\mathbf{y} - \boldsymbol{\mu}_y\right), \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx}\right) \tag{8.4.19}$$

Result 8.5 (Gaussian average of a quadratic function)**.**

$$\left\langle \mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} \right\rangle_{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})} = \boldsymbol{\mu}^\mathsf{T}\mathbf{A}\boldsymbol{\mu} + \mathrm{trace}\left(\mathbf{A}\boldsymbol{\Sigma}\right) \tag{8.4.20}$$

8.4.2 Conditioning as system reversal

For a joint Gaussian distribution $p(\mathbf{x}, \mathbf{y})$, consider the conditional $p(\mathbf{x}|\mathbf{y})$. The formula for this Gaussian is given in equation (8.4.19). An equivalent and useful way to write this result is to consider a 'reversed' linear system of the form

$$\mathbf{x} = \overleftarrow{\mathbf{A}}\mathbf{y} + \overleftarrow{\boldsymbol{\eta}}, \qquad \text{where} \quad \overleftarrow{\boldsymbol{\eta}} \sim \mathcal{N}\left(\overleftarrow{\boldsymbol{\eta}}\middle|\overleftarrow{\boldsymbol{\mu}}, \overleftarrow{\boldsymbol{\Sigma}}\right) \tag{8.4.21}$$

and show that the marginal over the 'reverse' noise $\overleftarrow{\boldsymbol{\eta}}$ is equivalent to conditioning. That is, for a Gaussian

$$p(\mathbf{x}|\mathbf{y}) = \int \delta\left(\mathbf{x} - \overleftarrow{\mathbf{A}}\mathbf{y} - \overleftarrow{\boldsymbol{\eta}}\right) p(\overleftarrow{\boldsymbol{\eta}}), \qquad p(\overleftarrow{\boldsymbol{\eta}}) = \mathcal{N}\left(\overleftarrow{\boldsymbol{\eta}}\middle|\overleftarrow{\boldsymbol{\mu}}, \overleftarrow{\boldsymbol{\Sigma}}\right) \tag{8.4.22}$$

for suitably defined $\overleftarrow{\mathbf{A}}$, $\overleftarrow{\boldsymbol{\mu}}$, $\overleftarrow{\boldsymbol{\Sigma}}$. To show this, we need to make the statistics of $\mathbf{x}$ under this linear system match those given by the conditioning operation, (8.4.19). The mean and covariance of the linear system equation (8.4.21) are given by

$$\boldsymbol{\mu}_x = \overleftarrow{\mathbf{A}}\mathbf{y} + \overleftarrow{\boldsymbol{\mu}}, \qquad \boldsymbol{\Sigma}_{xx} = \overleftarrow{\boldsymbol{\Sigma}}. \tag{8.4.23}$$

We can make these match equation (8.4.19) by setting

$$\overleftarrow{\mathbf{A}} = \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}, \qquad \overleftarrow{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx}, \qquad \overleftarrow{\boldsymbol{\mu}} = \boldsymbol{\mu}_x - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\mu}_y \tag{8.4.24}$$

This means that we can write an explicit linear system of the form equation (8.4.21) where the parameters are given in terms of the statistics of the original system. This is particularly useful in deriving results in inference with Linear Dynamical Systems, section(24.3).

8.4.3 Whitening and centering

For a set of data $\mathbf{x}^1, \ldots, \mathbf{x}^N$, with $\dim \mathbf{x}^n = D$, we can transform this data to $\mathbf{y}^1, \ldots, \mathbf{y}^N$ with zero mean using *centering*:

$$\mathbf{y}^n = \mathbf{x}^n - \mathbf{m} \tag{8.4.25}$$

where the mean $\mathbf{m}$ of the data is given by

$$\mathbf{m} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}^n \tag{8.4.26}$$

Furthermore, we can transform to values $\mathbf{z}^1, \ldots, \mathbf{z}^N$ that have zero mean and unit covariance using *whitening*

$$\mathbf{z}^n = \mathbf{S}^{-\frac{1}{2}} \left(\mathbf{x}^n - \mathbf{m} \right) \tag{8.4.27}$$

where the covariance $\mathbf{S}$ of the data is given by

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \left(\mathbf{x}^n - \mathbf{m} \right) \left(\mathbf{x}^n - \mathbf{m} \right)^\mathsf{T} \tag{8.4.28}$$

An equivalent approach is to compute the SVD decomposition of the matrix of centered datapoints

$$\mathbf{U}\mathbf{S}\mathbf{V}^\mathsf{T} = \mathbf{Y}, \qquad \mathbf{Y} = \left[\mathbf{y}^1, \ldots, \mathbf{y}^N \right] \tag{8.4.29}$$

then for the $D \times N$ matrix

$$\mathbf{Z} = \sqrt{N} \mathrm{diag} \left(1/S_{1,1}, \ldots, 1/S_{D,D} \right) \mathbf{U}^\mathsf{T} \mathbf{Y} \tag{8.4.30}$$

the columns of $\mathbf{Z} = \left(\mathbf{z}^1, \ldots, \mathbf{z}^N \right)$ have zero mean and unit covariance, see exercise(8.32).

Result 8.6 (Entropy of a Gaussian)**.** The differential entropy of a multivariate Gaussian $p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma} \right)$ is

$$H(\mathbf{x}) \equiv - \left\langle \log p(\mathbf{x}) \right\rangle_{p(\mathbf{x})} = \frac{1}{2} \log \det \left(2\pi \boldsymbol{\Sigma} \right) + \frac{D}{2} \tag{8.4.31}$$

where $D = \dim \mathbf{x}$. Note that the entropy is independent of the mean $\boldsymbol{\mu}$.

8.5 Exponential Family

A theoretically convenient class of distributions are the exponential family, which contains many standard distributions, including the Gaussian, Gamma, Poisson, Dirichlet, Wishart, Multinomial.

Definition 8.29 (Exponential Family)**.** For a distribution on a (possibly multidimensional) variable x (continuous or discrete) an *exponential family* model is of the form

$$p(x|\boldsymbol{\theta}) = h(x) \exp \left(\sum_i \eta_i \left(\boldsymbol{\theta} \right) T_i(x) - \psi(\boldsymbol{\theta}) \right) \tag{8.5.1}$$

$\boldsymbol{\theta}$ are the parameters, $T_i(x)$ the test statistics, and $\psi\left(\boldsymbol{\theta} \right)$ is the log partition function that ensures normalisation

$$\psi\left(\boldsymbol{\theta} \right) = \log \int_x h(x) \exp \left(\sum_i \eta_i \left(\boldsymbol{\theta} \right) T_i(x) \right). \tag{8.5.2}$$

One can always transform the parameters to the form $\boldsymbol{\eta}\left(\boldsymbol{\theta} \right) = \boldsymbol{\theta}$ in which case the distribution is in *canonical form*:

$$p(x|\boldsymbol{\theta}) = h(x) \exp \left(\boldsymbol{\theta}^\mathsf{T} \mathbf{T}(x) - \psi\left(\boldsymbol{\theta} \right) \right) \tag{8.5.3}$$

For example the univariate Gaussian can be written

$$\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right) = \exp\left(-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{\mu^2}{2\sigma^2} - \frac{1}{2}\log 2\pi\sigma^2\right). \tag{8.5.4}$$ @@

Defining $t_1(x) = x$, $t_2(x) = -x^2/2$ and , $\theta_1 = \mu$, $\theta_2 = \sigma^2$, $h(x) = 1$, then

$$\eta_1(\boldsymbol{\theta}) = \frac{\theta_1}{\theta_2}, \qquad \eta_2(\boldsymbol{\theta}) = \frac{1}{\theta_2}, \qquad \psi(\boldsymbol{\theta}) = \frac{1}{2}\left(\frac{\theta_1^2}{\theta_2} + \log 2\pi\theta_2\right) \tag{8.5.5}$$ @@

Note that the parameterisation is not necessarily unique – we can for example rescale the functions $T_i(x)$ and inversely scale η_i by the same amount to arrive at an equivalent representation.

8.5.1 Conjugate priors

For an exponential family likelihood

$$p(x|\boldsymbol{\theta}) = h(x)\exp\left(\boldsymbol{\theta}^\mathsf{T}\mathbf{T}(x) - \psi(\boldsymbol{\theta})\right) \tag{8.5.6}$$

and prior with hyperparameters $\boldsymbol{\alpha}, \gamma$,

$$p(\boldsymbol{\theta}|\boldsymbol{\alpha}, \gamma) \propto \exp\left(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\alpha} - \gamma\psi(\boldsymbol{\theta})\right) \tag{8.5.7}$$

the posterior is

$$p(\boldsymbol{\theta}|x, \boldsymbol{\alpha}, \gamma) \propto p(x|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\alpha}, \gamma) \propto \exp\left(\boldsymbol{\theta}^\mathsf{T}\left[\mathbf{T}(x) + \boldsymbol{\alpha}\right] - \left[\gamma + 1\right]\psi(\boldsymbol{\theta})\right) \tag{8.5.8}$$

$$= p(\boldsymbol{\theta}|\mathbf{T}(x) + \boldsymbol{\alpha}, 1 + \gamma) \tag{8.5.9}$$

so that the prior, equation (8.5.7), is conjugate for the exponential family likelihood equation (8.5.6); that is, the posterior is of the same form as the prior, but with modified hyperparameters. Whilst the likelihood is in the exponential family, the conjugate prior is not necessarily in the exponential family.

8.6 Learning distributions

For a distribution $p(x|\theta)$, parameterised by θ, and data $\mathcal{X} = \{x^1, \ldots, x^N\}$, learning corresponds to inferring the θ that best explains the data $\mathcal{X}$. There are various criteria for defining this:

Bayesian Methods In this one examines the posterior $p(\theta|\mathcal{X}) \propto p(\mathcal{X}|\theta)p(\theta)$. This gives rise to a distribution over θ. The Bayesian method itself says nothing about how to best summarise this posterior.

Maximum A posteriori This is a summarisation of the posterior, that is

$$\theta^{MAP} = \underset{\theta}{\text{argmax}}\; p(\theta|\mathcal{X}) \tag{8.6.1}$$

Maximum Likelihood Under a flat prior, $p(\theta) = \mathsf{const.}$, the MAP solution is equivalent to setting θ to that value that maximises the likelihood of observing the data

$$\theta^{ML} = \underset{\theta}{\text{argmax}}\; p(\mathcal{X}|\theta) \tag{8.6.2}$$

Moment Matching Based on an empirical estimate of a moment (say the mean), θ is set such that the moment (or moments) of the distribution matches the empirical moment.

Pseudo Likelihood For multivariate $\mathbf{x} = (x_1, \ldots, x_D)$, one sets the parameters based on @@

$$\theta = \underset{\theta}{\text{argmax}} \sum_{n=1}^{N}\sum_{i=1}^{D} \log p(x_i^n|x_{\setminus i}^n, \theta) \tag{8.6.3}$$

The pseudo-likelihood method is sometimes used when the full likelihood $p(\mathbf{x}|\theta)$ is difficult to compute.

In seeking the 'best' single parameter θ, we are often required to carry out a numerical optimisation. This is not necessarily a trivial step, and considerable effort is often spent either in attempting to define models for which the resulting computational difficulties are minimal, or in finding good approximations that find useful optima of complex objective functions, see section(A.5).

In this book we focus on the Bayesian methods and maximum likelihood. We first reiterate some of the basic ground covered in section(1.3), in which the Bayesian and maximum likelihood methods are related.

Definition 8.30. Prior, Likelihood and Posterior

For data $\mathcal{X}$ and variable θ, Bayes' rule tells us how to update our prior beliefs about the variable θ in light of the data to a posterior belief:

$$\underbrace{p(\theta|\mathcal{X})}_{\text{posterior}} = \frac{\overbrace{p(\mathcal{X}|\theta)}^{\text{likelihood}}\overbrace{p(\theta)}^{\text{prior}}}{\underbrace{p(\mathcal{X})}_{\text{evidence}}} \tag{8.6.4}$$

The *evidence* is also called the *marginal likelihood*. Note that the term 'evidence' is (rather unfortunately) used for both the marginal likelihood of observations and the observations themselves.

The term 'likelihood' is used for the probability that a model generates observed data. More fully, if we condition on the model M, we have

$$p(\theta|\mathcal{X}, M) = \frac{p(\mathcal{X}|\theta, M)p(\theta|M)}{p(\mathcal{X}|M)}$$

where we see the role of the likelihood $p(\mathcal{X}|\theta, M)$ and *model likelihood* $p(\mathcal{X}|M)$.

The *most probable a posteriori* (*MAP*) setting is that which maximises the posterior,

$$\theta^{MAP} = \underset{\theta}{\text{argmax}}\ p(\theta|\mathcal{X}, M) \tag{8.6.5}$$

For a 'flat prior', $p(\theta|M)$ being a constant, the MAP solution is equivalent to *maximum likelihood* namely that θ that maximises $p(\mathcal{X}|\theta, M)$,

$$\theta^{ML} = \underset{\theta}{\text{argmax}}\ p(\mathcal{X}|\theta, M) \tag{8.6.6}$$

Definition 8.31 (conjugacy)**.** If the posterior is of the same parametric form as the prior, then we call the prior the conjugate distribution for the likelihood distribution. That is, for a prior on parameter θ, with hyperparameter α, $p(\theta|\alpha)$, the posterior given data $\mathcal{D}$ is the same form as the prior, but with updated hyperparameters, $p(\theta|\mathcal{D}, \alpha) = p(\theta|\alpha')$.

Definition 8.32 (Independent and Identically distributed)**.** For a variable x, and a set of i.i.d. observations, $x^1, \ldots, x^N$, conditioned on θ, we assume there is no dependence between the observations

$$p(x^1, \ldots, x^N|\theta) = \prod_{n=1}^{N} p(x^n|\theta) \tag{8.6.7}$$

For non-Bayesian methods which return a single value for θ, based on the data $\mathcal{X}$, it is interesting to know how 'good' the procedure is. Concepts that can help in this case are 'bias' and 'consistency'. The bias

measures if the estimate of θ is correct 'on average'. The property of an estimator such that the parameter θ converges to the true model parameter θ^0 as the sequence of data increases is termed *consistency*.

Definition 8.33 (Unbiased estimator)**.** Given data $\mathcal{X} = \{x^1, \ldots, x^N\}$, formed from i.i.d. samples of a distribution $p(x|\theta)$ we can use the data $\mathcal{X}$ to estimate the parameter θ that was used to generate the data. The estimator is a function of the data, which we write $\hat{\theta}(\mathcal{X})$. For an *unbiased (parameter) estimator*

$$\left\langle \hat{\theta}(\mathcal{X}) \right\rangle_{p(\mathcal{X}|\theta)} = \theta \tag{8.6.8}$$

More generally, one can consider any function of the distribution $p(x)$, with scalar value θ, for example the mean $\theta = \langle x \rangle_{p(x)}$. Then $\hat{\theta}(\mathcal{X})$ is an unbiased estimator of θ with respect to the data distribution $\tilde{p}(\mathcal{X})$ if $\left\langle \hat{\theta}(\mathcal{X}) \right\rangle_{\tilde{p}(\mathcal{X})} = \theta$.

A classical example for estimator bias are those of the mean and variance. Let

$$\hat{\mu}(\mathcal{X}) = \frac{1}{N} \sum_{n=1}^{N} x^n \tag{8.6.9}$$

This is an unbiased estimator of the mean $\langle x \rangle_{p(x)}$ since, for iid data

$$\langle \hat{\mu}(\mathcal{X}) \rangle_{p(\mathcal{X})} = \frac{1}{N} \sum_{n=1}^{N} \langle x^n \rangle_{p(x^n)} = \frac{1}{N} N \langle x \rangle_{p(x)} = \langle x \rangle_{p(x)} \tag{8.6.10}$$

On the other hand, consider the estimator of the variance,

$$\hat{\sigma}^2(\mathcal{X}) = \frac{1}{N} \sum_{n=1}^{N} (x^n - \hat{\mu}(\mathcal{X}))^2 \tag{8.6.11}$$

This is biased since (omitting a few lines of algebra)

$$\left\langle \hat{\sigma}^2(\mathcal{X}) \right\rangle_{p(\mathcal{X})} = \frac{1}{N} \sum_{n=1}^{N} \left\langle (x^n - \hat{\mu}(\mathcal{X}))^2 \right\rangle = \frac{N-1}{N} \sigma^2 \tag{8.6.12}$$

8.7 Properties of Maximum Likelihood

A crude summary of the posterior is given by a distribution with all its mass in a single most likely state, $\delta\left(\theta, \theta^{MAP}\right)$, see definition(8.30) . In making such an approximation, potentially useful information concerning the reliability of the parameter estimate is lost. In contrast the full posterior reflects our beliefs about the range of possibilities and their associated credibilities.

The term 'maximum likelihood' refers to the parameter θ for which the observed data is most likely to be generated by the model. One can motivate MAP from a decision theoretic perspective. If we assume a utility that is zero for all but the correct θ,

$$U(\theta_{true}, \theta) = \mathbb{I}\left[\theta_{true} = \theta\right] \tag{8.7.1}$$

then the expected utility of θ is

$$U(\theta) = \sum_{\theta_{true}} \mathbb{I}\left[\theta_{true} = \theta\right] p(\theta_{true}|\mathcal{X}) = p(\theta|\mathcal{X}) \tag{8.7.2}$$

This means that the maximum utility decision is to return that θ with the highest posterior value.

When a 'flat' prior $p(\theta) = \text{const.}$ is used the MAP parameter assignment is equivalent to the maximum likelihood setting

$$\theta^{ML} = \underset{\theta}{\text{argmax}}\ p(\mathcal{X}|\theta) \tag{8.7.3}$$

Since the logarithm is a strictly increasing function, then for a positive function $f(\theta)$

$$\theta_{opt} = \underset{\theta}{\text{argmax}}\ f(\theta) \Leftrightarrow \theta_{opt} = \underset{\theta}{\text{argmax}}\ \log f(\theta) \tag{8.7.4}$$

so that the MAP parameters can be found either by optimising the MAP objective or, equivalently, its logarithm,

$$\log p(\theta|\mathcal{X}) = \log p(\mathcal{X}|\theta) + \log p(\theta) - \log p(\mathcal{X}) \tag{8.7.5}$$

where the normalisation constant, $p(\mathcal{X})$, is not a function of θ. The log likelihood is convenient since, under the i.i.d. assumption, it is a summation of data terms,

$$\log p(\theta|\mathcal{X}) = \sum_n \log p(x^n|\theta) + \log p(\theta) - \log p(\mathcal{X}) \tag{8.7.6}$$

so that quantities such as derivatives of the log-likelihood *w.r.t.* θ are straightforward to compute.

8.7.1 Training assuming the correct model class

Consider a dataset $\mathcal{X} = \{x^n, n = 1, \ldots, N\}$ generated from an underlying parametric model $p(x|\theta^0)$. Our interest is to fit a model $p(x|\theta)$ of the same form as the correct underlying model $p(x|\theta^0)$ and examine if, in the limit of a large amount of data, the parameter θ learned by maximum likelihood matches the correct parameter θ^0. Our derivation below is non-rigorous, but highlights the essence of the argument.

Assuming the data is i.i.d., the scaled log likelihood is

$$L(\theta) \equiv \frac{1}{N} \log p(\mathcal{X}|\theta) = \frac{1}{N} \sum_{n=1}^{N} \log p(x^n|\theta) \tag{8.7.7}$$

In the limit $N \to \infty$, the sample average can be replaced by an average with respect to the distribution generating the data

$$L(\theta) \overset{N\to\infty}{=} \langle \log p(x|\theta) \rangle_{p(x|\theta^0)} = -\text{KL}\big(p(x|\theta^0)|p(x|\theta)\big) + \big\langle \log p(x|\theta^0) \big\rangle_{p(x|\theta^0)} \tag{8.7.8}$$

Up to a negligible constant, this is the Kullback-Leibler divergence between two distributions in x, just with different parameter settings. The θ that maximises $L(\theta)$ is that which minimises the Kullback-Leibler divergence, namely $\theta = \theta^0$. In the limit of a large amount of data we can therefore, in principle, learn the correct parameters (assuming we know the correct model class). That is, maximum likelihood is a consistent estimator.

8.7.2 Training when the assumed model is incorrect

We write $q(x|\theta)$ for the assumed model, and $p(x|\phi)$ for the correct generating model. Repeating the above calculations in the case of the assumed model being correct, we have that, in the limit of a large amount of data, the scaled log likelihood is

$$L(\theta) = \langle \log q(x|\theta) \rangle_{p(x|\phi)} = -\text{KL}(p(x|\phi)|q(x|\theta)) + \langle \log p(x|\phi) \rangle_{p(x|\phi)} \tag{8.7.9}$$

Since q and p are not of the same form, setting θ to ϕ does not necessarily minimise $\text{KL}(p(x|\phi)|q(x|\theta))$, and therefore does not necessarily optimize $L(\theta)$.

8.7.3 Maximum likelihood and the empirical distribution

Given a dataset of discrete variables $\mathcal{X} = \{x^1, \ldots, x^N\}$ we define the empirical distribution as

$$q(x) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}[x = x^n] \tag{8.7.10}$$

in the case that x is a vector of variables,

$$\mathbb{I}[x = x^n] = \prod_i \mathbb{I}[x_i = x_i^n] \tag{8.7.11}$$

The Kullback-Leibler divergence between the empirical distribution $q(x)$ and a distribution $p(x)$ is

$$\mathrm{KL}(q|p) = \langle \log q(x) \rangle_{q(x)} - \langle \log p(x) \rangle_{q(x)} \tag{8.7.12}$$

Our interest is the functional dependence of $\mathrm{KL}(q|p)$ on p. Since the entropic term $\langle \log q(x) \rangle_{q(x)}$ is independent of $p(x)$ we may consider this constant and focus on the second term alone. Hence

$$\mathrm{KL}(q|p) = -\langle \log p(x) \rangle_{q(x)} + \mathsf{const.} = -\frac{1}{N} \sum_{n=1}^{N} \log p(x^n) + \mathsf{const.} \tag{8.7.13}$$

We recognise $\sum_{n=1}^{N} \log p(x^n)$ as the log likelihood under the model $p(x)$, assuming that the data is i.i.d. This means that setting parameters by maximum likelihood is equivalent to setting parameters by minimising the Kullback-Leibler divergence between the empirical distribution and the parameterised distribution. In the case that $p(x)$ is unconstrained, the optimal choice is to set $p(x) = q(x)$, namely the maximum likelihood optimal distribution corresponds to the empirical distribution.

8.8 Learning a Gaussian

Given the importance of the Gaussian distribution, it is instructive to explicitly consider maximum likelihood and Bayesian methods for fitting a Gaussian to data.

8.8.1 Maximum likelihood training

Given a set of training data $\mathcal{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$, drawn from a Gaussian $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with unknown mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, how can we find these parameters? Assuming the data are drawn i.i.d. the log likelihood is

$$L(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \equiv \sum_{n=1}^{N} \log p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{1}{2} \sum_{n=1}^{N} (\mathbf{x}^n - \boldsymbol{\mu})^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\mathbf{x}^n - \boldsymbol{\mu}) - \frac{N}{2} \log \det(2\pi\boldsymbol{\Sigma}) \tag{8.8.1}$$

Optimal $\boldsymbol{\mu}$

Taking the partial derivative with respect to the vector $\boldsymbol{\mu}$ we obtain the vector derivative

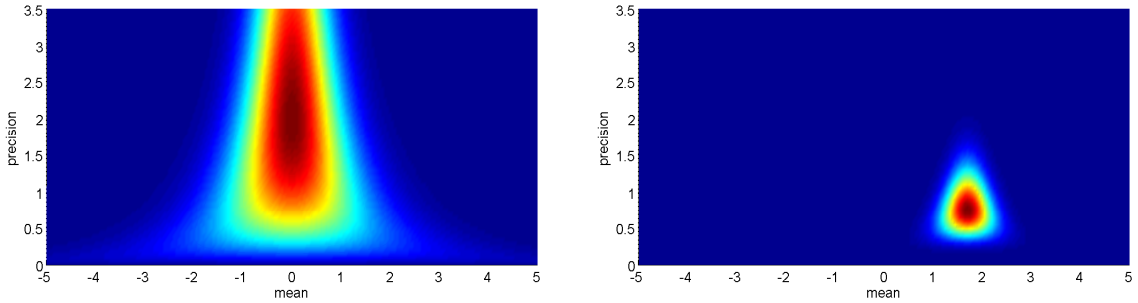
$$\nabla_{\boldsymbol{\mu}} L(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \boldsymbol{\Sigma}^{-1} (\mathbf{x}^n - \boldsymbol{\mu}) \tag{8.8.2}$$

Equating to zero gives that at the optimum of the log likelihood,

$$\sum_{n=1}^{N} \boldsymbol{\Sigma}^{-1} \mathbf{x}^n = N \boldsymbol{\mu} \boldsymbol{\Sigma}^{-1} \tag{8.8.3}$$

and therefore, optimally $\boldsymbol{\mu}$ is given by the sample mean

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}^n \tag{8.8.4}$$



(a) Prior (b) Posterior

Figure 8.8: Bayesian approach to inferring the mean and precision (inverse variance) of a Gaussian based on $N = 10$ randomly drawn datapoints. **(a)**: A Gauss-Gamma prior with $\mu_0 = 0$, $\alpha = 2$, $\beta = 1$, $\gamma = 1$. **(b)**: Gauss-Gamma posterior conditional on the data. For comparison, the sample mean of the data is 1.87 and maximum likelihood optimal variance is 1.16 (computed using the N normalisation). The 10 datapoints were drawn from a Gaussian with mean 2 and variance 1. See `demoGaussBayes.m`.

Optimal $\boldsymbol{\Sigma}$

The derivative of L with respect to the matrix $\boldsymbol{\Sigma}$ requires more work. It is convenient to isolate the dependence on the covariance, and also parameterise using the inverse covariance, $\boldsymbol{\Sigma}^{-1}$,

$$L = -\frac{1}{2}\text{trace}\left(\boldsymbol{\Sigma}^{-1}\underbrace{\sum_{n=1}^{N}\left(\mathbf{x}^n - \boldsymbol{\mu}\right)\left(\mathbf{x}^n - \boldsymbol{\mu}\right)^{\mathsf{T}}}_{\equiv \mathbf{M}}\right) + \frac{N}{2}\log\det\left(2\pi\boldsymbol{\Sigma}^{-1}\right) \tag{8.8.5}$$

Using $\mathbf{M} = \mathbf{M}^{\mathsf{T}}$, we obtain

$$\frac{\partial}{\partial \boldsymbol{\Sigma}^{-1}} L = -\frac{1}{2}\mathbf{M} + \frac{N}{2}\boldsymbol{\Sigma} \tag{8.8.6}$$

Equating the derivative to the zero matrix and solving for $\boldsymbol{\Sigma}$ gives the sample covariance

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{x}^n - \boldsymbol{\mu}\right)\left(\mathbf{x}^n - \boldsymbol{\mu}\right)^{\mathsf{T}} \tag{8.8.7}$$

Equations (8.8.4) and (8.8.7) define the maximum likelihood solution mean and covariance for training data $\mathcal{X}$. Consistent with our previous results, in fact these equations simply set the parameters to their sample statistics of the empirical distribution. That is, the mean is set to the sample mean of the data and the covariance to the sample covariance.

8.8.2 Bayesian inference of the mean and variance

For simplicity we here deal only with the univariate case. Assuming i.i.d. data the likelihood is

$$p(\mathcal{X}|\mu, \sigma^2) = \frac{1}{\left(2\pi\sigma^2\right)^{N/2}}\exp\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N}\left(x^n - \mu\right)^2\right) \tag{8.8.8}$$

For a Bayesian treatment, we require the posterior of the parameters

$$p(\mu, \sigma^2|\mathcal{X}) \propto p(\mathcal{X}|\mu, \sigma^2)p(\mu, \sigma^2) = p(\mathcal{X}|\mu, \sigma^2)p(\mu|\sigma^2)p(\sigma^2) \tag{8.8.9}$$

Our aim is to find conjugate priors for the mean and variance. A convenient choice for a prior on the mean μ is that it is a Gaussian centred on μ_0:

$$p(\mu|\mu_0, \sigma_0^2) = \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(-\frac{1}{2\sigma_0^2}\left(\mu_0 - \mu\right)^2\right) \tag{8.8.10}$$

The posterior is then

$$p(\mu,\sigma^2|\mathcal{X}) \propto \frac{1}{\sigma_0}\frac{1}{\sigma^N}\exp\left(-\frac{1}{2\sigma_0^2}(\mu_0-\mu)^2 - \frac{1}{2\sigma^2}\sum_n (x^n-\mu)^2\right)p(\sigma^2) \tag{8.8.11}$$

It is convenient to write this in the form

$$p(\mu,\sigma^2|\mathcal{X}) = p(\mu|\sigma^2,\mathcal{X})p(\sigma^2|\mathcal{X}) \tag{8.8.12}$$

Since equation (8.8.11) has quadratic contributions in μ in the exponent, the conditional posterior $p(\mu|\sigma^2,\mathcal{X})$ is Gaussian. To identify this Gaussian we multiply out the terms in the exponent to arrive at

$$\exp\left(-\frac{1}{2}\left(a\mu^2 - 2b\mu + c\right)\right) \tag{8.8.13}$$

with

$$a = \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2}, \quad b = \frac{\mu_0}{\sigma_0^2} + \frac{\sum_n x^n}{\sigma^2}, \quad c = \frac{\mu_0^2}{\sigma_0^2} + \sum_n \frac{(x^n)^2}{\sigma^2} \tag{8.8.14}$$

Using the identity

$$a\mu^2 - 2b\mu + c = a\left(\mu - \frac{b}{a}\right)^2 + \left(c - \frac{b^2}{a}\right) \tag{8.8.15}$$

we can write

$$p(\mu,\sigma^2|\mathcal{X}) \propto \underbrace{\sqrt{a}\exp\left(-\frac{1}{2}a\left(\mu-\frac{b}{a}\right)^2\right)}_{p(\mu|\mathcal{X},\sigma^2)}\underbrace{\frac{1}{\sqrt{a}}\exp\left(-\frac{1}{2}\left(c-\frac{b^2}{a}\right)\right)\frac{1}{\sigma_0}\frac{1}{\sigma^N}p(\sigma^2)}_{p(\sigma^2|\mathcal{X})} \tag{8.8.16}$$

We encounter a difficulty in attempting to find a conjugate prior for σ^2 because the term b^2/a is not a simple expression of σ^2. For this reason we constrain

$$\sigma_0^2 \equiv \gamma\sigma^2 \tag{8.8.17}$$

for some fixed hyperparameter γ. Defining the constants

$$\tilde{a} = \frac{1}{\gamma} + N, \quad \tilde{b} = \frac{\mu_0}{\gamma} + \sum_n x^n, \quad \tilde{c} = \frac{\mu_0^2}{\gamma} + \sum_n (x^n)^2 \tag{8.8.18}$$

we have

$$c - \frac{b^2}{a} = \frac{1}{\sigma^2}\left(\tilde{c} - \frac{\tilde{b}^2}{\tilde{a}}\right) \tag{8.8.19}$$

Using this expression in equation (8.8.16) we obtain

$$p(\sigma^2|\mathcal{X}) \propto \left(\sigma^2\right)^{-N/2}\exp\left(-\frac{1}{2\sigma^2}\left(\tilde{c} - \frac{\tilde{b}^2}{\tilde{a}}\right)\right)p(\sigma^2) \tag{8.8.20}$$

An inverse Gamma distribution for the prior $p(\sigma^2)$ is therefore conjugate. For a Gauss-Inverse-Gamma prior:

$$p(\mu,\sigma^2) = \mathcal{N}\left(\mu|\mu_0,\gamma\sigma^2\right) InvGam\left(\sigma^2|\alpha,\beta\right) \tag{8.8.21}$$

the posterior is also *Gauss-Inverse-Gamma* with

$$p(\mu,\sigma^2|\mathcal{X}) = \mathcal{N}\left(\mu\Big|\frac{\tilde{b}}{\tilde{a}},\frac{\sigma^2}{\tilde{a}}\right) InvGam\left(\sigma^2|\alpha+\frac{N}{2},\beta+\frac{1}{2}\left(\tilde{c}-\frac{\tilde{b}^2}{\tilde{a}}\right)\right) \tag{8.8.22}$$

8.8.3 Gauss-Gamma distribution

It is common to use a prior on the *precision*, defined as the inverse variance

$$\lambda \equiv \frac{1}{\sigma^2} \tag{8.8.23}$$

If we then use a Gamma prior

$$p(\lambda|\alpha,\beta) = Gam(\lambda|\alpha,\beta) = \frac{1}{\beta^\alpha \Gamma(\alpha)} \lambda^{\alpha-1} e^{-\lambda/\beta} \tag{8.8.24}$$

the posterior will be

$$p(\lambda|\mathcal{X},\alpha,\beta) = Gam\left(\lambda|\alpha + N/2, \tilde{\beta}\right) \tag{8.8.25}$$

where

$$\frac{1}{\tilde{\beta}} = \frac{1}{\beta} + \frac{1}{2}\left(\tilde{c} - \frac{\tilde{b}^2}{\tilde{a}}\right) \tag{8.8.26}$$

The *Gauss-Gamma* prior distribution

$$p(\mu,\lambda|\mu_0,\alpha,\beta,\gamma) = \mathcal{N}\left(\mu|\mu_0,\gamma\lambda^{-1}\right) Gam(\lambda|\alpha,\beta) \tag{8.8.27}$$

is therefore the conjugate prior for a Gaussian with unknown mean μ and precision λ. The posterior for this prior is a Gauss-Gamma distribution with parameters

$$p(\mu,\lambda|\mathcal{X},\mu_0,\alpha,\beta,\gamma) = \mathcal{N}\left(\mu\Big|\frac{\tilde{b}}{\tilde{a}}, \frac{1}{\tilde{a}\lambda}\right) Gam\left(\lambda|\alpha + N/2, \tilde{\beta}\right) \tag{8.8.28}$$

The marginal $p(\mu|\mathcal{X},\mu_0,\alpha,\beta,\gamma)$ is a Student's t-distribution. An example of a Gauss-Gamma prior/posterior is given in fig(8.8). The maximum likelihood solution is recovered in the limit of a 'flat' prior $\mu_0 = 0, \gamma \to \infty, \alpha = 1/2, \beta \to \infty$, see exercise(8.23). The unbiased estimators for the mean and variance are given using the prior $\mu_0 = 0, \gamma \to \infty, \alpha = 1, \beta \to \infty$, exercise(8.24).

For the multivariate case, the extension of these techniques uses a multivariate Gaussian distribution for the conjugate prior on the mean, and an Inverse Wishart distribution for the conjugate prior on the covariance[137].

8.9 Summary

- Classical univariate distributions include the exponential, Gamma, Beta, Gaussian and Poisson.
- A classical distribution of distributions is the Dirichlet distribution.
- Multivariate distributions are often difficult to deal with computationally. A special case is the multivariate Gaussian, for which marginals and normalisation constants can be computed in time cubic in the number of variables in the model.
- A useful measure of the difference between distributions is the Kullback-Leibler divergence.
- Bayes' rule enables us to achieve parameter learning by translating a priori parameter belief into a posterior parameter belief based on observed data.
- Bayes' rule itself says nothing about how best to summarise the posterior distribution.
- Conjugate distributions are such that the prior and posterior are from the same distribution, just with different parameters.
- Maximum likelihood corresponds to a simple summarisation of the posterior under a flat prior.

- Provided that we are using the correct model class, maximum likelihood will learn the optimal parameters in the limit of a large amount of data – otherwise there is no such guarantee.

8.10 Code

`demoGaussBayes.m`: Bayesian fitting of a univariate Gaussian
`logGaussGamma.m`: Plotting routine for a Gauss-Gamma distribution

8.11 Exercises

Exercise 8.1. *In a public lecture, the following phrase was uttered by a Professor of Experimental Psychology: 'In a recent data survey, 90% of people claim to have above average intelligence, which is clearly nonsense!' [Audience Laughs]. Is it theoretically possible for 90% of people to have above average intelligence? If so, give an example, otherwise explain why not. What about above median intelligence?*

Exercise 8.2. *Consider the distribution defined on real variables x, y:*

$$p(x,y) \propto (x^2+y^2)^2 e^{-x^2-y^2}, \qquad dom(x) = dom(y) = \{-\infty \dots \infty\} \tag{8.11.1}$$

Show that $\langle x \rangle = \langle y \rangle = 0$. Furthermore show that x and y are uncorrelated, $\langle xy \rangle = \langle x \rangle \langle y \rangle$. Whilst x and y are uncorrelated, show that they are nevertheless dependent.

Exercise 8.3. *For a variable x with $dom(x) = \{0, 1\}$, and $p(x = 1) = \theta$, show that in n independent draws $x_1, \dots, x_n$ from this distribution, the probability of observing k states 1 is the Binomial distribution*

$$\binom{n}{k} \theta^k (1-\theta)^{n-k} \tag{8.11.2}$$

Exercise 8.4 (Normalisation constant of a Gaussian)**.** *The normalisation constant of a Gaussian distribution is related to the integral*

$$I = \int_{-\infty}^{\infty} e^{-\frac{1}{2}x^2} dx \tag{8.11.3}$$

By considering

$$I^2 = \int_{-\infty}^{\infty} e^{-\frac{1}{2}x^2} dx \int_{-\infty}^{\infty} e^{-\frac{1}{2}y^2} dy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\frac{1}{2}(x^2+y^2)} dxdy \tag{8.11.4}$$

and transforming to polar coordinates,

$$x = r\cos\theta, \quad y = r\sin\theta, \quad dxdy \to rdrd\theta, \quad r = 0, \dots, \infty, \quad \theta = 0, \dots, 2\pi$$

show that

1. $I = \sqrt{2\pi}$
2. $\int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} dx = \sqrt{2\pi\sigma^2}$

Exercise 8.5. *For a univariate Gaussian distribution, show that*

1. $\mu = \langle x \rangle_{\mathcal{N}(x|\mu,\sigma^2)}$
2. $\sigma^2 = \left\langle (x-\mu)^2 \right\rangle_{\mathcal{N}(x|\mu,\sigma^2)}$

Exercise 8.6. *Using*

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = \text{trace}\left(\mathbf{A}\mathbf{x}\mathbf{x}^T\right) \tag{8.11.5}$$

derive result(8.5).

Exercise 8.7. *Show that the marginal of a Dirichlet distribution is another Dirichlet distribution:*

$$\int_{\theta_j} \text{Dirichlet}\left(\theta|\mathbf{u}\right) = \text{Dirichlet}\left(\theta_{\setminus j}|\mathbf{u}_{\setminus j}\right) \tag{8.11.6}$$

Exercise 8.8. *For a Beta distribution, show that*

$$\left\langle x^k \right\rangle = \frac{B(\alpha+k,\beta)}{B(\alpha,\beta)} = \frac{(\alpha+k-1)(\alpha+k-2)\ldots(\alpha)}{(\alpha+\beta+k-1)(\alpha+\beta+k)\ldots(\alpha+\beta)} \tag{8.11.7}$$

where we used $\Gamma(x+1) = x\Gamma(x)$.

Exercise 8.9. *For the moment generating function* $g(t) \equiv \left\langle e^{tx} \right\rangle_{p(x)}$*, show that*

$$\lim_{t\to 0} \frac{d^k}{dt^k} g(t) = \left\langle x^k \right\rangle_{p(x)} \tag{8.11.8}$$

Exercise 8.10 (Change of variables)**.** *Consider a one dimensional continuous random variable* x *with corresponding* $p(x)$*. For a variable* $y = f(x)$*, where* $f(x)$ *is a monotonic function, show that the distribution of* y *is*

$$p(y) = p(x)\left(\frac{df}{dx}\right)^{-1}, \quad x = f^{-1}(y) \tag{8.11.9}$$

More generally, for vector variables, and $\mathbf{y} = \mathbf{f}(\mathbf{x})$*, we have:*

$$p(\mathbf{y}) = p(\mathbf{x} = \mathbf{f}^{-1}(\mathbf{y}))\left|\det\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\right|^{-1} \tag{8.11.10}$$

where the Jacobian matrix has elements

$$\left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]_{ij} = \frac{\partial f_i(\mathbf{x})}{\partial x_j} \tag{8.11.11}$$

Exercise 8.11 (Normalisation of a Multivariate Gaussian)**.** *Consider*

$$I = \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}\right)\right) d\mathbf{x} \tag{8.11.12}$$

By using the transformation

$$\mathbf{z} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\left(\mathbf{x}-\boldsymbol{\mu}\right) \tag{8.11.13}$$

show that

$$I = \sqrt{\det\left(2\pi\boldsymbol{\Sigma}\right)} \tag{8.11.14}$$

Exercise 8.12. *Consider the partitioned matrix*

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \tag{8.11.15}$$

for which we wish to find the inverse $\mathbf{M}^{-1}$*. We assume that* $\mathbf{A}$ *is* $m \times m$ *and invertible, and* $\mathbf{D}$ *is* $n \times n$ *and invertible. By definition, the partitioned inverse*

$$\mathbf{M}^{-1} = \begin{pmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{pmatrix} \tag{8.11.16}$$

must satisfy

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{pmatrix} = \begin{pmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_n \end{pmatrix} \tag{8.11.17}$$

where $\mathbf{I}_m$ *is the* $m \times m$ *identity matrix, and* $\mathbf{0}$ *the zero matrix of the same dimension as* D*. Using the above, derive the results*

$$\begin{array}{ll} \mathbf{P} = \left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} & \mathbf{Q} = -\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1} \\ \mathbf{R} = -\mathbf{D}^{-1}\mathbf{C}\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} & \mathbf{S} = \left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1} \end{array} \tag{8.11.18}$$

Exercise 8.13. *Show that for Gaussian distribution* $p(x) = \mathcal{N}\left(x|\mu, \sigma^2\right)$ *the skewness and kurtosis are both zero.*

Exercise 8.14. *Consider a small interval of time* δt *and let the probability of an event occurring in this small interval be* $\theta\delta t$*. Derive a distribution that expresses the probability of at least one event in an interval from 0 to* t*.*

Exercise 8.15. *Consider a vector variable* $\mathbf{x} = (x_1, \ldots, x_n)^\mathsf{T}$ *and set of functions defined on each component of* x*,* $\phi_i(x_i)$*. For example for* $\mathbf{x} = (x_1, x_2)^\mathsf{T}$ *we might have*

$$\phi_1(x_1) = -|x_1|, \quad \phi_2(x_2) = -x_2^2 \tag{8.11.19}$$

Consider the distribution

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z}\exp\left(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\phi}(\mathbf{x})\right) \tag{8.11.20}$$

where $\phi(\mathbf{x})$ *is a vector function with* i^{th} *component* $\phi_i(x_i)$*, and* $\boldsymbol{\theta}$ *is a parameter vector. Each component is tractably integrable in the sense that*

$$\int_{-\infty}^{\infty} \exp\left(\theta_i \phi_i(x_i)\right) dx_i \tag{8.11.21}$$

can be computed either analytically or to an acceptable numerical accuracy. Show that

1. $x_i \perp\!\!\!\perp x_j$.
2. *The normalisation constant* Z *can be tractably computed.*
3. *Consider the transformation*

$$\mathbf{x} = \mathbf{M}\mathbf{y} \tag{8.11.22}$$

for an invertible matrix $\mathbf{M}$*. Show that the distribution* $p(\mathbf{y}|\mathbf{M}, \boldsymbol{\theta})$ *is tractable (its normalisation constant is known), and that, in general,* $y_i \not\perp\!\!\!\perp y_j$*. Explain the significance of this in deriving tractable multivariate distributions.*

Exercise 8.16. *Show that we may reparameterise the Beta distribution, definition(8.23) by writing the parameters* α *and* β *as functions of the mean* m *and variance* s *using*

$$\alpha = \beta\gamma, \qquad \gamma \equiv m/(1-m) \tag{8.11.23}$$

$$\beta = \frac{1}{1+\gamma}\left(\frac{\gamma}{s\left(1+\gamma\right)^2} - 1\right) \tag{8.11.24}$$

Exercise 8.17. *Consider the function*

$$f(\gamma + \alpha, \beta, \theta) \equiv \theta^{\gamma+\alpha-1}\left(1-\theta\right)^{\beta-1} \tag{8.11.25}$$

show that

$$\lim_{\gamma\to 0}\frac{\partial}{\partial\gamma}f(\gamma+\alpha,\beta,\theta)=\theta^{\alpha-1}\left(1-\theta\right)^{\beta-1}\log\theta \tag{8.11.26}$$

and hence that

$$\int\theta^{\alpha-1}\left(1-\theta\right)^{\beta-1}\log\theta d\theta=\lim_{\gamma\to 0}\frac{\partial}{\partial\gamma}\int f(\gamma+\alpha,\beta,\theta)d\theta=\frac{\partial}{\partial\alpha}\int f(\alpha,\beta,\theta)d\theta \tag{8.11.27}$$

Using this result, show therefore that

$$\left\langle\log\theta\right\rangle_{B(\theta|\alpha,\beta)}=\frac{\partial}{\partial\alpha}\log B(\alpha,\beta) \tag{8.11.28}$$

where $B(\alpha,\beta)$ is the Beta function. Show additionally that

$$\left\langle\log\left(1-\theta\right)\right\rangle_{B(\theta|\alpha,\beta)}=\frac{\partial}{\partial\beta}\log B(\alpha,\beta) \tag{8.11.29}$$

Using the fact that

$$B(\alpha,\beta)=\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{8.11.30}$$

where $\Gamma(x)$ is the gamma function, relate the above averages to the digamma function*, defined as*

$$\psi(x)=\frac{d}{dx}\log\Gamma(x) \tag{8.11.31}$$

Exercise 8.18. *Using a similar 'generating function' approach as in exercise(8.17), explain how to compute*

$$\left\langle\log\theta_i\right\rangle_{\text{Dirichlet}(\boldsymbol{\theta}|\mathbf{u})} \tag{8.11.32}$$

Exercise 8.19. *Consider the function*

$$f(x)=\int_0^\infty\delta\left(\sum_{i=1}^n\theta_i-x\right)\prod_i\theta_i^{u_i-1}d\theta_1\ldots d\theta_n \tag{8.11.33}$$

Show that the Laplace transform $\tilde{f}(s)\equiv\int_0^\infty e^{-sx}f(x)dx$ is

$$\tilde{f}(s)=\prod_{i=1}^n\left\{\int_0^\infty e^{-s\theta_i}\theta_i^{u_i-1}d\theta_i\right\}=\frac{1}{s^{\sum_i u_i}}\prod_{i=1}^n\Gamma\left(u_i\right) \tag{8.11.34}$$

By using that the inverse Laplace transform of $1/s^{1+q}$ is $x^q/\Gamma(1+q)$, show that

$$f(x)=\frac{\prod_{i=1}^n\Gamma\left(u_i\right)}{\Gamma\left(\sum_i u_i\right)}x^{\sum_i u_i} \tag{8.11.35}$$

Hence show that the normalisation constant of a Dirichlet distribution with parameters $\mathbf{u}$ is given by

$$\frac{\prod_{i=1}^n\Gamma\left(u_i\right)}{\Gamma\left(\sum_i u_i\right)} \tag{8.11.36}$$

Exercise 8.20. *Derive the formula for the differential entropy of a multi-variate Gaussian.*

Exercise 8.21. *Show that for a gamma distribution $Gam\left(x|\alpha,\beta\right)$ the mode is given by*

$$x^*=\left(\alpha-1\right)\beta \tag{8.11.37}$$

provided that $\alpha\geq 1$.

Exercise 8.22. *Consider a distribution $p(x|\theta)$ and a distribution $p(x|\theta+\delta)$ for small δ.*

1. *Take the Taylor expansion of*

$$KL(p(x|\theta)|p(x|\theta+\delta)) \tag{8.11.38}$$

for small δ and show that this is equal to

$$-\frac{\delta^2}{2}\left\langle \frac{\partial^2}{\partial\theta^2}\log p(x|\theta)\right\rangle_{p(x|\theta)} \tag{8.11.39}$$

2. *More generally for a distribution parameterised by a vector with elements $\theta_i+\delta_i$, show that a small change in the parameter results in*

$$\sum_{i,j}\frac{\delta_i\delta_j}{2}F_{ij} \tag{8.11.40}$$

where the Fisher Information *matrix is defined as*

$$F_{ij} = -\left\langle \frac{\partial^2}{\partial\theta_i\partial\theta_j}\log p(x|\theta)\right\rangle_{p(x|\theta)} \tag{8.11.41}$$

3. *Show that the Fisher information matrix is positive semidefinite by expressing it equivalently as*

$$F_{ij} = \left\langle \frac{\partial}{\partial\theta_i}\log p(x|\theta)\frac{\partial}{\partial\theta_j}\log p(x|\theta)\right\rangle_{p(x|\theta)} \tag{8.11.42}$$

Exercise 8.23. *Consider the joint prior distribution*

$$p(\mu,\lambda|\mu_0,\alpha,\beta,\gamma) = \mathcal{N}\left(\mu|\mu_0,\gamma\lambda^{-1}\right)Gam\left(\lambda|\alpha,\beta\right) \tag{8.11.43}$$

Show that for $\mu_0=0$, $\gamma\to\infty$, $\beta\to\infty$, then the prior distribution becomes 'flat' (independent of μ and λ) for $\alpha=1/2$. Show that for these settings the mean and variance that jointly maximise the posterior equation (8.8.28) are given by the standard maximum likelihood settings

$$\mu_* = \frac{1}{N}\sum_n x^n, \qquad \sigma_*^2 = \frac{1}{N}\sum_n\left(x^n-\mu_*\right)^2 \tag{8.11.44}$$

Exercise 8.24. *Show that for equation (8.8.28) in the limit $\mu_0=0,\gamma\to\infty,\alpha=1,\beta\to\infty$, the jointly optimal mean and variance obtained from*

$$\underset{\mu,\lambda}{\text{argmax}}\; p(\mu,\lambda|\mathcal{X},\mu_0,\alpha,\beta,\gamma) \tag{8.11.45}$$

is given by

$$\mu_* = \frac{1}{N}\sum_n x^n, \qquad \sigma_*^2 = \frac{1}{N+1}\sum_n\left(x^n-\mu_*\right)^2 \tag{8.11.46}$$

where $\sigma_*^2 = 1/\lambda_*$. Note that these correspond to the standard 'unbiased' estimators of the mean and variance.

Exercise 8.25. *For the Gauss-Gamma posterior $p(\mu,\lambda|\mu_0,\alpha,\beta,\mathcal{X})$ given in equation (8.8.28) compute the marginal posterior $p(\mu|\mu_0,\alpha,\beta,\mathcal{X})$. What is the mean of this distribution?*

Exercise 8.26. *This exercise concerns the derivation of equation (8.4.15).*

1. *By considering*

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x} \tag{8.11.47}$$

$$\propto \int \exp\left(-\frac{1}{2}\left(\mathbf{y}-\mathbf{M}\mathbf{x}-\boldsymbol{\mu}\right)^\intercal\boldsymbol{\Sigma}^{-1}\left(\mathbf{y}-\mathbf{M}\mathbf{x}-\boldsymbol{\mu}\right)-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_x\right)^\intercal\boldsymbol{\Sigma}_x^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_x\right)\right)d\mathbf{x} \tag{8.11.48}$$

show that

$$p(\mathbf{y}) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\intercal\boldsymbol{\Sigma}^{-1}\mathbf{y}\right)\int\exp\left(-\frac{1}{2}\mathbf{x}^\intercal\mathbf{A}\mathbf{x}+\mathbf{x}^\intercal\left(\mathbf{B}\mathbf{y}+\mathbf{c}\right)\right) \tag{8.11.49}$$

for suitably defined $\mathbf{A},\mathbf{B},\mathbf{c}$.

2. *Using equation (8.4.13), show that*

$$p(\mathbf{y}) \propto \exp\left(-\frac{1}{2}\mathbf{y}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\mathbf{y} - \frac{1}{2}\left(\mathbf{B}\mathbf{y}+\mathbf{c}\right)^{\mathsf{T}}\mathbf{A}^{-1}\left(\mathbf{B}\mathbf{y}+\mathbf{c}\right)\right)$$

3. *This establishes that $p(\mathbf{y})$ is Gaussian. We now need to find the mean and covariance of this Gaussian. We can do this by the lengthly process of completing the square. Alternatively, we can appeal to*

$$\langle \mathbf{y} \rangle = \mathbf{M}\langle \mathbf{x} \rangle + \langle \boldsymbol{\eta} \rangle = \mathbf{M}\boldsymbol{\mu}_x + \boldsymbol{\mu}$$

By considering

$$\left\langle \left(\mathbf{y} - \langle \mathbf{y} \rangle\right)\left(\mathbf{y} - \langle \mathbf{y} \rangle\right)^{\mathsf{T}} \right\rangle = \left\langle \left(\mathbf{M}\mathbf{x} + \boldsymbol{\eta} - \mathbf{M}\boldsymbol{\mu}_x - \boldsymbol{\mu}\right)\left(\mathbf{M}\mathbf{x} + \boldsymbol{\eta} - \mathbf{M}\boldsymbol{\mu}_x - \boldsymbol{\mu}\right)^{\mathsf{T}} \right\rangle \tag{8.11.50}$$

and the independence of $\mathbf{x}$ and $\boldsymbol{\eta}$, derive the formula for the covariance of $p(\mathbf{y})$.

Exercise 8.27. *Consider the multivariate Gaussian distribution $p(\mathbf{x}) \sim \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ on the vector $\mathbf{x}$ with components $x_1, \ldots, x_n$:*

$$p(\mathbf{x}) = \frac{1}{\sqrt{\det\left(2\pi\boldsymbol{\Sigma}\right)}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \tag{8.11.51}$$

Calculate $p(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. Hint: make use of equation (8.4.19).

Exercise 8.28. *Observations $y_0, \ldots, y_{n-1}$ are noisy i.i.d. measurements of an underlying variable x with $p(x) \sim \mathcal{N}\left(x|0, \sigma_0^2\right)$ and $p(y_i|x) \sim \mathcal{N}\left(y_i|x, \sigma^2\right)$ for $i = 0, \ldots, n-1$. Show that $p(x|y_0, \ldots, y_{n-1})$ is Gaussian with mean*

$$\mu = \frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2}\overline{y} \tag{8.11.52}$$

where $\overline{y} = (y_0 + y_1 + \ldots + y_{n-1})/n$ and variance σ_n^2 such that

$$\frac{1}{\sigma_n^2} = \frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}. \tag{8.11.53}$$

Exercise 8.29. *Consider a set of data $\mathcal{X} = x^1, \ldots, x^N$ where each x^n is independently drawn from a Gaussian with known mean μ and unknown variance σ^2. Assume a gamma distribution prior on $\tau = 1/\sigma^2$,*

$$p(\tau) = Gam^{is}(\tau|a, b) \tag{8.11.54}$$

1. *Show that the posterior distribution is*

$$p(\tau|\mathcal{X}) = Gam^{is}\left(\tau|a + \frac{N}{2}, b + \frac{1}{2}\sum_{n=1}^{N}\left(x^n - \mu\right)^2\right) \tag{8.11.55}$$

2. *Show that the distribution for x is*

$$p(x|\mathcal{X}) = \int p(x|\tau)p(\tau|\mathcal{X})d\tau = Student\left(x|\mu, \lambda = \frac{a'}{b'}, \nu = 2a'\right) \tag{8.11.56}$$

where $a' = a + \frac{1}{2}N$, $b' = b + \frac{1}{2}\sum_{n=1}^{N}\left(x^n - \mu\right)^2$.

Exercise 8.30. *The Poisson distribution is a discrete distribution on the non-negative integers, with*

$$p(x) = \frac{e^{-\lambda}\lambda^x}{x!} \qquad x = 0,\ 1,\ 2, \ldots \tag{8.11.57}$$

You are given a sample of n observations $x_1, \ldots, x_n$ independently drawn from this distribution. Determine the maximum likelihood estimator of the Poisson parameter λ.

Exercise 8.31. *For a Gaussian mixture model*

$$p(\mathbf{x}) = \sum_i p_i \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right), \qquad p_i > 0, \sum_i p_i = 1 \tag{8.11.58}$$

show that $p(\mathbf{x})$ *has mean*

$$\langle \mathbf{x} \rangle = \sum_i p_i \boldsymbol{\mu}_i \tag{8.11.59}$$

and covariance

$$\sum_i p_i \left(\boldsymbol{\Sigma}_i + \boldsymbol{\mu}_i \boldsymbol{\mu}_i^T\right) - \sum_i p_i \boldsymbol{\mu}_i \sum_j p_j \boldsymbol{\mu}_j^T \tag{8.11.60}$$

Exercise 8.32. *Show that for the whitened data matrix, given in equation (8.4.30),* $\mathbf{Z}\mathbf{Z}^T = N\mathbf{I}$.

Exercise 8.33. *Consider a uniform distribution* $p_i = 1/N$ *defined on states* $i = 1, \ldots, N$. *Show that the entropy of this distribution is*

$$H = -\sum_{i=1}^{N} p_i \log p_i = \log N \tag{8.11.61}$$

and that therefore as the number of states N *increases to infinity, the entropy diverges to infinity.*

Exercise 8.34. *Consider a continuous distribution* $p(x)$, $x \in [0, 1]$. *We can form a discrete approximation with probabilities* p_i *to this continuous distribution by identifying a continuous value* i/N *for each state* $i = 1, \ldots, N$. *With this*

$$p_i = \frac{p(i/N)}{\sum_i p(i/N)} \tag{8.11.62}$$

show that the entropy $H = -\sum_i p_i \log p_i$ *is given by*

$$H = -\frac{1}{\sum_i p(i/N)} \sum_i p(i/N) \log p(i/N) + \log \sum_i p(i/N) \tag{8.11.63}$$

Since for a continuous distribution

$$\int_0^1 p(x) dx = 1 \tag{8.11.64}$$

a discrete approximation of this integral into bins of size $1/N$ *gives*

$$\frac{1}{N} \sum_{i=1}^{N} p(i/N) = 1 \tag{8.11.65}$$

Hence show that for large N,

$$H \approx -\int_0^1 p(x) \log p(x) dx + \textit{const.} \tag{8.11.66}$$

where the constant tends to infinity as $N \to \infty$. *Note that this result says that as a continuous distribution has essentially an infinite number of states, the amount of uncertainty in the distribution is infinite (alternatively, we would need an infinite number of bits to specify a continuous value). This motivates the definition of the differential entropy, which neglects the infinite constant of the limiting case of the discrete entropy.*

Exercise 8.35. *Consider two multivariate Gaussians* $\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right)$ *and* $\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\right)$.

1. *Show that the log product of the two Gaussians is given by*

$$-\frac{1}{2}\mathbf{x}^T\left(\boldsymbol{\Sigma}_1^{-1}+\boldsymbol{\Sigma}_2^{-1}\right)\mathbf{x}+\mathbf{x}^T\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1+\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\right)-\frac{1}{2}\left(\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1+\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\right)-\frac{1}{2}\log\det\left(2\pi\boldsymbol{\Sigma}_1\right)\det\left(2\pi\boldsymbol{\Sigma}_2\right)$$

2. *Defining* $\mathbf{A}=\boldsymbol{\Sigma}_1^{-1}+\boldsymbol{\Sigma}_2^{-1}$ *and* $\mathbf{b}=\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1+\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2$ *we can write the above as*

$$-\frac{1}{2}\left(\mathbf{x}-\mathbf{A}^{-1}\mathbf{b}\right)^T\mathbf{A}\left(\mathbf{x}-\mathbf{A}^{-1}\mathbf{b}\right)+\frac{1}{2}\mathbf{b}^T\mathbf{A}^{-1}\mathbf{b}-\frac{1}{2}\left(\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1+\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\right)-\frac{1}{2}\log\det\left(2\pi\boldsymbol{\Sigma}_1\right)\det\left(2\pi\boldsymbol{\Sigma}_2\right)$$

Writing $\boldsymbol{\Sigma}=\mathbf{A}^{-1}$ *and* $\boldsymbol{\mu}=\mathbf{A}^{-1}\mathbf{b}$ *show that the product of two Gaussians is a Gaussian with covariance*

$$\boldsymbol{\Sigma}=\boldsymbol{\Sigma}_1\left(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2\right)^{-1}\boldsymbol{\Sigma}_2 \tag{8.11.67}$$

mean

$$\boldsymbol{\mu}=\boldsymbol{\Sigma}_1\left(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2\right)^{-1}\boldsymbol{\mu}_2+\boldsymbol{\Sigma}_2\left(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2\right)^{-1}\boldsymbol{\mu}_1 \tag{8.11.68}$$

and log prefactor

$$\frac{1}{2}\mathbf{b}^T\mathbf{A}^{-1}\mathbf{b}-\frac{1}{2}\left(\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1+\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\right)-\frac{1}{2}\log\det\left(2\pi\boldsymbol{\Sigma}_1\right)\det\left(2\pi\boldsymbol{\Sigma}_2\right)+\frac{1}{2}\log\det\left(2\pi\boldsymbol{\Sigma}\right)$$

3. *Show that this can be written as*

$$\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1\right)\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2\right)=\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}\right)\frac{\exp\left(-\frac{1}{2}\left(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2\right)^T\mathbf{S}^{-1}\left(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2\right)\right)}{\sqrt{\det\left(2\pi\mathbf{S}\right)}} \tag{8.11.69}$$

where $\mathbf{S}=\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2$.

Exercise 8.36. *Show that*

$$\frac{\partial}{\partial\theta}\left\langle\log p(x|\theta)\right\rangle_{p(x|\theta^0)}|_{\theta=\theta^0}=0 \tag{8.11.70}$$

Exercise 8.37. *Using* $\left\langle f^2(x)\right\rangle-\left\langle f(x)\right\rangle^2\geq 0$*, and a suitably chosen function* $f(x)$*, show that*

$$\left\langle\frac{p(x)}{q(x)}\right\rangle_{p(x)}\geq 1 \tag{8.11.71}$$

for distributions $q(x)$ *and* $p(x)$*. This is related to the* $\alpha=2$ *divergence.*

Exercise 8.38. *Show that for any* α*,* $D_\alpha(p|p)=0$ *and that* $D_\alpha(p|q)\geq 0$ *for any distributions* $p(x)$*,* $q(x)$*.*

Exercise 8.39. *Show that for two* D *dimensional Gaussians,*

$$2KL(\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1\right)|\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2\right))=\mathrm{trace}\left(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right)+\left(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2\right)^T\boldsymbol{\Sigma}_2^{-1}\left(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2\right)+\log\det\left(\boldsymbol{\Sigma}_2\boldsymbol{\Sigma}_1^{-1}\right)-D$$

Exercise 8.40. *For data pairs* $(x^n,y^n),n=1,\ldots,N$*, the correlation can be considered a measure of the extent to which a linear relationship holds between* x *and* y*. For simplicity, we consider that both* x *and* y *have zero mean, and wish to consider the validity of the linear relation*

$$x=\alpha y \tag{8.11.72}$$

A measure of the discrepancy of this linear assumption is given by

$$E(\alpha)\equiv\sum_{n=1}^{N}\left(x^n-\alpha y^n\right)^2 \tag{8.11.73}$$

1. *Show that the α that minimises $E(\alpha)$ is given by*

$$\alpha^* = \frac{c}{\sigma_y^2} \tag{8.11.74}$$

where

$$c = \frac{1}{N}\sum_n x^n y^n, \qquad \sigma_y^2 = \frac{1}{N}\sum_n (y^n)^2 \tag{8.11.75}$$

2. *A measure of the 'linearity' between x and y is then*

$$\frac{E(\alpha^*)}{E(0)} \tag{8.11.76}$$

which is 1 when there is no linear relation ($\alpha = 0$) and 0 when there is a perfect linear relation (since then $E(\alpha^*) = 0$). Show that the correlation coefficient, definition(8.7), is given by

$$\rho = \sqrt{1 - \frac{E(\alpha^*)}{E(0)}} \tag{8.11.77}$$

3. *Defining the vectors $\mathbf{x} = (x^1, \ldots, x^N)^\mathsf{T}$, $\mathbf{y} = (y^1, \ldots, y^N)^\mathsf{T}$, show that the correlation coefficient is the cosine of the angle between $\mathbf{x}$ and $\mathbf{y}$,*

$$\rho = \frac{\mathbf{x}^\mathsf{T}\mathbf{y}}{|\mathbf{x}|\,|\mathbf{y}|} \tag{8.11.78}$$

and hence show that $-1 \leq \rho \leq 1$.

4. *Show that for the more general relation*

$$x = \alpha y + \gamma \tag{8.11.79}$$

for constant offset γ, setting γ to minimise

$$E(\alpha, \gamma) \equiv \sum_{n=1}^{N} (x^n - \alpha y^n - \gamma)^2 \tag{8.11.80}$$

has the effect of simply replacing x^n by $x^n - \bar{x}$ and y^n by $y^n - \bar{y}$, where $\bar{x}$ and $\bar{y}$ are the mean of the x and y data respectively.

Exercise 8.41. *For variables x, y, and $z = x + y$, show that the correlation coefficients are related by $\rho_{x,z} \geq \rho_{x,y}$. With reference to the correlation coefficient as the angle between two vectors, explain why $\rho_{x,z} \geq \rho_{x,y}$ is geometrically obvious.*

Exercise 8.42. *Consider a 'Boltzman machine' distribution on binary variables $x_i \in \{0, 1\}$, $i = 1, \ldots, D$*

$$p(\mathbf{x}|\mathbf{W}) = \frac{1}{Z_p(\mathbf{W})} \exp\left(\mathbf{x}^\mathsf{T}\mathbf{W}\mathbf{x}\right) \tag{8.11.81}$$

and that we wish to fit another distribution q of the same form to p

$$q(\mathbf{x}|\mathbf{U}) = \frac{1}{Z_q(\mathbf{U})} \exp\left(\mathbf{x}^\mathsf{T}\mathbf{U}\mathbf{x}\right) \tag{8.11.82}$$

1. *Show that*

$$\underset{\mathbf{U}}{\text{argmin}}\ KL(p|q) = \underset{\mathbf{U}}{\text{argmax}}\ \text{trace}\,(\mathbf{U}\mathbf{C}) - \log Z_q(\mathbf{U}), \qquad C_{ij} \equiv \langle x_i x_j \rangle_p \tag{8.11.83}$$

2. *Hence show that knowing the 'cross moment' matrix $\mathbf{C}$ of p is sufficient to fully specify p.*

3. *Generalise the above result to all models in the exponential family.*

++ **Exercise 8.43.** *Consider a distribution $\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ defined jointly over two vectors $\mathbf{x}$ and $\mathbf{y}$ of potentially differing dimensions,*

$$\mathbf{z} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \tag{8.11.84}$$

with corresponding mean and partitioned covariance

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{pmatrix} \tag{8.11.85}$$

where $\boldsymbol{\Sigma}_{yx} \equiv \boldsymbol{\Sigma}_{xy}^T$.

1. *By considering $p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{x}, \mathbf{y})$, show that*

$$p(\mathbf{x}|\mathbf{y}) \propto \exp -\frac{1}{2}\left\{(\mathbf{x} - \boldsymbol{\mu}_x)^T \mathbf{P} (\mathbf{x} - \boldsymbol{\mu}_x) + 2(\mathbf{x} - \boldsymbol{\mu}_x)^T \mathbf{Q} (\mathbf{y} - \boldsymbol{\mu}_y)\right\} \tag{8.11.86}$$

where

$$\begin{pmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{Q}^T & \mathbf{S} \end{pmatrix} \tag{8.11.87}$$

2. *By completing the square, show that*

$$p(\mathbf{x}|\mathbf{y}) \propto \exp -\frac{1}{2}\left\{(\mathbf{x} - \boldsymbol{\mu}_x + \mathbf{P}^{-1}\mathbf{Q}(\mathbf{y} - \boldsymbol{\mu}_y))^T \mathbf{P} (\mathbf{x} - \boldsymbol{\mu}_x + \mathbf{P}^{-1}\mathbf{Q}(\mathbf{y} - \boldsymbol{\mu}_y))\right\} \tag{8.11.88}$$

3. *Using partitioned matrix inversion, show that*

$$\boldsymbol{\Sigma}_{yx}\mathbf{P} + \boldsymbol{\Sigma}_{yy}\mathbf{Q}^T = \mathbf{0} \tag{8.11.89}$$

and that therefore

$$\mathbf{P}^{-1}\mathbf{Q} = -\boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1} \tag{8.11.90}$$

Show also that

$$\mathbf{P} = \left(\boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx}\right)^{-1} \tag{8.11.91}$$

4. *Hence show that*

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(\mathbf{y} - \boldsymbol{\mu}_y), \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx}\right) \tag{8.11.92}$$

Exercise 8.44. *As for the question above, consider the joint Gaussian distribution $p(\mathbf{z})$. Consider*

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y}) d\mathbf{y} = \int p(\mathbf{z}) d\mathbf{y} \tag{8.11.93}$$

1. *Show that*

$$p(\mathbf{x}) \propto \exp -\frac{1}{2}\left\{(\mathbf{x} - \boldsymbol{\mu}_x)^T \mathbf{P} (\mathbf{x} - \boldsymbol{\mu}_x)\right\} \int \exp -\frac{1}{2}\left(2(\mathbf{x} - \boldsymbol{\mu}_x)^T \mathbf{Q} (\mathbf{y} - \boldsymbol{\mu}_y) + (\mathbf{y} - \boldsymbol{\mu}_y)^T \mathbf{S} (\mathbf{y} - \boldsymbol{\mu}_y)\right) d\mathbf{y} \tag{8.11.94}$$

2. *By completing the square, show that*

$$2(\mathbf{x} - \boldsymbol{\mu}_x)^T \mathbf{Q} (\mathbf{y} - \boldsymbol{\mu}_y) + (\mathbf{y} - \boldsymbol{\mu}_y)^T \mathbf{S} (\mathbf{y} - \boldsymbol{\mu}_y) \tag{8.11.95}$$

$$= \left(\mathbf{y} - \boldsymbol{\mu}_y + \mathbf{S}^{-1}\mathbf{Q}^T(\mathbf{x} - \boldsymbol{\mu}_x)\right)^T \mathbf{S} \left(\mathbf{y} - \boldsymbol{\mu}_y + \mathbf{S}^{-1}\mathbf{Q}^T(\mathbf{x} - \boldsymbol{\mu}_x)\right) - (\mathbf{x} - \boldsymbol{\mu}_x)^T \mathbf{Q}\mathbf{S}^{-1}\mathbf{Q}^T(\mathbf{x} - \boldsymbol{\mu}_x) \tag{8.11.96}$$

3. *Using the fact that*

$$\int \exp -\frac{1}{2}\left\{\left(\mathbf{y}-\boldsymbol{\mu}_y+\mathbf{S}^{-1}\mathbf{Q}^T\left(\mathbf{x}-\boldsymbol{\mu}_x\right)\right)^T\mathbf{S}\left(\mathbf{y}-\boldsymbol{\mu}_y+\mathbf{S}^{-1}\mathbf{Q}^T\left(\mathbf{x}-\boldsymbol{\mu}_x\right)\right)\right\}d\mathbf{y}=\sqrt{\det\left(2\pi\mathbf{S}^{-1}\right)} \tag{8.11.97}$$

show that

$$p(\mathbf{x})\propto\exp-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_x\right)^T\left(\mathbf{P}-\mathbf{Q}\mathbf{S}^{-1}\mathbf{Q}^T\right)\left(\mathbf{x}-\boldsymbol{\mu}_x\right) \tag{8.11.98}$$

4. *Since*

$$\begin{pmatrix}\boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy}\\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy}\end{pmatrix}=\begin{pmatrix}\mathbf{P} & \mathbf{Q}\\ \mathbf{Q}^T & \mathbf{S}\end{pmatrix}^{-1} \tag{8.11.99}$$

Use partitioned matrix inversion to show that

$$\boldsymbol{\Sigma}_{xx}=\left(\mathbf{P}-\mathbf{Q}\mathbf{S}^{-1}\mathbf{Q}^T\right)^{-1} \tag{8.11.100}$$

5. *Hence show that*

$$p(\mathbf{x})=\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_x,\boldsymbol{\Sigma}_{xx}\right) \tag{8.11.101}$$

Exercise 8.45. *Consider a not necessarily square matrix* $\mathbf{T}$ *and Gaussian distributed* $\mathbf{x}$*, with* $p(\mathbf{x})=\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}\right)$*. We are interested in the distribution of the variable* $\mathbf{y}=\mathbf{T}\mathbf{x}$*, where* $\dim\mathbf{y}\leq\dim\mathbf{x}$*.*
Define

$$\mathbf{z}=\begin{pmatrix}\mathbf{y}\\ \tilde{\mathbf{x}}\end{pmatrix}=\underbrace{\begin{pmatrix}\mathbf{T}\\ \tilde{\mathbf{I}}\end{pmatrix}}_{\mathbf{M}}\mathbf{x} \tag{8.11.102}$$

where $\tilde{\mathbf{I}}$ *is any matrix such that* $\mathbf{M}$ *is square and invertible.*

1. *Show that*

$$p(\mathbf{z})=\int p(\mathbf{z}|\mathbf{x})p(\mathbf{x})d\mathbf{x}=\int\delta\left(\mathbf{z}-\mathbf{M}\mathbf{x}\right)p(\mathbf{x})d\mathbf{x}\propto\exp-\frac{1}{2}\left(\mathbf{M}^{-1}\mathbf{z}-\boldsymbol{\mu}\right)^T\boldsymbol{\Sigma}^{-1}\left(\mathbf{M}^{-1}\mathbf{z}-\boldsymbol{\mu}\right) \tag{8.11.103}$$

2. *Show that*

$$p(\mathbf{z})=\mathcal{N}\left(\mathbf{z}|\mathbf{M}\boldsymbol{\mu},\mathbf{M}\boldsymbol{\Sigma}\mathbf{M}^T\right) \tag{8.11.104}$$

3. *Using* $p(\mathbf{y})=\int p(\mathbf{z})d\tilde{\mathbf{x}}$ *and the result that the marginal of a Gaussian is a Gaussian (see above exercise), show that*

$$p(\mathbf{y})=\mathcal{N}\left(\mathbf{y}|\mathbf{T}\boldsymbol{\mu},\mathbf{T}\boldsymbol{\Sigma}\mathbf{T}^T\right) \tag{8.11.105}$$

CHAPTER 9

Learning as Inference

In previous chapters we largely assumed that all distributions are fully specified for the inference tasks. In machine learning and related fields, however, the distributions need to be learned on the basis of data. Learning is then the problem of integrating data with domain knowledge of the model environment. In this chapter we discuss how learning can be phrased as an inference problem.

9.1 Learning as Inference

9.1.1 Learning the bias of a coin

Consider data expressing the results of tossing a coin. We write $v^n = 1$ if on toss n the coin comes up heads, and $v^n = 0$ if it is tails. Our aim is to estimate the probability θ that the coin will be a head, $p(v^n = 1|\theta) = \theta$ – called the 'bias' of the coin. For a fair coin, $\theta = 0.5$. The variables in this environment are $v^1, \ldots, v^N$ and θ and we require a model of the probabilistic interaction of the variables, $p(v^1, \ldots, v^N, \theta)$. Assuming there is no dependence between the observed tosses, except through θ, we have the belief network

$$p(v^1, \ldots, v^N, \theta) = p(\theta) \prod_{n=1}^{N} p(v^n|\theta) \tag{9.1.1}$$

which is depicted in fig(9.1). The assumption that each observation is independent and identically distributed is called the i.i.d. assumption.

Learning refers to using the observations $v^1, \ldots, v^N$ to infer θ. In this context, our interest is

$$p(\theta|v^1, \ldots, v^N) = \frac{p(v^1, \ldots, v^N, \theta)}{p(v^1, \ldots, v^N)} = \frac{p(v^1, \ldots, v^N|\theta)p(\theta)}{p(v^1, \ldots, v^N)} \tag{9.1.2}$$

We still need to fully specify the prior $p(\theta)$. To avoid complexities resulting from continuous variables, we'll consider a discrete θ with only three possible states, $\theta \in \{0.1, 0.5, 0.8\}$. Specifically, we assume

$$p(\theta = 0.1) = 0.15, \quad p(\theta = 0.5) = 0.8, \quad p(\theta = 0.8) = 0.05 \tag{9.1.3}$$

as shown in fig(9.2a). This prior expresses that we have 80% belief that the coin is 'fair', 5% belief the coin is biased to land heads (with $\theta = 0.8$), and 15% belief the coin is biased to land tails (with $\theta = 0.1$). The distribution of θ given the data and our beliefs is

$$p(\theta|v^1, \ldots, v^N) \propto p(\theta) \prod_{n=1}^{N} p(v^n|\theta) = p(\theta) \prod_{n=1}^{N} \theta^{\mathbb{I}[v^n=1]} (1-\theta)^{\mathbb{I}[v^n=0]} \tag{9.1.4}$$

$$\propto p(\theta) \theta^{\sum_{n=1}^{N} \mathbb{I}[v^n=1]} (1-\theta)^{\sum_{n=1}^{N} \mathbb{I}[v^n=0]} \tag{9.1.5}$$

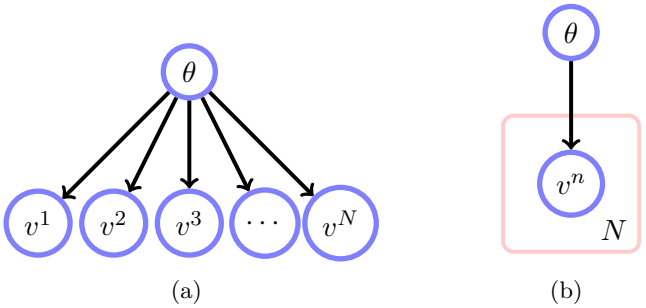


Figure 9.1: **(a)**: Belief network for coin tossing model. **(b)**: *Plate* notation equivalent of (a). A plate replicates the quantities inside the plate a number of times as specified in the plate.

In the above $\sum_{n=1}^{N} \mathbb{I}[v^n = 1]$ is the number of occurrences of heads, which we more conveniently denote as N_H. Likewise, $\sum_{n=1}^{N} \mathbb{I}[v^n = 0]$ is the number of tails, N_T. Hence

$$p(\theta|v^1, \ldots, v^N) \propto p(\theta)\theta^{N_H}(1-\theta)^{N_T} \tag{9.1.6}$$

For an experiment with $N_H = 2$, $N_T = 8$, the posterior distribution is

$$p(\theta = 0.1|\mathcal{V}) = k \times 0.15 \times 0.1^2 \times 0.9^8 = k \times 6.46 \times 10^{-4} \tag{9.1.7}$$
$$p(\theta = 0.5|\mathcal{V}) = k \times 0.8 \times 0.5^2 \times 0.5^8 = k \times 7.81 \times 10^{-4} \tag{9.1.8}$$
$$p(\theta = 0.8|\mathcal{V}) = k \times 0.05 \times 0.8^2 \times 0.2^8 = k \times 8.19 \times 10^{-8} \tag{9.1.9}$$

where $\mathcal{V}$ is shorthand for $v^1, \ldots, v^N$. From the normalisation requirement we have $1/k = 6.46 \times 10^{-4} + 7.81 \times 10^{-4} + 8.19 \times 10^{-8} = 0.0014$, so that

$$p(\theta = 0.1|\mathcal{V}) = 0.4525, \quad p(\theta = 0.5|\mathcal{V}) = 0.5475, \quad p(\theta = 0.8|\mathcal{V}) = 0.0001 \tag{9.1.10}$$

as shown in fig(9.2b). These are the 'posterior' parameter beliefs. In this case, if we were asked to choose a single *a posteriori* most likely value for θ, it would be $\theta = 0.5$, although our confidence in this is low since the posterior belief that $\theta = 0.1$ is also appreciable. This result is intuitive since, even though we observed more Tails than Heads, our prior belief was that it was more likely the coin is fair.

Repeating the above with $N_H = 20$, $N_T = 80$, the posterior changes to

$$p(\theta = 0.1|\mathcal{V}) \approx 1 - 1.93 \times 10^{-6}, \quad p(\theta = 0.5|\mathcal{V}) \approx 1.93 \times 10^{-6}, \quad p(\theta = 0.8|\mathcal{V}) \approx 2.13 \times 10^{-35} \tag{9.1.11}$$

fig(9.2c), so that the posterior belief in $\theta = 0.1$ dominates. This is reasonable since in this situation, there are so many more tails than heads that this is unlikely to occur from a fair coin. Even though we *a priori* thought that the coin was fair, *a posteriori* we have enough evidence to change our minds. @@

9.1.2 Making decisions

In itself, the Bayesian posterior merely represents our beliefs and says nothing about how best to summarise these beliefs. In situations in which decisions need to be taken under uncertainty we need to additionally specify what the utility of any decision is, as in chapter(7).

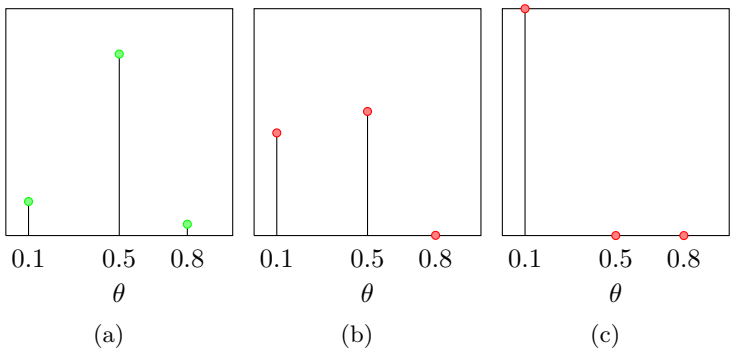


Figure 9.2: **(a)**: Prior encoding our beliefs about the amount the coin is biased to heads. **(b)**: Posterior having seen $N_H = 2$ heads and $N_T = 8$ tails. **(c)**: Posterior having seen $N_H = 20$ heads and $N_T = 80$ tails. Assuming any value of $0 \leq \theta \leq 1$ is possible the ML setting is $\theta = 0.2$ in both $N_H = 2, N_T = 8$ and $N_H = 20, N_T = 80$.

In the coin tossing scenario where θ is assumed to be either $0.1, 0.5$ or 0.8, we setup a decision problem as follows: If we correctly state the bias of the coin we gain 10 points; being incorrect, however, loses 20 points. We can write this using

$$U(\theta, \theta^0) = 10\mathbb{I}\left[\theta = \theta^0\right] - 20\mathbb{I}\left[\theta \neq \theta^0\right] \tag{9.1.12}$$

where θ^0 is the true value for the bias. The expected utility of the decision that the coin is $\theta = 0.1$ is

$$\begin{aligned} U(\theta = 0.1) = U(\theta = 0.1, \theta^0 = 0.1)p(\theta^0 = 0.1|\mathcal{V}) \\ + U(\theta = 0.1, \theta^0 = 0.5)p(\theta^0 = 0.5|\mathcal{V}) + U(\theta = 0.1, \theta^0 = 0.8)p(\theta^0 = 0.8|\mathcal{V}) \end{aligned} \tag{9.1.13}$$

Plugging in the numbers from equation (9.1.10), we obtain

$$U(\theta = 0.1) = 10 \times 0.4525 - 20 \times 0.5475 - 20 \times 0.0001 = -6.4270 \tag{9.1.14}$$

Similarly

$$U(\theta = 0.5) = 10 \times 0.5475 - 20 \times 0.4525 - 20 \times 0.0001 = -3.5770 \tag{9.1.15}$$

and

$$U(\theta = 0.8) = 10 \times 0.0001 - 20 \times 0.4525 - 20 \times 0.5475 = -19.999 \tag{9.1.16}$$

The best (that with the highest utility) is to say that the coin is unbiased, $\theta = 0.5$.

Repeating the above calculations for $N_H = 20, N_T = 80$, we arrive at

$$U(\theta = 0.1) = 10 \times (1 - 1.93 \times 10^{-6}) - 20\left(1.93 \times 10^{-6} + 2.13 \times 10^{-35}\right) = 9.9999 \tag{9.1.17}$$

$$U(\theta = 0.5) = 10 \times 1.93 \times 10^{-6} - 20\left(1 - 1.93 \times 10^{-6} + 2.13 \times 10^{-35}\right) \approx -20.0 \tag{9.1.18}$$

$$U(\theta = 0.8) = 10 \times 2.13 \times 10^{-35} - 20\left(1 - 1.93 \times 10^{-6} + 1.93 \times 10^{-6}\right) \approx -20.0 \tag{9.1.19}$$

so that the best decision in this case is to choose $\theta = 0.1$.

As more information about the distribution $p(v, \theta)$ becomes available the posterior $p(\theta|\mathcal{V})$ becomes increasingly peaked, aiding our decision making process.

9.1.3 A continuum of parameters

In section(9.1.1) we considered only three possible values for θ. Here we discuss a continuum of parameters.

Using a flat prior

We first examine the case of a 'flat' or *uniform* prior $p(\theta) = k$ for some constant k. For continuous variables, normalisation requires

$$\int p(\theta) d\theta = 1 \tag{9.1.20}$$

Since θ represents a probability we must have $0 \leq \theta \leq 1$,

$$\int_0^1 p(\theta) d\theta = k = 1 \tag{9.1.21}$$

Repeating the previous calculations with this flat continuous prior, we have

$$p(\theta|\mathcal{V}) = \frac{1}{c}\theta^{N_H}(1-\theta)^{N_T} \tag{9.1.22}$$

where c is a constant to be determined by normalisation,

$$c = \int_0^1 \theta^{N_H}(1-\theta)^{N_T} d\theta \equiv B(N_H + 1, N_T + 1) \tag{9.1.23}$$

where $B(\alpha, \beta)$ is the Beta function. See fig(9.3) for an example.

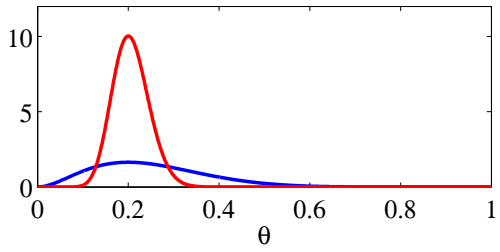


Figure 9.3: Posterior $p(\theta|\mathcal{V})$ assuming a flat prior on θ. (blue) $N_H = 2$, $N_T = 8$ and (red) $N_H = 20$, $N_T = 80$. In both cases, the most probable state of the posterior (maximum a posteriori) is 0.2, which makes intuitive sense, since the fraction of Heads to Tails in both cases is 0.2. Where there is more data, the posterior is more certain and sharpens around the most probable value.

Using a conjugate prior

Determining the normalisation constant of a continuous distribution requires that the integral of the unnormalised posterior can be carried out. For the coin tossing case, it is clear that if the prior is of the form of a Beta distribution, then the posterior will be of the same parametric form. For prior

$$p(\theta) = \frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1} \tag{9.1.24}$$

the posterior is

$$p(\theta|\mathcal{V}) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}\theta^{N_H}(1-\theta)^{N_T} \tag{9.1.25}$$

so that

$$p(\theta|\mathcal{V}) = B(\theta|\alpha+N_H,\beta+N_T) \tag{9.1.26}$$

The prior and posterior are of the same form (both Beta distributions) but simply with different parameters. Hence the Beta distribution is 'conjugate' to the Binomial distribution.

9.1.4 Decisions based on continuous intervals

To illustrate the use of continuous variables in decision making, we consider a simple decision problem. The result of a coin tossing experiment is $N_H = 2$ heads and $N_T = 8$ tails. You now need to make a decision: you win 10 dollars if you correctly guess which way the coin is biased – towards heads or tails. If your guess is incorrect, you lose a million dollars. What is your decision? (Assume an uninformative prior).

We need two quantities, θ for our guess and θ^0 for the truth. Then the utility of saying Heads is

$$U(\theta > 0.5, \theta^0 > 0.5)p(\theta^0 > 0.5|\mathcal{V}) + U(\theta > 0.5, \theta^0 < 0.5)p(\theta^0 < 0.5|\mathcal{V}) \tag{9.1.27}$$

In the above,

$$p(\theta^0 < 0.5|\mathcal{V}) = \int_0^{0.5} p(\theta^0|\mathcal{V})d\theta^0 \tag{9.1.28}$$

$$= \frac{1}{B(\alpha+N_H,\beta+N_T)}\int_0^{0.5}\theta^{\alpha+N_H-1}(1-\theta)^{\beta+N_T-1}\,d\theta \tag{9.1.29}$$

$$\equiv I_{0.5}(\alpha+N_H,\beta+N_T) \tag{9.1.30}$$

where $I_x(a,b)$ is the *regularised incomplete Beta function*. For the case of $N_H = 2, N_T = 8$, under a flat prior,

$$p(\theta^0 < 0.5|\mathcal{V}) = I_{0.5}(N_H+1, N_T+1) = 0.9673 \tag{9.1.31}$$

Since the events are exclusive, $p(\theta^0 \geq 0.5|\mathcal{V}) = 1 - 0.9673 = 0.0327$. Hence the expected utility of saying heads is more likely is

$$10 \times 0.0327 - 1000000 \times 0.9673 = -9.673 \times 10^5. \tag{9.1.32}$$

Similarly, the utility of saying tails is more likely can be computed to be

$$10 \times 0.9673 - 1000000 \times 0.0327 = -3.269 \times 10^4. \tag{9.1.33}$$

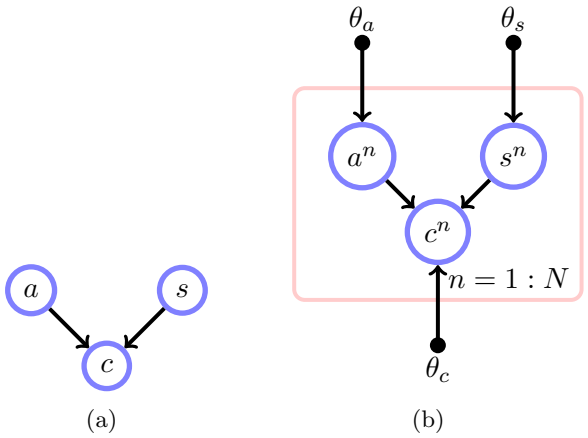


Figure 9.4: **(a)**: A model for the relationship between lung Cancer, Asbestos exposure and Smoking. **(b)**: Plate notation replicating the observed n datapoints with the CPTs tied across all datapoints.

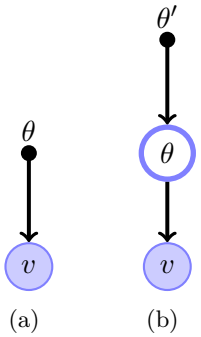


Figure 9.5: **(a)**: Standard ML learning. The best parameter θ is found by maximising the probability that the model generates the observed data $\theta_{opt} = \arg\max_\theta p(v|\theta)$. **(b)**: ML-II learning. In cases where we have a prior preference for the parameters θ, but with unspecified hyperparameter θ', we can find θ' by $\theta'_{opt} = \arg\max_{\theta'} p(v|\theta') = \arg\max_{\theta'} \langle p(v|\theta)\rangle_{p(\theta|\theta')}$.

Since the expected utility of deciding 'tails' is highest, we are better off taking the decision that the coin is more likely to come up tails.

If we modify the above so that we lose 100 million dollars if we guess tails when in fact it is heads, the expected utility of saying tails would be -3.27×10^6. In this case, even though we are more confident that the coin is likely to come up tails, we would pay such a penalty of making a mistake in saying tails, that it is in fact better to say heads.

9.2 Bayesian methods and ML-II

Consider a parameterised distribution $p(v|\theta)$, for which we wish to the learn the optimal parameters θ given some data. The model $p(v|\theta)$ is depicted in fig(9.5a), where a dot indicates that no distribution is present on that variable. For a single observed datapoint v, setting θ by maximum likelihood corresponds to finding the parameter θ that maximises $p(v|\theta)$.

In some cases we may have an idea about which parameters θ are more appropriate and can express this prior preference using a distribution $p(\theta)$. If the prior were fully specified, then there is nothing to 'learn' since $p(\theta|v)$ is now fully known. However, in many cases in practice, we are unsure of the exact parameter settings of the prior, and hence specify a parametersised prior using a distribution $p(\theta|\theta')$ with hyperparameter θ'. This is depicted in fig(9.5b). Learning then corresponds to finding the optimal θ' that maximises the likelihood $p(v|\theta') = \int_\theta p(v|\theta)p(\theta|\theta')$. This is known as an ML-II procedure since it corresponds to maximum likelihood, but at the higher, hyperparameter level[34, 198]. By treating the parameters θ as variables, one can view this then as learning under hidden variables, for which the methods of chapter(11) are applicable. We will encounter examples of this ML-II procedure later, for example in section(18.1.2).

a	s	c
1	1	1
1	0	0
0	1	1
0	1	0
1	1	1
0	0	0
1	0	1

Figure 9.6: A database containing information about the Asbestos exposure (1 signifies exposure), being a Smoker (1 signifies the individual is a smoker), and lung Cancer (1 signifies the individual has lung Cancer). Each row contains the information for each of the seven individuals in the database.

9.3 Maximum Likelihood Training of Belief Networks

Consider the following model of the relationship between exposure to asbestos (a), being a smoker (s) and the incidence of lung cancer (c)

$$p(a, s, c) = p(c|a, s)p(a)p(s) \tag{9.3.1}$$

which is depicted in fig(9.4a). Each variable is binary, $\text{dom}(a) = \{0, 1\}$, $\text{dom}(s) = \{0, 1\}$, $\text{dom}(c) = \{0, 1\}$. We assume that there is no direct relationship between Smoking and exposure to Asbestos. This is the kind of assumption that we may be able to elicit from medical experts. Furthermore, we assume that we have a list of patient records, fig(9.6), where each row represents a patient's data. To learn the table entries $p(c|a, s)$ we can do so by counting the number of times variable c is in state 1 for each of the 4 parental states of a and s:

$$\begin{array}{ll} p(c = 1|a = 0, s = 0) = 0, & p(c = 1|a = 0, s = 1) = 0.5 \\ p(c = 1|a = 1, s = 0) = 0.5 & p(c = 1|a = 1, s = 1) = 1 \end{array} \tag{9.3.2}$$

Similarly, based on counting, $p(a = 1) = 4/7$, and $p(s = 1) = 4/7$. These three CPTs then complete the full distribution specification.

Setting the CPT entries in this way by counting the relative number of occurrences corresponds mathematically to maximum likelihood learning under the i.i.d. assumption, as we show below.

Maximum likelihood corresponds to counting

For a BN there is a constraint on the form of $p(x)$, namely

$$p(x) = \prod_{i=1}^{K} p(x_i|\text{pa}\,(x_i)) \tag{9.3.3}$$

To compute the maximum likelihood setting of each term $p(x_i|\text{pa}\,(x_i))$, as shown in section(8.7.3), we can equivalently minimise the Kullback-Leibler divergence between the empirical distribution $q(x)$ and $p(x)$. For the BN $p(x)$, and empirical distribution $q(x)$ we have

$$\text{KL}(q|p) = -\left\langle \sum_{i=1}^{K} \log p\,(x_i|\text{pa}\,(x_i)) \right\rangle_{q(x)} + \text{const.} = -\sum_{i=1}^{K} \langle \log p\,(x_i|\text{pa}\,(x_i)) \rangle_{q(x_i,\text{pa}(x_i))} + \text{const.} \tag{9.3.4}$$

This follows using the general result

$$\langle f(\mathcal{X}_i) \rangle_{q(\mathcal{X})} = \langle f(\mathcal{X}_i) \rangle_{q(\mathcal{X}_i)} \tag{9.3.5}$$

which says that if the function f only depends on a subset of the variables, we only need to know the marginal distribution of this subset of variables in order to carry out the average.

Since $q(x)$ is fixed, we can add on entropic terms in q and equivalently mimimize

$$\text{KL}(q|p) = \sum_{i=1}^{K} \left[\langle \log q(x_i|\text{pa}\,(x_i)) \rangle_{q(x_i,\text{pa}(x_i))} - \langle \log p\,(x_i|\text{pa}\,(x_i)) \rangle_{q(x_i,\text{pa}(x_i))} \right] + \text{const.} \tag{9.3.6}$$ @@

$$= \sum_{i=1}^{K} \langle \text{KL}(q(x_i|\text{pa}\,(x_i))|p(x_i|\text{pa}\,(x_i))) \rangle_{q(\text{pa}(x_i))} + \text{const.} \tag{9.3.7}$$ @@

The final line is a positive weighted sum of individual Kullback-Leibler divergences. The minimal Kullback-Leibler setting, and that which corresponds to maximum likelihood, is therefore

$$p(x_i|\text{pa}\,(x_i)) = q(x_i|\text{pa}\,(x_i)) \tag{9.3.8}$$

In terms of the original data, this is

$$p(x_i = \mathsf{s}|\text{pa}\,(x_i) = \mathsf{t}) \propto \sum_{n=1}^{N} \mathbb{I}\left[x_i^n = \mathsf{s}, \text{pa}\,(x_i^n) = \mathsf{t}\right] \tag{9.3.9}$$

This expression corresponds to the intuition that the table entry $p(x_i|\text{pa}\,(x_i))$ can be set by counting the number of times the state $\{x_i = \mathsf{s}, \text{pa}\,(x_i) = \mathsf{t}\}$ occurs in the dataset (where t is a vector of parental states). The table is then given by the relative number of counts of being in state s compared to the other states s', for fixed joint parental state t.

An alternative method to derive this intuitive result is to use Lagrange multipliers, see exercise(9.4). For reader less comfortable with the above Kullback-Leibler derivation, a more direct example is given below which makes use of the notation

$$\sharp\left(x_1 = s_1, x_2 = s_2, x_3 = s_3, \ldots\right) \tag{9.3.10}$$

to denote the number of times that states $x_1 = s_1, x_2 = s_2, x_3 = s_3, \ldots$ occur together in the training data. See also section(10.1) for further examples.

Example 9.1. We wish to learn the table entries of the distribution $p(x_1, x_2, x_3) = p(x_1|x_2, x_3)p(x_2)p(x_3)$. We address here how to find the CPT entry $p(x_1 = 1|x_2 = 1, x_3 = 0)$ using maximum likelihood. For i.i.d. data, the contribution from $p(x_1|x_2, x_3)$ to the log likelihood is

$$\sum_n \log p(x_1^n|x_2^n, x_3^n)$$

The number of times $p(x_1 = 1|x_2 = 1, x_3 = 0)$ occurs in the log likelihood is $\sharp\left(x_1 = 1, x_2 = 1, x_3 = 0\right)$, the number of such occurrences in the training set. Since (by the normalisation constraint) $p(x_1 = 0|x_2 = 1, x_3 = 0) = 1 - p(x_1 = 1|x_2 = 1, x_3 = 0)$, the total contribution of $p(x_1 = 1|x_2 = 1, x_3 = 0)$ to the log likelihood is

$$\begin{aligned}\sharp\left(x_1 = 1, x_2 = 1, x_3 = 0\right) \log p(x_1 = 1|x_2 = 1, x_3 = 0) \\ + \sharp\left(x_1 = 0, x_2 = 1, x_3 = 0\right) \log\left(1 - p(x_1 = 1|x_2 = 1, x_3 = 0)\right)\end{aligned} \tag{9.3.11}$$

Using $\theta \equiv p(x_1 = 1|x_2 = 1, x_3 = 0)$ we have

$$\sharp\left(x_1 = 1, x_2 = 1, x_3 = 0\right) \log\theta + \sharp\left(x_1 = 0, x_2 = 1, x_3 = 0\right) \log\left(1 - \theta\right) \tag{9.3.12}$$

Differentiating the above expression *w.r.t.* θ and equating to zero gives

$$\frac{\sharp\left(x_1 = 1, x_2 = 1, x_3 = 0\right)}{\theta} - \frac{\sharp\left(x_1 = 0, x_2 = 1, x_3 = 0\right)}{1 - \theta} = 0 \tag{9.3.13}$$

The solution for optimal θ is then

$$p(x_1 = 1|x_2 = 1, x_3 = 0) = \frac{\sharp\left(x_1 = 1, x_2 = 1, x_3 = 0\right)}{\sharp\left(x_1 = 1, x_2 = 1, x_3 = 0\right) + \sharp\left(x_1 = 0, x_2 = 1, x_3 = 0\right)}, \tag{9.3.14}$$

corresponding to the intuitive counting procedure.

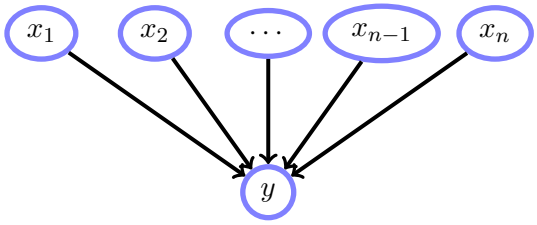


Figure 9.7: A variable y with a large number of parents $x_1, \ldots, x_n$ requires the specification of an exponentially large number of entries in the conditional probability $p(y|x_1, \ldots, x_n)$. One solution to this difficulty is to parameterise the conditional, $p(y|x_1, \ldots, x_n, \theta)$.

Conditional probability functions

Consider a binary variable y with n binary parental variables, $\mathbf{x} = (x_1, \ldots, x_n)$, see fig(9.7). There are 2^n entries in the CPT of $p(y|x)$ so that it is infeasible to explicitly store these entries for even moderate values of n. To reduce the complexity of this CPT we may constrain the form of the table. For example, one could use a function

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^\mathsf{T}\mathbf{x}}} \tag{9.3.15}$$

where we only need to specify the n-dimensional parameter vector $\mathbf{w}$.

In this case, rather than using maximum likelihood to learn the entries of the CPTs directly, we instead learn the value of the parameter $\mathbf{w}$. Since the number of parameters in $\mathbf{w}$ is small (n, compared with 2^n in the unconstrained case), we also have some hope that with a small number of training examples we can learn a reliable value for $\mathbf{w}$.

Example 9.2. Consider the following 3 variable model $p(x_1, x_2, x_3) = p(x_1|x_2, x_3)p(x_2)p(x_3)$, where $x_i \in \{0, 1\}, i = 1, 2, 3$. We assume that the CPT is parameterised using $\theta = (\theta_1, \theta_2)$ with

$$p(x_1 = 1|x_2, x_3, \theta) \equiv e^{-\theta_1^2 - \theta_2^2(x_2 - x_3)^2} \tag{9.3.16}$$

One may verify that the above probability is always positive and lies between 0 and 1. Due to normalisation, we must have

$$p(x_1 = 0|x_2, x_3) = 1 - p(x_1 = 1|x_2, x_3) \tag{9.3.17}$$

For unrestricted $p(x_2)$ and $p(x_3)$, the maximum likelihood setting is $p(x_2 = 1) \propto \sharp(x_2 = 1)$, and $p(x_3 = 1) \propto \sharp(x_3 = 1)$. The contribution to the log likelihood from the term $p(x_1|x_2, x_3, \theta)$, assuming i.i.d. data, is

$$L(\theta_1, \theta_2) = \sum_{n=1}^{N} \mathbb{I}[x_1^n = 1]\left(-\theta_1^2 - \theta_2^2(x_2^n - x_3^n)^2\right) + \mathbb{I}[x_1^n = 0] \log\left(1 - e^{-\theta_1^2 - \theta_2^2(x_2^n - x_3^n)^2}\right) \tag{9.3.18}$$

This objective function needs to be optimised numerically to find the best θ_1 and θ_2. The gradient is

$$\frac{dL}{d\theta_1} = \sum_{n=1}^{N} -2\mathbb{I}[x_1^n = 1]\,\theta_1 + 2\mathbb{I}[x_1^n = 0]\,\frac{\theta_1 e^{-\theta_1^2 - \theta_2^2(x_2^n - x_3^n)^2}}{1 - e^{-\theta_1^2 - \theta_2^2(x_2^n - x_3^n)^2}} \tag{9.3.19}$$

$$\frac{dL}{d\theta_2} = \sum_{n=1}^{N} -2\mathbb{I}[x_1^n = 1]\,\theta_2\,(x_2^n - x_3^n)^2 + 2\theta_2\mathbb{I}[x_1^n = 0]\,\frac{(x_2^n - x_3^n)^2 e^{-\theta_1^2 - \theta_2^2(x_2^n - x_3^n)^2}}{1 - e^{-\theta_1^2 - \theta_2^2(x_2^n - x_3^n)^2}} \tag{9.3.20}$$

The gradient can be used as part of a standard optimisation procedure (such as conjugate gradients, see section(A.5)) to find the maximum likelihood parameters θ_1, θ_2.

9.4 Bayesian Belief Network Training

An alternative to maximum likelihood training of a BN is to use a Bayesian approach in which we maintain a distribution over parameters. We continue with the Asbestos, Smoking, Cancer scenario,

$$p(a, c, s) = p(c|a, s)p(a)p(s) \tag{9.4.1}$$

as represented in fig(9.4a). So far we've only specified the independence structure, but not the entries of the tables $p(c|a, s)$, $p(a)$, $p(s)$. Given a set of visible observations, $\mathcal{V} = \{(a^n, s^n, c^n), n = 1, \ldots, N\}$, we would like to learn appropriate distributions for the table entries.

To begin we need a notation for the table entries. With all variables binary we have parameters such as

$$p(a = 1|\theta_a) = \theta_a, \quad p(c = 1|a = 0, s = 1, \theta_c) = \theta_c^{0,1} \tag{9.4.2}$$

and similarly for the remaining parameters $\theta_c^{1,1}, \theta_c^{0,0}, \theta_c^{1,0}$. For our example, the parameters are

$$\theta_a, \theta_s, \underbrace{\theta_c^{0,0}, \theta_c^{0,1}, \theta_c^{1,0}, \theta_c^{1,1}}_{\theta_c} \tag{9.4.3}$$

In the following section, section(9.4.1), we describe first useful independence assumptions on the general form of the prior variables, before making a specific numerical prior specification in section(9.4.2).

9.4.1 Global and local parameter independence

In Bayesian learning of BNs, we need to specify a prior on the joint table entries. Since in general dealing with multi-dimensional continuous distributions is computationally problematic, it is useful to specify only uni-variate distributions in the prior. As we show below, this has a pleasing consequence that for i.i.d. data the posterior also factorises into uni-variate distributions.

Global parameter independence

A convenient assumption is that the prior factorises over parameters. For our Asbestos, Smoking, Cancer example, we assume

$$p(\theta_a, \theta_s, \theta_c) = p(\theta_a)p(\theta_s)p(\theta_c) \tag{9.4.4}$$

Assuming the data is i.i.d., we then have the joint model

$$p(\theta_a, \theta_s, \theta_c, \mathcal{V}) = p(\theta_a)p(\theta_s)p(\theta_c)\prod_n p(a^n|\theta_a)p(s^n|\theta_s)p(c^n|s^n, a^n, \theta_c) \tag{9.4.5}$$

the belief network for which is given in fig(9.8.) A convenience of the factorised prior for a BN is that the posterior also factorises, since

$$\begin{aligned} p(\theta_a, \theta_s, \theta_c|\mathcal{V}) &\propto p(\theta_a, \theta_s, \theta_c, \mathcal{V}) \\ &= \left\{p(\theta_a)\prod_n p(a^n|\theta_a)\right\}\left\{p(\theta_s)\prod_n p(s^n|\theta_s)\right\}\left\{p(\theta_c)\prod_n p(c^n|s^n, a^n, \theta_c)\right\} \\ &\propto p(\theta_a|\mathcal{V}_a)p(\theta_s|\mathcal{V}_s)p(\theta_c|\mathcal{V}_c) \end{aligned} \tag{9.4.6}$$

so that one can consider each parameter posterior separately. In this case, 'learning' involves computing the posterior distributions $p(\theta_i|\mathcal{V}_i)$ where $\mathcal{V}_i$ is the set of training data restricted to the family of variable i.

The global independence assumption conveniently results in a posterior distribution that factorises over the conditional tables. However, the parameter θ_c is itself 4 dimensional. To simplify this we need to make a further assumption as to the structure of each local table.

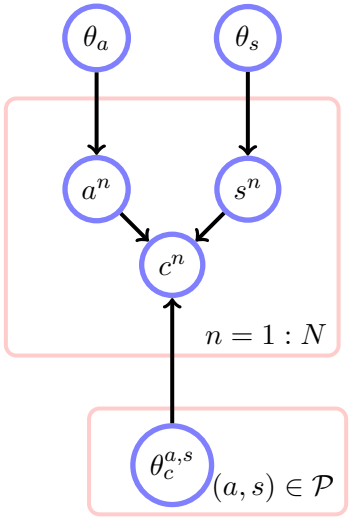


Figure 9.8: A Bayesian parameter model for the relationship between lung Cancer, Asbestos exposure and Smoking with factorised parameter priors. The global parameter independence assumption means that the prior over tables factorises into priors over each conditional probability table. The local independence assumption, which in this case comes into effect only for $p(c|a, s)$, means that $p(\theta_c)$ factorises in $\prod_{a,s\in\mathcal{P}} p(\theta_c^{a,s})$, where $\mathcal{P} = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$.

Local parameter independence

If we further assume that the prior for the table factorises over all states a, c:

$$p(\theta_c) = p(\theta_c^{0,0})p(\theta_c^{1,0})p(\theta_c^{0,1})p(\theta_c^{1,1}) \tag{9.4.7}$$

then the posterior is given by

$$\begin{aligned} p(\theta_c|\mathcal{V}_c) &\propto p(\mathcal{V}_c|\theta_c)p(\theta_c^{0,0})p(\theta_c^{1,0})p(\theta_c^{0,1})p(\theta_c^{1,1}) \\ &= \underbrace{\left[\theta_c^{0,0}\right]^{\sharp(a=0,s=0,c=1)}\left[1-\theta_c^{0,0}\right]^{\sharp(a=0,s=0,c=0)}p(\theta_c^{0,0})}_{\propto p(\theta_c^{0,0}|\mathcal{V}_c)}\underbrace{\left[\theta_c^{0,1}\right]^{\sharp(a=0,s=1,c=1)}\left[1-\theta_c^{0,1}\right]^{\sharp(a=0,s=1,c=0)}p(\theta_c^{0,1})}_{\propto p(\theta_c^{0,1}|\mathcal{V}_c)} \\ &\times \underbrace{\left[\theta_c^{1,0}\right]^{\sharp(a=1,s=0,c=1)}\left[1-\theta_c^{1,0}\right]^{\sharp(a=1,s=0,c=0)}p(\theta_c^{1,0})}_{\propto p(\theta_c^{1,0}|\mathcal{V}_c)}\underbrace{\left[\theta_c^{1,1}\right]^{\sharp(a=1,s=1,c=1)}\left[1-\theta_c^{1,1}\right]^{\sharp(a=1,s=1,c=0)}p(\theta_c^{1,1})}_{\propto p(\theta_c^{1,1}|\mathcal{V}_c)} \end{aligned} \tag{9.4.8}$$

so that the posterior also factorises over the parental states of the local conditional table.

Posterior marginal table

A marginal probability table is given by, for example,

$$p(c = 1|a = 1, s = 0, \mathcal{V}) = \int_{\theta_c} p(c = 1|a = 1, s = 0, \theta_c^{1,0})p(\theta_c|\mathcal{V}_c) \tag{9.4.9}$$

The integral over all the other tables in equation (9.4.9) is unity, and we are left with

$$p(c = 1|a = 1, s = 0, \mathcal{V}) = \int_{\theta_c^{1,0}} p(c = 1|a = 1, s = 0, \theta_c^{1,0})p(\theta_c^{1,0}|\mathcal{V}_c) = \int_{\theta_c^{1,0}} \theta_c^{1,0}p(\theta_c^{1,0}|\mathcal{V}_c) \tag{9.4.10}$$

9.4.2 Learning binary variable tables using a Beta prior

We continue the example of section(9.4.1) where all variables are binary, but using a continuous valued table prior. The simplest case is to start with $p(a|\theta_a)$ since this requires only a univariate prior distribution $p(\theta_a)$. The likelihood depends on the table variable via

$$p(a = 1|\theta_a) = \theta_a \tag{9.4.11}$$

so that the total likelihood term is

$$\theta_a^{\sharp(a=1)}\left(1 - \theta_a\right)^{\sharp(a=0)} \tag{9.4.12}$$

The posterior is therefore

$$p(\theta_a|\mathcal{V}_a) \propto p(\theta_a)\theta_a^{\sharp(a=1)}\left(1-\theta_a\right)^{\sharp(a=0)} \tag{9.4.13}$$

This means that if the prior is also of the form $\theta_a^{\alpha}\left(1-\theta_a\right)^{\beta}$ then conjugacy will hold, and the mathematics of integration will be straightforward. This suggests that the most convenient choice is a Beta distribution,

$$p(\theta_a) = B\left(\theta_a|\alpha_a,\beta_a\right) = \frac{1}{B(\alpha_a,\beta_a)}\theta_a^{\alpha_a-1}\left(1-\theta_a\right)^{\beta_a-1} \tag{9.4.14}$$

for which the posterior is also a Beta distribution:

$$p(\theta_a|\mathcal{V}_a) = B\left(\theta_a|\alpha_a+\sharp\left(a=1\right),\beta_a+\sharp\left(a=0\right)\right) \tag{9.4.15}$$

The marginal table is given by (following similar reasoning as for equation (9.4.10))

$$p(a=1|\mathcal{V}_a) = \int_{\theta_a} p(\theta_a|\mathcal{V}_a)\theta_a = \frac{\alpha_a+\sharp\left(a=1\right)}{\alpha_a+\sharp\left(a=1\right)+\beta_a+\sharp\left(a=0\right)} \tag{9.4.16}$$

using the result for the mean of a Beta distribution, definition(8.23).

The situation for the table $p(c|a,s)$ is slightly more complex since we need to specify a prior for each of the parental tables. As above, this is most convenient if we specify a Beta prior, one for each of the (four) parental states. Let's look at a specific table

$$p(c=1|a=1,s=0) \tag{9.4.17}$$

Assuming the local independence property, we have $p(\theta_c^{1,0}|\mathcal{V}_c)$ given by

$$B\left(\theta_c^{1,0}|\alpha_c(a=1,s=0)+\sharp\left(c=1,a=1,s=0\right),\beta_c(a=1,s=0)+\sharp\left(c=0,a=1,s=0\right)\right) \tag{9.4.18}$$

As before, the marginal probability table is then given by

$$p(c=1|a=1,s=0,\mathcal{V}_c) = \frac{\alpha_c(a=1,s=0)+\sharp\left(c=1,a=1,s=0\right)}{\alpha_c(a=1,s=0)+\beta_c(a=1,s=0)+\sharp\left(a=1,s=0\right)} \tag{9.4.19}$$

since $\sharp\left(a=1,s=0\right) = \sharp\left(c=0,a=1,s=0\right)+\sharp\left(c=1,a=1,s=0\right)$.

The prior parameters $\alpha_c(a,s)$ are called *hyperparameters*. A complete ignorance prior would correspond to setting $\alpha=\beta=1$, see fig(8.4).

It is instructive to examine this Bayesian solution under various conditions:

No data limit $N\to 0$ In the limit of no data, the marginal probability table corresponds to the prior, which is given in this case by

$$p(c=1|a=1,s=0) = \frac{\alpha_c(a=1,s=0)}{\alpha_c(a=1,s=0)+\beta_c(a=1,s=0)} \tag{9.4.20}$$

For a flat prior $\alpha=\beta=1$ for all states a,c, this would give a prior probability of $p(c=1|a=1,s=0)=0.5$.

Infinite data limit $N\to\infty$ In this limit the marginal probability tables are dominated by the data counts, since these will typically grow in proportion to the size of the dataset. This means that in the infinite (or very large) data limit,

$$p(c=1|a=1,s=0,\mathcal{V}) \to \frac{\sharp\left(c=1,a=1,s=0\right)}{\sharp\left(c=1,a=1,s=0\right)+\sharp\left(c=0,a=1,s=0\right)} \tag{9.4.21}$$

which corresponds to the maximum likelihood solution.

This effect that the large data limit of a Bayesian procedure corresponds to the maximum likelihood solution is general unless the prior has a pathologically strong effect.

Zero hyperparameter limit When $\alpha_c = \beta_c = 0$, the marginal table equation (9.4.19) corresponds to the maximum likelihood table setting for any amount of data. When $\alpha_c = \beta_c = 0$, the Beta distribution places mass 0.5 at 0 and mass 0.5 at 1. Note that this equivalence of the maximum likelihood solution with the marginal table under zero hyperparameter values contrasts with the equivalence of the MAP table under uniform hyperparameter values.

Example 9.3 (Asbestos-Smoking-Cancer)**.**

Consider the binary variable network

$$p(c, a, s) = p(c|a, s)p(a)p(s) \tag{9.4.22}$$

The data $\mathcal{V}$ is given in fig(9.6). Using a flat Beta prior $\alpha = \beta = 1$ for all conditional probability tables, the marginal posterior tables are given by

$$p(a = 1|\mathcal{V}) = \frac{1 + \sharp(a = 1)}{2 + N} = \frac{1 + 4}{2 + 7} = \frac{5}{9} \approx 0.556 \tag{9.4.23}$$

By comparison, the maximum likelihood setting is $4/7 = 0.571$. The Bayesian result is a little more cautious, which squares with our prior belief that any setting of the probability is equally likely, pulling the posterior towards 0.5.

Similarly,

$$p(s = 1|\mathcal{V}) = \frac{1 + \sharp(s = 1)}{2 + N} = \frac{1 + 4}{2 + 7} = \frac{5}{9} \approx 0.556 \tag{9.4.24}$$

and

$$p(c = 1|a = 1, s = 1, \mathcal{V}) = \frac{1 + \sharp(c = 1, a = 1, s = 1)}{2 + \sharp(c = 1, a = 1, s = 1) + \sharp(c = 0, a = 1, s = 1)} = \frac{1 + 2}{2 + 2} = \frac{3}{4} \tag{9.4.25}$$

$$p(c = 1|a = 1, s = 0, \mathcal{V}) = \frac{1 + \sharp(c = 1, a = 1, s = 0)}{2 + \sharp(c = 1, a = 1, s = 0) + \sharp(c = 0, a = 1, s = 0)} = \frac{1 + 1}{2 + 2} = \frac{1}{2} \tag{9.4.26}$$ @@

$$p(c = 1|a = 0, s = 1, \mathcal{V}) = \frac{1 + \sharp(c = 1, a = 0, s = 1)}{2 + \sharp(c = 1, a = 0, s = 1) + \sharp(c = 0, a = 0, s = 1)} = \frac{1 + 1}{2 + 2} = \frac{1}{2} \tag{9.4.27}$$

$$p(c = 1|a = 0, s = 0, \mathcal{V}) = \frac{1 + \sharp(c = 1, a = 0, s = 0)}{2 + \sharp(c = 1, a = 0, s = 0) + \sharp(c = 0, a = 0, s = 0)} = \frac{1 + 0}{2 + 1} = \frac{1}{3} \tag{9.4.28}$$

9.4.3 Learning multivariate discrete tables using a Dirichlet prior

The natural generalisation to our discussion of Bayesian learning of BNs is to consider variables that can take more than two states. In this case the natural conjugate prior is given by the Dirichlet distribution, which generalises the Beta distribution to more than two states. Again we assume throughout i.i.d. data and the local and global parameter prior independencies. Since under the global parameter independence assumption the posterior factorises over variables (as in equation (9.4.6)), we can concentrate on the posterior of a single variable.

No parents

Let's consider a variable v with $\text{dom}(v) = \{1, \ldots, I\}$. If we denote the probability of v being in state i by θ_i, *i.e.* $p(v = i|\boldsymbol{\theta}) = \theta_i$, the contribution to the posterior from a datapoint v^n is

$$p(v^n|\boldsymbol{\theta}) = \prod_{i=1}^{I} \theta_i^{\mathbb{I}[v^n=i]}, \qquad \sum_{i=1}^{I} \theta_i = 1 \tag{9.4.29}$$

so that the posterior for $\boldsymbol{\theta}$ given a dataset $\mathcal{V} = \left\{v^1, \ldots, v^N\right\}$

$$p(\boldsymbol{\theta}|\mathcal{V}) \propto p(\boldsymbol{\theta}) \prod_{n=1}^{N} \prod_{i=1}^{I} \theta_i^{\mathbb{I}[v^n=i]} = p(\boldsymbol{\theta}) \prod_{i=1}^{I} \theta_i^{\sum_{n=1}^{N} \mathbb{I}[v^n=i]} \tag{9.4.30}$$

It is convenient to use a Dirichlet prior distribution with hyperparameters $\mathbf{u}$

$$p(\boldsymbol{\theta}) = \text{Dirichlet}\left(\boldsymbol{\theta}|\mathbf{u}\right) \propto \prod_{i=1}^{I} \theta_i^{u_i - 1} \tag{9.4.31}$$

Using this prior the posterior becomes

$$p(\boldsymbol{\theta}|\mathcal{V}) \propto \prod_{i=1}^{I} \theta_i^{u_i-1} \prod_{i=1}^{I} \theta_i^{\sum_{n=1}^{N} \mathbb{I}[v^n=i]} = \prod_{i=1}^{I} \theta_i^{u_i - 1 + \sum_{n=1}^{N} \mathbb{I}[v^n=i]} \tag{9.4.32}$$

which means that the posterior is given by

$$p(\boldsymbol{\theta}|\mathcal{V}) = \text{Dirichlet}\left(\boldsymbol{\theta}|\mathbf{u} + \mathbf{c}\right) \tag{9.4.33}$$

where $\mathbf{c}$ is a count vector with components

$$c_i = \sum_{n=1}^{N} \mathbb{I}\left[v^n = i\right] \tag{9.4.34}$$

being the number of times state i was observed in the training data.

The marginal table is given by integrating

$$p(v = i|\mathcal{V}) = \int_{\boldsymbol{\theta}} p(v = i|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{V}) = \int_{\theta_i} \theta_i p(\theta_i|\mathcal{V}) \tag{9.4.35}$$

The single-variable marginal distribution of a Dirichlet is a Beta distribution,

$$p(\theta_i|\mathcal{V}) = B\left(\theta_i|u_i + c_i, \sum_{j \neq i} u_j + c_j\right). \tag{9.4.36}$$

The marginal table is then given by the mean of the Beta distribution

$$p(v = i|\mathcal{V}) = \frac{u_i + c_i}{\sum_j u_j + c_j} \tag{9.4.37}$$

which generalises the binary state formula equation (9.4.16).

Parents

To deal with the general case of a variable v with parents $\text{pa}\left(v\right)$ we denote the probability of v being in state i, conditioned on the parents being in state j as

$$p(v = i|\text{pa}\left(v\right) = j, \boldsymbol{\theta}) = \theta_i(v; j) \tag{9.4.38}$$

a	s	c
1	1	2
1	0	0
0	1	1
0	1	0
1	1	2
0	0	0
1	0	1

Figure 9.9: A database of patient records about Asbestos exposure (1 signifies exposure), being a Smoker (1 signifies the individual is a smoker), and lung Cancer (0 signifies no cancer, 1 signifies early stage cancer, 2 signifies late state cancer). Each row contains the information for each of the seven individuals in the database.

where $\sum_i \theta_i(v;j) = 1$. This forms the components of a vector $\boldsymbol{\theta}(v;j)$. Note that if v has K parents then the number of parental states S will be exponential in K.

Writing $\boldsymbol{\theta}(v) = [\boldsymbol{\theta}(v;1), \ldots, \boldsymbol{\theta}(v;S)]$, local (parental state) independence means

$$p(\boldsymbol{\theta}(v)) = \prod_j p(\boldsymbol{\theta}(v;j)) \tag{9.4.39}$$

and global independence means

$$p(\boldsymbol{\theta}) = \prod_v p(\boldsymbol{\theta}(v)) \tag{9.4.40}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}(v), v = 1, \ldots, V)$ represents the combined table of all the variables.

Parameter posterior

Thanks to the global parameter independence assumption the posterior factorises, with one posterior table per variable. Each posterior table for a variable v depends only on the data $\mathcal{D}(v)$ of the family of the variable. Assuming a Dirichlet distribution prior

$$p(\boldsymbol{\theta}(v;j)) = \text{Dirichlet}\left(\boldsymbol{\theta}(v;j)|\mathbf{u}(v;j)\right) \tag{9.4.41}$$

the posterior is also Dirichlet

$$p(\boldsymbol{\theta}(v)|\mathcal{D}(v)) = \prod_j \text{Dirichlet}\left(\boldsymbol{\theta}(v;j)|\mathbf{u}'(v;j)\right) \tag{9.4.42}$$

where the hyperparameter prior term is updated by the observed counts,

$$u'_i(v;j) \equiv u_i(v;j) + \sharp\left(v = i, \text{pa}\left(v\right) = j\right). \tag{9.4.43}$$

By analogy with the no-parents case, the marginal table is given by

$$p(v = i|\text{pa}\left(v\right) = j, \mathcal{D}(v)) \propto u'_i(v;j). \tag{9.4.44}$$

Example 9.4. Consider the $p(c|a,s)p(s)p(a)$ asbestos example with $\text{dom}(a) = \text{dom}(s) = \{0,1\}$, except now with the variable c taking three states, $\text{dom}(c) = \{0,1,2\}$, accounting for different kinds of cancer, see fig(9.9). The marginal table under a Dirichlet prior is then given by, for example

$$p(c = 0|a = 1, s = 1, \mathcal{V}) = \frac{u_0(a = 1, s = 1) + \sharp\left(c = 0, a = 1, s = 1\right)}{\sum_{i\in\{0,1,2\}} u_i(a = 1, s = 1) + \sharp\left(c = i, a = 1, s = 1\right)} \tag{9.4.45}$$

Assuming a flat Dirichlet prior, which corresponds to setting all components of $\mathbf{u}$ to 1, this gives

$$p(c = 0|a = 1, s = 1, \mathcal{V}) = \frac{1+0}{3+2} = \frac{1}{5} \tag{9.4.46}$$

$$p(c = 1|a = 1, s = 1, \mathcal{V}) = \frac{1+0}{3+2} = \frac{1}{5} \tag{9.4.47}$$

$$p(c = 2|a = 1, s = 1, \mathcal{V}) = \frac{1+2}{3+2} = \frac{3}{5} \tag{9.4.48}$$

and similarly for the other three tables $p(c|a = 1, s = 0), p(c|a = 0, s = 1), p(c|a = 1, s = 1)$.

Algorithm 9.1 PC algorithm for skeleton learning.

Start with a complete undirected graph G on the set $\mathcal{V}$ of all vertices.
$i = 0$
repeat
  for $x \in \mathcal{V}$ **do**
    for $y \in Adj\{x\}$ **do**
      Determine if there is a subset $\mathcal{S}$ of size i of the neighbours of x (not including y) for which $x \perp\!\!\!\perp y | \mathcal{S}$. If this set exists remove the $x - y$ link from the graph G and set $\mathcal{S}_{xy} = \mathcal{S}$.
    end for
  end for
  $i = i + 1$.
until all nodes have $\leq i$ neighbours.

Model likelihood

For a variable v, and i.i.d. data $\mathcal{D}(v) = \{(v^n|\text{pa}(v^n)), n = 1, \ldots, N\}$ for the family of this variable,

$$\prod_n p(v^n|\text{pa}(v^n)) = \int_{\boldsymbol{\theta}(v)} p(\boldsymbol{\theta}(v)) \prod_n p(v^n|\text{pa}(v^n), \boldsymbol{\theta}(v)) \tag{9.4.49}$$

$$= \int_{\boldsymbol{\theta}(v)} \left\{ \prod_j \frac{1}{Z(\mathbf{u}(v;j))} \prod_i \theta_i(v;j)^{u_i(v;j)-1} \right\} \prod_n \prod_j \prod_i \theta_i(v;j)^{\mathbb{I}[v^n=i, \text{pa}(v^n)=j]} \tag{9.4.50}$$

$$= \prod_j \frac{1}{Z(\mathbf{u}(v;j))} \int_{\boldsymbol{\theta}(v;j)} \prod_i \theta_i(v;j)^{u_i(v;j)-1+\sharp(v=i, \text{pa}(v)=j)} \tag{9.4.51}$$

$$= \prod_j \frac{Z(\mathbf{u}'(v;j))}{Z(\mathbf{u}(v;j))} \tag{9.4.52}$$

where $Z(\mathbf{u})$ is the normalisation constant of a Dirichlet distribution with hyperparameters $\mathbf{u}$; $\mathbf{u}'$ is as given in equation (9.4.43).

For a belief network on variables $\mathbf{v} = (v_1, \ldots, v_D)$ the joint probability of all variables factorises into the local probabilities of each variable conditioned on its parents. The likelihood of a complete set of i.i.d. data $\mathcal{D} = \{\mathbf{v}^1, \ldots, \mathbf{v}^N\}$ is then given by:

$$p(\mathcal{D}) = \prod_k \prod_n p(v_k^n|\text{pa}(v_k^n)) = \prod_k \prod_j \frac{Z(\mathbf{u}'(v_k;j))}{Z(\mathbf{u}(v_k;j))} \tag{9.4.53}$$

where $\mathbf{u}'$ is given by equation (9.4.43). Expression (9.4.53) can be written explicitly in terms of Gamma functions, see exercise(9.9). In the above expression in general the number of parental states differs for each variable v_k, so that implicit in the above formula is that the state product over j goes from 1 to the number of parental states of variable v_k. Due to the local and global parameter independence assumptions, the @@ logarithm of the model likelihood is a sum of terms, one for each variable v_k and parental configuration j. This is called the *likelihood decomposable* property.

9.5 Structure learning

Up to this point we have assumed that we are given both the structure of the distribution and a dataset $\mathcal{D}$. A more complex task is when we need to learn the structure of the network as well. We'll consider the case in which the data is complete (*i.e.* there are no missing observations). Since for D variables, there is an exponentially large number (in D) of BN structures, it's clear that we cannot search over all possible structures. For this reason structure learning is a computationally challenging problem and we must rely on constraints and heuristics to help guide the search. Whilst in general structure learning is intractable, a celebrated tractable special case is when the network is constrained to have at most one parent, see section(9.5.4).

For all but the sparsest networks, estimating the dependencies to any accuracy requires a large amount of data, making testing of dependencies difficult. Consider the following simple situation of two independent variables, $p(x, y) = p(x)p(y)$. Based on a finite sample from this joint distribution $\mathcal{D} = \{(x^n, y^n), n = 1, \ldots, N\}$, we want to try to understand if x is independent of y. One way to do this is to compute the empirical mutual information $I(x, y)$; if this is zero then, empirically, x and y are independent. However, for a finite amount of data, two variables will typically have non-zero mutual information, so that a threshold needs to be set to decide if the measured dependence is significant under the finite sample, see section(9.5.2).

Other complexities arise from the concern that a Belief or Markov Network on the visible variables alone may not be a parsimonious way to represent the observed data if, for example, there may be latent variables which are driving the observed dependencies. We will not enter into such issues in our discussion here and limit the presentation to two central approaches, one which attempts to make a network structure consistent with local empirical dependencies (the PC algorithm), and one which builds a structure that is most probable for the global data (network scoring).

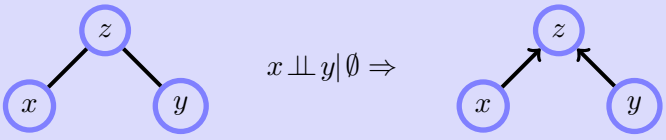
9.5.1 PC algorithm

The PC algorithm[275] first learns the skeleton of a graph, after which edges may be oriented to form a (partially oriented) DAG. The procedure to learn the skeleton is based on using the empirical data to test if two variables are independent. A variety of approaches can be used to ascertain independence, as described in section(9.5.2).

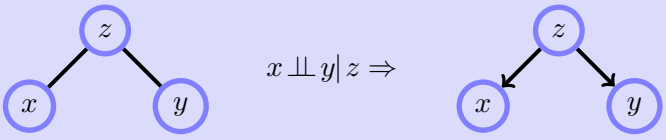
The PC algorithm begins at the first round with a complete skeleton G and attempts to remove as many links as possible. At the first step we test all pairs $x \perp\!\!\!\perp y \,|\, \emptyset$. If an x and y pair are deemed independent then the link $x - y$ is removed from the complete graph. One repeats this for all the pairwise links. In the second round, for the remaining graph, one examines each $x - y$ link and conditions on a single neighbour z of x. If $x \perp\!\!\!\perp y \,|\, z$ then remove the link $x - y$. One repeats in this way through all the variables. At each round the number of neighbours in the conditioning set is increased by one. See algorithm(9.1), fig(9.10)[1] and `demoPCoracle.m`. A refinement of this algorithm, known as NPC for necessary path PC[277] limits the number of independence checks to remove inconsistencies resulting from the empirical estimates of conditional mutual information.

Given a learned skeleton, a partial DAG can be constructed using algorithm(9.2). Note that this is necessary since the undirected graph G is a skeleton – not a belief network of the independence assumptions discovered. For example, we may have a graph G with $x - z - y$ in which the $x - y$ link was removed on the basis $x \perp\!\!\!\perp y \,|\, \emptyset \rightarrow \mathcal{S}_{xy} = \emptyset$. As a MN the graph $x - z - y$ (graphically) implies $x \not\perp\!\!\!\perp y$, although this is inconsistent with the discovery in the first round $x \perp\!\!\!\perp y$. This is the reason for the orientation part: for consistency, we must have $x \rightarrow z \leftarrow y$, for which $x \perp\!\!\!\perp y$ and $x \not\perp\!\!\!\perp y \,|\, z$, see example(9.5). See also fig(9.11).

Example 9.5 (Skeleton orienting).



If x is (unconditionally) independent of y, it must be that z is a collider since otherwise marginalising over z would introduce a dependence between x and y.



If x is independent of y conditioned on z, z must not be a collider. Any other orientation is appropriate.

[1] This example appears in [162] and [224] – thanks also to Serafín Moral for his online notes.

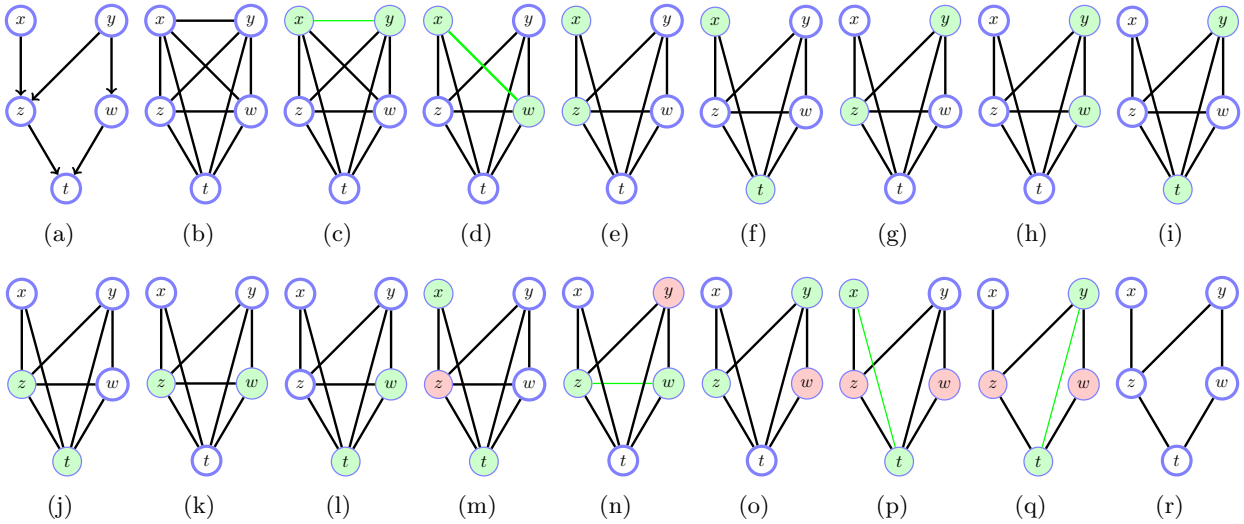


Figure 9.10: PC algorithm. **(a)**: The BN from which data is assumed generated and against which conditional independence tests will be performed. **(b)**: The initial skeleton is fully connected. **(c-l)**: In the first round ($i = 0$) all the pairwise mutual informations $x \perp\!\!\!\perp y | \emptyset$ are checked, and the link between x and y removed if deemed independent (green line). **(m-o)**: $i = 1$. We now look at connected subsets on three variables x, y, z of the remaining graph, removing the link $x - y$ if $x \perp\!\!\!\perp y | z$ is true. Not all steps are shown. **(p,q)**: $i = 2$. We now examine all $x \perp\!\!\!\perp y | \{a, b\}$. The algorithm terminates after this round (when i gets incremented to 3) since there are no nodes with 3 or more neighbours. **(r)**: Final skeleton. During this process the sets $S_{x,y} = \emptyset, S_{x,w} = \emptyset, S_{z,w} = y, S_{x,t} = \{z, w\}, S_{y,t} = \{z, w\}$ were found. See also `demoPCoracle.m`

Algorithm 9.2 Skeleton orientation algorithm (returns a DAG).

1: **Unmarried Collider**: Examine all undirected links $x - z - y$. If $z \notin \mathcal{S}_{xy}$ set $x \to z \leftarrow y$.
2: **repeat**
3: $x \to z - y \Rightarrow x \to z \to y$
4: For $x - y$, if there is a directed path from x to y orient $x \to y$
5: If for $x - z - y$ there is a w such that $x \to w$, $y \to w$, $z - w$ then orient $z \to w$
6: **until** No more edges can be oriented.
7: The remaining edges can be arbitrarily oriented provided that the graph remains a DAG and no additional colliders are introduced.

Example 9.6. In fig(9.10) we describe the processes of the PC algorithm in learning the structure for a belief network on the variables x, y, z, w, t. In this case, rather than using data to assess independence, we assume that we have access to an 'oracle' that can correctly answer any independence question put to it. In practice, of course, we will not be so fortunate! Once the skeleton has been found, we then orient the skeleton, as in fig(9.11).

9.5.2 Empirical independence

Mutual information test

Given data we can obtain an estimate of the conditional mutual information by using the empirical distribution $p(x, y, z)$ estimated by simply counting occurrences in the data. In practice, however, we only have a finite amount of data to estimate the empirical distribution. This means that for data sampled from a distribution for which the variables truly are independent, the empirical mutual information will nevertheless typically be greater than zero. An issue therefore is what threshold to use for the empirical conditional mutual information to decide if this is sufficiently far from zero to be caused by dependence. A frequentist approach is to compute the distribution of the conditional mutual information and then see where the

sample value is compared to the distribution. According to [179], under the null hypothesis that the variables are independent, $2N\mathrm{MI}(x;y|z)$ is Chi-square distributed with $(X-1)(Y-1)Z$ degrees of freedom with $\dim(x)=X$, $\dim(y)=Y$, $\dim(z)=Z$. This can then be used to form a hypothesis test; if the sample value of the empirical mutual information is 'significantly' in the tails of the chi-square distribution, we deem that the variables are conditionally dependent. This classical approach can work well for large amounts of data, but is less effective in the case of small amounts of data. An alternative pragmatic approach is to estimate the threshold based on empirical samples of the MI under controlled independent/dependent conditions – see `demoCondindepEmp.m` for a comparison of these approaches.

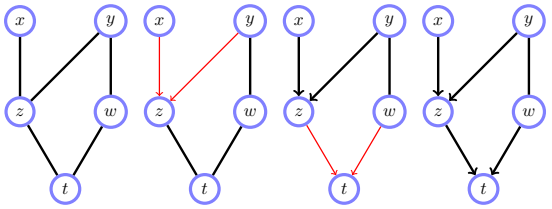


Figure 9.11: Skeleton orientation algorithm. **(a)**: The skeleton along with $S_{x,y}=\emptyset, S_{x,w}=\emptyset, S_{z,w}=y, S_{x,t}=\{z,w\}, S_{y,t}=\{z,w\}$. **(b)**: $z\notin S_{x,y}$, so form collider. **(c)**: $t\notin S_{z,w}$, so form collider. **(d)**: Final partially oriented DAG. The remaining edge may be oriented as desired, without violating the DAG condition. See also `demoPCoracle.m`.

Bayesian conditional independence test

A Bayesian approach to testing for independence can be made by comparing the likelihood of the data under the independence hypothesis, versus the likelihood under the dependent hypothesis. For the independence hypothesis, fig(9.12a), we have a joint distribution over variables and parameters:

$$p(x,y,z,\theta|\mathcal{H}_{indep})=p(x|z,\theta_{x|z})p(y|z,\theta_{y|z})p(z|\theta_z)p(\theta_{x|z})p(\theta_{y|z})p(\theta_z) \tag{9.5.1}$$

For categorical distributions, it is convenient to use a prior Dirichlet $(\theta|u)$ on the parameters θ, assuming also local as well as global parameter independence. For a set of assumed i.i.d. data $(\mathcal{X},\mathcal{Y},\mathcal{Z})=(x^n,y^n,z^n)\,,n=1,\ldots,N$, the likelihood is then given by integrating over the parameters θ:

$$p(\mathcal{X},\mathcal{Y},\mathcal{Z}|\mathcal{H}_{indep})=\int_\theta p(\theta|\mathcal{H}_{indep})\prod_n p(x^n,y^n,z^n|\theta,\mathcal{H}_{indep}) \qquad @@$$

Thanks to conjugacy, this is straightforward and gives the expression

$$p(\mathcal{X},\mathcal{Y},\mathcal{Z}|\mathcal{H}_{indep})=\frac{Z(u_z+\sharp(z))}{Z(u_z)}\prod_z\frac{Z(u_{x|z}+\sharp(x,z))}{Z(u_{x|z})}\frac{Z(u_{y|z}+\sharp(y,z))}{Z(u_{y|z})} \tag{9.5.2}$$

where $u_{x|z}$ is a hyperparameter matrix of pseudo counts for each state of x given each state of z. $Z(v)$ is the normalisation constant of a Dirichlet distribution with vector parameter v.

For the dependent hypothesis, fig(9.12b), we have

$$p(x,y,z,\theta|\mathcal{H}_{dep})=p(x,y,z|\theta_{x,y,z})p(\theta_{x,y,z}) \tag{9.5.3}$$

The likelihood is then

$$p(\mathcal{X},\mathcal{Y},\mathcal{Z}|\mathcal{H}_{dep})=\frac{Z(u_{x,y,z}+\sharp(x,y,z))}{Z(u_{x,y,z})} \tag{9.5.4}$$

Assuming each hypothesis is equally likely, for a Bayes' Factor

$$\frac{p(\mathcal{X},\mathcal{Y},\mathcal{Z}|\mathcal{H}_{indep})}{p(\mathcal{X},\mathcal{Y},\mathcal{Z}|\mathcal{H}_{dep})} \tag{9.5.5}$$

greater than 1, we assume that conditional independence holds; otherwise we assume the variables are conditionally dependent. `demoCondindepEmp.m` suggests that the Bayesian hypothesis test tends to outperform the conditional mutual information approach, particularly in the small sample size case, see fig(9.13).

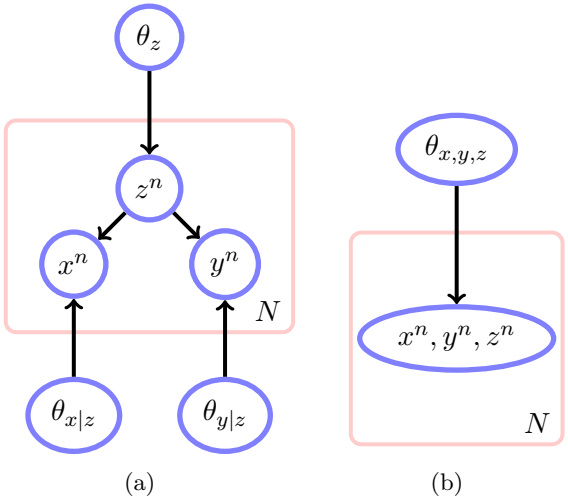


Figure 9.12: Bayesian conditional independence test using Dirichlet priors on the tables. **(a)**: A model $\mathcal{H}_{indep}$ for conditional independence $x \perp\!\!\!\perp y \,|\, z$. **(b)**: A model $\mathcal{H}_{dep}$ for conditional dependence $x \top\!\!\!\top y \,|\, z$. By computing the likelihood of the data under each model, a numerical score for the validity of the conditional independence assumption can be formed. See `demoCondindepEmp.m`.

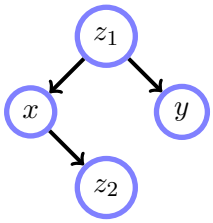


Figure 9.13: Conditional independence test of $x \perp\!\!\!\perp y \,|\, z_1, z_2$ with x, y, z_1, z_2 having $3, 2, 4, 2$ states respectively. From the oracle belief network shown, in each experiment the tables are drawn at random and 20 examples are sampled to form a dataset. For each dataset a test is carried out to determine if x and y are independent conditioned on z_1, z_2 (the correct answer being that they are independent). Over 500 experiments, the Bayesian conditional independence test correctly states that the variables are conditionally independent 74% of the time, compared with only 50% accuracy using the chi-square mutual information test. See `demoCondindepEmp.m`.

9.5.3 Network scoring

An alternative to local methods such as the PC algorithm is to evaluate the whole network structure on the set of variables $\mathbf{v}$. That is we wish to ascertain how well a belief network with a particular structure $p(\mathbf{v}) = \prod_k p(v_k|\text{pa}\,(v_k))$ fits the data. In a probabilistic context, given a model structure M, we wish to compute $p(M|\mathcal{D}) \propto p(\mathcal{D}|M)p(M)$. Some care is needed here since we have to first 'fit' each model with parameters θ, $p(\mathbf{v}|\theta, M)$ to the data $\mathcal{D}$. If we do this using maximum likelihood alone, with no constraints on θ, we will always end up favouring that model M with the most complex structure (assuming $p(M) = \text{const.}$). This can be remedied by using the Bayesian technique

$$p(\mathcal{D}|M) = \int_\theta p(\mathcal{D}|\theta, M)p(\theta|M) \tag{9.5.6}$$

In the case of directed networks, as we saw in section(9.4), the assumptions of local and global parameter independence make the integrals tractable. For a discrete state network and Dirichlet priors, we have $p(\mathcal{D}|M)$ given explicitly by the *Bayesian Dirichlet score* equation (9.4.53). First we specify the hyperparameters $\mathbf{u}(v; j)$, and then search over structures M, to find the one with the best score $p(\mathcal{D}|M)$. The simplest setting for the hyperparameters is set them all to unity[70]. Another setting is the 'uninformative prior'[56]

$$u_i(v; j) = \frac{\alpha}{\dim\,(v)\,\dim\,(\text{pa}\,(v))} \tag{9.5.7}$$

where $\dim\,(x)$ is the number of states of the variable(s) x, giving rise to the *BDeu* score, for an 'equivalent sample size' parameter α. A discussion of these settings is given in [142] under the concept of likelihood equivalence, namely that two networks which are Markov equivalent should have the same score. How dense the resulting network is can be sensitive to α[279, 267, 278]. Including an explicit prior $p(M)$ on the networks to favour those with sparse connections is also a sensible idea, for which one considers the modified score $p(\mathcal{D}|M)p(M)$.

Searching over structures is a computationally demanding task. However, since the log-score decomposes into additive terms involving only the family of each variable v, we can compare two networks differing in a single edge efficiently since when we make an adjustment within a family, no other terms outside the family

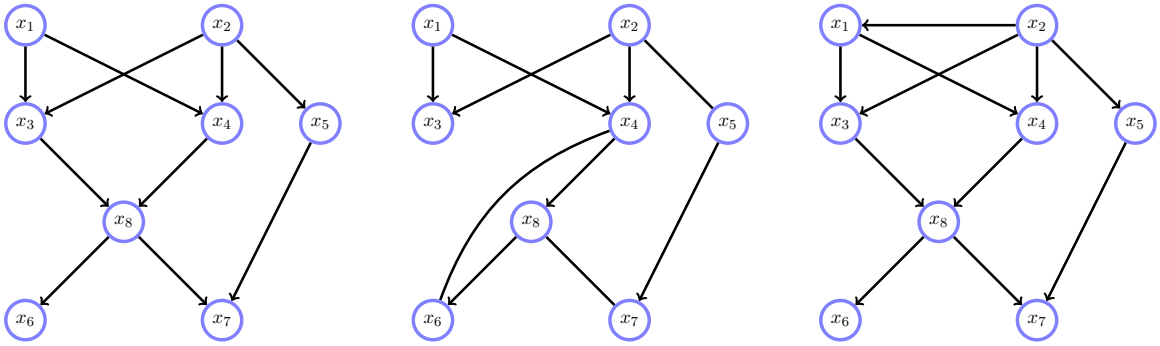


Figure 9.14: Learning the structure of a Bayesian network. **(a)**: The correct structure in which all variables are binary. The ancestral order is $x_2, x_1, x_5, x_4, x_3, x_8, x_7, x_6$. The dataset is formed from 1000 samples from this network. **(b)**: The learned structure based on the PC algorithm using the Bayesian empirical conditional independence test. Undirected edges may be oriented arbitrarily (provided the graph remains acyclic). **(c)**: The learned structure based on the Bayes Dirichlet network scoring method. See `demoPCdata.m` and `demoBDscore.m`.

are affected. This means that, given a candidate family, we can find which parents in this family should connect to the child; this computation can be carried out for all families independently. To help find the best families, search heuristics based on local addition/removal/reversal of edges [70, 142] that increase the score are popular[142]. In `learnBayesNet.m` we simplify the problem for demonstration purposes in which we assume we know the ancestral order of the variables, and also the maximal number of parents of each variable. In practice, it is unlikely that a large number of parents will influence a variable; even if this were the case, we would in general require an amount of data exponential in the number of parents to ascertain this. One could in principle approach this by assuming parametric forms for such large tables, although this is not common in practice.

Example 9.7 (PC algorithm versus network scoring)**.** In fig(9.14) we compare the PC algorithm with network scoring based (with Dirichlet hyperparameters set to unity) on 1000 samples from a known belief network. The PC algorithm conditional independence test is based on the Bayesian factor (9.5.5) in which Dirichlet priors with $u = 0.1$ were used throughout.

In the network scoring approach, in general there is no need to assume an ancestral ordering. However, here we assume that we know the correct ancestral ordering, and also limit the number of parents of each variable to be at most two. In this case, we can then easily search over all possible graph structures, choosing that with the highest posterior score. This proceeds by for example looking at the contribution of variable x_7 and its family. Since, according to the given ancestral ordering, $x_1, x_2, x_3, x_4, x_5, x_8$ are possible parents of x_7, in principle we need to search over all the 2^6 parental configurations for this family. However, since we assume that there are only maximally two parents, this reduces to $1 + \binom{6}{1} + \binom{6}{2} = 22$ parental configurations. We can perform this optimisation for the parental structure of variable x_7 independently of the parental structure of the other variables, thanks to the likelihood decomposable property of the network score. Similarly, we carry out this optimisation for the other variables separately, based on their possible parents according to the ancestral order.

In fig(9.14) the network scoring technique outperforms the PC algorithm. This is partly explained by the network scoring technique being provided with the correct ancestral order and the constraint that each variable has maximally two parents.

Algorithm 9.3 Chow-Liu Trees

1: **for** $i = 1$ to D **do**
2: **for** $j = 1$ to D **do**
3: Compute the mutual information for the pair of variables x_i, x_j: $w_{ij} = \mathrm{MI}(x_i; x_j)$.
4: **end for**
5: **end for**
6: For the undirected graph $\mathcal{G}$ with edge weights w, find a maximum weight undirected spanning tree $\mathcal{T}$.
7: Choose an arbitrary variable as the root node of the tree $\mathcal{T}$.
8: Form a directed tree by orienting all edges away from the root node.

9.5.4 Chow-Liu Trees

Consider a multivariate distribution $p(x)$ that we wish to approximate with a distribution $q(x)$. Furthermore, we constrain the approximation $q(x)$ to be a Belief Network in which each node has at most one parent, see fig(9.15). First we assume that we have chosen a particular labelling of the D variables so that children have higher parent indices than their parents. The DAG single parent constraint then means

$$q(x) = \prod_{i=1}^{D} q(x_i|x_{pa(i)}), \qquad pa(i) < i, \quad \text{or } pa(i) = \emptyset \tag{9.5.8}$$

where $pa(i)$ is the single parent index of node i. To find the best approximating distribution q in this constrained class, we may minimise the Kullback-Leibler divergence

$$\mathrm{KL}(p|q) = \langle \log p(x) \rangle_{p(x)} - \sum_{i=1}^{D} \left\langle \log q(x_i|x_{pa(i)}) \right\rangle_{p(x_i, x_{pa(i)})} \tag{9.5.9}$$

Since $p(x)$ is fixed, the first term is constant. By adding a term $\left\langle \log p(x_i|x_{pa(i)}) \right\rangle_{p(x_i, x_{pa(i)})}$ that depends on $p(x)$ alone, we can write

$$\mathrm{KL}(p|q) = \text{const.} - \sum_{i=1}^{D} \left\langle \left\langle \log q(x_i|x_{pa(i)}) \right\rangle_{p(x_i|x_{pa(i)})} - \left\langle \log p(x_i|x_{pa(i)}) \right\rangle_{p(x_i|x_{pa(i)})} \right\rangle_{p(x_{pa(i)})} \tag{9.5.10}$$

This enables us to recognise that, up to a negligible constant, the overall Kullback-Leibler divergence is a positive sum of individual Kullback-Leibler divergences so that the optimal setting is therefore

$$q(x_i|x_{pa(i)}) = p(x_i|x_{pa(i)}) \tag{9.5.11}$$

Plugging this solution into equation (9.5.9) and using $\log p(x_i|x_{pa(i)}) = \log p(x_i, x_{pa(i)}) - \log p(x_{pa(i)})$ we obtain

$$\mathrm{KL}(p|q) = \text{const.} - \sum_{i=1}^{D} \left\langle \log p(x_i, x_{pa(i)}) \right\rangle_{p(x_i, x_{pa(i)})} + \sum_{i=1}^{D} \left\langle \log p(x_{pa(i)}) \right\rangle_{p(x_{pa(i)})} \tag{9.5.12}$$

We still need to find the optimal parental structure $pa(i)$ that minimises the above expression. If we add and subtract an entropy term we can write

$$\mathrm{KL}(p|q) = -\sum_{i=1}^{D} \left\langle \log p(x_i, x_{pa(i)}) \right\rangle_{p(x_i, x_{pa(i)})} + \sum_{i=1}^{D} \left\langle \log p(x_{pa(i)}) \right\rangle_{p(x_{pa(i)})} + \sum_{i=1}^{D} \langle \log p(x_i) \rangle_{p(x_i)} - \sum_{i=1}^{D} \langle \log p(x_i) \rangle_{p(x_i)} + \text{const.} \tag{9.5.13}$$

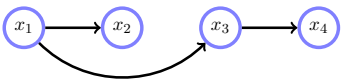


Figure 9.15: A Chow-Liu Tree in which each variable x_i has at most one parent. The variables may be indexed such that $1 \leq i \leq D$.

For two variables x_i and x_j and distribution $p(x_i, x_j)$, the *mutual information* definition(8.13) can be written as

$$\mathrm{MI}(x_i; x_j) = \left\langle \log \frac{p(x_i, x_j)}{p(x_i)p(x_j)} \right\rangle_{p(x_i,x_j)} \tag{9.5.14}$$

which can be seen as the Kullback-Leibler divergence $\mathrm{KL}(p(x_i, x_j)|p(x_i)p(x_j))$ and is therefore non-negative. Using this, equation (9.5.13) is

$$\mathrm{KL}(p|q) = -\sum_{i=1}^{D} \mathrm{MI}\big(x_i; x_{pa(i)}\big) - \sum_{i=1}^{D} \langle \log p(x_i) \rangle_{p(x_i)} + \mathsf{const.} \tag{9.5.15}$$

Since our task is to find the optimal parental indices $pa(i)$, and the entropic term $\sum_i \langle \log p(x_i) \rangle_{p(x_i)}$ of the fixed distribution $p(x)$ is independent of this mapping, finding the optimal mapping is equivalent to maximising the summed mutual informations

$$\sum_{i=1}^{D} \mathrm{MI}\big(x_i; x_{pa(i)}\big) \tag{9.5.16}$$

under the constraint that $pa(i) < i$. Since we also need to choose the optimal initial labelling of the variables as well, the problem is equivalent to computing all the pairwise mutual informations

$$w_{ij} = \mathrm{MI}(x_i; x_j) \tag{9.5.17}$$

and then finding a maximal spanning tree for the graph with edge weights w (see `spantree.m`)[65]. Once found, we need to identify a directed tree with at most one parent. This is achieved by choosing any node and then orienting edges consistently away from this node.

Maximum likelihood Chow-Liu trees

If $p(x)$ is the empirical distribution

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} \delta\left(x, x^n\right) \tag{9.5.18}$$

then

$$\mathrm{KL}(p|q) = \mathsf{const.} - \frac{1}{N} \sum_{n} \log q(x^n) \tag{9.5.19}$$

Hence the distribution q that minimises $\mathrm{KL}(p|q)$ is equivalent to that which maximises the likelihood of the data. This means that if we use the mutual information found from the empirical distribution, with

$$p(x_i = \mathsf{a}, x_j = \mathsf{b}) \propto \sharp\left(x_i = \mathsf{a}, x_j = \mathsf{b}\right) \tag{9.5.20}$$

then the Chow-Liu tree produced corresponds to the maximum likelihood solution amongst all single-parent trees. An outline of the procedure is given in algorithm(9.3). An efficient algorithm for sparse data is also available[206].

Remark 9.1 (Learning Tree structured Belief Networks)**.** The Chow-Liu algorithm pertains to the discussion in section(9.5) on learning the structure of Belief networks from data. Under the special constraint that each variable has at most one parent, the Chow-Liu algorithm returns the maximum likelihood structure to fit the data.

9.6 Maximum Likelihood for Undirected models

Consider a Markov network $p(\mathcal{X})$ defined on (not necessarily maximal) cliques $\mathcal{X}_c \subseteq \mathcal{X}, c = 1, \ldots, C$ with clique parameters $\theta = (\theta_1, \ldots, \theta_C)$

$$p(\mathcal{X}|\theta) = \frac{1}{Z(\theta)} \prod_c \phi_c(\mathcal{X}_c|\theta_c) \tag{9.6.1}$$

The term

$$Z(\theta) = \sum_{\mathcal{X}} \prod_c \phi_c(\mathcal{X}_c|\theta_c) \tag{9.6.2}$$

ensures normalisation, with the notation $\sum_{\mathcal{X}}$ indicating a summation over all states of the set of variables $\mathcal{X}$. Given a set of data, $\{\mathcal{X}^n, n = 1, \ldots, N\}$, and assuming i.i.d. data, the log likelihood is

$$L(\theta) = \sum_n \log p(\mathcal{X}^n|\theta) = \sum_n \sum_c \log \phi_c(\mathcal{X}_c^n|\theta_c) - N \log Z(\theta) \tag{9.6.3}$$

Our interest is to find the parameters that maximise the log likelihood $L(\theta)$. In general learning the optimal parameters $\theta_c, c = 1, \ldots, C$ is awkward since they are coupled via $Z(\theta)$. Unlike the BN, the objective function does not split into a set of isolated parameter terms and in general we need to resort to numerical methods. In special cases, however, exact results still apply, in particular when the MN is decomposable and no constraints are placed on the form of the clique potentials, as we discuss in section(9.6.3). More generally, however, gradient based techniques may be used and also give insight into properties of the maximum likelihood solution.

9.6.1 The likelihood gradient

The gradient of the log likelihood with respect to a clique parameter θ_c is given by

$$\frac{\partial}{\partial \theta_c} L(\theta) = \sum_n \frac{\partial}{\partial \theta_c} \log \phi_c(\mathcal{X}_c^n|\theta_c) - N \left\langle \frac{\partial}{\partial \theta_c} \log \phi_c(\mathcal{X}_c|\theta_c) \right\rangle_{p(\mathcal{X}_c|\theta)} . \tag{9.6.4}$$

This is obtained by using the result

$$\frac{\partial}{\partial \theta_c} \log Z(\theta) = \frac{1}{Z(\theta)} \sum_{\mathcal{X}} \frac{\partial}{\partial \theta_c} \phi_c(\mathcal{X}_c|\theta_c) \prod_{c' \neq c} \phi_{c'}(\mathcal{X}_{c'}|\theta_{c'}) = \left\langle \frac{\partial}{\partial \theta_c} \log \phi_c(\mathcal{X}_c|\theta_c) \right\rangle_{p(\mathcal{X}_c|\theta)} . \tag{9.6.5}$$

The gradient can then be used as part of a standard numerical optimisation package.

Exponential form potentials

A common form of parameterisation is to use an exponential form

$$\phi_c(\mathcal{X}_c) = \exp\left(\boldsymbol{\theta}_c^\mathsf{T} \boldsymbol{\psi}_c(\mathcal{X}_c)\right) \tag{9.6.6}$$

where $\boldsymbol{\theta}_c$ are the vector parameters and $\boldsymbol{\psi}_c(\mathcal{X}_c)$ is a fixed 'feature function' defined on the variables of clique c. From equation (9.6.4) to find the gradient, we need

$$\frac{\partial}{\partial \theta_c} \log \phi_c(\mathcal{X}_c|\theta_c) = \frac{\partial}{\partial \boldsymbol{\theta}_c} \boldsymbol{\theta}_c^\mathsf{T} \boldsymbol{\psi}_c(\mathcal{X}_c) = \boldsymbol{\psi}_c(\mathcal{X}_c). \tag{9.6.7}$$

Using this in equation (9.6.4), we find that the $L(\theta)$ has a zero derivative when

$$\frac{1}{N} \sum_n \boldsymbol{\psi}_c(\mathcal{X}_c^n) = \langle \boldsymbol{\psi}_c(\mathcal{X}_c) \rangle_{p(\mathcal{X}_c)} . \tag{9.6.8}$$

Hence the maximum likelihood solution satisfies that the empirical average of a feature function matches the average of the feature function with respect to the model. By defining the empirical distribution on the clique variables $\mathcal{X}_c$ as

$$\epsilon(\mathcal{X}_c) \equiv \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}\left[\mathcal{X}_c = \mathcal{X}_c^n\right] \tag{9.6.9}$$

we can write equation (9.6.8) more compactly as

$$\langle \boldsymbol{\psi}_c(\mathcal{X}_c)\rangle_{\epsilon(\mathcal{X}_c)} = \langle \boldsymbol{\psi}_c(\mathcal{X}_c)\rangle_{p(\mathcal{X}_c)}\,. \tag{9.6.10}$$

An example of learning such an exponential form is given in example(9.8). We return to learning the parameters of these models in section(9.6.4).

Example 9.8 (Boltzmann Machine learning)**.** We define the BM as

$$p(\mathbf{v}|\mathbf{W}) = \frac{1}{Z(\mathbf{W})}e^{\frac{1}{2}\mathbf{v}^\mathsf{T}\mathbf{W}\mathbf{v}}, \qquad Z(\mathbf{W}) = \sum_{\mathbf{v}} e^{\frac{1}{2}\mathbf{v}^\mathsf{T}\mathbf{W}\mathbf{v}} \tag{9.6.11}$$

for symmetric $\mathbf{W}$ and binary variables $\mathrm{dom}(v_i) = \{0, 1\}$. Given a set of training data, $\mathcal{D} = \left\{\mathbf{v}^1, \ldots, \mathbf{v}^N\right\}$, the log likelihood is

$$L(\mathbf{W}) = \frac{1}{2}\sum_{n=1}^{N} (\mathbf{v}^n)^\mathsf{T}\mathbf{W}\mathbf{v}^n - N\log Z(\mathbf{W}) \tag{9.6.12}$$

Differentiating *w.r.t.* w_{ij}, $i \neq j$ and w_{ii} we have the gradients

$$\frac{\partial L}{\partial w_{ij}} = \sum_{n=1}^{N}\left(v_i^n v_j^n - \langle v_i v_j\rangle_{p(\mathbf{v}|\mathbf{W})}\right), \qquad \frac{\partial L}{\partial w_{ii}} = \frac{1}{2}\sum_{n=1}^{N}\left(v_i^n - \langle v_i\rangle_{p(\mathbf{v}|\mathbf{W})}\right). \tag{9.6.13}$$

A simple algorithm to optimise the weight matrix $\mathbf{W}$ is to use gradient ascent,

$$w_{ij}^{new} = w_{ij}^{old} + \eta_1\frac{\partial L}{\partial w_{ij}}, \qquad w_{ii}^{new} = w_{ii}^{old} + \eta_2\frac{\partial L}{\partial w_{ii}} \tag{9.6.14}$$

for learning rates $\eta_1, \eta_2 > 0$. The intuitive interpretation is that learning will stop (the gradient is zero) when the second order statistics of the model $\langle v_i v_j\rangle_{p(\mathbf{v}|\mathbf{W})}$ match those of the empirical distribution, $\sum_n v_i^n v_j^n/N$. BM learning however is difficult since $\langle v_i v_j\rangle_{p(\mathbf{v}|\mathbf{W})}$ is typically computationally intractable for an arbitrary interaction matrix $\mathbf{W}$ and therefore needs to be approximated. Indeed, one cannot compute the likelihood $L(\mathbf{W})$ exactly for a general matrix $\mathbf{W}$ so that monitoring performance is also difficult.

9.6.2 General tabular clique potentials

For unconstrained clique potentials we have a separate table for each of the states defined on the clique. In writing the log likelihood, it is convenient to use the identity

$$\phi_c(\mathcal{X}_c^n) = \prod_{\mathcal{Y}_c}\phi_c(\mathcal{Y}_c)^{\mathbb{I}[\mathcal{Y}_c = \mathcal{X}_c^n]} \tag{9.6.15}$$

where the product is over all states of potential c. This expression follows since the indicator is zero for all but the single observed state $\mathcal{X}_c^n$. The log likelihood is then

$$L(\theta) = \sum_c\sum_{\mathcal{Y}_c}\sum_n \mathbb{I}\left[\mathcal{Y}_c = \mathcal{X}_c^n\right]\log\phi_c(\mathcal{Y}_c) - N\log Z(\phi) \tag{9.6.16}$$

where

$$Z(\phi) = \sum_{\mathcal{Y}_c} \prod_c \phi_c(\mathcal{Y}_c) \tag{9.6.17}$$

Differentiating the log likelihood with respect to a specific table entry $\phi_c(\mathcal{Y}_c)$ we obtain

$$\frac{\partial}{\partial \phi_c(\mathcal{Y}_c)} L(\theta) = \sum_n \mathbb{I}\left[\mathcal{Y}_c = \mathcal{X}_c^n\right] \frac{1}{\phi_c(\mathcal{Y}_c)} - N \frac{p(\mathcal{Y}_c)}{\phi_c(\mathcal{Y}_c)} \tag{9.6.18}$$

Equating to zero, and rewriting in terms of the variables $\mathcal{X}$, the maximum likelihood solution is obtained when

$$p(\mathcal{X}_c) = \epsilon(\mathcal{X}_c) \tag{9.6.19}$$

where the empirical distribution is defined in equation (9.6.9). That is, the unconstrained optimal maximum likelihood solution is given by setting the clique potentials such that the marginal distribution on each clique $p(\mathcal{X}_c)$ matches the empirical distribution on each clique $\epsilon(\mathcal{X}_c)$. Note that this only describes the form that the optimal maximum likelihood solution should take, and doesn't give us a closed form expression for setting the tables. To find the optimal tables in this case would still require a numerical procedure, such as gradient based methods based on equation (9.6.18), or the IPF method described below.

Iterative proportional fitting

According to the general result of equation (9.6.19) the maximum likelihood solution is such that the clique marginals match the empirical marginals. Assuming that we can absorb the normalisation constant into an arbitrarily chosen clique, we can drop explicitly representing the normalisation constant. For a clique c, the requirement that the marginal of p matches the empirical marginal on the variables in the clique is

$$\phi(\mathcal{X}_c) \sum_{\mathcal{X}_{\setminus c}} \prod_{d \neq c} \phi(\mathcal{X}_d) = \epsilon(\mathcal{X}_c) \tag{9.6.20}$$

Given an initial setting for the potentials we can then update $\phi(\mathcal{X}_c)$ to satisfy the above marginal requirement,

$$\phi^{new}(\mathcal{X}_c) = \frac{\epsilon(\mathcal{X}_c)}{\sum_{\mathcal{X}_{\setminus c}} \prod_{d \neq c} \phi(\mathcal{X}_d)} \tag{9.6.21}$$

which is required for each of the states of $\mathcal{X}_c$. By multiplying and dividing the right hand side by $\phi(\mathcal{X}_c)$ this is equivalent to ascertaining if

$$\phi^{new}(\mathcal{X}_c) = \frac{\phi(\mathcal{X}_c)\epsilon(\mathcal{X}_c)}{p(\mathcal{X}_c)} \tag{9.6.22}$$

This is a so-called Iterative Proportional Fitting (IPF) update and corresponds to a coordinate-wise optimisation of the log likelihood in which the coordinate corresponds to $\phi_c(\mathcal{X}_c)$, with all other parameters fixed. In this case this conditional optimum is analytically given by the above setting. One proceeds by selecting another potential to update, and continues updating until some convergence criterion is met. Note that in general, with each update, the marginal $p(\mathcal{X}_c)$ needs to be recomputed; computing these marginals may be expensive unless the width of the junction tree formed from the graph is suitably limited.

9.6.3 Decomposable Markov networks

Whilst for general Markov networks we require numerical methods to find the maximum likelihood solution, there is an important special case for which we can find the optimal tables very easily. If the MN corresponding is decomposable, then we know (from the junction tree representation) that we can express the distribution in the form of a product of local marginals divided by the separator distributions

$$p(\mathcal{X}) = \frac{\prod_c p(\mathcal{X}_c)}{\prod_s p(\mathcal{X}_s)} \tag{9.6.23}$$

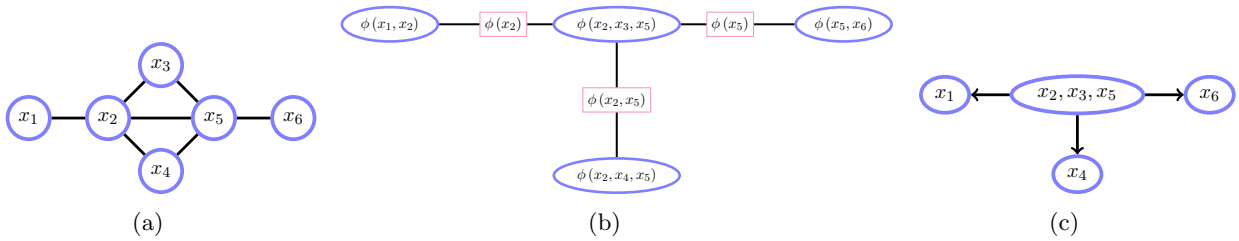


Figure 9.16: **(a)**: A decomposable Markov network. **(b)**: A junction tree for (a). **(c)**: Set chain for (a) formed by choosing clique x_2, x_3, x_5 as root and orienting edges consistently away from the root. Each separator is absorbed into its child clique to form the set chain.

Algorithm 9.4 Learning of an unconstrained decomposable Markov network using maximum likelihood. We have a triangulated (decomposable) Markov network on cliques $\phi_c(\mathcal{X}_c)$, $c = 1, \ldots, C$ and the empirical marginal distributions on all cliques and separators, $\epsilon(\mathcal{X}_c)$, $\epsilon(\mathcal{X}_s)$

1: Form a junction tree from the cliques.
2: Initialise each clique $\phi_c(\mathcal{X}_c)$ to $\epsilon(\mathcal{X}_c)$ and each separator $\phi_s(\mathcal{X}_s)$ to $\epsilon(\mathcal{X}_s)$.
3: Choose a root clique on the junction tree and orient edges consistently away from this root.
4: For this oriented junction tree, divide each clique by its parent separator.
5: Return the new potentials on each clique as the maximum likelihood solution.

By reabsorbing the separators into the numerator terms, we can form a set chain distribution, section(6.8)

$$p(\mathcal{X}) = \prod_c p(\mathcal{X}_c|\mathcal{X}_{\setminus c}) \tag{9.6.24}$$

Since this is directed, and provided no constraint is placed on the tables, the maximum likelihood solution to learning the tables is given by assigning each set chain factor $p(\mathcal{X}_c|\mathcal{X}_{\setminus c})$ based on counting the instances in the dataset[183], see `learnMarkovDecom.m`. The procedure is perhaps best explained by an example, as given below. See algorithm(9.4) for a general description.

Example 9.9. Given a dataset $\{\mathcal{X}^n, n = 1, \ldots, N\}$, with corresponding empirical distribution $\epsilon(\mathcal{X})$, we wish to fit by maximum likelihood a MN of the form

$$p(x_1, \ldots, x_6) = \frac{1}{Z}\phi(x_1, x_2)\phi(x_2, x_3, x_5)\phi(x_2, x_4, x_5)\phi(x_5, x_6) \tag{9.6.25}$$

where the potentials are unconstrained tables, see fig(9.16a). Since the graph is decomposable, we know it admits a factorisation of clique potentials divided by the separators:

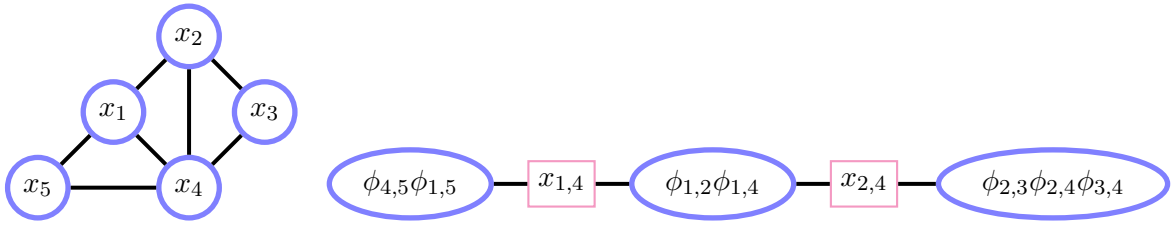
$$p(x_1, \ldots, x_6) = \frac{p(x_1, x_2)p(x_2, x_3, x_5)p(x_2, x_4, x_5)p(x_5, x_6)}{p(x_2)p(x_2, x_5)p(x_5)} \tag{9.6.26}$$

We can convert this to a set chain by reabsorbing the denominators into numerator terms, see section(6.8). For example, by choosing the clique x_2, x_3, x_5 as root, we can write

$$p(x_1, \ldots, x_6) = \underbrace{p(x_1|x_2)}_{\phi(x_1,x_2)}\underbrace{p(x_2, x_3, x_5)}_{\phi(x_2,x_3,x_5)}\underbrace{p(x_4|x_2, x_5)}_{\phi(x_2,x_4,x_5)}\underbrace{p(x_6|x_5)}_{\phi(x_5,x_6)} \tag{9.6.27}$$

where we identified the factors with clique potentials, and the normalisation constant Z is unity, see fig(9.16b). The advantage is that in this representation, the clique potentials are independent since the distribution is a BN on cluster variables. The log likelihood for an i.i.d. dataset is

$$L = \sum_n \log p(x_1^n|x_2^n) + \log p(x_2^n, x_3^n, x_5^n) + \log p(x_4^n|x_2^n, x_5^n) + \log p(x_6^n|x_5^n) \tag{9.6.28}$$



(a) (b)

Figure 9.17: **(a)**: Interpreted as a Markov network, the graph represents the distribution $\phi(x_1, x_4, x_5)\phi(x_1, x_2, x_4)\phi(x_2, x_4, x_3)$. As a pairwise MN, the graph represents $\phi(x_4, x_5)\phi(x_1, x_4)\phi(x_4, x_5)\phi(x_1, x_2)\phi(x_2, x_4)\phi(x_2, x_3)\phi(x_3, x_4)$. **(b)**: A junction tree for the pairwise MN in (a). We have a choice where to place the pairwise cliques, and this is one valid choice, using the shorthand $\phi_{a,b} = \phi_{a,b}(x_a, x_b)$ and $x_{a,b} = \{x_a, x_b\}$.

where each of the terms is an independent parameter of the model. The maximum likelihood solution then corresponds (as for the BN case) to simply setting each factor to the empirical distribution

$$\phi(x_1, x_2) = \epsilon(x_1|x_2), \quad \phi(x_2, x_3, x_5) = \epsilon(x_2, x_3, x_5), \quad \phi(x_2, x_4, x_5) = \epsilon(x_4|x_2, x_5), \quad \phi(x_5, x_6) = \epsilon(x_6|x_5) \tag{9.6.29}$$

Constrained decomposable Markov networks

If there are no constraints on the forms of the maximal clique potentials of the Markov network, as we've seen, learning is straightforward. Here our interest is when the functional form of the maximal clique is constrained to be a product of potentials on smaller cliques[2]:

$$\phi_c(\mathcal{X}_c) = \prod_i \phi_c^i(\mathcal{X}_c^i) \tag{9.6.30}$$

with no constraint being placed on the non-maximal clique potentials $\phi_c^i(\mathcal{X}_c^i)$. In general, in this case one cannot write down directly the maximum likelihood solution for the non-maximal clique potentials $\phi_c^i(\mathcal{X}_c^i)$.

Consider the graph in fig(9.17) which we consider a pairwise MN. In this case, the clique potentials are constrained, so that we cannot simply write down the solution, as we did in the unconstrained decomposable case. Since the graph is decomposable, there are however, computational savings that can be made in this case[11]. For an empirical distribution ϵ, maximum likelihood requires that all the pairwise marginals of the MN match the corresponding marginals obtained from ϵ. As explained in fig(9.17) we have a choice as to which junction tree clique each potential is assigned to, with one valid choice being given in fig(9.17b).

Let's consider updating the potentials within the $1, 2, 4$ clique. Keeping the potentials of the other cliques $\phi_{4,5}\phi_{1,5}$ and $\phi_{2,3}\phi_{2,4}\phi_{3,4}$ fixed we can update the potentials $\phi_{1,2}$, $\phi_{1,4}$. Using a bar to denote fixed potentials, the marginal requirement that the MN marginal $p(x_1, x_2, x_4)$ matches the empirical marginal $\epsilon(x_1, x_2, x_4)$ can be written in shorthand as

$$p_{1,2,4} = \epsilon_{1,2,4} \tag{9.6.31}$$

We can express this requirement in terms of the pairs of variables x_1, x_2 and x_1, x_4 within the $\phi_{1,2}\phi_{1,4}$ clique as

$$p_{1,2} = \epsilon_{1,2}, \qquad p_{1,4} = \epsilon_{1,4} \tag{9.6.32}$$

[2] A Boltzmann machine is of this form since any unconstrained binary pairwise potentials can be converted into a BM. For other cases in which the ϕ_c^i are constrained, then Iterative scaling may be used in place of IPF.

Algorithm 9.5 Efficient Iterative Proportional Fitting. Given a set of ϕ_i, $i = 1, \ldots, I$ and a corresponding set of reference (empirical) marginal distributions on the variables of each potential, ϵ_i, we aim to set all ϕ such that all marginals of the Markov network match the given empirical marginals.

Given a Markov network on potentials ϕ_i, $i = 1, \ldots, I$, triangulate the graph and form the cliques $\mathcal{C}_1, \ldots, \mathcal{C}_C$.
Assign potentials to cliques. Thus each clique has a set of associated potentials $\mathcal{F}_c$
Initialise all potentials (for example to unity).
repeat
    Choose a clique c as root.
    Propagate messages towards the root and compute the separators on the boundary of the root.
    repeat
        Choose a potential ϕ_i in clique c, $i \in \mathcal{F}_c$.
        Perform an IPF update for ϕ_i, given fixed boundary separators and other potentials in c.
    until Potentials in clique c converge.
until All Markov network marginals converge to the reference marginals.

Taking the first marginal requirement, this means

$$p_{1,2} = \sum_{x_3,x_4,x_5} \bar{\phi}_{1,5}\bar{\phi}_{4,5}\phi_{1,4}\phi_{1,2}\bar{\phi}_{2,4}\bar{\phi}_{2,3}\bar{\phi}_{3,4} = \epsilon_{1,2} \tag{9.6.33}$$

which can be expressed as

$$\sum_{x_4} \underbrace{\left(\sum_{x_5} \bar{\phi}_{1,5}\bar{\phi}_{4,5} \right)}_{\gamma_{1,4}} \phi_{1,4}\phi_{1,2} \underbrace{\left(\sum_{x_3} \bar{\phi}_{2,4}\bar{\phi}_{2,3}\bar{\phi}_{3,4} \right)}_{\gamma_{2,4}} = \epsilon_{1,2} \tag{9.6.34}$$

The 'messages' $\gamma_{1,4}$ and $\gamma_{1,2}$ are the boundary separator tables when we choose the central clique as root and carry out absorption towards the root. Given these fixed messages we can then perform IPF updates of the root clique using

$$\phi_{1,2}^{new} = \frac{\epsilon_{1,2}}{\sum_{x_4} \gamma_{1,4}\phi_{1,4}\gamma_{2,4}} \tag{9.6.35}$$

After making this update, we can subsequently update $\phi_{1,4}$ similarly using the constraint

$$\sum_{x_2} \underbrace{\left(\sum_{x_5} \bar{\phi}_{1,5}\bar{\phi}_{4,5} \right)}_{\gamma_{1,4}} \phi_{1,4}\phi_{1,2} \underbrace{\left(\sum_{x_3} \bar{\phi}_{2,4}\bar{\phi}_{2,3}\bar{\phi}_{3,4} \right)}_{\gamma_{2,4}} = \epsilon_{1,4} \tag{9.6.36}$$

so that

$$\phi_{1,4}^{new} = \frac{\epsilon_{1,4}}{\sum_{x_2} \gamma_{1,4}\phi_{1,2}\gamma_{2,4}} \tag{9.6.37}$$

We then iterate these updates until convergence within this $1, 2, 4$ clique. Given converged updates for this clique, we can choose another clique as root, propagate towards the root and compute the separator cliques on the boundary of the root. Given these fixed boundary clique potentials we again perform IPF within the clique.

This 'efficient' IPF procedure is described more generally in algorithm(9.5) for an empirical distribution ϵ. More generally, IPF minimises the Kullback-Leibler divergence between a given reference distribution ϵ and the Markov network. See `demoIPFeff.m` and `IPF.m`.

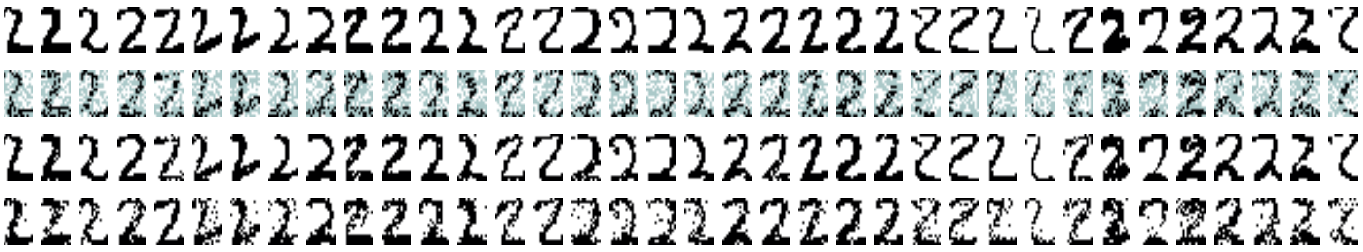
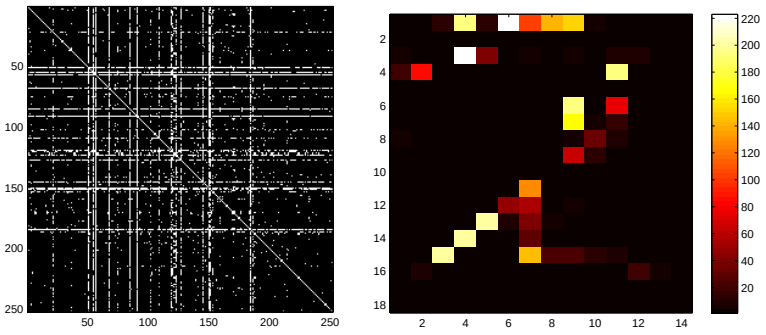


Figure 9.18: Learning digits (from Simon Lucas' algoval system) using a Markov network. Top row: the 36 training examples. Each example is a binary image on 18×14 pixels. Second row: the training data with 50% missing pixels (grey represents a missing pixel). Third row: Reconstructions from the missing data using a thin-junction-tree MN with maximum clique size 15. Bottom row: Reconstructions using a thin-junction-tree Boltzmann machine with maximum clique size 15, trained using efficient IPF.



(a) (b)

Figure 9.19: **(a)**: Based on the pairwise empirical entropies $H(x_i, x_j)$ edges are ordered, high entropy edges first. Shown is the adjacency matrix of the resulting Markov network whose junction tree has cliques ≤ 15 in size (white represents an edge). **(b)**: Indicated are the number of cliques that each pixel is a member of, indicating a degree of importance. Note that the lowest clique membership value is 1, so that each pixel is a member of at least one clique.

Example 9.10 (Learning with a structured Markov network)**.** In this example we aim to fit a Markov network to data, constrained so that inference in the Markov network is computationally cheap by ensuring that the junction tree of the Markov network has limited clique sizes.

In fig(9.18) 36 examples of $18 \times 14 = 252$ binary pixel handwritten twos are presented, forming the training set from which we wish to fit a Markov network. First all pairwise empirical entropies $H(x_i, x_j), i, j = 1, \ldots, 252$ were computed and used to rank edges, with highest entropy edges ranked first. Edges were included in a graph G, highest ranked first, provided the triangulated G had all cliques less than size 15. This resulted in 238 unique cliques and an adjacency matrix for the triangulated G as presented in fig(9.19a). In fig(9.19b) the number of times that a pixel appears in the 238 cliques is shown, and indicates the degree of importance of each pixel in distinguishing between the 36 examples. Two models were then trained and used to compute the most likely reconstruction based on missing data $p(x_{missing}|x_{visible})$.

The first model was a Markov network on the maximal cliques of the graph, for which essentially no training is required, and the settings for each clique potential can be obtained as explained in algorithm(9.4). The model makes 3.8% errors in reconstruction of the missing pixels. Note that the unfortunate effect of reconstructing a white pixel surrounded by black pixels is an effect of the limited training data. With larger amounts of data the model would recognise that such effects do not occur.

In the second model, the same maximal cliques were used, but the maximal clique potentials restricted to be the product of all pairwise two-cliques within the maximal clique. This is equivalent to using a structured Boltzmann machine, and was trained using the efficient IPF approach of algorithm(9.5). The corresponding reconstruction error is 20%. This performance is worse than the first model since the Boltzmann machine is a more constrained Markov network and struggles to represent the data well. See `demoLearnThinMNDigit.m`.

9.6.4 Exponential form potentials

For exponential form potentials

$$\phi_c(\mathcal{X}_c) = \exp\left(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{\psi}_c(\mathcal{X}_c)\right) \tag{9.6.38}$$

we saw in section(9.6.1) how to compute the derivatives for use in standard numerical optimisation procedures. In the following section we outline another popular numerical technique.

Iterative scaling

We consider Markov networks of the exponential form

$$p(\mathcal{X}|\theta) = \frac{1}{Z(\theta)}\prod_c e^{\theta_c f_c(\mathcal{X}_c)} \tag{9.6.39}$$

where the 'feature functions' $f_c(\mathcal{X}_c) \geq 0$ and c ranges of the non-maximal cliques $\mathcal{X}_c \subset \mathcal{X}$ (Note that equation (9.6.38) can be written in this form by having multiple potentials within the same clique.). The normalisation requirement is

$$Z(\theta) = \sum_{\mathcal{X}}\prod_c \exp\left(\theta_c f_c(\mathcal{X}_c)\right) \tag{9.6.40}$$

A maximum likelihood training algorithm for a Markov network, somewhat analogous to the EM approach of section(11.2) can be derived as follows[33]. Consider the bound, for positive x:

$$\log x \leq x - 1 \Rightarrow -\log x \geq 1 - x \tag{9.6.41}$$

Hence

$$-\log\frac{Z(\theta)}{Z(\theta^{old})} \geq 1 - \frac{Z(\theta)}{Z(\theta^{old})} \Rightarrow -\log Z(\theta) \geq -\log Z(\theta^{old}) + 1 - \frac{Z(\theta)}{Z(\theta^{old})} \tag{9.6.42}$$

Then we can write a bound on the log likelihood

$$\frac{1}{N}L(\theta) \geq \frac{1}{N}\sum_{c,n}\theta_c f_c(\mathcal{X}_c^n) - \log Z(\theta^{old}) + 1 - \frac{Z(\theta)}{Z(\theta^{old})} \tag{9.6.43}$$

As it stands, the bound (9.6.43) is in general not straightforward to optimise since the parameters of each potential are coupled through the $Z(\theta)$ term. For convenience it is useful to first reparmameterise and write

$$\theta_c = \underbrace{\theta_c - \theta_c^{old}}_{\alpha_c} + \theta_c^{old} \tag{9.6.44}$$

Then

$$Z(\theta) = \sum_{\mathcal{X}}\exp\left(\sum_c f_c(\mathcal{X}_c)\theta_c\right) = \sum_{\mathcal{X}}\exp\left(\sum_c f_c(\mathcal{X}_c)\theta_c^{old}\right)\exp\left(\sum_c f_c(\mathcal{X}_c)\alpha_c\right) \tag{9.6.45}$$

One can decouple this using an additional bound derived by first considering:

$$\exp\left(\sum_c \alpha_c f_c(\mathcal{X}_c)\right) = \exp\left(\sum_c p_c\left[\alpha_c\sum_d f_d(\mathcal{X}_d)\right]\right) \tag{9.6.46}$$

where

$$p_c \equiv \frac{f_c(\mathcal{X}_c)}{\sum_d f_d(\mathcal{X}_d)} \tag{9.6.47}$$

Since $p_c \geq 0$ and $\sum_c p_c = 1$ we may apply Jensen's inequality to give

$$\exp\left(\sum_c \alpha_c f_c(\mathcal{X}_c)\right) \leq \sum_c p_c \exp\left(\sum_d f_d(\mathcal{X}_d)\alpha_c\right) \tag{9.6.48}$$

Hence

$$Z(\theta) \leq \sum_{\mathcal{X}} \exp\left(\sum_c f_c(\mathcal{X}_c)\theta_c^{old}\right) \sum_c p_c \exp\left(\alpha_c \sum_f f_d(\mathcal{X}_c)\right) \tag{9.6.49}$$

Plugging this bound into (9.6.43) we have

$$\frac{1}{N}L(\theta) \geq \sum_c \underbrace{\left\{\frac{1}{N}\sum_n f_c(\mathcal{X}_c^n)\theta_c - \left\langle p_c \exp\left(\alpha_c \sum_d f_d(\mathcal{X}_c)\right)\right\rangle_{p(\mathcal{X}|\theta^{old})}\right\}}_{LB(\theta_c)} + 1 - \log Z(\theta^{old}) \tag{9.6.50}$$

The term in curly brackets contains the potential parameters θ_c in an uncoupled fashion. Differentiating with respect to θ_c the gradient of each lower bound is given by

$$\frac{\partial LB(\theta_c)}{\partial \theta_c} = \frac{1}{N}\sum_n f_c(\mathcal{X}_c^n) - \left\langle f_c(\mathcal{X}_c)\exp\left(\left(\theta_c - \theta_c^{old}\right)\sum_d f_d(\mathcal{X}_d)\right)\right\rangle_{p(\mathcal{X}|\theta^{old})} \tag{9.6.51}$$

This can be used as part of a gradient based optimisation procedure to learn the parameters θ_c. Intuitively, the parameters converge when the empirical average of the functions f match the average of the functions with respect to samples drawn from the distribution, in line with our general condition for maximum likelihood optimal solution.

In general, there would appear to be little advantage in using the above procedure over the general gradient approach in section(9.6.1) [212]. However, in the special case that the functions sum to 1, $\sum_c f_c(\mathcal{X}_c) = 1$, the zero of the gradient (9.6.51) can be found analytically, giving the iterative scaling (IS) update

$$\theta_c = \theta_c^{old} + \log \frac{1}{N}\sum_n f_c(\mathcal{X}_c^n) - \log \langle f_c(\mathcal{X}_c)\rangle_{p(\mathcal{X}_c|\theta^{old})} \tag{9.6.52}$$

The constraint that the features f_c need to be non-negative can be relaxed at the expense of additional variational parameters, see exercise(9.12).

If the junction tree formed from this exponential form Markov network has limited tree width, computational savings can be made by performing IPF over the cliques of the junction tree and updating the parameters θ within each clique using IS[11]. This is a modified version of the constrained decomposable case. See also [291] for a unified treatment of propagation and scaling on junction trees.

9.6.5 Conditional random fields

For an input x and output y, a CRF is defined by a conditional distribution [283, 181]

$$p(y|x) = \frac{1}{Z(x)}\prod_k \phi_k(y, x) \tag{9.6.53}$$

for (positive) potentials $\phi_k(y, x)$. To make learning more straightforward, the potentials are usually defined as $\exp(\lambda_k f_k(y, x))$ for fixed functions $f(y, x)$ and parameters λ_k. In this case the distribution of the output conditioned on the input is

$$p(y|x, \lambda) = \frac{1}{Z(x, \lambda)}\prod_k \exp(\lambda_k f_k(y, x)) \tag{9.6.54}$$

CRFs can also be viewed simply as Markov networks with exponential form potentials, as in section(9.6.4). Equation(9.6.54) is equivalent to equation (9.6.39) where the parameters θ are here denoted by λ and the variables x are here denoted by y. In the CRF case the inputs x simply have the effect of determining the feature $f_k(y, x)$.

For an i.i.d. dataset of input-outputs, $\mathcal{D} = \{(x^n, y^n), n = 1, \ldots, N\}$, training based on conditional maximum likelihood requires the maximisation of

$$L(\lambda) \equiv \sum_{n=1}^{N} \log p(y^n|x^n, \lambda) = \sum_{n=1}^{N} \sum_k \lambda_k f_k(y^n, x^n) - \log Z(x^n, \lambda) \tag{9.6.55}$$

In general no closed form solution for the optimal λ exists and this needs to be determined numerically. The methods we've discussed, such as iterative scaling may be readily adapted to this optimisation problem, although in practice gradient based techniques are to be preferred[212]. For completeness we describe gradient based training. The gradient has components

$$\frac{\partial}{\partial \lambda_i} L = \sum_n \left(f_i(y^n, x^n) - \langle f_i(y, x^n) \rangle_{p(y|x^n, \lambda)} \right) \tag{9.6.56}$$

The terms $\langle f_i(y, x^n) \rangle_{p(y|x^n, \lambda)}$ can be problematic and their tractability depends on the structure of the potentials. For a multivariate y, provided the structure of the cliques defined on subsets of y is singly-connected, then computing the average is generally tractable. More generally, provided the cliques of the resulting junction tree have limited width, then exact marginals are available. An example of this is given for a linear-chain CRF in section(23.4.3) – see also example(9.11) below.

Another quantity often useful for numerical optimisation is the Hessian which has components

$$\frac{\partial^2}{\partial \lambda_i \partial \lambda_j} L = \sum_n \left(\langle f_i(y, x^n) \rangle \langle f_j(y, x^n) \rangle - \langle f_i(y, x^n) f_j(y, x^n) \rangle \right) \tag{9.6.57}$$

where the averages above are with respect to $p(y|x^n, \lambda)$. This expression is a (negated) sum of covariance elements, and is therefore negative (semi) definite. Hence the function $L(\lambda)$ is concave and has only a single global optimum. In practice CRFs often have many thousands if not millions of parameters λ so that computing the Newton update associated with the inverse of the Hessian may be prohibitively expensive. In this case alternative optimisation methods such as conjugate gradients are preferable.

In practice regularisation terms are often added to prevent overfitting (see section(13.2.2) for a discussion of regularisation) . Using a term

$$-\sum_k c_k^2 \lambda_k^2 \tag{9.6.58}$$

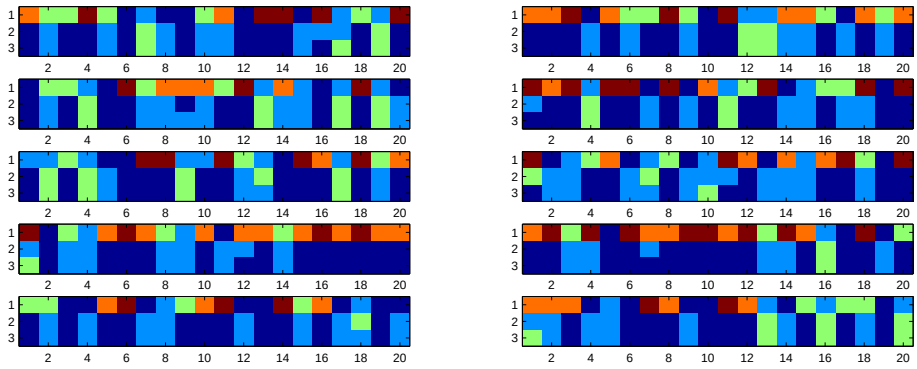
for positive regularisation constants c_k^2 discourages the weights λ from being too large. This term is also negative definite and hence the overall objective function remains concave.

Once trained a CRF can be used for predicting the output distribution for a novel input x^*. The most likely output y^* is equivalently given by

$$y^* = \underset{y}{\text{argmax}} \ \log p(y|x^*) = \underset{y}{\text{argmax}} \ \sum_k \lambda_k f_k(y, x^*) - \log Z(x^*, \lambda) \tag{9.6.59}$$

Since the normalisation term is independent of y, finding the most likely output is equivalent to

$$y^* = \underset{y}{\text{argmax}} \ \sum_k \lambda_k f_k(y, x^*) \tag{9.6.60}$$



(a) (b)

Figure 9.20: **(a)**: Training results for a linear chain CRF. There are 5 training sequences, one per subpanel. In each subpanel the top row corresponds to the input sequence $x_{1:20}$, $x_t \in \{1, \ldots, 5\}$ (each state represented by a different colour). The middle row is the correct output sequence $y_{1:20}$, $y_t \in \{1, 2, 3\}$ (each state represented by a different colour). Together the input and output sequences make the training data $\mathcal{D}$. The bottom row contains the most likely output sequence given the trained CRF, $\arg\max_{y_{1:20}} p(y_{1:20}|x_{1:20}, \mathcal{D})$. **(b)**: Five additional test input sequences (top), correct output sequence (middle) and predicted output sequence (bottom).

Natural language processing

In a natural language processing application, x_t might represent a word and y_t a corresponding linguistic tag ('noun','verb', *etc.*). A more suitable form in this case is to constrain the CRF to be of the form

$$\exp\left(\sum_k \mu_k g_k(y_t, y_{t-1}) + \sum_l \rho_l h_l(y_t, x_t)\right) \tag{9.6.61}$$

for binary functions g_k and h_l and parameters μ_k and ρ_l. The grammatical structure of tag-tag transitions is encoded in $g_k(y_t, y_{t-1})$ and linguistic tag information in $h_k(y_t, x_t)$, with the importance of these being determined by the corresponding parameters[181]. In this case inference of the marginals $\langle y_t y_{t-1}|x_{1:T}\rangle$ is straightforward since the factor graph corresponding to the inference problem is a linear chain.

Variants of the linear chain CRF are used heavily in natural language processing, including part-of-speech tagging and machine translation (in which the input sequence x represents a sentence say in English and the output sequence y the corresponding translation into French). See, for example, [230].

Example 9.11 (Linear chain CRF)**.** We consider a CRF with $X = 5$ input states and $Y = 3$ output states of the form

$$p(y_{1:T}|x_{1:T}) = \prod_{t=2}^{T} e^{\sum_k \mu_k g_k(y_t, y_{t-1}) + \sum_l \rho_l h_l(y_t, x_t)} \tag{9.6.62}$$

Here the binary functions $g_k(y_t, y_{t-1}) = \mathbb{I}[y_t = a_k]\mathbb{I}[y_{t-1} = b_k]$, $k = 1, \ldots, 9$, $a_k \in \{1, 2, 3\}$, $b_k \in \{1, 2, 3\}$, simply index the transitions between two consecutive outputs. The binary functions $h_l(y_t, x_t) = \mathbb{I}[y_t = a_l]\mathbb{I}[x_t = c_l]$, $l = 1, \ldots, 15$, $a_l \in \{1, 2, 3\}$, $c_l \in \{1, 2, 3, 4, 5\}$, index the translation of the input to the output. There are therefore $9 + 15 = 24$ parameters in total. In fig(9.20) we plot the training and test results based on a small set of data. As we see, the model learns to predict the output well, both for the training and test data. The training of the CRF is obtained using 50 iterations of gradient ascent with a learning rate of 0.1. See `demoLinearCRF.m`.

9.6.6 Pseudo likelihood

Consider a MN on variables $\mathbf{x}$ with $\dim(\mathbf{x}) = D$ of the form

$$p(\mathbf{x}|\theta) = \frac{1}{Z}\prod_c \phi_c(\mathcal{X}_c|\theta_c) \tag{9.6.63}$$

For all but specially constrained ϕ_c, the partition function Z will be intractable and the likelihood of a set of i.i.d. data intractable as well. A surrogate is to use the *pseudo likelihood* of each variable conditioned on all other variables (which is equivalent to conditioning on only the variable's neighbours for a MN)

$$L'(\theta) = \sum_{n=1}^{N}\sum_{i=1}^{D} \log p(x_i^n|\mathbf{x}_{\setminus i}^n|\theta) \tag{9.6.64}$$

The terms $p(x_i^n|\mathbf{x}_{\setminus i}^n|\theta)$ are usually straightforward to work out since they require finding the normalisation of a univariate distribution only. In this case the gradient can be computed exactly, and learning of the parameters θ carried out. In general, the solution found does not correspond to the maximum likelihood solution. However, at least for some special cases, such as the Boltzmann machine, this forms a consistent estimator[153].

9.6.7 Learning the structure

Learning the structure of a Markov network can also be based on independence tests, as for belief networks in section(9.5). A criterion for finding a MN on a set of nodes $\mathcal{X}$ is to use the fact that no edge exits between x and y if, conditioned on all other nodes, x and y are deemed independent. This is the pairwise Markov property described in section(4.2.1). By checking $x \perp\!\!\!\perp y|\,\mathcal{X}\setminus\{x,y\}$ for every pair of variables x and y, this edge deletion approach in principle reveals the structure of the network[237]. For learning the structure from an oracle, this method is sound. However, a practical difficulty in the case where the independencies are determined from data is that checking if $x \perp\!\!\!\perp y|\,\mathcal{X}\setminus\{x,y\}$ requires in principle enormous amounts of data. The reason for this is that the conditioning selects only those parts of the dataset consistent with the conditioning. In practice this will result in very small numbers of remaining datapoints, and estimating independencies on this basis is unreliable.

The Markov boundary criterion[237] uses the local Markov property, section(4.2.1), namely that conditioned on its neighbours, a variable is independent of all other variables in the graph. By starting with a variable x and an empty neighbourhood set, one can progressively include neighbours, testing if their inclusion renders the remaining non-neighbours independent of x. A difficultly with this is that, if one doesn't have the correct Markov boundary, then including a variable in the neighbourhood set may be deemed necessary. To see this, consider a network which corresponds to a linear chain and that x is at the edge of the chain. In this case, only the nearest neighbour of x is in the Markov boundary of x. However, if this nearest neighbour were not currently in the set, then any other non-nearest neighbour would be included, even though this is not strictly required. To counter this, the neighbourhood variables included in the neighbourhood of x may be later removed if they are deemed superfluous to the boundary[113].

In cases where specific constraints are imposed, such as learning structures whose resulting triangulation has a bounded tree-width, whilst still formally difficult, approximate procedures are available[276].

In terms of network scoring methods for undirected networks, computing a score is hampered by the fact that the parameters of each clique become coupled in the normalisation constant of the distribution. This issue can be addressed using hyper Markov priors[80].

9.7 Summary

- For discrete belief networks, it is particularly convenient to use a Dirichlet parameter prior since this is conjugate to the categorical distribution.

- Provided we assume local and global parameter independence, the posterior over belief network tables factorises.
- Learning the structure of a belief network is more complex. The PC algorithm uses local independence tests to decide if two variables should be linked. A global alternative is based on using a network scoring method such as the model likelihood of a network structure under a Dirichlet prior.
- Learning the maximum likelihood parameters of a decomposable Markov network is straightforward and can be achieved by counting.
- For non-decomposable Markov networks, no closed form solution exists. The maximum likelihood criterion is equivalent to ensuring that clique marginals match empirical marginals. The iterative proportional fitting algorithm is a technique to set the tables to ensure these marginals match.
- For Markov networks parameterised using feature functions, iterative scaling is a maximum likelihood technique that enables individual parameter updates to be made. Gradient based approaches are also straightforward and popular in conditional random fields.

9.8 Code

`condindepEmp.m`: Bayes test and Mutual Information for empirical conditional independence
`condMI.m`: Conditional Mutual Information
`condMIemp.m`: Conditional Mutual Information of Empirical distribution
`MIemp.m`: Mutual Information of Empirical distribution

9.8.1 PC algorithm using an oracle

This demo uses an oracle to determine $x \perp\!\!\!\perp y | z$, rather than using data to determine the empirical dependence. The oracle is itself a belief network. For the partial orientation only the first 'unmarried collider' rule is implemented.
`demoPCoracle.m`: Demo of PC algorithm with an oracle
`PCskeletonOracle.m`: PC algorithm using an oracle
`PCorient.m`: Orient a skeleton

9.8.2 Demo of empirical conditional independence

For half of the experiments, the data is drawn from a distribution for which $x \perp\!\!\!\perp y | z$ is true. For the other half of the experiments, the data is drawn from a random distribution for which $x \perp\!\!\!\perp y | z$ is false. We then measure the fraction of experiments for which the Bayes test correctly decides $x \perp\!\!\!\perp y | z$. We also measure the fraction of experiments for which the Mutual Information test correctly decides $x \perp\!\!\!\perp y | z$, based on setting the threshold equal to the median of all the empirical conditional mutual information values. A similar empirical threshold can also be obtained for the Bayes' factor (although this is not strictly kosher in the pure Bayesian spirit since one should in principle set the threshold to zero). The test based on the assumed chi-squared distributed MI is included for comparison, although it seems to be impractical in these small data cases.
`demoCondIndepEmp.m`: Demo of empirical conditional independence based on data

9.8.3 Bayes Dirichlet structure learning

It is interesting to compare the result of `demoPCdata.m` with `demoBDscore.m`.
`PCskeletonData.m`: PC algorithm using empirical conditional independence
`demoPCdata.m`: Demo of PC algorithm with data
`BDscore.m`: Bayes Dirichlet (BD) score for a node given parents
`learnBayesNet.m`: Given an ancestral order and maximal parents, learn the network
`demoBDscore.m`: Demo of structure learning

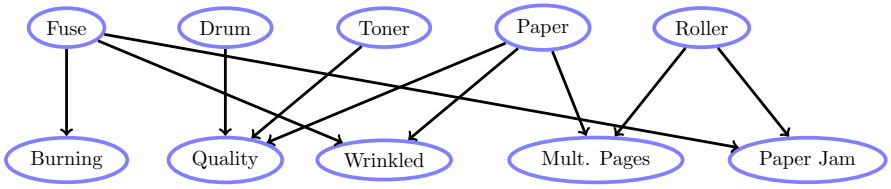


Figure 9.21: Printer Nightmare belief network. All variables are binary. The upper variables without parents are possible problems (diagnoses), and the lower variables consequences of problems (faults).

9.9 Exercises

Exercise 9.1 (Printer Nightmare)**.** *Cheapco is, quite honestly, a pain in the neck. Not only did they buy a dodgy old laser printer from StopPress and use it mercilessly, but try to get away with using substandard components and materials. Unfortunately for StopPress, they have a contract to maintain Cheapco's old warhorse, and end up frequently sending the mechanic out to repair the printer. They decide to make a statistical model of Cheapco's printer, so that they will have a reasonable idea of the fault based only on the information that Cheapco's secretary tells them on the phone. In that way, StopPress hopes to be able to send out to Cheapco only a junior repair mechanic, having most likely diagnosed the fault over the phone.*

Based on the manufacturer's information, StopPress has a good idea of the dependencies in the printer, and what is likely to directly affect other printer components. The belief network in fig(9.21) represents these assumptions. However, the specific way that Cheapco abuse their printer is a mystery, so that the exact probabilistic relationships between the faults and problems is idiosyncratic to Cheapco. StopPress has the following table of faults in which each column represents a visit.

fuse assembly malfunction	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1
drum unit	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0
toner out	1	1	0	0	0	1	0	1	0	0	0	1	0	0	0
poor paper quality	1	0	1	0	1	0	1	0	1	1	0	1	1	0	0
worn roller	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1
burning smell	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
poor print quality	1	1	1	0	1	1	0	1	0	0	1	1	0	0	0
wrinkled pages	0	0	1	0	0	0	0	0	1	0	0	0	1	1	1
multiple pages fed	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1
paper jam	0	0	1	1	0	0	1	1	1	1	0	0	0	1	0

1. *The above table is contained in* `printer.mat`*. Learn all table entries on the basis of maximum likelihood.*

2. *Program the belief network using the tables maximum likelihood tables and* BRMLtoolbox*. Compute the probability that there is a fuse assembly malfunction given that the secretary complains there is a burning smell and that the paper is jammed, and that there are no other problems.*

3. *Repeat the above calculation using a Bayesian method in which a flat Beta prior is used on all tables. Continue to use these tables for the remainder of this question.* ++

4. *Given the above information from the secretary, what is the most likely joint diagnosis over the diagnostic variables – that is the joint most likely* $p(Fuse, Drum, Toner, Paper, Roller|\mathsf{evidence})$*? Use the max-absorption method on the associated junction tree.*

5. *Compute the joint most likely state of the distribution*

$$p(Fuse, Drum, Toner, Paper, Roller|\mathsf{burning\ smell}, \mathsf{paper\ jammed})$$

Explain how to compute this efficiently using the max-absorption method.

Exercise 9.2. *Consider data* $x^n, n = 1, \ldots, N$*. Show that for a Gaussian distribution, the maximum likelihood estimator of the mean is* $\hat{m} = \frac{1}{N}\sum_{n=1}^{N} x^n$ *and variance is* $\hat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^{N}(x^n - \hat{m})^2$*.*

Exercise 9.3. *A training set consists of one dimensional examples from two classes. The training examples from class 1 are*

$$0.5, 0.1, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.35, 0.25 \tag{9.9.1}$$

and from class 2 are

$$0.9, 0.8, 0.75, 1.0 \tag{9.9.2}$$

Fit a (one dimensional) Gaussian using Maximum Likelihood to each of these two classes. Also estimate the class probabilities p_1 and p_2 using maximum likelihood. What is the probability that the test point $x = 0.6$ belongs to class 1?

Exercise 9.4. *For a set of N observations (training data), $\mathcal{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$, and independently gathered observations, the log likelihood for a belief network to generate $\mathcal{X}$ is*

$$\log p(\mathcal{X}) = \sum_{n=1}^{N} \sum_{i=1}^{K} \log p\left(x_i^n | \mathrm{pa}\left(x_i^n\right)\right) \tag{9.9.3}$$

We define the notation

$$\theta_s^i(t) = p(x_i = s | \mathrm{pa}\left(x_i\right) = t) \tag{9.9.4}$$

representing the probability that variable x_i is in state s given the parents of variable x_i are in the vector of states $\mathbf{t}$. Using a Lagrangian

$$L \equiv \sum_{n=1}^{N} \sum_{i=1}^{K} \log p\left(x_i^n | \mathrm{pa}\left(x_i^n\right)\right) + \sum_{i=1}^{K} \sum_{t^i} \lambda_{t^i}^i \left(1 - \sum_s \theta_s^i\left(t^i\right)\right) \tag{9.9.5}$$

Show that the maximum likelihood setting of $\theta_s^i(\mathbf{t})$ is

$$\theta_s^j(t^j) = \frac{\sum_{n=1}^{N} \mathbb{I}\left[x_j^n = s\right] \mathbb{I}\left[\mathrm{pa}\left(x_j^n\right) = t^j\right]}{\sum_{n=1}^{N} \sum_s \mathbb{I}\left[x_j^n = s\right] \mathbb{I}\left[\mathrm{pa}\left(x_j^n\right) = t^j\right]} \tag{9.9.6}$$

Exercise 9.5 (Conditional Likelihood training)**.** *Consider a situation in which we partition observable variables into disjoint sets x and y and that we want to find the parameters that maximize the* conditional likelihood*,*

$$CL(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log p(y^n | x^n, \theta), \tag{9.9.7}$$

for a set of training data $\{(x^n, y^n), n = 1, \ldots, N\}$. All data is assumed generated from the same distribution $p(x, y|\theta^0) = p(y|x, \theta^0)p(x|\theta^0)$ for some unknown parameter θ^0. In the limit of a large amount of i.i.d. training data, does $CL(\theta)$ have an optimum at θ^0?

Exercise 9.6 (Moment Matching)**.** *One way to set parameters of a distribution is to match the moments of the distribution to the empirical moments. This sometimes corresponds to maximum likelihood (for the Gaussian distribution for example), though generally this is not consistent with maximum likelihood.*
For data with mean m and variance s, show that to fit a Beta distribution $B(x|\alpha, \beta)$ by moment matching, we use

$$\alpha = \frac{m(m - m^2 - s)}{s}, \quad \beta = \alpha \frac{1 - m}{m} \tag{9.9.8}$$

Does this correspond to maximum likelihood?

Exercise 9.7. *For i.i.d. data $0 \leq x^n \leq 1$, $n = 1, \ldots, N$, generated from a Beta distribution $B(x|a, b)$, show that the log likelihood is given by*

$$L(a, b) \equiv (a-1)\sum_{n=1}^{N} \log x^n + (b-1)\sum_{n=1}^{N} \log(1-x^n) - N \log B(a, b) \tag{9.9.9}$$

where $B(a, b)$ is the Beta function. Show that the derivatives are

$$\frac{\partial}{\partial a} L = \sum_{n=1}^{N} \log x^n - N\psi(a) + N\psi(a+b), \quad \frac{\partial}{\partial b} L = \sum_{n=1}^{N} \log(1-x^n) - N\psi(b) + N\psi(a+b) \tag{9.9.10}$$ @@

where $\psi(x) \equiv d \log \Gamma(x)/dx$ is the digamma function, and suggest a method to learn the parameters a, b.

Exercise 9.8. *Consider the Boltzmann machine as defined in example(9.8). Write down the pseudo likelihood for a set of i.i.d. data $\mathbf{v}^1, \ldots, \mathbf{v}^N$ and derive the gradient of this with respect to w_{ij}, $i \neq j$.*

Exercise 9.9. *Show that the model likelihood equation (9.4.53) can be written explicitly as*

$$p(\mathcal{D}|M) = \prod_k \prod_j \frac{\Gamma\left(\sum_i u_i(v_k; j)\right)}{\Gamma\left(\sum_i u_i'(v_k; j)\right)} \prod_i \left[\frac{\Gamma\left(u_i'(v_k; j)\right)}{\Gamma\left(u_i(v_k; j)\right)}\right] \tag{9.9.11}$$

Exercise 9.10. *Define the set $\mathcal{N}$ as consisting of 8 node belief networks in which each node has at most 2 parents. For a given ancestral order a, the restricted set is written $\mathcal{N}_a$*

1. *How many belief networks are in $\mathcal{N}_a$?*
2. *What is the computational time to find the optimal member of $\mathcal{N}_a$ using the Bayesian Dirichlet score, assuming that computing the BD score of any member of $\mathcal{N}_a$ takes 1 second and bearing in mind the decomposability of the BD score.*
3. *Estimate the time required to find the optimal member of $\mathcal{N}$?* @@

Exercise 9.11. *For the Markov network*

$$p(x, y, z) = \frac{1}{Z}\phi_1(x, y)\phi_2(y, z) \tag{9.9.12}$$

derive an iterative scaling algorithm to learn the unconstrained tables $\phi_1(x, y)$ and $\phi_2(x, y)$ based on a set of i.i.d. data $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$.

Exercise 9.12. *In section(9.6.4) we considered maximum likelihood learning of a Markov network $p(\mathcal{X}) \propto \prod_c \phi_c(\mathcal{X}_c)$ with parameters θ_c and potentials of the form*

$$\phi_c(\mathcal{X}_c) = \exp\left(\theta_c f_c(\mathcal{X}_c)\right) \tag{9.9.13}$$

with the constraint $f_c(\mathcal{X}_c) \geq 0$. Our interest here is to drop this positive constraint on $f_c(\mathcal{X}_c)$. By considering

$$\sum_c \theta_c f_c(\mathcal{X}_c) = \sum_c p_c \frac{\theta_c f_c(\mathcal{X}_c)}{p_c} \tag{9.9.14}$$

for auxiliary variables $p_c > 0$ such that $\sum_c p_c = 1$, explain how to derive a form of iterative scaling training algorithm for general f_c in which each parameter θ_c can be updated separately.

Exercise 9.13. *Write a* MATLAB *routine* `A = ChowLiu(X)` *where X is a $D \times N$ data matrix containing a multivariate datapoint on each column that returns a Chow-Liu maximum likelihood tree for X. The tree structure is to be returned in the sparse matrix A. You may find the routine* `spantree.m` *useful. The file* `ChowLiuData.mat` *contains a data matrix for 10 variables. Use your routine to find the maximum likelihood Chow Liu tree, and draw a picture of the resulting DAG with edges oriented away from variable 1.*

Exercise 9.14. *Show that for graph with $N > 1$ nodes, there are at least* ++

$$\prod_{n=1}^{N} 2^{n-1} = 2^{N(N-1)/2}$$

and less than $N! 2^{N(N-1)/2}$ valid DAGs.

CHAPTER 10

Naive Bayes

So far we've discussed methods in some generality without touching much on how we might use the methods in a practical setting. Here we discuss one of the simplest methods that is widely used in practice to classify data. This is a useful junction since it enables us to discuss the issues of parameter learning from data and also (constrained) structure learning.

10.1 Naive Bayes and Conditional Independence

We shall discuss machine learning concepts in some detail in chapter(13). Here, we require only the intuitive concept of classification, which means giving a discrete label to an input. For example, one might wish to classify an input image into one of two classes - male or female. Naive Bayes (NB) is a popular classification method and aids our discussion of conditional independence, overfitting and Bayesian methods. In NB, we form a joint model of a D-dimensional attribute (input) vector $\mathbf{x}$ and the corresponding class label c

$$p(\mathbf{x}, c) = p(c) \prod_{i=1}^{D} p(x_i|c) \tag{10.1.1}$$

whose belief network is depicted in fig(10.1a). Coupled with a suitable choice for each conditional distribution $p(x_i|c)$, we can then use Bayes' rule to form a classifier for a novel input vector $\mathbf{x}^*$:

$$p(c|\mathbf{x}^*) = \frac{p(\mathbf{x}^*|c)p(c)}{p(\mathbf{x}^*)} = \frac{p(\mathbf{x}^*|c)p(c)}{\sum_c p(\mathbf{x}^*|c)p(c)} \tag{10.1.2}$$

In practice it is common to consider only two classes $\text{dom}(c) = \{0, 1\}$. The theory we describe below is valid for any number of classes c, though our examples are restricted to the binary class case. Also, the attributes x_i are often taken to be binary, as we shall do initially below as well. The extension to more than two attribute states, or continuous attributes is straightforward.

Example 10.1. EZsurvey.org partitions radio station listeners into two groups – the 'young' and 'old'. They assume that, given the knowledge that a customer is either 'young' or 'old', this is sufficient to determine whether or not a customer will like a particular radio station, independent of their likes or dislikes for any other stations:

$$p(r_1, r_2, r_3, r_4|age) = p(r_1|age)p(r_2|age)p(r_3|age)p(r_4|age) \tag{10.1.3}$$

where each of the variables r_1, r_2, r_3, r_4 can take the states like or dislike, and the 'age' variable can take the value young or old. Thus the information about the age of the customer determines the individual

radio station preferences without needing to know anything else. To complete the specification, given that a customer is young, she has a 95% chance to like Radio1, a 5% chance to like Radio2, a 2% chance to like Radio3 and a 20% chance to like Radio4. Similarly, an old listener has a 3% chance to like Radio1, an 82% chance to like Radio2, a 34% chance to like Radio3 and a 92% chance to like Radio4. They know that 90% of the listeners are old.

Given this model, and the fact that a new customer likes Radio1, and Radio3, but dislikes Radio2 and Radio4, what is the probability that the new customer is young? This is given by

$$p(\text{young}|r_1=\text{like}, r_2=\text{dislike}, r_3=\text{like}, r_4=\text{dislike}) = \frac{p(r_1=\text{like}, r_2=\text{dislike}, r_3=\text{like}, r_4=\text{dislike}|\text{young})p(\text{young})}{\sum_{age} p(r_1=\text{like}, r_2=\text{dislike}, r_3=\text{like}, r_4=\text{dislike}|age)p(age)} \tag{10.1.4}$$

Using the naive Bayes structure, the numerator above is given by

$$p(r_1=\text{like}|\text{young})p(r_2=\text{dislike}|\text{young})p(r_3=\text{like}|\text{young})p(r_4=\text{dislike}|\text{young})p(\text{young}) \tag{10.1.5}$$

Plugging in the values we obtain

$$0.95 \times 0.95 \times 0.02 \times 0.8 \times 0.1 = 0.0014$$

The denominator is given by this value plus the corresponding term evaluated assuming the customer is old,

$$0.03 \times 0.18 \times 0.34 \times 0.08 \times 0.9 = 1.3219 \times 10^{-4}$$

Which gives

$$p(\text{young}|r_1=\text{like}, r_2=\text{dislike}, r_3=\text{like}, r_4=\text{dislike}) = \frac{0.0014}{0.0014 + 1.3219 \times 10^{-4}} = 0.9161 \tag{10.1.6}$$

10.2 Estimation using Maximum Likelihood

Learning the table entries for NB is a straightforward application of the more general BN learning discussed in section(9.3). For a fully observed dataset, maximum likelihood learning of the table entries corresponds to counting the number of occurrences in the training data, as we show below. This is a useful exercise to reinforce and make more concrete the general theory of section(9.3).

10.2.1 Binary attributes

Consider a dataset $\{(\mathbf{x}^n, c^n), n = 1, \ldots, N\}$ of binary attributes, $x_i^n \in \{0, 1\}$, $i = 1, \ldots, D$ and associated class label c^n. The number of datapoints from class $c = 0$ is denoted n_0 and the number from class $c = 1$ is denoted n_1. For each attribute of the two classes, we need to estimate the values $p(x_i = 1|c) \equiv \theta_i^c$. The

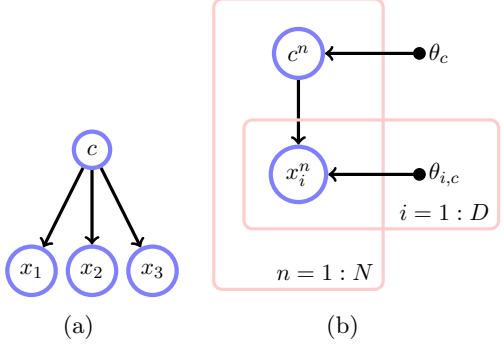


Figure 10.1: Naive Bayes classifier. **(a)**: The central assumption is that given the class c, the attributes x_i are independent. **(b)**: Assuming the data is i.i.d., maximum likelihood learns the optimal parameters θ_c of the distribution $p(c)$ and the parameters $\theta_{i,c}$ of the class-dependent attribute distributions $p(x_i|c)$.

other probability, $p(x_i = 0|c)$ is given by the normalisation requirement, $p(x_i = 0|c) = 1 - p(x_i = 1|c) = 1 - \theta_i^c$.

Based on the NB conditional independence assumption the probability of observing a vector $\mathbf{x}$ can be compactly written[1]

$$p(\mathbf{x}|c) = \prod_{i=1}^{D} p(x_i|c) = \prod_{i=1}^{D} (\theta_i^c)^{x_i} (1 - \theta_i^c)^{1-x_i} \tag{10.2.1}$$

In the above expression, x_i is either 0 or 1 and hence each i term contributes a factor θ_i^c if $x_i = 1$ or $1 - \theta_i^c$ if $x_i = 0$. Together with the assumption that the training data is i.i.d. generated, the log likelihood of the attributes and class labels is

$$L = \sum_n \log p(\mathbf{x}^n, c^n) = \sum_n \log p(c^n) \prod_i p(x_i^n|c^n) \tag{10.2.2}$$

$$= \left\{ \sum_{i,n} x_i^n \log \theta_i^{c^n} + (1 - x_i^n) \log(1 - \theta_i^{c^n}) \right\} + n_0 \log p(c = 0) + n_1 \log p(c = 1) \tag{10.2.3}$$

This can be written more explicitly in terms of the parameters as

$$L = \sum_{i,n} \left\{ \mathbb{I}\left[x_i^n = 1, c^n = 0\right] \log \theta_i^0 + \mathbb{I}\left[x_i^n = 0, c^n = 0\right] \log(1 - \theta_i^0) + \mathbb{I}\left[x_i^n = 1, c^n = 1\right] \log \theta_i^1 \right.$$
$$\left. + \mathbb{I}\left[x_i^n = 0, c^n = 1\right] \log(1 - \theta_i^1) \right\} + n_0 \log p(c = 0) + n_1 \log p(c = 1) \tag{10.2.4}$$

We can find the maximum likelihood optimal θ_i^c by differentiating *w.r.t.* θ_i^c and equating to zero, giving

$$\theta_i^c = p(x_i = 1|c) = \frac{\sum_n \mathbb{I}\left[x_i^n = 1, c^n = c\right]}{\sum_n \mathbb{I}\left[x_i^n = 0, c^n = c\right] + \mathbb{I}\left[x_i^n = 1, c^n = c\right]} \tag{10.2.5}$$

$$= \frac{\text{number of times } x_i = 1 \text{ for class c}}{\text{number of datapoints in class c}} \tag{10.2.6}$$

Similarly, optimising equation (10.2.3) with respect to $p(c)$ gives

$$p(c) = \frac{\text{number of times class c occurs}}{\text{total number of data points}} \tag{10.2.7}$$

These results are consistent with our general theory in section(9.3) that maximum likelihood corresponds to setting tables by counting.

Classification boundary

We classify a novel input $\mathbf{x}^*$ as class 1 if

$$p(c = 1|\mathbf{x}^*) > p(c = 0|\mathbf{x}^*) \tag{10.2.8}$$

Using Bayes' rule and writing the log of the above expression, this is equivalent to

$$\log p(\mathbf{x}^*|c = 1) + \log p(c = 1) - \log p(\mathbf{x}^*) > \log p(\mathbf{x}^*|c = 0) + \log p(c = 0) - \log p(\mathbf{x}^*) \tag{10.2.9}$$

From the definition of the classifier, this is equivalent to (the normalisation constant $-\log p(\mathbf{x}^*)$ can be dropped from both sides)

$$\sum_i \log p(x_i^*|c = 1) + \log p(c = 1) > \sum_i \log p(x_i^*|c = 0) + \log p(c = 0) \tag{10.2.10}$$

[1] This makes use of the general notation that any quantity raised to the power 0 is 1, *i.e.* $x^0 \equiv 1$. Unfortunately, there is a potential general mathematical notational confusion with x^y meaning possibly x raised to the power y, or alternatively that y is simply indexing a set of x variables. This potential conflict will not hopefully arise too often, and can most often be resolved by reference to the meaning of the symbols involved.

Using the binary encoding $x_i \in \{0, 1\}$, we therefore classify $\mathbf{x}^*$ as class 1 if

$$\sum_i \left\{x_i^* \log \theta_i^1 + (1 - x_i^*) \log(1 - \theta_i^1)\right\} + \log p(c = 1) > \sum_i \left\{x_i^* \log \theta_i^0 + (1 - x_i^*) \log(1 - \theta_i^0)\right\} + \log p(c = 0) \tag{10.2.11}$$

This decision rule can be expressed in the form: classify $\mathbf{x}^*$ as class 1 if $\sum_i w_i x_i^* + a > 0$ for some suitable choice of weights w_i and constant a, see exercise(10.4). The interpretation is that $\mathbf{w}$ specifies a hyperplane in the attribute space and $\mathbf{x}^*$ is classified as 1 if it lies on the positive side of the hyperplane.

Example 10.2 (Are they Scottish?)**.** Consider the following vector of binary attributes:

$$\text{(shortbread, lager, whiskey, porridge, football)} \tag{10.2.12}$$

A vector $\mathbf{x} = (1, 0, 1, 1, 0)^\mathsf{T}$ would describe that a person likes shortbread, does not like lager, drinks whiskey, eats porridge, and has not watched England play football. Together with each vector $\mathbf{x}$, there is a label *nat* describing the nationality of the person, $\text{dom}(nat) = \{\mathsf{scottish}, \mathsf{english}\}$, see fig(10.2).

We wish to classify the vector $\mathbf{x} = (1, 0, 1, 1, 0)^\mathsf{T}$ as either scottish or english. Using Bayes' rule:

$$p(\mathsf{scottish}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathsf{scottish})p(\mathsf{scottish})}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathsf{scottish})p(\mathsf{scottish})}{p(\mathbf{x}|\mathsf{scottish})p(\mathsf{scottish}) + p(\mathbf{x}|\mathsf{english})p(\mathsf{english})} \tag{10.2.13}$$

By maximum likelihood the 'prior' class probability $p(\mathsf{scottish})$ is given by the fraction of people in the database that are Scottish, and similarly $p(\mathsf{english})$ is given as the fraction of people in the database that are English. This gives $p(\mathsf{scottish}) = 7/13$ and $p(\mathsf{english}) = 6/13$.

For $p(\mathbf{x}|nat)$ under the Naive Bayes assumption:

$$p(\mathbf{x}|nat) = p(x_1|nat)p(x_2|nat)p(x_3|nat)p(x_4|nat)p(x_5|nat) \tag{10.2.14}$$

so that knowing whether not someone is Scottish, we don't need to know anything else to calculate the probability of their likes and dislikes. Based on the table in fig(10.2) and using maximum likelihood we have:

$$\begin{array}{llll} p(x_1 = 1|\mathsf{english}) & = 1/2 & p(x_1 = 1|\mathsf{scottish}) & = 1 \\ p(x_2 = 1|\mathsf{english}) & = 1/2 & p(x_2 = 1|\mathsf{scottish}) & = 4/7 \\ p(x_3 = 1|\mathsf{english}) & = 1/3 & p(x_3 = 1|\mathsf{scottish}) & = 3/7 \\ p(x_4 = 1|\mathsf{english}) & = 1/2 & p(x_4 = 1|\mathsf{scottish}) & = 5/7 \\ p(x_5 = 1|\mathsf{english}) & = 1/2 & p(x_5 = 1|\mathsf{scottish}) & = 3/7 \end{array} \tag{10.2.15}$$

For $\mathbf{x} = (1, 0, 1, 1, 0)^\mathsf{T}$, we get

$$p(\mathsf{scottish}|\mathbf{x}) = \frac{1 \times \frac{3}{7} \times \frac{3}{7} \times \frac{5}{7} \times \frac{4}{7} \times \frac{7}{13}}{1 \times \frac{3}{7} \times \frac{3}{7} \times \frac{5}{7} \times \frac{4}{7} \times \frac{7}{13} + \frac{1}{2} \times \frac{1}{2} \times \frac{1}{3} \times \frac{1}{2} \times \frac{1}{2} \times \frac{6}{13}} = 0.8076 \tag{10.2.16}$$

Since this is greater than 0.5, we would classify this person as being Scottish.

Small data counts

In example(10.2), consider trying to classify the vector $\mathbf{x} = (0, 1, 1, 1, 1)^\mathsf{T}$. In the training data, all Scottish people say they like shortbread. This means that for this particular $\mathbf{x}$, $p(\mathbf{x}, \mathsf{scottish}) = 0$, and therefore that we make the extremely confident classification $p(\mathsf{scottish}|\mathbf{x}) = 0$. This demonstrates a difficulty using maximum likelihood with sparse data. One way to ameliorate this is to smooth the probabilities, for example by adding a small number to the frequency counts of each attribute. This ensures that there are

0	1	1	1	0	0
0	0	1	1	1	0
1	1	0	0	0	0
1	1	0	0	0	1
1	0	1	0	1	0

(a)

1	1	1	1	1	1	1
0	1	1	1	1	0	0
0	0	1	0	0	1	1
1	0	1	1	1	1	0
1	1	0	0	1	0	0

(b)

Figure 10.2: **(a)**: English tastes for 6 people over attributes $(shortbread, lager, whiskey, porridge, football)$. Each column represents the tastes of an individual. **(b)**: Scottish tastes for 7 people.

no zero probabilities in the model. An alternative is to use a Bayesian approach that discourages extreme probabilities, as discussed in section(10.3).

Potential pitfalls with encoding

In many off-the-shelf packages implementing Naive Bayes, binary attributes are assumed. In practice, however, the case of non-binary attributes often occurs. Consider the following attribute : age. In a survey, a person's age is marked down using the variable $a \in 1, 2, 3$. $a = 1$ means the person is between 0 and 10 years old, $a = 2$ means the person is between 10 and 20 years old, $a = 3$ means the person is older than 20. One way to transform the variable a into a binary representation would be to use three binary variables (a_1, a_2, a_3) with $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$ representing $a = 1, a = 2, a = 3$ respectively. This is called $1 - of - M$ *coding* since only 1 of the binary variables is active in encoding the M states. By construction, this means that the variables a_1, a_2, a_3 are dependent – for example, if we know that $a_1 = 1$, we know that $a_2 = 0$ and $a_3 = 0$. Regardless of any class conditioning, these variables will always be dependent, contrary to the assumption of Naive Bayes. A correct approach is to use variables with more than two states, as explained in section(10.2.2).

10.2.2 Multi-state variables

The extension of the above method to class variables c with more than two states is straightforward. We concentrate here therefore on extending the attribute variables to having more than two states. For a variable x_i with states, $\mathrm{dom}(x_i) = \{1, \ldots, S\}$, the likelihood of observing a state $x_i = s$ is denoted

$$p(x_i = s|c) = \theta_s^i(c) \tag{10.2.17}$$

with $\sum_s p(x_i = s|c) = 1$. The class conditional likelihood of generating the i.i.d. data $\mathcal{D} = (\mathbf{x}^n, c^n), n = 1, \ldots, N$ is

$$\prod_{n=1}^{N} p(\mathbf{x}^n|c^n) = \prod_{n=1}^{N} \prod_{i=1}^{D} \prod_{s=1}^{S} \prod_{c=1}^{C} \theta_s^i(c)^{\mathbb{I}\left[x_i^n = s\right]\mathbb{I}[c^n = c]}. \tag{10.2.18}$$

The effect of the indicators is that only those terms $\theta_s^i(c)$ survive for which there are attributes i in state s for class c. This then gives the class conditional log-likelihood

$$L(\theta) = \sum_{n=1}^{N} \sum_{i=1}^{D} \sum_{s=1}^{S} \sum_{c=1}^{C} \mathbb{I}\left[x_i^n = s\right] \mathbb{I}\left[c^n = c\right] \log \theta_s^i(c) \tag{10.2.19}$$

We can optimize with respect to the parameters θ using a Lagrange multiplier (one for each of the attributes i and classes c) to ensure normalisation. This gives the Lagrangian

$$\mathcal{L}(\theta, \lambda) = \sum_{n=1}^{N} \sum_{i=1}^{D} \sum_{s=1}^{S} \sum_{c=1}^{C} \mathbb{I}\left[x_i^n = s\right] \mathbb{I}\left[c^n = c\right] \log \theta_s^i(c) + \sum_{c=1}^{C} \sum_{i=1}^{D} \lambda_i^c \left(1 - \sum_{s=1}^{S} \theta_s^i(c)\right) \tag{10.2.20}$$

To find the optimum of this function we may differentiate with respect to $\theta_s^i(c)$ and equate to zero. Solving the resulting equation we obtain

$$\sum_{n=1}^{N} \frac{\mathbb{I}\left[x_i^n = s\right] \mathbb{I}\left[c^n = c\right]}{\theta_s^i(c)} = \lambda_i^c \tag{10.2.21}$$

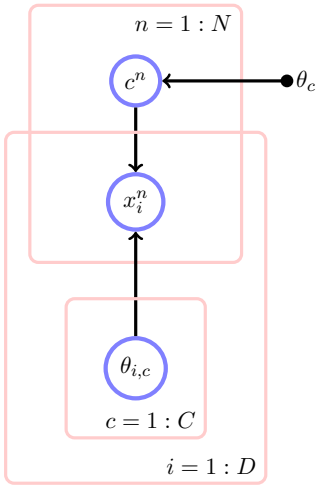


Figure 10.3: Bayesian Naive Bayes with a factorised prior on the class conditional attribute probabilities $p(x_i = s|c)$. For simplicity we assume that the class probability $\theta_c \equiv p(c)$ is learned with maximum likelihood, so that no distribution is placed over this parameter.

Hence, by normalisation,

$$\theta_s^i(c) = p(x_i = s|c) = \frac{\sum_n \mathbb{I}\left[x_i^n = s\right] \mathbb{I}\left[c^n = c\right]}{\sum_{s',n'} \mathbb{I}\left[x_i^{n'} = s'\right] \mathbb{I}\left[c^{n'} = c\right]} \tag{10.2.22}$$

The maximum likelihood setting for the parameter $p(x_i = s|c)$ equals the relative number of times that attribute i is in state s for class c.

10.2.3 Text classification

Consider a set of documents about politics, and another set about sport. Our interest is to make a method that can automatically classify a new document as pertaining to either sport or politics. We search through both sets of documents to find say the 100 most commonly occurring words (not including so-called 'stop words' such as 'a' or 'the'). Each document is then represented by a 100 dimensional vector representing the number of times that each of the words occurs in that document – the so called *bag of words* representation (this is a crude representation of the document since it discards word order). A Naive Bayes model specifies a distribution of these number of occurrences $p(x_i|c)$, where x_i is the count of the number of times word i appears in documents of type c. One can achieve this using either a multistate representation (as discussed in section(10.2.2)) or using a continuous x_i to represent the relative frequency of word i in the document. In the latter case $p(x_i|c)$ could be conveniently modelled using for example a Beta distribution.

Despite the simplicity of naive Bayes, it can classify novel documents surprisingly well[138] (although naturally the real practical methods work with many more attributes and choose them wisely). Intuitively a potential justification for the conditional independence assumption is that if we know that a document is about politics, this is a good indication of the kinds of other words we will find in the document. Because Naive Bayes is a reasonable classifier in this sense, and has minimal storage and fast training, it has been applied to time and storage critical applications, such as automatically classifying webpages into types[310], and spam filtering[8]. It also forms one of the simplest yet most commonly used basic machine learning classification routines.

10.3 Bayesian Naive Bayes

As we saw in the previous section, naive Bayes can be a powerful method for classification, yet can be overly zealous in the case of small counts. If a single attribute i has no counts for class c then, irrespective of the other attributes, the classifier will say that $\mathbf{x}$ cannot be from class c. This happens because the product of 0 with anything else remains 0. To counter overconfidence effect, we can use a simple Bayesian method.

Given a dataset $\mathcal{D} = \{(\mathbf{x}^n, c^n), n = 1, \ldots, N\}$, we predict the class c of an input $\mathbf{x}$ using

$$p(c|\mathbf{x}, \mathcal{D}) \propto p(\mathbf{x}, \mathcal{D}, c) \propto p(\mathbf{x}|\mathcal{D}, c)p(c|\mathcal{D}) \tag{10.3.1}$$ @@

For convenience we will simply set $p(c|\mathcal{D})$ using maximum likelihood

$$p(c|\mathcal{D}) = \frac{1}{N}\sum_n \mathbb{I}\left[c^n = c\right] \tag{10.3.2}$$

However, as we've seen, setting the parameters of $p(\mathbf{x}|\mathcal{D}, c)$ using maximum likelihood training can yield over-confident predictions in the case of sparse data. A Bayesian approach that addresses this difficulty uses priors on the probabilities $p(x_i = s|c) \equiv \theta_s^i(c)$ to discourage extreme values. The model is depicted in fig(10.3).

The prior

Writing $\boldsymbol{\theta}^i(c) = \left(\theta_1^i(c), \ldots, \theta_S^i(c)\right)$ as the vector of probabilities and $\theta = \left\{\boldsymbol{\theta}^i(c), i = 1, \ldots, D, c = 1, \ldots, C\right\}$, we make the global factorisation assumption (see section(9.4)) and use a prior

$$p(\theta) = \prod_{i,c} p(\boldsymbol{\theta}^i(c)) \tag{10.3.3}$$

We consider discrete x_i each of which take states from $1, \ldots, S$. In this case $p(x_i = s|c)$ corresponds to a categorical distribution, for which the conjugate prior is a Dirichlet distribution. Under the factorised prior assumption (10.3.3) we define a prior for each attribute i and class c,

$$p(\boldsymbol{\theta}^i(c)) = \text{Dirichlet}\left(\boldsymbol{\theta}^i(c)|\mathbf{u}^i(c)\right) \tag{10.3.4}$$

where $\mathbf{u}^i(c)$ is the hyperparameter vector of the Dirichlet distribution for table $p(x_i|c)$.

The posterior

Consistent with our general Bayesian BN training result in section(9.4), the parameter posterior factorises

$$p(\theta(c^*)|\mathcal{D}) = \prod_i p(\boldsymbol{\theta}^i(c^*)|\mathcal{D}) \tag{10.3.5}$$

where

$$p(\boldsymbol{\theta}^i(c^*)|\mathcal{D}) \propto p(\boldsymbol{\theta}^i(c^*)) \prod_{n:c^n=c^*} p(x_i^n|\boldsymbol{\theta}^i(c^*)) \tag{10.3.6}$$

By conjugacy, the posterior for class c^* is a Dirichlet distribution,

$$p(\boldsymbol{\theta}^i(c^*)|\mathcal{D}) = \text{Dirichlet}\left(\boldsymbol{\theta}^i(c^*)|\hat{\mathbf{u}}^i(c^*)\right) \tag{10.3.7}$$

where the vector $\hat{\mathbf{u}}^i(c^*)$ has components

$$\left[\hat{\mathbf{u}}^i(c^*)\right]_s = u_s^i(c^*) + \sum_{n:c^n=c^*} \mathbb{I}\left[x_i^n = s\right] \tag{10.3.8}$$

For Dirichlet hyperparameters $\mathbf{u}^i(c^*)$ the above equation updates the hyperparameter by the number of times variable i is in state s for class c^* data. A common default setting is to take all components of $\mathbf{u}$ to be 1.

Classification

The posterior class distribution for a novel input $\mathbf{x}^*$ is given by

$$p(c^*|\mathbf{x}^*, \mathcal{D}) \propto p(c^*, \mathbf{x}^*, \mathcal{D}) \propto p(c^*|\mathcal{D})p(\mathbf{x}^*|\mathcal{D}, c^*) = p(c^*|\mathcal{D}) \prod_i p(x_i^*|\mathcal{D}, c^*) \tag{10.3.9}$$

To compute $p(x_i^*|\mathcal{D}, c^*)$ we use

$$p(x_i^* = s|\mathcal{D}, c^*) = \int_{\boldsymbol{\theta}^i(c^*)} p(x_i^* = s, \boldsymbol{\theta}^i(c^*)|\mathcal{D}, c^*) = \int_{\boldsymbol{\theta}^i(c^*)} p(x_i^* = s|\boldsymbol{\theta}^i(c^*))p(\boldsymbol{\theta}^i(c^*)|\mathcal{D}) \tag{10.3.10}$$

$$= \int_{\boldsymbol{\theta}^i(c^*)} \theta_s^i(c^*)p(\boldsymbol{\theta}^i(c^*)|\mathcal{D}) \tag{10.3.11}$$

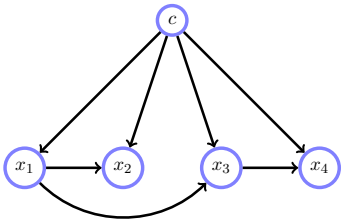


Figure 10.4: Tree Augmented Naive (TAN) Bayes. Each variable x_i has at most one parent. The maximum likelihood optimal TAN structure is computed using a modified Chow-Liu algorithm in which the conditional mutual information $\mathrm{MI}(x_i; x_j|c)$ is computed for all i, j. A maximum weight spanning tree is then found and turned into a directed graph by orienting the edges outwards from a chosen root node. The table entries can then be read off using the usual maximum likelihood counting argument.

Using the general identity

$$\int \theta_s \mathrm{Dirichlet}\left(\boldsymbol{\theta}|\mathbf{u}\right) d\boldsymbol{\theta} = \frac{1}{Z(\mathbf{u})} \int \prod_{s'} \theta_{s'}{}^{u_{s'}-1+\mathbb{I}[s'=s]} d\boldsymbol{\theta} = \frac{Z(\mathbf{u}')}{Z(\mathbf{u})} \tag{10.3.12}$$

where $Z(\mathbf{u})$ is the normalisation constant of the distribution $\mathrm{Dirichlet}\left(\cdot|\mathbf{u}\right)$ and

$$u'_s = \begin{cases} u_s & s \neq s' \\ u_s + 1 & s = s' \end{cases} \tag{10.3.13}$$

we obtain

$$p(c^*|\mathbf{x}^*, \mathcal{D}) \propto p(c^*|\mathcal{D}) \prod_i \frac{Z(\mathbf{u}^{*i}(c^*))}{Z(\hat{\mathbf{u}}^i(c^*))} \tag{10.3.14}$$

where

$$u^{*i}_s(c^*) = \hat{u}^i_s(c^*) + \mathbb{I}\left[x^*_i = s\right] \tag{10.3.15}$$

Example 10.3 (Bayesian Naive Bayes)**.** Repeating the previous analysis for the 'Are they Scottish?' data from example(10.2), the probability under a uniform Dirichlet prior for all the tables, gives a value of 0.764 for the probability that $(1, 0, 1, 1, 0)$ is Scottish, compared with a value of 0.8076 under the standard Naive Bayes assumption. See `demoNaiveBayes.m`. @@

10.4 Tree Augmented Naive Bayes

A natural extension of Naive Bayes is to relax the assumption that the attributes are independent given the class:

$$p(\mathbf{x}|c) \neq \prod_{i=1}^{D} p(x_i|c) \tag{10.4.1}$$

The question then arises – which structure should we choose for $p(\mathbf{x}|c)$? As we saw in section(9.5), learning a structure is computationally infeasible for all but very small numbers of attributes. A practical algorithm therefore requires a specific form of constraint on the structure. In section(9.5.4) we saw that we can learn single-parent tree-structured networks efficiently. Below we extend this to learning class dependent tree networks for classification.

10.4.1 Learning tree augmented Naive Bayes networks

For a distribution $p(x_1, \ldots, x_D|c)$ of the form of a tree structure with a single-parent constraint, see fig(10.4), we can readily find the class conditional maximum likelihood solution by computing the Chow-Liu tree for each class. One then adds links from the class node c to each variable and learns the class conditional probabilities from c to x, which can be read off for maximum likelihood using the usual counting argument. Note that this would generally result in a different Chow-Liu tree for each class.

Practitioners typically constrain the network to have the same structure for all classes. The maximum likelihood objective under the TAN constraint then corresponds to maximising the *conditional mutual information*[105]

$$\mathrm{MI}(x_i; x_j|c) = \langle \mathrm{KL}(p(x_i, x_j|c)|p(x_i|c)p(x_j|c)) \rangle_{p(c)} \tag{10.4.2}$$

see exercise(10.7). Once the structure is learned one subsequently sets parameters by maximum likelihood counting. Techniques to prevent overfitting are discussed in [105] and can be addressed using Dirichlet priors, as for the simpler Naive Bayes structure.

One can readily consider less restrictive structures than single-parent Belief Networks. However, finding optimal BN structures is generally computationally infeasible and heuristics are required to limit the search space.

10.5 Summary

- Naive Bayes is a simple class-conditional generative model of data that can be used to form a simple classifier.
- Bayesian training of the parameters is straightforward.
- An extension of the standard naive Bayes' model is to consider attributes with at most a single parent attribute (in addition to the class label). Finding the maximum likelihood optimal tree augmented structure is straightforward and corresponds to a maximum spanning tree problem with weights given by the class conditional mutual information.

10.6 Code

`NaiveBayesTrain.m`: Naive Bayes trained with Maximum Likelihood
`NaiveBayesTest.m`: Naive Bayes test
`NaiveBayesDirichletTrain.m`: Naive Bayes trained with Bayesian Dirichlet
`NaiveBayesDirichletTest.m`: Naive Bayes testing with Bayesian Dirichlet
`demoNaiveBayes.m`: Demo of Naive Bayes

10.7 Exercises

Exercise 10.1. *A local supermarket specializing in breakfast cereals decides to analyze the buying patterns of its customers. They make a small survey asking 6 randomly chosen people their age (older or younger than 60 years) and which of the breakfast cereals (Cornflakes, Frosties, Sugar Puffs, Branflakes) they like. Each respondent provides a vector with entries 1 or 0 corresponding to whether they like or dislike the cereal. Thus a respondent with* (1101) *would like Cornflakes, Frosties and Branflakes, but not Sugar Puffs. The older than 60 years respondents provide the following data* $(1000), (1001), (1111), (0001)$*. The younger than 60 years old respondents responded* $(0110), (1110)$*. A novel customer comes into the supermarket and says she only likes Frosties and Sugar Puffs. Using naive Bayes trained with maximum likelihood, what is the probability that she is younger than 60?*

Exercise 10.2. *A psychologist does a small survey on 'happiness'. Each respondent provides a vector with entries 1 or 0 corresponding to whether they answer 'yes' to a question or 'no', respectively. The question vector has attributes*

$$\mathbf{x} = (rich, married, healthy) \tag{10.7.1}$$

Thus, a response $(1, 0, 1)$ *would indicate that the respondent was 'rich', 'unmarried', 'healthy'. In addition, each respondent gives a value* $c = 1$ *if they are content with their lifestyle, and* $c = 0$ *if they are not. The following responses were obtained from people who claimed also to be 'content' :* $(1, 1, 1), (0, 0, 1), (1, 1, 0), (1, 0, 1)$ *and for 'not content':* $(0, 0, 0), (1, 0, 0), (0, 0, 1), (0, 1, 0), (0, 0, 0)$.

1. *Using Naive Bayes, what is the probability that a person who is 'not rich', 'married' and 'healthy' is 'content'?*

2. *What is the probability that a person who is 'not rich' and 'married' is 'content'? (That is, we do not know whether or not they are 'healthy').*

3. *Consider the following vector of attributes :*

$$x_1 = 1 \text{ if customer is younger than 20 } ; x_1 = 0 \text{ otherwise} \tag{10.7.2}$$

$$x_2 = 1 \text{ if customer is between 20 and 30 years old } ; x_2 = 0 \text{ otherwise} \tag{10.7.3}$$

$$x_3 = 1 \text{ if customer is older than 30 } ; x_3 = 0 \text{ otherwise} \tag{10.7.4}$$

$$x_4 = 1 \text{ if customer walks to work } ; x_4 = 0 \text{ otherwise} \tag{10.7.5}$$

Each vector of attributes has an associated class label 'rich' or 'poor'. Point out any potential difficulties with using your previously described approach to training using naive Bayes. Hence describe how to extend your previous naive Bayes method to deal with this dataset.

Exercise 10.3. *Whizzco decide to make a text classifier. To begin with they attempt to classify documents as either sport or politics. They decide to represent each document as a (row) vector of attributes describing the presence or absence of words.*

$$\mathbf{x} = (\text{goal}, \text{football}, \text{golf}, \text{defence}, \text{offence}, \text{wicket}, \text{office}, \text{strategy}) \tag{10.7.6}$$

Training data from sport documents and from politics documents is represented below in MATLAB *using a matrix in which each row represents the 8 attributes.*

```
xP=[1 0 1 1 1 0 1 1;  % Politics
    0 0 0 1 0 0 1 1;
    1 0 0 1 1 0 1 0;
    0 1 0 0 1 1 0 1;
    0 0 0 1 1 0 1 1;
    0 0 0 1 1 0 0 1]
```

```
xS=[1 1 0 0 0 0 0 0; % Sport
    0 0 1 0 0 0 0 0;
    1 1 0 1 0 0 0 0;
    1 1 0 1 0 0 0 1;
    1 1 0 1 1 0 0 0;
    0 0 0 1 0 1 0 0;
    1 1 1 1 1 0 1 0]
```

Using a maximum likelihood naive Bayes classifier, what is the probability that the document $\mathbf{x} = (1, 0, 0, 1, 1, 1, 1, 0)$ *is about politics?*

Exercise 10.4. *A Naive Bayes Classifier for binary attributes* $x_i \in \{0, 1\}$ *is parameterised by* $\theta_i^1 = p(x_i = 1|class = 1)$, $\theta_i^0 = p(x_i = 1|class = 0)$, *and* $p_1 = p(class = 1)$ *and* $p_0 = p(class = 0)$. *Show that the decision to classify a datapoint* $\mathbf{x}$ *as class 1 holds if* $\mathbf{w}^T\mathbf{x} + b > 0$ *for some* $\mathbf{w}$ *and* b, *and state explicitly* $\mathbf{w}$ *and* b *as a function of* $\boldsymbol{\theta}^1, \boldsymbol{\theta}^0, p_1, p_0$.

Exercise 10.5. *This question concerns spam filtering. Each email is represented by a vector*

$$x = (x_1, \ldots, x_D) \tag{10.7.7}$$

where $x_i \in \{0, 1\}$. *Each entry of the vector indicates if a particular symbol or word appears in the email. The symbols/words are*

$$money, cash, !!!, viagra, \ldots, \text{etc.} \tag{10.7.8}$$

so that for example $x_2 = 1$ if the word 'cash' appears in the email. The training dataset consists of a set of vectors along with the class label c, where $c = 1$ indicates the email is spam, and $c = 0$ not spam. Hence, the training set consists of a set of pairs $(x^n, c^n), n = 1, \ldots, N$. The naive Bayes model is given by

$$p(c, x) = p(c) \prod_{i=1}^{D} p(x_i|c) \tag{10.7.9}$$

1. *Derive expressions for the parameters of this model in terms of the training data using maximum likelihood. Assume that the data is independent and identically distributed*

$$p(c^1, \ldots, c^N, x^1, \ldots, x^N) = \prod_{n=1}^{N} p(c^n, x^n) \tag{10.7.10}$$

Explicitly, the parameters are

$$p(c = 1), p(x_i = 1|c = 1), p(x_i = 1|c = 0), i = 1, \ldots, D \tag{10.7.11}$$

2. *Given a trained model $p(x, c)$, explain how to form a classifier $p(c|x)$.*

3. *If 'viagra' never appears in the spam training data, discuss what effect this will have on the classification for a new email that contains the word 'viagra'. Explain how you might counter this effect. Explain how a spammer might try to fool a naive Bayes spam filter.*

Exercise 10.6. *For a distribution $p(x, c)$ and an approximation $q(x, c)$, show that when $p(x, c)$ corresponds to the empirical distribution, finding $q(x, c)$ that minimises the Kullback-Leibler divergence*

$$KL(p(x, c)|q(x, c)) \tag{10.7.12}$$

corresponds to maximum likelihood training of $q(x, c)$ assuming i.i.d. data.

Exercise 10.7. *Consider a distribution $p(x, c)$ and a Tree Augmented approximation*

$$q(x, c) = q(c) \prod_i q(x_i|x_{pa(i)}, c), \qquad pa(i) < i \quad or \; pa(i) = \emptyset \tag{10.7.13}$$

Show that for the optimal $q(x, c)$ constrained as above, the solution $q(x, c)$ that minimises $KL(p(x, c)|q(x, c))$ when plugged back into the Kullback-Leibler expression gives, as a function of the parental structure,

$$KL(p(x, c)|q(x, c)) = -\sum_i \left\langle \log \frac{p(x_i, x_{pa(i)}|c)}{p(x_{pa(i)}|c)p(x_i|c)} \right\rangle_{p(x_i, x_{pa(i)}, c)} + \textit{const.} \tag{10.7.14}$$

This shows that under the single-parent constraint and that each tree $q(x|c)$ has the same structure, minimising the Kullback-Leibler divergence is equivalent to maximising the sum of conditional mutual information terms. From exercise(10.6), we know therefore that this also corresponds to the maximum likelihood setting for the parental structure. This can then be achieved by finding a maximal weight spanning tree, as in the case of the Chow-Liu tree.

CHAPTER 11

Learning with Hidden Variables

In many models some variables are not directly observed, but are latent or 'hidden'. This can also occur when some data are not observed. In this chapter we discuss methods for learning in the presence of such missing information, and in particular develop the Expectation-Maximisation algorithm and related variants.

11.1 Hidden Variables and Missing Data

In practice data entries are often missing resulting in incomplete information to specify a likelihood. Observational variables may be split into *visible* (those for which we actually know the state) and *missing* (those whose states would nominally be known but are missing for a particular datapoint).

Another scenario in which not all variables in the model are observed are the so-called hidden or *latent variable models*. In this case there are variables which are essential for the model description but never observed. For example, the underlying physics of a model may contain latent processes which are essential to describe the model, but cannot be directly measured.

11.1.1 Why hidden/missing variables can complicate proceedings

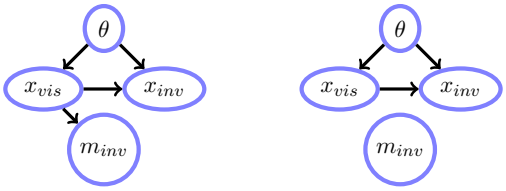
In learning the parameters of models as previously described in chapter(9), we assumed we have complete information to define all variables of the joint model of the data $p(x|\theta)$. Consider the Asbestos-Smoking-Cancer network of section(9.3). Using the multivariate variable $x = (a, s, c)$, if patient n has a complete record, the likelihood of this record is

$$p(x^n|\theta) = p(a^n, s^n, c^n|\theta) = p(c^n|a^n, s^n, \theta_c)p(a^n|\theta_a)p(s^n|\theta_s) \tag{11.1.1}$$

which is factorised in terms of the table entry parameters. We exploited this property to show that table entries θ can be learned by considering only local information, both in the maximum likelihood and Bayesian frameworks.

Now consider the case that for some of the patients, only partial information is available. For example, for patient n with incomplete record $x^n = (c = 1, s = 1)$ it is known that the patient has cancer and is a smoker, but whether or not they had exposure to asbestos is unknown. Since we can only use the 'visible' available information it would seem reasonable (see section(11.1.2)) to assess parameters using the marginal likelihood

$$p(x^n|\theta) = \sum_a p(a, s^n, c^n|\theta) = \sum_a p(c^n|a, s^n, \theta_c)p(a|\theta_a)p(s^n|\theta_s) \tag{11.1.2}$$



(a) (b)

Figure 11.1: **(a)**: Missing at random assumption. The mechanism that generates the missing data does not depend on either the parameter θ of the model, or on the value of the missing data. **(b)**: Missing completely at random assumption. The mechanism generating the missing data is completely independent of the model. Note that in both cases the direction of the arrow between x_{vis} and x_{inv} is irrelevant.

Using the marginal likelihood, however, may result in computational difficulties since the likelihood equation (11.1.2) cannot now be factorised into a product of the separate table parameters of the form $f_s(\theta_s)f_a(\theta_a)f_c(\theta_c)$. In this case the maximisation of the likelihood is more complex since the parameters of different tables are coupled.

A similar complication holds for Bayesian learning. As we saw in section(9.4.1), under a prior factorised over each CPT θ, the posterior is also factorised. However, the missing variable introduces dependencies in the posterior parameter distribution, making the posterior more complex. In both the maximum likelihood and Bayesian cases, one has a well defined likelihood function of the table parameters/posterior.

Note that missing data does not always make the parameter posterior non-factorised. For example, if the cancer state is unobserved above, because cancer is a collider with no descendants, the conditional distribution simply sums to 1, and one is left with a factor dependent on a and another on s.

11.1.2 The missing at random assumption

Under what circumstances is it valid to use the marginal likelihood to assess parameters? We partition the variables x into those that are 'visible', x_{vis} and 'invisible', x_{inv}, so that the set of all variables can be written $x = [x_{vis}, x_{inv}]$. For the visible variables we have an observed state $x_{vis} = \mathsf{v}$, whereas the state of the invisible variables is unknown. To complete the picture we also need to model the process that informs when data will be missing. We use an indicator $m_{inv} = 1$ to denote that the state of the invisible variables is unknown and require then a model $p(m_{inv}|x_{vis}, x_{inv}, \theta)$. Then for a datapoint which contains both visible and invisible information,

$$p(x_{vis} = \mathsf{v}, m_{inv} = 1|\theta) = \sum_{x_{inv}} p(x_{vis} = \mathsf{v}, x_{inv}, m_{inv} = 1|\theta) \tag{11.1.3}$$

$$= \sum_{x_{inv}} p(m_{inv} = 1|x_{vis} = \mathsf{v}, x_{inv}, \theta)p(x_{vis} = \mathsf{v}, x_{inv}|\theta) \tag{11.1.4}$$

If we assume that the mechanism which generates invisible data is independent of the parameter and the missing value x_{inv}

$$p(m_{inv} = 1|x_{vis} = \mathsf{v}, x_{inv}, \theta) = p(m_{inv} = 1|x_{vis} = \mathsf{v}) \tag{11.1.5}$$

then

$$p(x_{vis} = \mathsf{v}, m_{inv} = 1|\theta) = p(m_{inv} = 1|x_{vis} = \mathsf{v}) \sum_{x_{inv}} p(x_{vis} = \mathsf{v}, x_{inv}|\theta) \tag{11.1.6}$$

$$= p(m_{inv} = 1|x_{vis} = \mathsf{v})p(x_{vis} = \mathsf{v}|\theta) \tag{11.1.7}$$

Only the term $p(x_{vis} = \mathsf{v}|\theta)$ conveys information about the parameter θ of the model. Therefore, provided the mechanism by which the data is missing depends only on the visible states, we may simply use the marginal likelihood to assess parameters. This is called the *missing at random* (MAR) assumption, see fig(11.1).

Example 11.1 (Not missing at random)**.** EZsurvey.org stop men on the street and ask them their favourite colour. All men whose favourite colour is pink decline to respond to the question – for any other colour, all men respond to the question. Based on the data, EZsurvey.org produce a histogram of men's favourite colour, based on the likelihood of the visible data alone, confidently stating that none of them likes pink. For simplicity, we assume there are only three colours, blue, green and pink. EZsurvey.org attempts to find the histogram with probabilities $\theta_b, \theta_g, \theta_p$ with $\theta_b + \theta_g + \theta_p = 1$. Each respondent produces a visible response x with $\text{dom}(x) = \{\text{blue}, \text{green}, \text{pink}\}$, otherwise $m = 1$ if there is no response. Three men are asked their favourite colour, giving data

$$\left\{x^1, x^2, x^3\right\} = \{\text{blue}, \text{missing}, \text{green}\} \tag{11.1.8}$$

Based on the likelihood of the visible data alone we have the log likelihood for i.i.d. data

$$L(\theta_b, \theta_g, \theta_p) = \log \theta_b + \log \theta_g + \lambda \left(1 - \theta_b - \theta_g - \theta_p\right) \tag{11.1.9}$$

where the last Lagrange term ensures normalisation. Maximising the expression we arrive at

$$\theta_b = \frac{1}{2}, \quad \theta_g = \frac{1}{2}, \quad \theta_p = 0 \tag{11.1.10}$$

Hence EZsurvey.org arrive at the erroneous conclusion that no men like pink, even though the reality is that some men like pink – they just don't want to say so. The unreasonable result that EZsurvey.org produce is due to not accounting correctly for the mechanism which produces the data. In this case the data is not MAR since whether or not the data is missing depends on the state of the missing variable.

The correct mechanism that generates the data (including the missing data is)

$$p(x^1 = \text{blue}|\theta) p(m^2 = 1|\theta) p(x^3 = \text{green}|\theta) = \theta_b \theta_p \theta_g = \theta_b \left(1 - \theta_b - \theta_g\right) \theta_g \tag{11.1.11}$$

where we used $p(m^2 = 1|\theta) = \theta_p$ since the probability that a datapoint is missing is the same as the probability that the favourite colour is pink. Maximising the likelihood, we arrive at

$$\theta_b = \frac{1}{3}, \theta_g = \frac{1}{3}, \theta_p = \frac{1}{3} \tag{11.1.12}$$

as we would expect. On the other hand if there is another visible variable, t, denoting the time of day, and the probability that men respond to the question depends only on the time t alone (for example the probability of having missing data is high during rush hour), then we may indeed treat the missing data as missing at random.

A stronger assumption than MAR is that the missing data mechanism is completely independent of all other model processes

$$p(m_{inv} = 1|x_{vis} = \mathsf{v}, x_{inv}, \theta) = p(m_{inv} = 1) \tag{11.1.13}$$

which is called *missing completely at random*. This applies for example to latent variable models in which the variable state is always missing, independent of anything else.

11.1.3 Maximum likelihood

Throughout the remaining discussion we will assume any missing data is MAR or missing completely at random. We partition the variables into 'visible variables v for which we know the states and 'hidden' variables h whose states will not be observed. For maximum likelihood we may then learn model parameters θ by optimising the on the visible variables alone

$$p(v|\theta) = \sum_h p(v, h|\theta) \tag{11.1.14}$$

with respect to θ.

11.1.4 Identifiability issues

The marginal likelihood objective function depends on the parameters only through $p(v|\theta)$, so that equivalent parameter solutions may exist. For example, consider a latent variable model with distribution

$$p(x_1, x_2|\theta) = \theta_{x_1,x_2} \tag{11.1.15}$$

in which variable x_2 is never observed. This means that the marginal likelihood only depends on the entry $p(x_1|\theta) = \sum_{x_2} \theta_{x_1,x_2}$. Given a maximum likelihood solution θ^*, we can then always find an equivalent maximum likelihood solution θ' provided (see exercise(11.9))

$$\sum_{x_2} \theta'_{x_1,x_2} = \sum_{x_2} \theta^*_{x_1,x_2} \tag{11.1.16}$$

In other cases there is an inherent symmetry in the parameter space of the marginal likelihood. For example, consider the network over binary variables

$$p(c, a, s) = p(c|a, s)p(a)p(s) \tag{11.1.17}$$

We assume we know the table for $p(s)$ and that our aim is to learn the asbestos table $\hat{p}(a = 1)$ and cancer tables

$$\hat{p}(c = 1|a = 1, s = 1), \quad \hat{p}(c = 1|a = 1, s = 0), \quad \hat{p}(c = 1|a = 0, s = 1), \quad \hat{p}(c = 1|a = 0, s = 0) \tag{11.1.18}$$

where we used a '$\hat{}$' to denote that these are parameter estimates.

We assume that we have missing data such that the states of variable a are never observed. In this case an equivalent solution (in the sense that it has the same marginal likelihood) is given by interchanging the states of a:

$$\hat{p}'(a = 0) = \hat{p}(a = 1) \tag{11.1.19}$$

and the four tables

$$\begin{aligned} &\hat{p}'(c = 1|a = 0, s = 1) = \hat{p}(c = 1|a = 1, s = 1), \qquad &&\hat{p}'(c = 1|a = 0, s = 0) = \hat{p}(c = 1|a = 1, s = 0) \\ &\hat{p}'(c = 1|a = 1, s = 1) = \hat{p}(c = 1|a = 0, s = 1), \qquad &&\hat{p}'(c = 1|a = 1, s = 0) = \hat{p}(c = 1|a = 0, s = 0) \end{aligned} \tag{11.1.20}$$

A similar situation occurs in a more general setting in which the state of a variable is consistently unobserved (mixture models are a case in point) yielding an inherent symmetry in the solution space. A well known characteristic of maximum likelihood algorithms is that 'jostling' occurs in the initial stages of training in which these symmetric solutions compete.

11.2 Expectation Maximisation

The EM algorithm is a convenient and general purpose iterative approach to maximising the likelihood under missing data/hidden variables[85, 203]. It is generally straightforward to implement and can achieve large jumps in parameter space, particularly in the initial iterations.

11.2.1 Variational EM

The key feature of the EM algorithm is to form an alternative objective function for which the parameter coupling effect discussed in section(11.1.1) is removed, meaning that individual parameter updates can be achieved, akin to the case of fully observed data. The way this works is to replace the marginal likelihood with a lower bound – it is this lower bound that has the useful decoupled form.

We first consider a single variable pair (v, h), where v stands for 'visible' and h for 'hidden'. The model of the data is then $p(v, h|\theta)$ and our interest is to set θ by maximising the marginal likelihood $p(v|\theta)$. To derive the

Algorithm 11.1 Expectation Maximisation. Compute Maximum Likelihood value for data with hidden variables. Input: a distribution $p(x|\theta)$ and dataset $\mathcal{V}$. Returns ML candidate θ.

$t = 0$ ▷ Iteration counter
Choose an initial setting for the parameters θ^0. ▷ Initialisation
while θ not converged (or likelihood not converged) **do**
  $t \leftarrow t + 1$
  for $n = 1$ to N **do** ▷ Run over all datapoints
    $q_t^n(h^n|v^n) = p(h^n|v^n, \theta^{t-1})$ ▷ E step
  end for
  $\theta^t = \arg\max_\theta \sum_{n=1}^N \langle \log p(h^n, v^n|\theta) \rangle_{q_t^n(h^n|v^n)}$ ▷ M step
end while
return θ^t ▷ The max likelihood parameter estimate.

bound on the marginal likelihood, consider the Kullback-Leibler divergence (which is always non-negative) between a 'variational' distribution $q(h|v)$ and the parametric model $p(h|v, \theta)$:

$$\mathrm{KL}(q(h|v)|p(h|v,\theta)) \equiv \langle \log q(h|v) - \log p(h|v,\theta) \rangle_{q(h|v)} \geq 0 \tag{11.2.1}$$

The term 'variational' refers to the fact that this distribution will be a parameter of an optimisation problem. Using $p(h|v,\theta) = p(h,v|\theta)/p(v|\theta)$ and the fact that $p(v|\theta)$ does not depend on h,

$$\mathrm{KL}(q(h|v)|p(h|v,\theta)) = \langle \log q(h|v) \rangle_{q(h|v)} - \langle \log p(h,v|\theta) \rangle_{q(h|v)} + \log p(v|\theta) \geq 0 \tag{11.2.2}$$

Rearranging, we obtain a bound on the marginal likelihood[1]

$$\log p(v|\theta) \geq \underbrace{-\langle \log q(h|v) \rangle_{q(h|v)}}_{\text{Entropy}} + \underbrace{\langle \log p(h,v|\theta) \rangle_{q(h|v)}}_{\text{Energy}} \tag{11.2.3}$$

The energy term is also called the 'expected complete data log likelihood'. The bound is potentially useful since the θ dependent energy term is similar in form to the fully observed case, except that terms with missing data have their log likelihood weighted by a prefactor. Equation(11.2.3) is a marginal likelihood bound for a single training example. Under the i.i.d. assumption, the log likelihood of all training data $\mathcal{V} = \{v^1, \ldots, v^N\}$ is the sum of the individual log likelihoods:

$$\log p(\mathcal{V}|\theta) = \sum_{n=1}^N \log p(v^n|\theta) \tag{11.2.4}$$

Summing over the training data, we obtain a bound on the log (marginal) likelihood

$$\log p(\mathcal{V}|\theta) \geq \tilde{L}(\{q\}, \theta) \equiv \underbrace{-\sum_{n=1}^N \langle \log q(h^n|v^n) \rangle_{q(h^n|v^n)}}_{\text{entropy}} + \underbrace{\sum_{n=1}^N \langle \log p(h^n, v^n|\theta) \rangle_{q(h^n|v^n)}}_{\text{energy}} \tag{11.2.5}$$

Note that the bound $\tilde{L}(\{q\}, \theta)$ is exact (that is, the right hand side is equal to the log likelihood) when we set $q(h^n|v^n) = p(h^n|v^n, \theta), n = 1, \ldots, N$.

The bound depends both on θ and the set of variational distributions $\{q\}$. Our aim is then to try to optimise this bound *w.r.t.* θ and $\{q\}$; in doing so we will push up the lower bound, and hopefully increase thereby the likelihood itself. A simple iterative procedure to optimise the bound is to first fix θ and then optimise *w.r.t.* $\{q\}$, and then fix $\{q\}$ and optimise the bound *w.r.t.* θ. These are known as the 'E' and 'M' steps and are repeated until convergence:

E-step For fixed θ, find the distributions $q(h^n|v^n), n = 1, \ldots, N$ that maximise equation (11.2.5).

M-step For fixed $q(h^n|v^n), n = 1, \ldots, N$, find the parameters θ that maximise equation (11.2.5).

[1] This is analogous to a standard partition function bound in statistical physics, from where the terminology 'energy' and 'entropy' hails.

11.2.2 Classical EM

In the variational E-step above, the fully optimal setting is

$$q(h^n|v^n) = p(h^n|v^n, \theta) \tag{11.2.6}$$

Since q is fixed during the M-step, performing the M-step optimisation is equivalent to maximising the energy term alone, see algorithm(11.1). From here onwards we shall use the term 'EM' to refer to the classical EM algorithm, unless otherwise stated. It is important to note that the EM algorithm cannot generally guarantee to find the fully optimal maximum likelihood solution and can get trapped in local optima, as discussed in example(11.2).

Example 11.2 (EM for a one-parameter model)**.** We consider a model small enough that we can plot fully the evolution of the EM algorithm. The model is on a single visible variable v with $\mathrm{dom}(v) = \mathbb{R}$ and single two-state hidden variable h with $\mathrm{dom}(h) = \{1, 2\}$. We define a model $p(v, h|\theta) = p(v|h, \theta)p(h)$ with

$$p(v|h, \theta) = \frac{1}{\sqrt{\pi}} e^{-(v-\theta h)^2} \tag{11.2.7}$$

and $p(h = 1) = p(h = 2) = 0.5$. For an observation $v = 2.75$ our interest is to find the parameter θ that optimises the likelihood

$$p(v = 2.75|\theta) = \sum_{h=1,2} p(v = 2.75|h, \theta)p(h) = \frac{1}{2\sqrt{\pi}} \left(e^{-(2.75-\theta)^2} + e^{-(2.75-2\theta)^2} \right) \tag{11.2.8}$$

The log likelihood is plotted in fig(11.2a) with optimum at $\theta = 1.325$. If the state of h were given, the log likelihood would be a single bump, rather than the more complex double bump in the case of missing data. To use the EM approach to find the optimum θ, we need to work out the energy

$$\langle \log p(v, h|\theta) \rangle_{q(h|v)} = \langle \log p(v|h, \theta) \rangle_{q(h|v)} + \langle \log p(h) \rangle_{q(h|v)} \tag{11.2.9}$$

$$= -\left\langle (v - \theta h)^2 \right\rangle_{q(h|v)} + \mathsf{const.} \tag{11.2.10}$$

There are two h states of the q distribution and using $q(h)$ as shorthand for $q(h|v)$, normalisation requires $q(1) + q(2) = 1$. Due to the normalisation, we can fully parameterise q using $q(2)$ alone. The EM procedure then iteratively optimises the lower bound

$$\log p(v = 2.75|\theta) \geq \tilde{L}(q(2), \theta)$$

$$\equiv -q(1) \log q(1) - q(2) \log q(2) - \sum_{h=1,2} q(h) \left(2.75 - \theta h\right)^2 + \mathsf{const.} \tag{11.2.11}$$

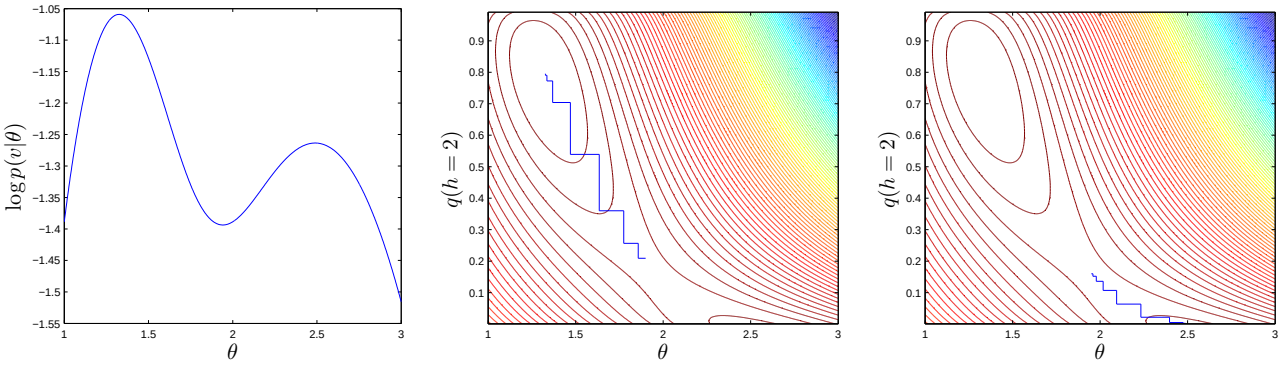
From an initial starting θ, the EM algorithm finds the q distribution that optimises $\tilde{L}(q(2), \theta)$ (E-step) and then updates θ (M-step). Depending on the initial θ, the solution found is either a global or local optimum of the likelihood, see fig(11.2).

The M-step is easy to work out analytically in this case with $\theta^{new} = v \left\langle h \right\rangle_{q(h)} / \left\langle h^2 \right\rangle_{q(h)}$. Similarly, the E-step sets $q^{new}(h) = p(h|v, \theta)$ so that

$$q^{new}(h = 2) = \frac{p(v = 2.75|h = 2, \theta)p(h = 2)}{p(v = 2.75)} = \frac{e^{-(2.75-2\theta)^2}}{e^{-(2.75-2\theta)^2} + e^{-(2.75-\theta)^2}} \tag{11.2.12}$$

where we used

$$p(v = 2.75) = p(v = 2.75|h = 1, \theta)p(h = 1) + p(v = 2.75|h = 2, \theta)p(h = 2) \tag{11.2.13}$$



(a) (b) (c)

Figure 11.2: **(a)**: The log likelihood for the model described in example(11.2). **(b)**: Contours of the lower bound $\tilde{L}(q(h=2),\theta)$. For an initial choice $\theta = 1.9$, successive updates of the E (vertical) and M (horizontal) steps are plotted, with the algorithm converging to the global optimum maximum likelihood setting for θ. **(c)**: Starting at $\theta = 1.95$, the EM algorithm converges to a local optimum.

Example 11.3. Consider a simple model

$$p(x_1, x_2|\theta) \tag{11.2.14}$$

where $\mathrm{dom}(x_1) = \mathrm{dom}(x_2) = \{1, 2\}$. Assuming an unconstrained distribution

$$p(x_1, x_2|\theta) = \theta_{x_1,x_2}, \qquad \theta_{1,1} + \theta_{1,2} + \theta_{2,1} + \theta_{2,2} = 1 \tag{11.2.15}$$

our aim is to learn θ from the data $\mathbf{x}^1 = (1,1), \mathbf{x}^2 = (1,?), \mathbf{x}^3 = (?,2)$. The energy term for the classical EM is

$$\log p(x_1 = 1, x_2 = 1|\theta) + \langle \log p(x_1 = 1, x_2|\theta) \rangle_{p(x_2|x_1=1,\theta^{old})} + \langle \log p(x_1, x_2 = 2|\theta) \rangle_{p(x_1|x_2=2,\theta^{old})} \tag{11.2.16}$$

Writing out fully each of the above terms on a separate line gives the energy

$$\log \theta_{1,1} \tag{11.2.17}$$

$$+p(x_2 = 1|x_1 = 1, \theta^{old}) \log \theta_{1,1} + p(x_2 = 2|x_1 = 1, \theta^{old}) \log \theta_{1,2} \tag{11.2.18}$$

$$+p(x_1 = 1|x_2 = 2, \theta^{old}) \log \theta_{1,2} + p(x_1 = 2|x_2 = 2, \theta^{old}) \log \theta_{2,2} \tag{11.2.19}$$

This expression resembles the standard log likelihood of fully observed data except that terms with missing data have their weighted log parameters. The parameters are conveniently decoupled in this bound (apart from the trivial normalisation constraint) so that finding the optimal parameters is straightforward. This is achieved by the M-step update which gives

$$\begin{array}{ll} \theta_{1,1} \propto 1 + p(x_2 = 1|x_1 = 1, \theta^{old}) & \theta_{1,2} \propto p(x_2 = 2|x_1 = 1, \theta^{old}) + p(x_1 = 1|x_2 = 2, \theta^{old}) \\ \theta_{2,1} = 0 & \theta_{2,2} \propto p(x_1 = 2|x_2 = 2, \theta^{old}) \end{array} \tag{11.2.20}$$

where $p(x_2|x_1, \theta^{old}) \propto \theta^{old}_{x_1,x_2}$ (E-step) *etc.* The E and M-steps are iterated till convergence.

The EM algorithm increases the likelihood

Whilst, by construction, the EM algorithm cannot decrease the lower bound on the likelihood, an important question is whether or not the log likelihood itself is necessarily increased by this procedure.

s	c
1	1
0	0
1	1
1	0
1	1
0	0
0	1

Figure 11.3: A database containing information about being a Smoker (1 signifies the individual is a smoker), and lung Cancer (1 signifies the individual has lung Cancer). Each row contains the information for an individual, so that there are 7 individuals in the database.

We use θ' for the new parameters, and θ for the previous parameters in two consecutive iterations. Using $q(h^n|v^n) = p(h^n|v^n, \theta)$ we see that as a function of the parameters, the lower bound for a single variable pair (v, h) depends on θ and θ':

$$LB(\theta'|\theta) \equiv -\left\langle \log p(h|v,\theta)\right\rangle_{p(h|v,\theta)} + \left\langle \log p(h,v|\theta')\right\rangle_{p(h|v,\theta)}. \tag{11.2.21}$$

From the definition of the lower bound, equation (11.2.3), we have

$$\log p(v|\theta') = LB(\theta'|\theta) + KL(p(h|v,\theta)|p(h|v,\theta')). \tag{11.2.22}$$

That is, the Kullback-Leibler divergence is the difference between the true log likelihood and the lower @@ bound.
We may then also write

$$\log p(v|\theta) = LB(\theta|\theta) + \underbrace{KL(p(h|v,\theta)|p(h|v,\theta))}_{0} \tag{11.2.23}$$

Hence

$$\log p(v|\theta') - \log p(v|\theta) = \underbrace{LB(\theta'|\theta) - LB(\theta|\theta)}_{\geq 0} + \underbrace{KL(p(h|v,\theta)|p(h|v,\theta'))}_{\geq 0} \tag{11.2.24}$$

The first assertion is true since, by definition of the M-step, we search for a θ' which has a higher value for the bound than our starting value θ. The second assertion is true by the non-negative property of the Kullback-Leibler divergence.

For more than a single datapoint, we simply sum each individual bound for $\log p(v^n|\theta)$. Hence we reach the important conclusion that the EM algorithm increases, not only the lower bound on the marginal likelihood, but the marginal likelihood itself (more correctly, EM cannot decrease these quantities).

Shared parameters and tables

It is often the case in models that parameters are shared between components of the model. The application of EM to this shared parameter case is essentially straightforward. According to the energy term, we need to identify all those terms in which the shared parameter occurs. The objective for the shared parameter is then the sum over all energy terms containing the shared parameter.

11.2.3 Application to Belief networks

Conceptually, the application of EM to training Belief Networks with missing data is straightforward. The battle is more notational than conceptual. We begin the development with an example, from which intuition about the general case can be gleaned.

Example 11.4. Consider the network.

$$p(a, c, s) = p(c|a, s)p(a)p(s) \tag{11.2.25}$$

for which we have a set of data, but that the states of variable a are never observed, see fig(11.3). Our goal is to learn the CPTs $p(c|a, s)$ and $p(a)$ and $p(s)$. To apply EM, algorithm(11.1) to this case, we first assume

initial parameters θ_a^0, θ_s^0, θ_c^0.

The first E-step, for iteration $t = 1$ then defines a set of distributions on the hidden variables (here the hidden variable is a). For notational convenience, we write $q_t^n(a)$ in place of $q_t^n(a|v^n)$. Then

$$q_{t=1}^{n=1}(a) = p(a|c = 1, s = 1, \theta^0), \qquad q_{t=1}^{n=2}(a) = p(a|c = 0, s = 0, \theta^0) \tag{11.2.26}$$

and so on for the 7 training examples, $n = 2, \ldots, 7$.

We now move to the first M-step. The energy term for any iteration t is:

$$E(\theta) = \sum_{n=1}^{7} \langle \log p(c^n|a^n, s^n) + \log p(a^n) + \log p(s^n) \rangle_{q_t^n(a)} \tag{11.2.27}$$

$$= \sum_{n=1}^{7} \left\{ \langle \log p(c^n|a^n, s^n) \rangle_{q_t^n(a)} + \langle \log p(a^n) \rangle_{q_t^n(a)} + \log p(s^n) \right\} \tag{11.2.28}$$

The final term is the log likelihood of the variable s, and $p(s)$ appears explicitly only in this term. Hence, the usual maximum likelihood rule applies, and $p(s = 1)$ is simply given by the relative number of times that $s = 1$ occurs in the database, giving $p(s = 1) = 4/7$, $p(s = 0) = 3/7$.

The contribution of parameter $p(a = 1)$ to the energy occurs in the terms

$$\sum_n \{q_t^n(a = 0) \log p(a = 0) + q_t^n(a = 1) \log p(a = 1)\} \tag{11.2.29}$$

which, using the normalisation constraint is

$$\log p(a = 0) \sum_n q_t^n(a = 0) + \log(1 - p(a = 0)) \sum_n q_t^n(a = 1) \tag{11.2.30}$$

Differentiating with respect to $p(a = 0)$ and solving for the zero derivative we get the M-step update for $p(a = 0)$

$$p(a = 0) = \frac{\sum_n q_t^n(a = 0)}{\sum_n q_t^n(a = 0) + \sum_n q_t^n(a = 1)} = \frac{1}{N} \sum_n q_t^n(a = 0) \tag{11.2.31}$$

That is, whereas in the standard maximum likelihood estimate, we would have the real counts of the data in the above formula, here they have been replaced with our guessed values $q_t^n(a = 0)$ and $q_t^n(a = 1)$.

A similar story holds for $p(c = 1|a = 0, s = 1)$. Again we need to consider that normalisation means that $p(c = 0|a = 0, s = 1)$ also contributes. The contribution of this term to the energy is from those data indices n for which $s = 1$

$$\sum_{n:c^n=1,s^n=1} q_t^n(a = 0) \log p(c = 1|a = 0, s = 1) + \sum_{n:c^n=0,s^n=1} q_t^n(a = 0) \log \left(1 - p(c = 1|a = 0, s = 1)\right)$$

Optimising with respect to $p(c = 1|a = 0, s = 1)$ gives

$$p(c = 1|a = 0, s = 1) = \frac{\sum_n \mathbb{I}\left[c^n = 1\right] \mathbb{I}\left[s^n = 1\right] q_t^n(a = 0)}{\sum_n \mathbb{I}\left[c^n = 1\right] \mathbb{I}\left[s^n = 1\right] q_t^n(a = 0) + \sum_n \mathbb{I}\left[c^n = 0\right] \mathbb{I}\left[s^n = 1\right] q_t^n(a = 0)} \tag{11.2.32}$$

For comparison, the setting in the complete data case is

$$p(c = 1|a = 0, s = 1) = \frac{\sum_n \mathbb{I}\left[c^n = 1\right] \mathbb{I}\left[s^n = 1\right] \mathbb{I}\left[a^n = 0\right]}{\sum_n \mathbb{I}\left[c^n = 1\right] \mathbb{I}\left[s^n = 1\right] \mathbb{I}\left[a^n = 0\right] + \sum_n \mathbb{I}\left[c^n = 0\right] \mathbb{I}\left[s^n = 1\right] \mathbb{I}\left[a^n = 0\right]} \tag{11.2.33}$$

There is an intuitive relationship between these updates: in the missing data case we replace the indicators by the assumed distributions q. Iterating the E and M steps, these parameters will converge to a local likelihood optimum.

11.2.4 General case

A belief network on a multivariate variable x takes the general form

$$p(x) = \prod_i p(x_i|\text{pa}\,(x_i)). \tag{11.2.34}$$

Some of the variables will be observed, and others hidden. We therefore partition the variable x into visible and hidden multivariate parts $x = (v, h)$. Given an i.i.d. dataset $\mathcal{V} = \{v^1, \ldots, v^N\}$ our interest is to learn the tables of $p(x)$ to maximise the likelihood on the visible data $\mathcal{V}$. For each datapoint index n, each multivariate variable $x^n = (v^n, h^n)$ decomposes into its visible and hidden parts; an upper variable index will typically the datapoint, and a lower index the variable number.

From equation (11.2.5), the form of the energy term for belief networks is

$$\sum_n \langle \log p(x^n) \rangle_{q_t(h^n|v^n)} = \sum_n \sum_i \langle \log p(x_i^n|\text{pa}\,(x_i^n)) \rangle_{q_t(h^n|v^n)} \tag{11.2.35}$$

Here t indexes the iteration count for the EM algorithm. It is useful to define the following notation:

$$q_t^n(x) = q_t(h^n|v^n)\delta(v, v^n) \tag{11.2.36}$$

This means that $q_t^n(x)$ sets the visible variables in the observed state, and defines a conditional distribution on the unobserved variables. We then define the mixture distribution

$$q_t(x) = \frac{1}{N}\sum_{n=1}^N q_t^n(x) \tag{11.2.37}$$

The energy term in the left hand side of equation (11.2.35) can be written more compactly using this notation as

$$\sum_n \langle \log p(x^n) \rangle_{q_t(h^n|v^n)} = N \langle \log p(x) \rangle_{q_t(x)} \tag{11.2.38}$$

To see this consider the right hand side of the above

$$N \langle \log p(x) \rangle_{q_t(x)} = N \sum_x [\log p(x)] \frac{1}{N} \sum_n q_t(h^n|v^n)\delta(v, v^n) = \sum_n \langle \log p(x^n) \rangle_{q_t(h^n|v^n)} \tag{11.2.39}$$

Using this compact notation for the energy, and the structure of the belief network, we can decompose the energy as

$$\langle \log p(x) \rangle_{q_t(x)} = \sum_i \langle \log p(x_i|\text{pa}\,(x_i)) \rangle_{q_t(x)} = \sum_i \left\langle \langle \log p(x_i|\text{pa}\,(x_i)) \rangle_{q_t(x_i|\text{pa}(x_i))} \right\rangle_{q_t(\text{pa}(x_i))} \tag{11.2.40}$$

This means that maximising the energy is equivalent to minimising

$$\sum_i \left\langle \langle \log q_t(x_i|\text{pa}\,(x_i)) \rangle_{q_t(x_i|\text{pa}(x_i))} - \langle \log p(x_i|\text{pa}\,(x_i)) \rangle_{q_t(x_i|\text{pa}(x_i))} \right\rangle_{q_t(\text{pa}(x_i))} \tag{11.2.41}$$

where we added the constant first term to make this into the form of a Kullback-Leibler divergence. Since this is a sum of independent Kullback-Leibler divergences, optimally the M-step is given by setting

$$p^{new}(x_i|\text{pa}\,(x_i)) = q_t(x_i|\text{pa}\,(x_i)) \tag{11.2.42}$$

In practice, storing the $q_t(x)$ over the states of all variables x is prohibitively expensive. Fortunately, since the M-step only requires the distribution on the family of each variable x_i, one only requires the local distributions $q_t^n(x_i, \text{pa}\,(x_i))$. We may therefore dispense with the global $q_t(x)$ and equivalently use

$$p^{new}(x_i|\text{pa}\,(x_i)) = \frac{\sum_n q_t^n(x_i, \text{pa}\,(x_i))}{\sum_{n'} q_t^{n'}(\text{pa}\,(x_i))} \tag{11.2.43}$$

Algorithm 11.2 EM for Belief Networks. Input: a BN DAG and dataset on the visible variables $\mathcal{V}$. Returns the maximum likelihood estimate of the tables $p(x_i|\text{pa}(x_i))$, $i = 1, \ldots, K$.

$t = 1$ ▷ Iteration counter
Set $p_t(x_i|\text{pa}(x_i))$ to initial values. ▷ Initialisation
while $p(x_i|\text{pa}(x_i))$ not converged (or likelihood not converged) **do**
  $t \leftarrow t + 1$
  for $n = 1$ to N **do** ▷ Run over all datapoints
    $q_t^n(x) = p_t(h^n|v^n)\,\delta(v, v^n)$ ▷ E step
  end for
  for $i = 1$ to K **do** ▷ Run over all variables
@@ $p_{t+1}(x_i|\text{pa}(x_i)) = \frac{\sum_{n=1}^{N} q_t^n(x_i, \text{pa}(x_i))}{\sum_{n'=1}^{N} q_t^{n'}(\text{pa}(x_i))}$ ▷ M step
  end for
end while
return $p_t(x_i|\text{pa}(x_i))$ ▷ The max likelihood parameter estimate.

Using the EM algorithm, the optimal setting for the E-step is to use $q_t(h^n|v^n) = p^{old}(h^n|v^n)$. With this notation, the EM algorithm can be compactly stated as in algorithm(11.2). See also `EMbeliefnet.m`. An illustration of the evolution of the log likelihood under EM iterations is given in fig(11.4). For readers less comfortable with the above KL-based derivation, we describe a more classical approach based on Lagrange multipliers for a specific case in example(11.5).

Example 11.5 (Another belief network example)**.** Consider a five variable distribution with discrete variables,

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1|x_2)p(x_2|x_3)p(x_3|x_4)p(x_4|x_5)p(x_5) \tag{11.2.44}$$

in which the variables x_2 and x_4 are consistently hidden in the training data, and training data for x_1, x_3, x_5 are always present. The distribution can be represented as a belief network

The M-step is given by maximising the energy. According to the general form for the energy, equation (11.2.35), we need to consider a variational distribution $q(h|v)$ on the hidden variables $h = (x_2, x_4)$ conditioned on the visible variables $v = (x_1, x_3, x_5)$. Using n as the datapoint index and t as the EM iteration counter, we require variational distributions for each datapoint

$$q_t(x_2^n, x_4^n|x_1^n, x_3^n, x_5^n) \tag{11.2.45}$$

To keep the notation more compact, we drop here the iteration counter index and write the above as simply $q^n(x_2, x_4)$. In this case, the contributions to the energy have the form

$$\sum_n \langle \log p(x_1^n|x_2)p(x_2|x_3^n)p(x_3^n|x_4)p(x_4|x_5^n)p(x_5^n)\rangle_{q^n(x_2,x_4)} \tag{11.2.46}$$

which may be written as

$$\sum_n \langle \log p(x_1^n|x_2)\rangle_{q^n(x_2,x_4)} + \sum_n \langle \log p(x_2|x_3^n)\rangle_{q^n(x_2,x_4)} + \sum_n \langle \log p(x_3^n|x_4)\rangle_{q^n(x_2,x_4)} + \sum_n \langle \log p(x_4|x_5^n)\rangle_{q^n(x_2,x_4)} + \sum_n \log p(x_5^n) \tag{11.2.47}$$

A useful property can now be exploited, namely that each term depends on only those hidden variables in the family that that term represents. Thus we may write

$$\sum_n \langle \log p(x_1^n|x_2) \rangle_{q^n(x_2)} + \sum_n \langle \log p(x_2|x_3^n) \rangle_{q^n(x_2)} + \sum_n \langle \log p(x_3^n|x_4) \rangle_{q^n(x_4)} + \sum_n \langle \log p(x_4|x_5^n) \rangle_{q^n(x_4)} + \sum_n \log p(x_5^n) \tag{11.2.48}$$

The final term can be set using maximum likelihood. Let us consider therefore a more difficult table, $p(x_1|x_2)$. When will the table entry $p(x_1 = i|x_2 = j)$ occur in the energy? This happens whenever x_1^n is in state i. Since there is a summation over all the states of variables x_2 (due to the average), there is also a term with variable x_2 in state j. Hence the contribution to the energy from terms of the form $p(x_1 = i|x_2 = j)$ is

$$\sum_n \mathbb{I}\left[x_1^n = i\right] q^n(x_2 = j) \log p(x_1 = i|x_2 = j) \tag{11.2.49}$$

where the indicator function $\mathbb{I}\left[x_1^n = i\right]$ equals 1 if x_1^n is in state i and is zero otherwise. To ensure normalisation of the table, we add a Lagrange term:

$$\sum_n \mathbb{I}\left[x_1^n = i\right] q^n(x_2 = j) \log p(x_1 = i|x_2 = j) + \lambda \left\{ 1 - \sum_k p(x_1 = k|x_2 = j) \right\} \tag{11.2.50}$$

Differentiating with respect to $p(x_1 = i|x_2 = j)$ and equating to zero we get

$$\sum_n \mathbb{I}\left[x_1^n = i\right] \frac{q^n(x_2 = j)}{p(x_1 = i|x_2 = j)} = \lambda \tag{11.2.51}$$

or

$$p(x_1 = i|x_2 = j) \propto \sum_n \mathbb{I}\left[x_1^n = i\right] q^n(x_2 = j). \tag{11.2.52}$$

Hence

$$p(x_1 = i|x_2 = j) = \frac{\sum_n \mathbb{I}\left[x_1^n = i\right] q^n(x_2 = j)}{\sum_{n,k} \mathbb{I}\left[x_1^n = k\right] q^n(x_2 = j)} \tag{11.2.53}$$

From the E-step we have

$$q^n(x_2 = j) = p^{old}(x_2 = j|x_1^n, x_3^n, x_5^n) \tag{11.2.54}$$

This optimal distribution is easy to compute since this is the marginal on the family, given some evidential variables. Hence, the M-step update for the table is

$$p^{new}(x_1 = i|x_2 = j) = \frac{\sum_n \mathbb{I}\left[x_1^n = i\right] p^{old}(x_2 = j|x_1^n, x_3^n, x_5^n)}{\sum_{n,k} \mathbb{I}\left[x_1^n = k\right] p^{old}(x_2 = j|x_1^n, x_3^n, x_5^n)} \tag{11.2.55}$$

If there were no hidden data, equation (11.2.55) would read

$$p^{new}(x_1 = i|x_2 = j) \propto \sum_n \mathbb{I}\left[x_1^n = i\right] \mathbb{I}\left[x_2^n = j\right] \tag{11.2.56}$$

All that we do, therefore, in the general EM case, is to replace those deterministic functions such as $\mathbb{I}\left[x_2^n = i\right]$ by their missing variable equivalents $p^{old}(x_2 = i|x_1^n, x_3^n, x_5^n)$.

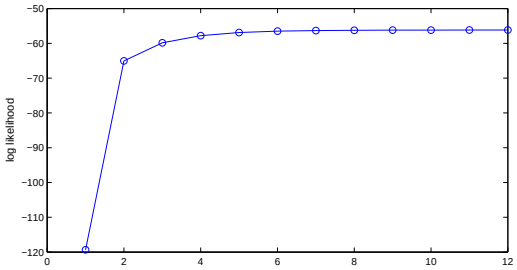


Figure 11.4: Evolution of the log-likelihood versus iterations under the EM training procedure (from solving the Printer Nightmare with missing data, exercise(11.1)). Note how rapid progress is made at the beginning, but convergence can be slow.

11.2.5 Convergence

Convergence of EM can be slow, particularly when the number of missing observations is greater than the number of visible observations. In practice, one often combines the EM with gradient based procedures to improve convergence, see section(11.6). Note also that the log likelihood is typically a non-convex function of the parameters. This means that there may be multiple local optima and the solution found often depends on the initialisation.

11.2.6 Application to Markov networks

Whilst our examples have been for belief networks, we may also apply the EM application to learning the parameters of Markov networks that have missing data. For a MN defined over visible and hidden variables with separate parameters θ_c for each clique c

$$p(v, h|\theta) = \frac{1}{Z(\theta)} \prod_c \phi_c(h, v|\theta_c) \tag{11.2.57}$$

the EM variational bound is

$$\log p(v|\theta) \geq H(q) + \sum_c \langle \log \phi_c(h, v|\theta_c) \rangle_{q(h)} - \log Z(\theta) \tag{11.2.58}$$

where $H(p)$ is the entropy function of a distribution, $H(p) \equiv -\langle \log p(x) \rangle_{p(x)}$. Whilst the bound decouples the clique parameters in the second term, the parameters are nevertheless coupled in the normalisation

$$Z(\theta) = \sum_{v,h} \prod_{c=1}^{C} \phi_c(h, v|\theta_c), \qquad \theta = (\theta_1, \ldots, \theta_C) \tag{11.2.59}$$

Because of this we cannot directly optimise the above bound on a parameter by parameter basis. One approach is to use an additional bound $\log Z(\theta)$ from above, as for iterative scaling, section(9.6.4) to decouple the clique parameters in Z; we leave the details as an exercise for the interested reader.

11.3 Extensions of EM

11.3.1 Partial M step

It is not necessary to find the full optimum of the energy term at each iteration. As long as one finds a parameter θ' which has a higher energy than that of the current parameter θ, then the conditions required in section(11.2.2) still hold, and the likelihood cannot decrease at each iteration.

11.3.2 Partial E-step

The E-step requires us to find the optimum of

$$\log p(\mathcal{V}|\theta) \geq -\sum_{n=1}^{N} \langle \log q(h^n|v^n) \rangle_{q(h^n|v^n)} + \sum_{n=1}^{N} \langle \log p(h^n, v^n|\theta) \rangle_{q(h^n|v^n)} \tag{11.3.1}$$

with respect to $q(h^n|v^n)$. The fully optimal setting is

$$q(h^n|v^n) = p(h^n|v^n) \tag{11.3.2}$$

For a guaranteed increase in likelihood at each iteration, from section(11.2.2) we required that this fully optimal setting of q is used. Unfortunately, therefore, one cannot in general guarantee that a partial E-step (in which one would only partially optimsise the lower bound with respect to q for fixed θ) would always increase the likelihood. Of course, it *is* guaranteed to increase the lower bound on the likelihood, though not the likelihood itself. We discuss some partial E-step scenarios below.

Intractable energy

The EM algorithm assumes that we can calculate

$$\langle \log p(h, v|\theta) \rangle_{q(h|v)} \tag{11.3.3}$$

However, there may be cases in which we cannot computationally carry out the averages with respect to the fully optimal form of q. In this case one may consider a restricted class $\mathcal{Q}$ of q-distributions for which the averages can be carried out. For example, one class for which the averages may become computationally tractable is the factorised distributions $q(h|v) = \prod_j q(h_j|v)$; another popular class are Gaussian q distributions. We can then find the best distribution in the class $\mathcal{Q}$ by using a numerical optimisation routine:

$$q^{opt} = \underset{q \in \mathcal{Q}}{\text{argmin}} \, \text{KL}(q(h)|p(h|v, \theta)) \tag{11.3.4}$$

Alternatively, one can assume a certain structured form for the q distribution, and learn the optimal factors of the distribution by free form functional calculus. This approach is taken for example in section(28.4.2).

Viterbi training

An extreme case of a partial E-step is to restrict $q(h^n|v^n)$ to a delta-function. In this case, the entropic term $\langle \log q(h^n|v^n) \rangle_{q(h^n|v^n)}$ is constant (zero for discrete h), so that the optimal delta function q is to set

$$q(h^n|v^n) = \delta\left(h^n, h^n_*\right) \tag{11.3.5}$$

where

$$h^n_* = \underset{h}{\text{argmax}} \, p(h, v^n|\theta) \tag{11.3.6}$$

When used in the energy, the average with respect to q is trivial and the energy becomes simply

$$\sum_{n=1}^{N} \log p(h^n_*, v^n|\theta) \tag{11.3.7}$$

The corresponding bound on the log-likelihood is then

$$\log p(\mathcal{V}|\theta) \geq H + \sum_{n=1}^{N} \log p(h^n_*, v^n|\theta) \tag{11.3.8}$$

where H is the entropy of the delta function (zero for discrete h).

As a partial justification for this technique, provided there is sufficient data, one might hope that the likelihood as a function of the parameter θ will be sharply peaked around the optimum value. This means that at convergence the approximation of the posterior $p(h|v, \theta^{opt})$ by a delta function will be reasonable, and an update of EM using Viterbi training will produce a new θ approximately the same as θ^{opt}. For any highly suboptimal θ, however, $p(h|v, \theta)$ may be far from a delta function, and therefore a Viterbi update is less reliable in terms of leading to an increase in the likelihood itself. This suggests that the initialisation of θ for Viterbi training is more critical than for the standard EM. Note that since Viterbi training corresponds to a partial E-step, EM training with this restricted class of q distribution is therefore only guaranteed to increase the lower bound on the log likelihood, not the likelihood itself.

This technique is popular in the speech recognition community for training HMMs, section(23.2), from where the terminology Viterbi training arises.

Stochastic EM

Another approximate $q(h^n|v^n)$ distribution that is popular is to use an empirical distribution formed by samples from the fully optimal distribution $p(h^n|v^n,\theta)$. That is one draws samples (see chapter(27) for a discussion on sampling) $h_1^n,\ldots,h_L^n$ from $p(h^n|v^n,\theta)$ and forms a q distribution

$$q(h^n|v^n) = \frac{1}{L}\sum_{l=1}^{L}\delta\left(h^n, h_l^n\right) \tag{11.3.9}$$

The energy then becomes proportional to

$$\sum_{n=1}^{N}\sum_{l=1}^{L}\log p(h_l^n, v^n|\theta) \tag{11.3.10}$$

so that, as in Viterbi training, the energy is always computationally tractable for this restricted q class. Provided that the samples from $p(h^n|v^n)$ are reliable, stochastic training will produce an energy function with (on average) the same characteristics as the true energy under the classical EM algorithm. This means that the solution obtained from stochastic EM should tend to that from classical EM as the number of samples increases.

11.4 A failure case for EM

Whilst the EM algorithm is very useful, there are some cases in which it does not work. Consider a likelihood of the form

@@ $$p(v|\theta) = \int_h p\left(v|h,\theta\right)p(h), \qquad \text{with} \qquad p\left(v|h,\theta\right) = \delta\left(v, f(h|\theta)\right) \tag{11.4.1}$$

If we attempt an EM approach for this, this will fail (see also exercise(7.8)). To see why this happens, the M-step sets

@@ $$\theta_{new} = \underset{\theta}{\text{argmax}}\ \left\langle \log p(v,h|\theta)\right\rangle_{p(h|v,\theta_{old})} = \underset{\theta}{\text{argmax}}\ \left\langle \log p(v|h,\theta)\right\rangle_{p(h|v,\theta_{old})} \tag{11.4.2}$$

where we used the fact that for this model $p(h)$ is independent of θ. In the case that $p\left(v|h,\theta\right) = \delta\left(v, f(h|\theta)\right)$ then

@@ $$p(h|v,\theta_{old}) \propto \delta\left(v, f(h|\theta_{old})\right)p(h) \tag{11.4.3}$$

so that optimising the energy gives the update

@@ $$\theta_{new} = \underset{\theta}{\text{argmax}}\ \left\langle \log \delta\left(v, f(h|\theta)\right)\right\rangle_{p(h|v,\theta_{old})} \tag{11.4.4}$$

@@ Since $p(h|v,\theta_{old})$ is zero everywhere except at that h^* for which $v = f(h^*|\theta_{old})$, then the energy term becomes $\log\delta\left(f(h^*|\theta_{old}), f(h^*|\theta)\right)$. This is effectively negative infinity if $\theta \neq \theta_{old}$ and zero when $\theta = \theta_{old}$. Hence $\theta = \theta_{old}$ is optimal and the EM algorithm fails to produce a meaningful parameter update.[2] This situation occurs in practice, and has been noted in particular in the context of Independent Component Analysis[240]. Whilst using a delta-function for the output is clearly extreme, a similar slowing down of parameter updates can occur when the term $p(v|h,\theta)$ becomes close to deterministic.

One can attempt to heal this behaviour by deriving an EM algorithm based on the distribution

@@ $$p_\epsilon(v,h|\theta) = (1-\epsilon)\delta\left(v, f(h|\theta)\right)p(h) + \epsilon n(v,h), \quad 0 \leq \epsilon \leq 1 \tag{11.4.5}$$

[2] For discrete variables and the Kronecker delta, the energy attains the maximal value of zero when $\theta = \theta_{old}$. In the case of continuous variables, however, the log of the Dirac delta function is not well defined. Considering the delta function as the limit of a narrow width Gaussian, for any small but finite width, the energy is largest when $\theta = \theta_{old}$.

where $n(v,h)$ is an arbitrary distribution. The original deterministic model corresponds to $p_0(v,h|\theta)$. Defining @@

$$p_\epsilon(v|\theta) = \int_h p_\epsilon(v,h|\theta), \qquad p_\epsilon(v|\theta) = (1-\epsilon)p_0(v|\theta) + \epsilon n(v) \tag{11.4.6}$$ @@

an EM algorithm for $p_\epsilon(v|\theta)$, $0 < \epsilon < 1$ cannot decrease the likelihood and therefore satisfies @@

$$p_\epsilon(v|\theta_{new}) - p_\epsilon(v|\theta_{old}) = (1-\epsilon)\left(p_0(v|\theta_{new}) - p_0(v|\theta_{old})\right) > 0 \tag{11.4.7}$$

which implies

$$p_0(v|\theta_{new}) - p_0(v|\theta_{old}) > 0 \tag{11.4.8}$$

This means that the EM algorithm for the non-deterministic case $0 < \epsilon < 1$ is guaranteed to increase the likelihood under the deterministic model $p_0(v|\theta)$ at each iteration (unless we are at convergence). See [108] for an application of this 'antifreeze' technique to learning Markov Decision Processes with EM.

11.5 Variational Bayes

Variational Bayes is analogous to EM in that it helps us to deal with hidden variables; however it is a Bayesian method that returns a posterior distribution on parameters, rather than a single best θ as given by maximum likelihood. To keep the notation simple, we'll initially assume only a single datapoint with observation v. Our interest is then the parameter posterior

$$p(\theta|v) \propto p(v|\theta)p(\theta) = \sum_h p(v,h|\theta)p(\theta) \tag{11.5.1}$$ @@

The VB approach assumes a factorised approximation of the joint hidden and parameter posterior, see fig(11.5):

$$p(h,\theta|v) \approx q(h)q(\theta) \tag{11.5.2}$$

The optimal settings for the factors $q(h)$ and $q(\theta)$ can be found by minimising the Kullback-Leibler divergence between $p(h,\theta|v)$ and $q(h)q(\theta)$ as discussed below.

A bound on the marginal likelihood

By minimising the KL divergence,

$$\mathrm{KL}(q(h)q(\theta)|p(h,\theta|v)) = \langle \log q(h) \rangle_{q(h)} + \langle \log q(\theta) \rangle_{q(\theta)} - \langle \log p(h,\theta|v) \rangle_{q(h)q(\theta)} \geq 0 \tag{11.5.3}$$

we arrive at the bound

$$\log p(v) \geq -\langle \log q(h) \rangle_{q(h)} - \langle \log q(\theta) \rangle_{q(\theta)} + \langle \log p(v,h,\theta) \rangle_{q(h)q(\theta)} \tag{11.5.4}$$

Minimizing the Kullback-Leibler divergence with respect to $q(\theta)$ and $q(h)$ is equivalent to obtaining the tightest lower bound on $\log p(v)$. A simple coordinate-wise procedure in which we first fix the $q(\theta)$ and solve for $q(h)$ and then vice versa is analogous to the E and M step of the EM algorithm:

E-step

$$q^{new}(h) = \underset{q(h)}{\mathrm{argmin}}\ \mathrm{KL}\Big(q(h)q^{old}(\theta)|p(h,\theta|v)\Big) \tag{11.5.5}$$

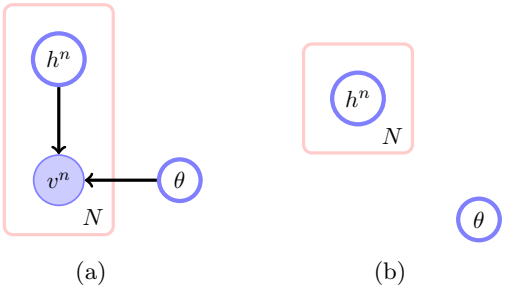
M-step

$$q^{new}(\theta) = \underset{q(\theta)}{\mathrm{argmin}}\ \mathrm{KL}(q^{new}(h)q(\theta)|p(h,\theta|v)) \tag{11.5.6}$$

For a set of observations $\mathcal{V}$ and hidden variables $\mathcal{H}$, the procedure is described in algorithm(11.3). For distributions $q(\mathcal{H})$ and $q(\theta)$ which are parameterised or otherwise constrained, the best distributions in the minimal KL sense are returned. In general, each iteration of VB is guaranteed to increase the bound on the marginal likelihood, but not the marginal likelihood itself. Like the EM algorithm, VB can (and often does) suffer from local maxima issues. This means that the converged solution can be dependent on the initialisation.

Algorithm 11.3 Variational Bayes.

$t = 0$ ▷ Iteration counter
Choose an initial distribution $q_0(\theta)$. ▷ Initialisation
while θ not converged (or likelihood bound not converged) **do**
  $t \leftarrow t + 1$
  $q_t(\mathcal{H}) = \arg\min_{q(\mathcal{H})} \mathrm{KL}(q(\mathcal{H})q_{t-1}(\theta)|p(\mathcal{H}, \theta|\mathcal{V}))$ ▷ E step
  $q_t(\theta) = \arg\min_{q(\theta)} \mathrm{KL}(q_t(\mathcal{H})q(\theta)|p(\mathcal{H}, \theta|\mathcal{V}))$ ▷ M step
end while
return $q_t(\theta)$ ▷ The posterior parameter approximation.



(a) (b)

Figure 11.5: **(a)**: Generic form of a model with hidden variables. **(b)**: A factorised posterior approximation used in Variational Bayes.

Unconstrained approximations

For fixed $q(\theta)$ the contribution to the KL divergence equation (11.5.3) from $q(h)$ is

$$\langle \log q(h) \rangle_{q(h)} - \langle \log p(v, h, \theta) \rangle_{q(h)q(\theta)} = \mathrm{KL}(q(h)|\tilde{p}(h)) + \text{const.} \tag{11.5.7}$$

where

$$\tilde{p}(h) \equiv \frac{1}{\tilde{Z}} \exp\left(\langle \log p(v, h, \theta) \rangle_{q(\theta)} \right) \tag{11.5.8}$$

and $\tilde{Z}$ is a normalising constant. Hence, for fixed $q(\theta)$, the E-step sets $q(h)$ to $\tilde{p}$,

$$q(h) \propto \exp \langle \log p(v, h, \theta) \rangle_{q(\theta)} \propto \exp \langle \log p(v, h|\theta) \rangle_{q(\theta)} \tag{11.5.9}$$

Similarly, for fixed $q(h)$, the M-step sets

$$q(\theta) \propto \exp \langle \log p(v, h, \theta) \rangle_{q(h)} = p(\theta) \exp \langle \log p(v, h|\theta) \rangle_{q(h)} \tag{11.5.10}$$

These E and M-step updates are iterated to convergence.

i.i.d. Data

Under the i.i.d. assumption, we obtain a bound on the marginal likelihood for the whole dataset $\mathcal{V} = \{v^1, \ldots, v^N\}$:

@@ $$\log p(\mathcal{V}) \geq \sum_n \left\{ - \langle \log q(h^n) \rangle_{q(h^n)} - \langle \log q(\theta) \rangle_{q(\theta)} + \langle \log p(v^n, h^n, \theta) \rangle_{q(h^n)q(\theta)} \right\} \tag{11.5.11}$$

The bound holds for any $q(h^n)$ and $q(\theta)$ but is tightest for the converged estimates from the VB procedure. For an i.i.d. dataset, it is straightforward to show that without loss of generality we may assume

$$q(h^1, \ldots, h^N) = \prod_n q(h^n) \tag{11.5.12}$$

Under this we arrive at algorithm(11.4).

Algorithm 11.4 Variational Bayes (i.i.d. data).

$t = 0$ ▷ Iteration counter
Choose an initial distribution $q_0(\theta)$. ▷ Initialisation
while θ not converged (or likelihood bound not converged) **do**
  $t \leftarrow t + 1$
  for $n = 1$ to N **do** ▷ Run over all datapoints
    $q_t^n(h^n) \propto \exp\left(\langle \log p(v^n, h^n|\theta)\rangle_{q_{t-1}(\theta)}\right)$ ▷ E step
  end for
  $q_t(\theta) \propto p(\theta) \exp\left(\sum_n \langle \log p(v^n, h^n|\theta)\rangle_{q_t^n(h^n)}\right)$ ▷ M step
end while
return $q_t^n(\theta)$ ▷ The posterior parameter approximation.

11.5.1 EM is a special case of variational Bayes

If we wish to find a summary of the parameter posterior corresponding to only the most likely point θ_*, then we may use a restricted $q(\theta)$ of the form

$$q(\theta) = \delta(\theta, \theta_*) \tag{11.5.13}$$

where θ_* is the single optimal value of the parameter. If we plug this assumption into equation (11.5.4) we obtain the bound

$$\log p(v|\theta_*) \geq -\langle \log q(h)\rangle_{q(h)} + \langle \log p(v, h, \theta_*)\rangle_{q(h)} + \text{const.} \tag{11.5.14}$$

The M-step is then given by

$$\theta_* = \underset{\theta}{\text{argmax}} \left(\langle \log p(v, h|\theta)\rangle_{q(h)} + \log p(\theta)\right) \tag{11.5.15}$$

For a flat prior $p(\theta) = \text{const.}$, this is therefore equivalent to energy maximisation in the EM algorithm. Using this single optimal value in the VB E-step update for $q(h^n)$ we have

$$q_t^n(h) \propto p(v, h|\theta_*) \propto p(h|v, \theta_*) \tag{11.5.16}$$

which is the standard E-step of EM. Hence EM is a special case of VB under a flat prior $p(\theta) = \text{const.}$ and delta function approximation of the parameter posterior.

11.5.2 An example: VB for the Asbestos-Smoking-Cancer network

In section(9.4) we showed how to apply Bayesian methods to train a belief network, giving rise to a posterior distribution over parameter. In our previous discussion, the data was fully observed. Here, however, we wish to revisit this case, now assuming however that some of the observations can be missing. This complicates the Bayesian analysis and motivates approximate methods such as VB.
Let's reconsider Bayesian learning in the binary variable asbestos-smoking-cancer network, as described in section(9.4)

$$p(a, c, s) = p(c|a, s)p(a)p(s) \tag{11.5.17}$$

in which we use a factorised parameter prior

$$p(\theta_c)p(\theta_a)p(\theta_s) \tag{11.5.18}$$

When all the data $\mathcal{V}$ is i.i.d. and observed, the parameter posterior factorises. However, as we discussed in section(11.1.1) if the state of a (asbestos) is not observed, the parameter posterior no longer factorises:

$$p(\theta_a, \theta_s, \theta_c|\mathcal{V}) \propto p(\theta_a)p(\theta_s)p(\theta_c)p(\mathcal{V}|\theta_a, \theta_s, \theta_c) \tag{11.5.19}$$

$$\propto p(\theta_a)p(\theta_s)p(\theta_c)\prod_n p(v^n|\theta_a, \theta_s, \theta_c) \tag{11.5.20}$$

$$\propto p(\theta_a)p(\theta_s)p(\theta_c)\prod_n p(s^n|\theta_s)\sum_{a^n} p(c^n|s^n, a^n, \theta_c)p(a^n|\theta_a) \tag{11.5.21}$$

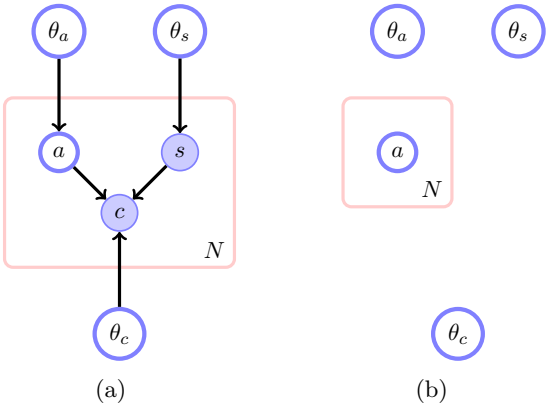


Figure 11.6: **(a)**: A model for the relationship between lung Cancer, Asbestos exposure and Smoking with factorised parameter priors. Variables c and s are observed, but variable a is consistently missing. **(b)**: A factorised parameter posterior approximation.

where the summation over a prevents the factorisation into a product of the individual table parameters. This means that it becomes more awkward to represent the posterior since we cannot exactly do this by using posterior distributions on each parameter table alone. This is a situation in which VB can be useful since it enables us to impose factorisations on the posterior. In VB we consider an approximation to the posterior over the parameters and latent variables $q(h)q(\theta)$. In this case $\theta = (\theta_a, \theta_s, \theta_c)$ and the latent variables are asbestos, a^n (one for each datapoint). For this example the visible variables are smoking and cancer: $\mathcal{V} = \{s^n, c^n\}, n = 1, \ldots, N$. The exact joint distribution over all these variables is then

$$p(\theta_a, \theta_s, \theta_c, a^1, \ldots, a^N|\mathcal{V}) \propto \underbrace{p(\theta_a)p(\theta_s)p(\theta_c)}_{\text{prior}} \underbrace{\prod_n p(c^n|s^n, a^n, \theta_c)p(s^n|\theta_s)p(a^n|\theta_a)}_{\text{posterior}} \tag{11.5.22}$$

In VB we make a factorised assumption, splitting the parameters and latent variables:

$$p(\theta, a^{1:N}|\mathcal{V}) \approx q(\theta)q(a^{1:N}) \tag{11.5.23}$$

From the general results in equation (11.5.9), we have (ignoring terms that are independent of a in the exponent)

@@ $$q(a^{1:N}) \propto \exp\left(\sum_n \langle \log p(c^n|s^n, a^n, \theta_c)\rangle_{q(\theta_c)} + \sum_n \langle \log p(a^n|\theta_a)\rangle_{q(\theta_a)}\right) \tag{11.5.24}$$

From this we see immediately that our approximation automatically factorises

$$q(a^{1:N}) = \prod_n q(a^n) \tag{11.5.25}$$

Similarly, from equation (11.5.10) we have

$$q(\theta) \propto p(\theta) \exp\left(\sum_n \langle \log p(c^n|s^n, a^n, \theta_c)p(s^n|\theta_s)p(a^n|\theta_a)\rangle_{q(a^n)}\right) \tag{11.5.26}$$

$$= p(\theta) \exp\left(\sum_n \langle \log p(c^n|s^n, a^n, \theta_c)\rangle_{q(a^n)}\right)\left\{\prod_n p(s^n|\theta_s)\right\} \exp\left(\sum_n \langle \log p(a^n|\theta_a)\rangle_{q(a^n)}\right) \tag{11.5.27}$$

Since the parameter prior is factorised, we can collect terms in θ_a, θ_s and θ_c and see that the VB assumption automatically results in a factorised parameter posterior approximation. Hence, our VB approximation is of the form, see fig(11.6),

$$p(\theta_a, \theta_s, \theta_c, a^1, \ldots, a^N|\mathcal{V}) \approx q(\theta_a)q(\theta_c)q(\theta_s)\prod_n q(a^n) \tag{11.5.28}$$

All that remains is to form E-step ($q(a^n)$ updates) and M-step ($q(\theta)$ updates), as described below.

M-step: $q(\theta)$ updates

From above we have

$$q(\theta_a) \propto p(\theta_a) \prod_n \exp \left\langle \log p(a^n|\theta_a) \right\rangle_{q(a^n)} \tag{11.5.29}$$

where

$$\left\langle \log p(a^n|\theta_a) \right\rangle_{q(a^n)} = q(a^n = 1) \log \theta_a + q(a^n = 0) \log (1 - \theta_a) \tag{11.5.30}$$

Hence

$$\exp \left\langle \log p(a^n|\theta_a) \right\rangle_{q(a^n)} = \theta_a^{q(a^n=1)} (1 - \theta_a)^{q(a^n=0)} \tag{11.5.31}$$

It is convenient to use a Beta distribution prior,

$$p(\theta_a) = B(\theta_a|\alpha, \beta) \propto \theta_a^{\alpha-1} (1 - \theta_a)^{\beta-1} \tag{11.5.32}$$

since the posterior approximation is then also a Beta distribution:

$$q(\theta_a) = B\left(\theta_a|\alpha + \sum_n q(a^n = 1), \beta + \sum_n q(a^n = 0)\right) \tag{11.5.33}$$

A similar calculation gives

$$q(\theta_s) = B\left(\theta_s|\alpha + \sum_n \mathbb{I}[s^n = 1], \beta + \sum_n \mathbb{I}[s^n = 0]\right) \tag{11.5.34}$$

Finally, we have a table for each of the parental states of c. For example

$$q(\theta_c(a = 0, s = 1)) = B\left(\theta_c|\alpha + \sum_n \mathbb{I}[s^n = 1] q(a^n = 0), \beta + \sum_n \mathbb{I}[s^n = 0] q(a^n = 1)\right) \tag{11.5.35}$$

These are reminiscent of the standard Bayesian equations, equation (9.4.15), except that the missing data counts have been replaced by q's.

E-step: $q(a^n)$ updates

From equation (11.5.24), we have

$$q(a^n) \propto \exp\left(\left\langle \log p(c^n|s^n, a^n, \theta_c) \right\rangle_{q(\theta_c)} + \left\langle \log p(a^n|\theta_a) \right\rangle_{q(\theta_a)}\right) \tag{11.5.36}$$

For example, if assume that for datapoint n, s is in state 1 and c in state 0, then

$$q(a^n = 1) \propto \exp \left\langle \log (1 - \theta_c(s = 1, a = 1)) \right\rangle_{q(\theta_c(s=1,a=1))} + \left\langle \log \theta_a \right\rangle_{q(\theta_a)} \tag{11.5.37}$$

and

$$q(a^n = 0) \propto \exp\left(\left\langle \log (1 - \theta_c(s = 1, a = 0)) \right\rangle_{q(\theta_c(s=1,a=1))} + \left\langle \log (1 - \theta_a) \right\rangle_{q(\theta_a)}\right) \tag{11.5.38}$$

These updates require the Beta distribution averages $\left\langle \log \theta \right\rangle_{B(\theta|\alpha,\beta)}$ and $\left\langle \log (1 - \theta) \right\rangle_{B(\theta|\alpha,\beta)}$; these are straightforward to compute, see exercise(8.17).

The complete VB procedure is then given by iterating equations (11.5.33,11.5.34,11.5.35) and (11.5.37,11.5.38) until convergence.

Given a converged factorised approximation, computing a marginal table such as $p(a = 1|\mathcal{V})$ is then straightforward under the approximation

$$p(a = 1|\mathcal{V}) \approx \int_{\theta_a} q(a = 1|\theta_a)q(\theta_a) = \frac{\alpha + \sum_n q(a^n = 1)}{\alpha + \sum_n q(a^n = 0) + \beta + \sum_n q(a^n = 1)} \tag{11.5.39}$$

The application of VB to learning the tables in arbitrarily structured BNs is a straightforward extension of the technique outlined here. Under the factorised approximation, $q(h, \theta) = q(h)q(\theta)$, one will always obtain a simple updating equation analogous to the full data case, but with the missing data replaced by variational approximations. Nevertheless, if a variable has many missing parents, the number of states in the average with respect to the q distribution can become intractable, and further constraints on the form of the approximation, or additional bounds are required.

One may readily extend the above to the case of Dirichlet distributions on multinomial variables, see exercise(11.5). Indeed, the extension to the exponential family is straightforward.

11.6 Optimising the Likelihood by Gradient Methods

The EM algorithm typically works well when the amount of missing information is small compared to the complete information. In this case EM exhibits approximately the same convergence as Newton based gradient method[256]. However, if the fraction of missing information approaches unity, EM can converge very slowly. In the case of continuous parameters θ, an alternative is to compute the gradient of the likelihood directly and use this as part of a standard continuous variable optimisation routine. The gradient is straightforward to compute using the following identity. Consider the log likelihood

$$L(\theta) = \log p(v|\theta) \tag{11.6.1}$$

The derivative can be written

$$\partial_\theta L(\theta) = \frac{1}{p(v|\theta)}\partial_\theta p(v|\theta) = \frac{1}{p(v|\theta)}\partial_\theta \int_h p(v, h|\theta) \tag{11.6.2}$$

At this point, we take the derivative inside the integral

$$\partial_\theta L(\theta) = \frac{1}{p(v|\theta)} \int_h \partial_\theta p(v, h|\theta) = \int_h p(h|v, \theta)\partial_\theta \log p(v, h|\theta) = \langle \partial_\theta \log p(v, h|\theta) \rangle_{p(h|v,\theta)} \tag{11.6.3}$$

where we used $\partial \log f(x) = (1/f(x))\partial f(x)$. The right hand side is the average of the derivative of the log complete likelihood. This is closely related to the derivative of the energy term in the EM algorithm, though note that the average here is performed with respect the current distribution parameters θ and not θ^{old} as in the EM case. Used in this way, computing the derivatives of latent variable models is relatively straightforward. These derivatives may then be used as part of a standard optimisation routine such as conjugate gradients[256].

11.6.1 Undirected models

Whilst equation (11.6.3) represents the general case, it is not always possible to easily compute the required averages. Consider an undirected model which contains both hidden and visible variables

$$p(v, h|\theta) = \frac{1}{Z(\theta)} \exp\left(\phi(v, h|\theta)\right) \tag{11.6.4}$$

For i.i.d. data, the log likelihood on the visible variables is (assuming discrete v and h)

$$L(\theta) = \sum_n \left(\log \sum_h \exp \phi(v^n, h|\theta) - \log \sum_{h,v} \exp \phi(v, h|\theta) \right) \tag{11.6.5}$$

which has gradient

$$\frac{\partial}{\partial\theta}L = \sum_n \left(\underbrace{\left\langle \frac{\partial}{\partial\theta}\phi(v^n, h|\theta) \right\rangle_{p(h|v^n,\theta)}}_{\text{clamped average}} - \underbrace{\left\langle \frac{\partial}{\partial\theta}\phi(v, h|\theta) \right\rangle_{p(h,v|\theta)}}_{\text{free average}} \right) \tag{11.6.6}$$

For a Markov Network p that is intractable (the partition function Z cannot be computed efficiently and averages *w.r.t.* p cannot be computed exactly), the gradient is particularly difficult to estimate since it is the difference of two averages, each of which needs to be estimated. Even getting the sign of the gradient correct can therefore be computationally difficult. For this reason learning in general unstructured Markov networks is particularly difficult (the unstructured Boltzmann machine with hidden units being a particular case in point).

11.7 Summary

- Provided the data is missing at random, we can safely learn parameters by maximising the likelihood of the observed data.
- Variational Expectation Maximisation is a general-purpose algorithm for maximum likelihood learning under missing information.
- The classical EM algorithm is a special case of V-EM and guarantees an improvement (non-decrease) in the likelihood at each iteration.
- Bayesian learning in the case of missing information is potentially problematic since the posterior is typically not factored according to the prior assumptions. In this case approximations are useful, such as variational Bayes, which assumes a factorisation between the parameters and the latent/missing variables.
- The gradient can be easily computed for latent variable models and may be used as part of an optimisation routine. This provides an alternative training approach in cases when EM is slow to converge.

11.8 Code

`demoEMchestclinic.m`: Demo of EM in learning the Chest Clinic Tables

In the demo code we take the original Chest Clinic network [185] and draw data samples from this network. Our interest is then to see if we can use the EM algorithm to estimate the tables based on the data (with some parts of the data missing at random). We assume that we know the correct BN structure, only that the CPTs are unknown. We assume the logic gate table is known, so we do not need to learn this.

`EMbeliefnet.m`: EM training of a Belief Network

The code implements maximum likelihood learning of BN tables based on data with possibly missing values.

11.9 Exercises

Exercise 11.1 (Printer Nightmare continued)**.** *Continuing with the BN given in fig(9.21), the following table represents data gathered on the printer, where ? indicates that the entry is missing. Each column represents a datapoint. Use the EM algorithm to learn all CPTs of the network.*

fuse assembly malfunction	?	?	?	1	0	0	?	0	?	0	0	?	1	?	1
drum unit	?	0	?	0	1	0	0	1	?	?	1	1	?	0	0
toner out	1	1	0	?	?	1	0	1	0	?	0	1	?	0	?
poor paper quality	1	0	1	0	1	?	1	0	1	1	?	1	1	?	0
worn roller	0	0	?	?	?	0	1	?	0	0	?	0	?	1	1
burning smell	0	?	?	1	0	0	0	0	0	?	0	?	1	0	?
poor print quality	1	1	1	0	1	1	0	1	0	0	1	1	?	?	0
wrinkled pages	0	0	1	0	0	0	?	0	1	?	0	0	1	1	1
multiple pages fed	0	?	1	0	?	0	1	0	1	?	0	0	?	0	1
paper jam	?	0	1	1	?	0	1	1	1	1	0	?	0	1	?

The table is contained in `EMprinter.mat`*, using states* $1, 2,$ `nan` *in place of* $0, 1, ?$ *(since* BRMLtoolbox *requires states to be numbered 1,2,....). Given no wrinkled pages, no burning smell and poor print quality, what is the probability there is a drum unit problem?*

Exercise 11.2. *Consider the following distribution over discrete variables,*

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1|x_2, x_4)p(x_2|x_3)p(x_3|x_4)p(x_4|x_5)p(x_5), \tag{11.9.1}$$

in which the variables x_2 and x_4 are consistently hidden in the training data, and training data for x_1, x_3, x_5 are always present. Derive the EM update for the table $p(x_1|x_2, x_4)$.

@@

Exercise 11.3. *Consider a simple two variable BN*

$$p(y, x) = p(y|x)p(x) \tag{11.9.2}$$

where both y and x are binary variables, $dom(x) = \{1, 2\}$, $dom(y) = \{1, 2\}$. You have a set of training data $\{(y^n, x^n), n = 1, \ldots, N\}$, in which for some cases x^n may be missing. We are specifically interested in learning the table $p(x)$ from this data. A colleague suggests that one can set $p(x)$ by simply looking at datapoints where x is observed, and then setting $p(x = 1)$ to be the fraction of observed x that is in state 1. Explain how this suggested procedure relates to maximum likelihood and EM.

Exercise 11.4. *Assume that a sequence $v_1, \ldots, v_T$, $v_t \in \{1, \ldots, V\}$ is generated by a Markov chain. For a single chain of length T, we have*

$$p(v_1, \ldots, v_T) = p(v_1) \prod_{t=1}^{T-1} p(v_{t+1}|v_t) \tag{11.9.3}$$

For simplicity, we denote the sequence of visible variables as

$$\mathbf{v} = (v_1, \ldots, v_T) \tag{11.9.4}$$

For a single Markov chain labelled by h,

$$p(\mathbf{v}|h) = p(v_1|h) \prod_{t=1}^{T-1} p(v_{t+1}|v_t, h) \tag{11.9.5}$$

In total there are a set of H such Markov chains ($h = 1, \ldots, H$). The distribution on the visible variables is therefore

$$p(\mathbf{v}) = \sum_{h=1}^{H} p(\mathbf{v}|h)p(h) \tag{11.9.6}$$

1. *There are a set of training sequences, $\mathbf{v}^n, n = 1, \ldots, N$. Assuming that each sequence $\mathbf{v}^n$ is independently and identically drawn from a Markov chain mixture model with H components, derive the Expectation Maximisation algorithm for training this model.*

2. *The file* `sequences.mat` *contains a set of fictitious bio-sequence in a cell array* `sequences{\acup{n}}(t)`*. Thus* `sequences{3}(:)` *is the third sequence,* `GTCTCCTGCCCTCTCTGAAC` *which consists of 20 timesteps. There are 20 such sequences in total. Your task is to cluster these sequences into two clusters, assuming that each cluster is modelled by a Markov chain. State which of the sequences belong together by assigning a sequence $\mathbf{v}^n$ to that state for which $p(h|\mathbf{v}^n)$ is highest. You may wish to use* `mixMarkov.m`*.*