Variational Message Passing and its Applications

John M. Winn
St John's College
Cambridge

A dissertation submitted in candidature for the degree of Doctor of Philosophy, University of Cambridge

Inference Group
Cavendish Laboratory
University of Cambridge

Submitted October 2003, revised January 2004

Declaration

I hereby declare that my dissertation entitled "Variational Message Passing and its Applications" is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any other university.

I further state that no part of my dissertation has already been or is being concurrently submitted for any such degree or diploma or other qualification.

Except where explicit reference is made to the work of others, this dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration. This dissertation does not exceed sixty thousand words in length.

Date: ..... Signed: .....

John M. Winn
St John's College
Cambridge
January 13th, 2004

Abstract

This thesis is concerned with the development of Variational Message Passing (VMP), an algorithm for automatically performing variational inference in a probabilistic graphical model. VMP allows learning and reasoning about a system to proceed directly from a given probabilistic model of that system. The utility of VMP has been demonstrated by solving problems in the domains of machine vision and bioinformatics. VMP dramatically simplifies the construction and testing of new variational models and readily allows a range of alternative models to be tested on a given problem.

In chapter 1, a probabilistic approach to automatic learning and reasoning is introduced. Belief propagation, an existing exact inference algorithm that uses message passing in a graphical model, is outlined, along with its limitations. These limitations lead to the need for approximate inference methods, including sampling methods and variational inference. The latter method of variational inference, which provides an analytical approximation to the posterior distribution, is described in detail.

Chapter 2 presents a novel framework for performing automatic variational inference in a wide range of probabilistic models. The core of the framework is the Variational Message Passing algorithm which is an analog of belief propagation that uses message passing within a graphical model to optimise an approximate variational distribution. A software package, called VIBES (Variational Inference in BayESian networks), is presented as an implementation of the VMP framework. A tutorial is included which demonstrates applying VIBES to a small data set.

Chapter 3 sees the framework being applied to the problem of modelling non-linear image manifolds such as those of face images and digits images. In chapter 4, the problems of DNA microarray image analysis and gene expression modelling are addressed, again using the VMP framework.

Chapter 5 extends Variational Message Passing by allowing variational distributions which retain part of the dependency structure of the original model. The resulting Structured VMP algorithm is shown to improve the quality of the approximate inference and hence widen the applicability of the framework. Conclusions and suggestions for future research directions are presented in Chapter 6.

Dedicated to the memory of Carol Melissa Smith and of my mother Barbara Winn.
They both possessed a joy in life and learning that will always be an inspiration.

Acknowledgements

I am very grateful to David MacKay and Chris Bishop for their guidance, ideas and inspiration. I would also like to thank Matthew Beal, Andrew Blake, Phil Cowans, Bill Fitzgerald, Zoubin Ghahramani, David Kreil, Neil Lawrence, Matthew Orton, Ed Ratzer, David Spiegelhalter, Mike Tipping, Hanna Wallach and Sebastian Wills for their advice, insight and many useful discussions.

Finally, I would like to thank Sara for her love and support and for giving me the strength to complete this doctorate.

This work was supported by Microsoft Research Cambridge, the Engineering Department and the Cavendish Laboratory.

Notation

Variables and sets of variables

$X, Y, Z \ldots$	Variables or corresponding nodes in a graph
$\mathbf{X}, \mathbf{Y}, \mathbf{Z} \ldots$	Sets of variables or corresponding sets of nodes in a graph
X_i	The ith variable in set $\mathbf{X}$ or the corresponding node
$X_i = x$	Variable X_i is in state x
$\{x_i\}_{i=1}^N$	The set $\{x_1, \ldots, x_N\}$
$\mathbf{X} \backslash \mathbf{Y}$	The variables in set $\mathbf{X}$ that are not in set $\mathbf{Y}$
$c, d \ldots$	Clusters (small subsets of variables)

Probability theory

$P(X \,\vert\, Y)$	The probability distribution over X given Y (also used to describe a conditional probability density)
$P(X = x \,\vert\, Y = y)$ or $P(x \,\vert\, y)$	The probability that variable X is in state x, given that Y is in state y
$X \sim P(X)$	X is distributed according to $P(X)$
$Q(X \,\vert\, Y)$	The probability distribution which is a variational approximation to $P(X \,\vert\, Y)$
$\mathrm{KL}(Q \,\vert\vert\, P)$	The Kullback-Leibler divergence between the distributions Q and P
$\langle f \rangle_Q$	The expectation of f under the probability distribution Q
$\mathbb{H}(P)$	The entropy of the distribution P
$\delta(X \,\vert\, x_0)$	The continuous probability distribution with the property $\int f(X)\ \delta(X \,\vert\, x_0)\, dX = f(x_0)$ or the discrete distribution that assigns probability 1 to $X = x_0$ and 0 to $X \neq x_0$
$\Phi(c), \Psi(d)$	Cluster potentials (non-negative functions of cluster variables)

Graphical models

pa_i	The variables or nodes corresponding to the parents of X_i in a directed graph
ch_i	The variables or nodes corresponding to the children of X_i in a directed graph
$\mathrm{cp}_i^{(j)}$	The set of parents of X_i excluding the parent X_j (the co-parents)
ne_i	The variables or nodes corresponding to the neighbours of X_i in a graph

Contents

List of Figures

Chapter 1

Inference in Graphical Models

In an increasingly connected and automated world, machines that are capable of learning and reasoning have moved from the realm of science fiction into that of practical necessity. There are an increasing number of situations where machine reasoning is the only form of reasoning that makes practical and economic sense. The commercial case is clear: machine reasoning is cheaper, more consistent, faster and available for longer hours than the human equivalent. The only limitation is on the capabilities of the machines – and there is growing economic pressure to push back these boundaries. Where, historically, the industrial revolution was about augmenting or replacing human labour with machine labour, we are on the verge of an *inference revolution* where machine reasoning augments or replaces human reasoning. Whilst the information revolution has been about communication, processing and storing of information, the inference revolution will be about learning and reasoning and hence the creation of new information.

Aside from economic arguments in favour of machine reasoning, there are many situations where human reasoning cannot be applied for practical reasons. There are times when there is too much information for a human to cope with, like reasoning about all the documents on the world wide web or interpreting the human gene sequence. Or there may be no humans available, such as when controlling a Mars rover or flying a spy plane over hostile territory. In addition, there are security applications where machines are preferable as they cannot be bribed or threatened, have no self-interest and are constantly alert. Some applications, like sorting post, benefit greatly from the speed at which a computer can reason and there are also tasks which humans would prefer not to do themselves, like filtering out junk email. Finally, in the domain of Human-Computer Interaction machine reasoning has the potential to transform the way we interact with our computers.

In all the applications areas mentioned above, some form of machine reasoning is already beginning to be used. Before an inference revolution can occur, however, we will need general purpose inference systems that can quickly be adapted (or self-adapt) to particular applications. This thesis is concerned with developing an inference framework, which is able to reason automatically about a range of domains. I illustrate this by showing how it can be

readily applied to problems in machine vision and bioinformatics. I hope that this framework may provide the first step towards a general purpose inference system that will enable the widespread use of machine-based reasoning.

1.1 Learning Machines

We are interested in making machines that can learn and reason. In order to do this, we have to find answers to the following fundamental questions:

1. How can a machine learn a representation of a system?
2. What form should this representation take?
3. How can we use it to make predictions or reason about the system?

There have been many different approaches to answering these questions. One approach, which was popular in early Artificial Intelligence research, is to represent knowledge by a set of *rules*. Reasoning is then carried out by applying *formal logic* [Winston 1992]. Typically, such systems cannot learn by themselves: instead humans are required to add or update the rules. A well known example of a rule-based system is the enormous CYC Knowledge Base [Lenat 1995] which contains over 60,000 rules (all of which were entered by hand!).

Unfortunately, such systems have not proven successful. Because each rule is taken to be absolutely true, there is no way to handle noisy observations, no way to resolve conflicting rules and no way to recover if a rule is in error. In short, there is no representation of the uncertainty in the rules and the corresponding conclusions. Ad-hoc methods of introducing 'evidence' for each conclusion has allowed limited success in highly constrained environments, such as the MYCIN medical diagnosis system [Shortcliffe 1976]. To make real progress, however, a rigorous way to represent uncertainty throughout the system is required.

1.2 Representing Uncertainty

A method for representing uncertainty must use more than just `true` or `false` values. It must have a way of stating how much the system believes that a particular statement is true. This can be achieved by using a continuous range of values to represent the belief, with `true` and `false` corresponding to the extrema of this range. Cox [1946] laid down the axioms for *probability theory* which uses *probabilities* to provide a mathematically rigorous way of representing degrees of belief. A probability $P(X = x)$ is a degree of belief that a discrete variable X has value x. A probability distribution $P(x)$ is a function that returns the probability that $X = x$ and is defined such that:

$$P(x) \geq 0 \tag{1.1}$$

$$\sum_x P(x) = 1, \tag{1.2}$$

which simply means that probabilities cannot be negative and that the total probability of all possible values of X is one (as it must have *some* value). If X is a continuous variable then $P(x)$ becomes a *probability density* which obeys

$$\int_x P(x)\, dx \quad = \quad 1, \tag{1.3}$$

where $P(x)\, dx$ is the probability that X lies between x and $x + dx$.

A *conditional* probability $P(x \,|\, y)$ is the probability that $X = x$ given that $Y = y$. A *joint* probability $P(x, y)$ is the probability that both $X = x$ and $Y = y$. These definitions both extend to probability densities. The relationship between the conditional and joint probabilities is given by

$$P(x \,|\, y)P(y) = P(x, y) = P(y \,|\, x)P(x). \tag{1.4}$$

This can be rearranged to give *Bayes's theorem*, first stated by Rev. T. Bayes [1763],

$$P(y \,|\, x) = \frac{P(x \,|\, y)P(y)}{P(x)}. \tag{1.5}$$

Bayes's theorem is an important result because it tells us how to update our *prior* belief $P(y)$ to a *posterior* belief $P(y \,|\, x)$ when we make the observation that X is equal to x.

Probability theory provides an alternative to using rules when answering our three fundamental questions. In a probabilistic approach, the representation of a system is a *probabilistic model* and learning and reasoning are then achieved by performing *probabilistic inference* within this model. In the rest of this chapter, I define what a probabilistic model is; describe several types of probabilistic model; explain how to learn a probabilistic model from observed data and give a number of ways of reasoning within a model by performing probabilistic inference.

1.3 Probabilistic Models

A probabilistic model $\mathcal{H}$ consists of:

- **a sample space**: a set of all possible outcomes of an event. For example, this could be the set of all possible configurations of a system with variables $\mathbf{X} = \{X_1, X_2, \dots\}$.
- **a probability distribution**: a function which assigns a probability $P(\mathbf{X} = \mathbf{x})$ to each possible outcome $\mathbf{x}$ in the sample space, such that the probabilities sum to unity. Strictly speaking, the probability distribution $P(\mathbf{X})$ should be written as $P(\mathbf{X} \,|\, \mathcal{H})$ as it is the probability distribution over $\mathbf{X}$ given that we believe the model $\mathcal{H}$ to be true. For compactness, this conditioning on the model is usually omitted.

Thus, a probabilistic model encodes our belief of how likely it is that the system is in any particular state.

Example 1.1: A Table-based Probabilistic Model

Consider the case of a system with n binary variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$. The sample space is the set of all possible configurations of these variables and thus contains 2^n possible outcomes. A simple way of specifying the distribution $P(X_1, \ldots, X_n)$ over these outcomes would be to use a table with 2^n entries containing the probabilities of each outcome. Clearly, such a table would be unfeasibly large for any but the smallest values of n.

Many real world problems have hundreds or thousands of variables. For example, a colour image from a digital camera contains many millions of measurements (three for every pixel). To define probabilistic models for these systems requires a much more compact representation than a table. We can make considerable savings if we can decompose the full joint distribution into a product of factors, each a function of small subsets of variables. To see how this can be done, consider the following example:

Example 1.2: Building a Lie Detector

Suppose we wanted to build a lie detecting machine to determine if a suspect is guilty of a crime. Let the binary variable G be true if the suspect is guilty. We will ask the suspect if he committed the crime and record the yes/no response R and let L represent whether the suspect is lying. The principle behind lie detectors is that most people will become stressed when attempting to deceive another person and that this can be detected by examining their physiological reaction. Let S be whether the suspect is stressed and B be some bio-physical measurements (heart rate, respiratory rate, skin resistance etc.). We can use the chain rule to write down a joint probability distribution over these five variables.

$$P(G, L, R, S, B) = P(G)P(L \mid G)P(R \mid G, L)P(S \mid G, L, R)P(B \mid G, L, R, S). \tag{1.6}$$

We assume that the suspect's stress levels depend only on whether he is lying and so $P(S \mid G, L, R)$ can be simplified to $P(S \mid L)$. Similarly, we assume that the bio-physical measurements depend only on whether the suspect is stressed, so $P(B \mid G, L, R, S)$ reduces to $P(B \mid S)$. Now we can rewrite our joint distribution as a product of factors each involving only a small subset of the variables:

$$P(G, L, R, S, B) = P(G)P(L \mid G)P(R \mid G, L)P(S \mid L)P(B \mid S). \tag{1.7}$$

In this example, we have exploited conditional independencies between variables in the model to factorise the joint distribution. The set of all such independencies defines the *dependency structure* of the model and the corresponding factorisation that can be applied to the joint distribution. Note that the dependency structure of the model only tells us which variables

appear in each factor. A full specification of the model would require each factor function (in this case, each conditional probability distribution) to be defined.

A vivid way to visualise a model's dependency structure is to use a *graph* to show which variables are directly dependent on each other. This had led to the widespread use of probabilistic models based on graphs, known as *graphical models*.

1.4 Graphical Models

A graph consists of a set of nodes and a set of edges that connect some pairs of nodes. In a graphical model, the graph represents a probabilistic model where the nodes correspond to variables, and edges show dependencies between variables. There are various types of graphical model which represent the dependency structure in different ways, including Bayesian networks [Pearl 1988], factor graphs [Frey et al. 1998], Markov random fields [Kinderman and Snell 1980] and chain graphs [Lauritzen and Wermuth 1989].

In addition to providing a visual representation of a probabilistic model, graphs can be represented by simple data structures and lead to efficient, decomposable algorithms.

1.4.1 Bayesian networks

In a Bayesian network, dependency structure is represented by a *directed acyclic graph* (DAG) – a graph where the edges are *directed* (marked with arrows) and there are no cycles (no closed paths following the directions of the edges). The edges are taken to represent causal relationships between the variables corresponding to the parent and child nodes.

In the lie detector example, the joint probability distribution was decomposed as a product of factors, each of the form $P(X \mid Y_1, Y_2, \dots)$. In a Bayesian network, each such factor is represented by making $Y_1, Y_2, \dots$ the parents of X in the graph. It follows from Equation 1.7 that the dependency structure of the lie detector example is represented by the Bayesian network shown in Figure 1.1. It is reasonable to suggest that most people would find such a graphical representation much more compelling and easier to understand than the corresponding decomposed joint distribution.

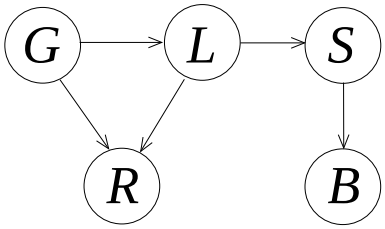


Figure 1.1: A Bayesian network corresponding to the probabilistic model used in the lie detector example on page 4. The variables are: G the guilt of the suspect, L whether the suspect is lying, R the subject's response, S whether the subject is stressed and B the subject's bio-physical measurements.

The edges in a Bayesian network represent conditional independence relationships in that each variable is independent of all ancestor variables conditioned on the state of its parents. For example, knowing a subject's biophysiological state tells us nothing about whether he is lying if we already know he is stressed. That is, B is independent of L conditioned on S.

For a general Bayesian network, the joint probability distribution is written as a product of the distributions for each variable, conditioned on the states of its parent variables:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathrm{pa}_i). \tag{1.8}$$

1.4.2 Factor graphs

A *factor graph* [Frey et al. 1998] is a form of graphical model that explicitly represents how the joint probability decomposes into a product of factor functions. In addition to having a node for each variable, a factor graph also contains a node for each factor function (shown as small black squares). Edges are used to connect each such *function node* to the variables the function depends on. Each edge therefore only connects nodes of different types.

The joint distribution of a general factor graph is given by:

$$P(X_1, \ldots, X_n) = \frac{1}{Z} \prod_{i} \Psi_i(c_i). \tag{1.9}$$

In this equation, each c_i is the small subset of variables that the corresponding factor function Ψ_i depends on. The normalisation constant Z is chosen to ensure that the distribution sums to unity, as required.

Equation 1.8 has the same form as Equation 1.9 and so the joint distribution for any Bayesian network can be represented by a factor graph (this is also true for Markov random fields, see Kschischang et al. [2001]). A Bayesian network may be converted to a factor graph by creating a function node for each variable node X_i and connecting it to that node and all its parents. The function Ψ_i associated with the function node is then the conditional probability $P(X_i \mid \mathrm{pa}_i)$. A factor graph for the lie detector example is shown in Figure 1.2a.

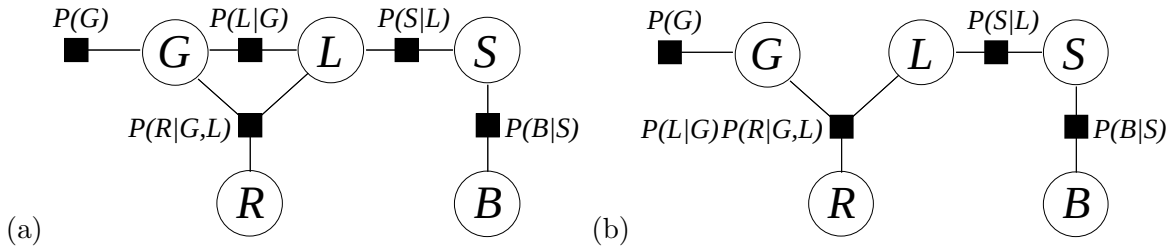


Figure 1.2: (a) A factor graph corresponding to the probabilistic model used in the lie detector. The Bayesian network of Figure 1.1 can be converted to this factor graph by creating a function node for each variable node and connecting it to that node and all its parents. (b) An equivalent factor graph which leads to the same joint probability distribution, but in which there are no cycles.

This method gives a factor graph corresponding to the given Bayesian network. It is not the only such factor graph because function nodes can be combined as needed, for example, to eliminate cycles in the graph. Figure 1.2b shows an alternate factor graph for the lie detector in which the cycle has been removed by combining two function nodes $P(L \mid G)$ and $P(R \mid G, L)$.

1.4.3 Form of local functions

The dependency structure of a model does not dictate the form of the local functions, that is, exactly how dependent variables depend on each other. To complete the definition of a graphical model, each local function must be defined.

For a Bayesian network, the local functions are the conditional probabilities $P(X_i \mid \mathrm{pa}_i)$. Frequently, probability distributions are chosen to be Gibbs distributions, defined as

$$P(\mathbf{X}) = \frac{1}{Z} \exp\left[\frac{-E(\mathbf{X})}{T}\right] \tag{1.10}$$

where E is an *energy function*, T is a temperature and Z is a normalisation constant. Gibbs distributions are used in statistical physics to model the probability of a system being in state $\mathbf{X}$ when its energy as a function of state is $E(\mathbf{X})$. When used in the context of machine learning, the temperature parameter is normally set to be 1.

Many commonly used distributions, including the Gaussian, Gamma, Binomial, Poisson and discrete distributions, belong to a family of Gibbs distributions known as the *exponential family* [Bernardo and Smith 1994, pp.197–202]. The form of a distribution in the exponential family is

$$P(\mathbf{X} \mid \mathbf{Y}) = \exp[\boldsymbol{\phi}(\mathbf{Y})^{\mathrm{T}} \mathbf{u}(\mathbf{X}) + f(\mathbf{X}) + g(\mathbf{Y})] \tag{1.11}$$

where $\mathbf{Y}$ is a vector of parameters. The vector $\boldsymbol{\phi}(\mathbf{Y})$ is called the *natural parameter* vector as it provides a consistent way of parameterising all exponential family distributions. Similarly, $\mathbf{u}(\mathbf{X})$ is called the *natural statistic* vector. The quantity $g(\mathbf{Y})$ acts as a normalisation factor and equals $-\log Z$. For example, a Gaussian distribution may be expressed in this form[1] as

$$\mathcal{N}(x \mid \mu, \gamma^{-1}) = \exp\left(\begin{bmatrix} \gamma\mu \\ -\frac{\gamma}{2} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} x \\ x^2 \end{bmatrix} + 0 + \frac{1}{2}(\log\gamma - \gamma\mu^2 - \log 2\pi)\right). \tag{1.12}$$

Suppose we have a conditional distribution $P(X \mid Y)$ and X is the parent of another variable W. The distribution $P(X \mid Y)$ is said to be *conjugate* if it has the same functional form, with respect to X, as $P(W \mid X)$. This property simplifies the application of Bayes's theorem, as will be shown later. If $P(X \mid Y)$ is in the exponential family form of Equation 1.11, then

[1] In this equation for the Gaussian distribution, γ is the inverse variance, also known as the *precision*. In the standard notation, $\sigma^2 = 1/\gamma$. Precisions will be used instead of variances throughout this thesis.

conjugacy means that $P(W \mid X)$ can be written as

$$P(W \mid X) = \exp[\boldsymbol{\psi}(W)^{\mathrm{T}}\mathbf{u}(X) + h(W)], \tag{1.13}$$

for some definition of the functions $\boldsymbol{\psi}(W)$ and $h(W)$.

A model where all the conditional distributions are both conjugate and in the exponential family is known as a *conjugate-exponential* model.

1.5 Learning a Probabilistic Model

Rather than specifying a probabilistic model explicitly ourselves, we want to learn a probabilistic model of a system directly from some observations of that system. Learning a probabilistic model consists of two stages: *model fitting* and *model selection*.

1.5.1 Model fitting

Suppose we have a probabilistic model $\mathcal{H}$ which defines a distribution $P(\mathbf{X})$ over a set of variables $\mathbf{X}$. The set $\mathbf{X}$ is divided into model *parameters* $\boldsymbol{\theta}$ and *observed data* $\mathbf{D}$. During model fitting, we assume that the model $\mathcal{H}$ is true and aim to learn the parameters given the observed data. Applying Bayes's theorem gives

$$\overbrace{P(\boldsymbol{\theta} \mid \mathbf{D}, \mathcal{H})}^{\text{posterior}} = \frac{\overbrace{P(\mathbf{D} \mid \boldsymbol{\theta}, \mathcal{H})}^{\text{likelihood}}\overbrace{P(\boldsymbol{\theta} \mid \mathcal{H})}^{\text{prior}}}{\underbrace{P(\mathbf{D} \mid \mathcal{H})}_{\text{evidence}}}, \tag{1.14}$$

which allows us to update our *prior* belief about the model parameters $P(\boldsymbol{\theta} \mid \mathcal{H})$ to a *posterior* belief $P(\boldsymbol{\theta} \mid \mathbf{D}, \mathcal{H})$ given the data $\mathbf{D}$. In this way, the parameters of the model have been learnt from the data. The *likelihood* function $P(\mathbf{D} \mid \boldsymbol{\theta}, \mathcal{H})$ simply defines how likely the observed data are given a particular setting of the model parameters $\boldsymbol{\theta}$. The *evidence* $P(\mathbf{D} \mid \mathcal{H})$ is the probability of the data given the choice of model (i.e. the evidence for that model). Learning the parameters of a model in this way is an example of *Bayesian inference*, which will be described further in Section 1.6

1.5.2 Model selection

The task of model selection is to determine which of a number of models $\mathcal{H}_1 \ldots \mathcal{H}_N$ is most plausible given the data. Again, we apply Bayes's theorem:

$$P(\mathcal{H}_i \mid \mathbf{D}) = \frac{P(\mathbf{D} \mid \mathcal{H}_i)P(\mathcal{H}_i)}{P(\mathbf{D})} \tag{1.15}$$

The prior over models $P(\mathcal{H}_i)$ must be selected subjectively. If there is no reason to favour one model over another, it is often chosen to be a uniform distribution, in which case the models

can be ranked by their evidence $P(\mathbf{D} \mid \mathcal{H}_i)$.

We can therefore learn a probabilistic model by proposing a number of models $\mathcal{H}_1 \ldots \mathcal{H}_N$, fitting them to the data and ranking them by their evidence. It is possible to generate new models automatically from a constrained set of models and find which model has the highest posterior probability. This is an example of *structure learning* as the dependency structure of the model is being learned (as opposed to just the model parameters).

1.6 Bayesian Inference

Suppose we have learned a probabilistic model $\mathcal{H}$ with variables $\mathbf{X}$ and wish to use it to reason about the system it is modelling. A subset of the variables $\mathbf{D}$ are observed to have values $\mathbf{d}$. The remaining variables $\mathbf{H}$ are known as hidden or *latent* variables. These latent variables may correspond to actual events that were simply not observed or that have not yet occurred, or they may be fictional – introduced just to improve the representational power of the model.

The task of reasoning or making predictions about the values of some of the latent variables $\mathbf{Z} \subseteq \mathbf{H}$ then corresponds to finding the conditional distribution $P(\mathbf{Z} \mid \mathbf{D} = \mathbf{d})$, also known as a *marginal* distribution as it involves integrating (marginalising) out the variables in $\mathbf{H}$ which are not in $\mathbf{Z}$. The process of computing this posterior is termed *Bayesian inference* (see MacKay [2003] for an excellent introduction to a range of Bayesian inference methods).

Let us assume for now that all variables in our model are discrete. Superficially, the calculation of posterior conditionals appears straightforward, given that the overall joint distribution is supplied by our model. Note that:

$$P(\mathbf{Z} \mid \mathbf{D} = \mathbf{d}) = \frac{P(\mathbf{Z}, \mathbf{D} = \mathbf{d})}{\sum_{\mathbf{Z}} P(\mathbf{Z}, \mathbf{D} = \mathbf{d})} \tag{1.16}$$

so we can compute $P(\mathbf{Z}, \mathbf{D} = \mathbf{d})$ for all possible values of $\mathbf{Z}$ and then normalise to get $P(\mathbf{Z} \mid \mathbf{D} = \mathbf{d})$. The distribution $P(\mathbf{Z}, \mathbf{D} = \mathbf{d})$ can be found by summing the overall joint distribution over all possible configurations of the remaining latent variables $\mathbf{W}$ (those which are in $\mathbf{H}$ but not in $\mathbf{Z}$):

$$P(\mathbf{Z}, \mathbf{D} = \mathbf{d}) = \sum_{\mathbf{W}} P(\mathbf{W}, \mathbf{Z}, \mathbf{D} = \mathbf{d}) \tag{1.17}$$

It is this marginalisation that can cause difficulties because the number of possible configurations will scale exponentially with the size of $\mathbf{W}$. For example, if $\mathbf{W}$ contains n binary variables then the number of configurations will be 2^n which would be intractable for large values of n. Additionally, computational complexity problems can occur if the dimensionality of $\mathbf{Z}$ is large.

The evaluation of this marginal distribution can be made more efficient by exploiting the dependency structure in the model. One technique for achieving this is *variable elimination*.

1.6.1 Variable elimination

Consider again the lie detector example, whose Bayesian network is shown in Figure 1.1 on page 5. Suppose we observe the response of the subject R to be r and find the values of bio-physical measurements B to be b. The variable of interest is the subject's guilt G and so we want to find the conditional distribution $P(G \mid R = r, B = b)$. Following the method above, we marginalise out L and S from the joint distribution in Equation 1.7:

$$P(G, R = r, B = b) = \sum_L \sum_S P(G, L, S, R = r, B = b) \quad (1.18)$$

$$= \sum_L \sum_S P(G)P(L \mid G)P(R = r \mid G, L)P(S \mid L)P(B = b \mid S) \quad (1.19)$$

The summations over L and S can be moved to the right

$$P(G, R = r, B = b) = P(G) \sum_L P(L \mid G)P(R = r \mid G, L) \sum_S P(S \mid L)P(B = b \mid S). \quad (1.20)$$

This process is called *variable elimination* because each summation eliminates one variable and replaces it with a function. For example, the summation over S eliminates all the terms in S from the equation.

Exploiting the dependency structure using variable elimination can significantly reduce the amount of computation that is required: the number of additions scales exponentially with the size of the largest factor function rather than exponentially in the total number of unobserved variables[2].

The computations performed in variable elimination consist of products of local factor functions and summations over local variables. If the graphical model is a tree, all these computations can be performed locally within the graph. This allows the use of a *message passing* algorithm in which the results of one local computation are summarised in a message (typically a short real-valued vector) and passed along an edge to be used in another local computation.

The first method developed to use a message passing algorithm for probabilistic inference was named *belief propagation* and was developed independently by Pearl [1986] and Lauritzen and Spiegelhalter [Spiegelhalter 1986; Lauritzen and Spiegelhalter 1988].

1.6.2 Belief propagation

In the belief propagation (BP) algorithm, the message passed from a node i to another node j can be interpreted as the node i's belief about which state node j is in. For example, if X_j is a discrete node then this would be a vector of the same dimensionality as X_j with each element being proportional to how much node i believes node j to be in the corresponding

[2]The computational saving is not apparent in this small example where the number of variables we are marginalising over is the same as the size of the largest factor function. However, in real models with hundreds of variables, inference is typically not computationally tractable without exploiting dependency structure.

state.

The belief propagation algorithm was originally defined on Bayesian networks. However, it can also be applied in a slightly different form to factor graphs (when it is known as the *Sum-Product algorithm* [Wiberg 1996; Kschischang et al. 2001]), which leads to simpler definitions for the messages. In the Sum-Product algorithm, the message from an unobserved variable node X_i to a function node f_j is defined as:

$$m_{X_i \to f_j} = \prod_{k \in \mathrm{ne}_i \setminus j} m_{f_k \to X_i} \tag{1.21}$$

where ne_i is the set of neighbours of node X_i. Thus, the message sent to a neighbouring function node is just the product of the messages received from all other neighbours (or a *combination* of the beliefs each neighbour has about the state of X_i). If the variable X_i is observed to have value x_i then the message is simply

$$m_{X_i \to f_j} = \delta(X_i \,|\, x_i), \tag{1.22}$$

where $\delta(X_i \,|\, x_i)$ is 1 if $X_i = x_i$ and 0 otherwise. The message from a function node f_i to a variable node X_j is

$$m_{f_i \to X_j} = \sum_{\mathrm{ne}_i \setminus j} f_i(\mathrm{ne}_i) \prod_{k \in \mathrm{ne}_i \setminus j} m_{X_k \to f_i}, \tag{1.23}$$

which can be interpreted as the belief in X_j that is consistent with the local function f_i given the beliefs in the state of all the other variables that the function depends on.

The algorithm is defined on *singly-connected* graphs (i.e. trees). Message passing takes place in two phases: `CollectEvidence` and `DistributeEvidence`, both of which operate with respect to a *root node* X_r. In `CollectEvidence`, messages are passed toward the root: each node sends a message towards the root node as soon as it is able to (as soon as it has received messages on all other edges). During `DistributeEvidence`, messages pass away from the root node in the exact reverse way so that, after both phases, a message has passed in both directions over every edge in the graph. At this point, the distribution over a variable X_i, conditioned on the observed variables $\mathbf{D} = \mathbf{d}$, is computed from the product of all messages received at that node,

$$P(X_i \,|\, \mathbf{D} = \mathbf{d}) = \frac{1}{Z} \prod_{j \in \mathrm{ne}_i} m_{f_j \to X_i}, \tag{1.24}$$

where Z normalises the distribution (as in Equation 1.16).

To understand this algorithm in more detail, consider applying it to the singly-connected lie detector factor graph of Figure 1.2b, with node G as the root node. Firstly, we apply `CollectEvidence` as shown in Figure 1.3a–c. Nodes corresponding to observed variables are shown shaded.

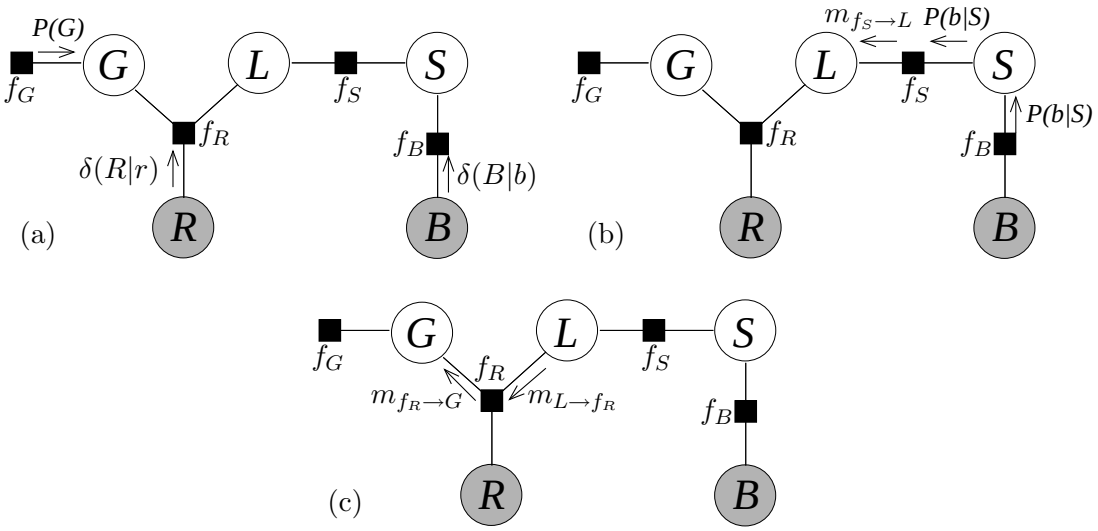


Figure 1.3: (a)-(c) Messages passed during the `CollectEvidence` phase of belief propagation as applied to the lie detector factor graph, with G as the root node. The nodes corresponding to the observed variables R and B are shown shaded.

(a) Initially, only nodes connected to just one edge can send a message: R, B and f_G. Both R and B are observed and so their outgoing messages are delta functions. The message from f_G is simply the corresponding local function $P(G)$.

(b) Each node that has received messages on all edges except the one towards G can send out a message on that edge. First f_B sends out a message $\sum_B P(B \,|\, S)\delta(B \,|\, b)$ which just equals $P(B = b \,|\, S)$. Then S sends a message to f_S which is equal to the message received from f_B (as there is only one term in the product in Equation 1.21). The function node f_S is consequently able to send the function-to-variable message

$$m_{f_S \to L} = \sum_S P(S \,|\, L)P(B = b \,|\, S). \tag{1.25}$$

(c) Finally, the node L sends out the same message it received $m_{L \to f_R} = m_{f_S \to L}$ allowing node f_R to send the message

$$m_{f_R \to G} = \sum_L P(L \,|\, G)P(R = r \,|\, G, L)m_{L \to f_R}. \tag{1.26}$$

At this point, the root node G has received its full complement of incoming messages and so we can use Equation 1.24 to calculate $P(G \,|\, R = r, B = b)$ as being proportional to the

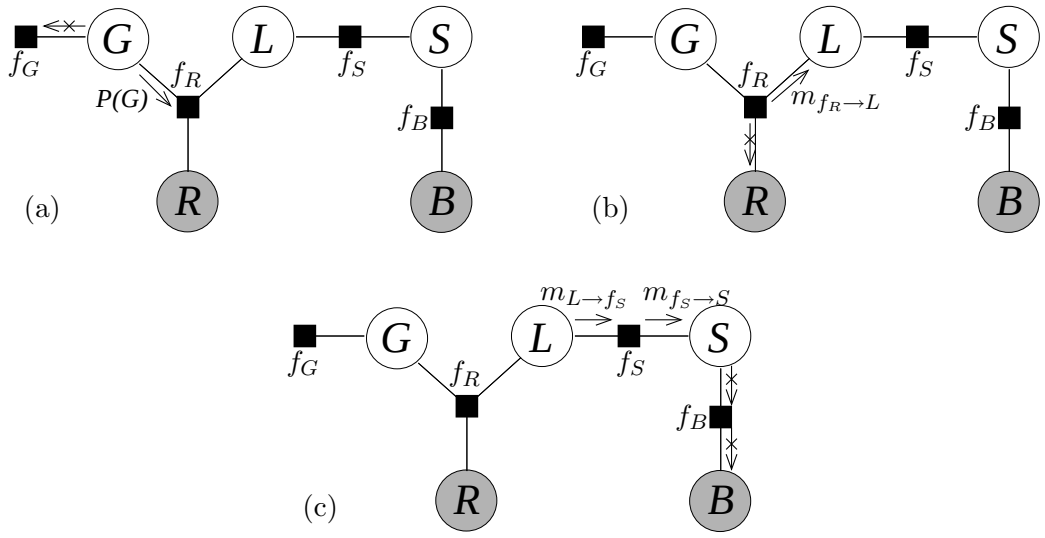


Figure 1.4: (a)-(c) Messages passed during `DistributeEvidence`. Messages marked with crosses are not computed as they are not needed to find the posterior over the latent variables.

product of these messages

$$\begin{aligned} P(G \,|\, R=r, B=b) &= \frac{1}{Z} m_{f_G \to G} m_{f_R \to G} \\ &= \frac{1}{Z} P(G) \sum_L P(L \,|\, G) P(R=r \,|\, G, L) m_{L \to f_R} \\ &\propto P(G) \sum_L P(L \,|\, G) P(R=r \,|\, G, L) \sum_S P(S \,|\, L) P(B=b \,|\, S). \end{aligned} \tag{1.27}$$

This is the same answer as given by variable elimination (Equation 1.20) but found using only local computations and messages sent along edges of the factor graph.

To find the posterior distributions over the remaining latent variables L and S, the `DistributeEvidence` phase must be completed, as shown in Figure 1.4a–c. As only messages into latent variables are needed, messages sent to observed nodes or singly-connected function nodes like f_G need not be computed.

(a) Messages are sent in the reverse direction and reverse order to previously and so we start with the messages sent from the root G. The message to f_G is not computed. The message to f_R is just $P(G)$.

(b) The message from f_R to R is not needed because R is observed. The message from f_R to L is

$$m_{f_R \to L} = \sum_G P(G) P(L \,|\, G) P(R=r \,|\, G, L). \tag{1.28}$$

(c) This message is propagated on through L to f_S. The message from f_S to S is

$$m_{f_S \to S} = \sum_L P(S \,|\, L) m_{L \to f_S}. \tag{1.29}$$

The remaining messages $m_{S\to f_B}$ and $m_{f_B\to B}$ are not computed since B is observed. The posterior distributions over L and S can be found using Equation 1.24,

$$\begin{aligned} P(L \mid R=r, B=b) &= \frac{1}{Z} m_{f_R\to L} m_{f_S\to L} \\ &\propto \sum_G P(G)P(L \mid G)P(R=r \mid G, L) \sum_S P(S \mid L)P(B=b \mid S) \quad (1.30) \\ P(S \mid R=r, B=b) &= \frac{1}{Z} m_{f_B\to S} m_{f_S\to S} \\ &\propto P(B=b \mid S) \sum_L P(S \mid L) \sum_G P(G)P(L \mid G)P(R=r \mid G, L). \quad (1.31) \end{aligned}$$

These are identical to the marginal posterior distributions that would be calculated using variable elimination. Belief propagation has provided marginal posterior distributions for all the latent variables in the graph in an efficient manner, using only local calculations and message passing.

The regular message-passing orderings imposed by `Collect-` and `DistributeEvidence` provide just one method of performing belief propagation. It is possible to use more flexible message passing schemes (see also Frey [1998, pp. 34–35]) in which:

- The network is *initialised* by performing `CollectEvidence` and `DistributeEvidence` with no observations. This initialises the marginals to the *a priori* probabilities.
- When an observation is made at a node, messages are *created* and *propagated* outwards (as if the node was the root in `DistributeEvidence`).
- Messages can be *buffered* so that a node can wait for several messages to be received before sending its own messages.

1.6.3 Inference in graphs with cycles

Belief propagation will perform exact inference only if the graph is singly-connected. However, there are other methods that allow inference in graphs with cycles.

Loopy Belief Propagation

Whilst belief propagation is only guaranteed to converge to give exact posterior marginals if the graph is singly-connected, it can be applied to graphs with cycles (in which case messages propagate continually around the loops[3]). This *loopy* belief propagation (LBP) will generally only give an approximate answer and convergence is not guaranteed. However, LBP has been applied successfully in channel coding [Gallager 1962, 1963; MacKay and Neal 1995; McEliece et al. 1997]. Recently, machine learning researchers have made some progress by considering LBP as finding fixed points in the Bethe free energy [Yedidia et al. 2002; Heskes 2002].

[3]The presence of loops means that the order in which messages are sent can affect the result. The need to specify such an ordering further complicates the use of loopy belief propagation.

Conversion to a Cluster Tree

Alternatively, if we can convert a graph with cycles into an equivalent acyclic graph then standard belief propagation can be applied. Cycles can be removed by combining appropriate clusters of variables into new compound variables.

Example 1.3: Removing a cycle by clustering two variables

Consider the Bayesian network of Figure 1.5a and corresponding cyclic factor graph Figure 1.5b. The set of variables $\{B, C\}$ can be clustered into a new compound variable so as to produce the singly-connected factor graph of Fig. 1.5c. If B and C are discrete variables with m and n states respectively then the new compound variable has mn states.

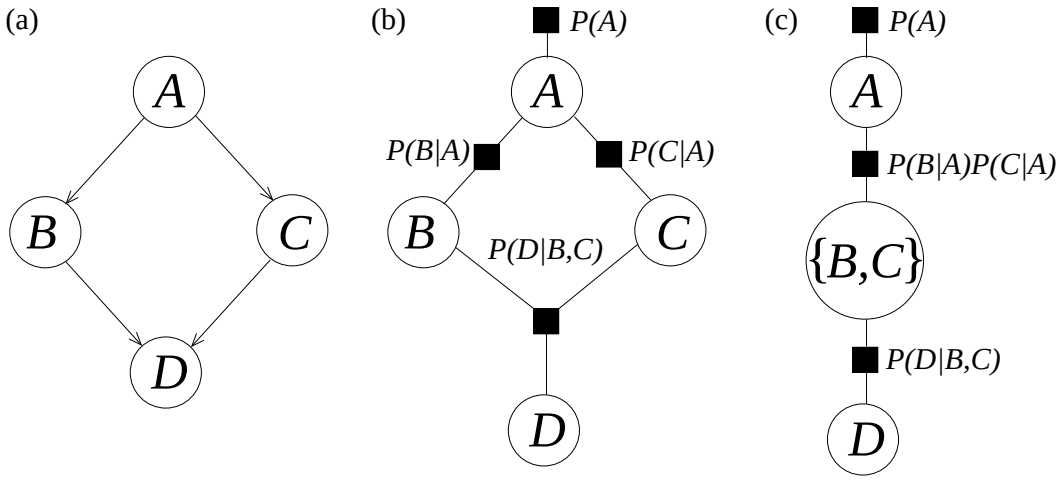


Figure 1.5: (a) A Bayesian network. (b) The corresponding cyclic factor graph. (c) The singly-connected factor graph created by combining the variables B and C into a two-variable cluster corresponding to a new compound variable.

Cutset conditioning

Cutset conditioning [Pearl 1988] provides another method for performing inference in multiply-connected graphs. It works by reducing a multiply-connected graph into a number of conditioned singly-connected graphs, each corresponding to a particular instantiation of a set of variables known as the *cutset*. To perform exact inference, inference must be performed separately in each of these singly-connected graphs. It follows that the computation for cutset conditioning scales exponentially in the size of the cutset and so exact cutset conditioning is not tractable for many models. Where this is the case, there are methods (for example, Darwiche [1995]) for approximating the exact answer by examining only some configurations of the cutset. However, such approximations have given good results only for graphs that are 'nearly singly-connected' and the method is unlikely to provide solutions for dense multiply-connected graphs.

1.6.4 Tractability of exact probabilistic inference

In belief propagation, the computation scales at worst exponentially with the size of the largest local function. When multiply-connected graphs are converted to singly-connected cluster trees, the introduction of the clusters removes some of the structure in the model and hence increases the computation required. Taken to the extreme, grouping all the variables together into a single cluster would remove any advantage given by the dependency structure of the model and reduce to the (usually intractable) situation of manipulating the full joint probability.

In practice, in many cases, it is computationally intractable to perform exact inference in either the singly-connected graph equivalent to one's model or using cutset conditioning. In fact, Dagum and Luby [1993] and Cooper [1990] have shown that, in general, probabilistic inference in Bayesian networks is NP-hard. Another problem to consider is when the variables are *continuous* rather than discrete. Bayesian inference then requires calculating marginals over continuous latent variables, leading to an integral form of Equation 1.17,

$$P(\mathbf{Z}, \mathbf{D} = \mathbf{d}) = \int_{\mathbf{W}} P(\mathbf{W}, \mathbf{Z}, \mathbf{D} = \mathbf{d})\, d\mathbf{W} \tag{1.32}$$

and similar integrals in the messages of belief propagation algorithms. The integrals are typically high-dimensional, non-linear and, with some exceptions (for example Williams and Rasmussen [1996]), non-analytic.

The intractability of exact inference in both discrete and continuous models has led to the development of a number of *approximate inference* techniques. These involve the use of algorithms that compute either approximations to the posterior or approximate expectations of functions under the posterior. Approximate inference methods include sampling methods and variational inference.

1.7 Sampling Methods

Instead of trying to determine the posterior exactly, sampling methods only attempt to obtain a number of samples from it (see MacKay [1998] for an introduction to sampling techniques). If enough independent samples $\{\mathbf{H}_i\}_{i=1}^{S} \sim P(\mathbf{H} \,|\, \mathbf{D})$ can be obtained, an expectation under $P(\mathbf{H} \,|\, \mathbf{D})$ can be approximated with a finite sum across the set of samples:

$$\langle f(\mathbf{H}) \rangle_{P(\mathbf{H} \,|\, \mathbf{D})} \approx \frac{1}{S} \sum_{i=1}^{S} f(\mathbf{H}_i). \tag{1.33}$$

In the limit $S \to \infty$, this approximation is guaranteed to converge to the exact value.

The challenge lies in finding a suitably representative set of a samples. The most widely-used techniques lie in the family of Markov Chain Monte Carlo (MCMC) methods [Neal 1993] which involve creating Markov chains that converge to the desired distribution. The

simplest MCMC algorithm is *Gibbs sampling* which has been successfully applied to Bayesian networks [Pearl 1987; Neal 1992] as well as other graphical models [Geman and Geman 1984; Ackley et al. 1985]. In Gibbs sampling, each successive state $\mathbf{H}^{(k)}$ is selected by modifying a single variable in the previous state $\mathbf{H}^{(k-1)}$. As this modification depends only on local variables in the graph, Gibbs sampling lends itself to graph-based models. This property has enabled the development of BUGS (Bayesian inference Using Gibbs Sampling) by Thomas et al. [1992]: a software package that allows Gibbs sampling to be performed automatically in almost arbitrary graphical models.

MCMC methods are computationally intensive, stochastic and have the problem that it is difficult to determine if the Markov chain has converged on the posterior distribution. In other words, it is hard to determine whether our samples are truly representative of the posterior. As algorithms such as Gibbs sampling take a random walk in the parameter space, it can take an extremely long time to move from one posterior mode to another. For these reasons, I will focus instead on an alternative approximate inference method: *variational inference*.

1.8 Variational Inference

Variational inference [Hinton and van Camp 1993; Neal and Hinton 1998; Jaakkola 1997; Jordan et al. 1998] is an approximate inference method that is deterministic (unlike sampling methods) and aims to optimise directly the accuracy of the approximate posterior distribution. Variational inference is also known as *variational free energy minimisation* or *ensemble learning*.

Historically, variational methods have been used as approximate methods in a number of fields including statistical mechanics [Feynman 1972], statistics [Rustagi 1976], quantum mechanics [Sakurai 1985], and finite element analysis [Bathe 1996]. Broadly speaking, the aim of variational approximation is to convert a complex problem into a simpler problem by decoupling degrees of freedom in the original problem. This decoupling is achieved by the addition of extra parameters, known as variational parameters. When applied to inference, this corresponds to using an approximating distribution that has a simpler dependency structure than that of the exact solution. The idea is that, conditioned on the observed data, certain latent variables may become approximately independent.

In variational inference, the posterior distribution over the latent variables $\mathbf{H}$ is approximated by a *variational distribution*:

$$P(\mathbf{H} \,|\, \mathbf{D}) \approx Q(\mathbf{H}) \tag{1.34}$$

The variational distribution $Q(\mathbf{H})$ is restricted to belong to a family of distributions of simpler form than $P(\mathbf{H} \,|\, \mathbf{D})$. This family is selected with the intention that Q can be made very similar to the true posterior. The difference between Q and this true posterior is measured in terms of a dissimilarity function $d(Q, P)$ and hence inference is performed by selecting the distribution

Q that minimises d. One choice of dissimilarity function where this minimisation is tractable is the *Kullback–Leibler divergence*.

1.8.1 Kullback–Leibler divergence

The Kullback–Leibler (KL) divergence [Kullback and Leibler 1951; Kullback 1959] will be used throughout this thesis as the measure of dissimilarity between the variational distribution and the true posterior. It is an entropy-like measure, defined as

$$\mathrm{KL}(Q \,||\, P) = \int_X Q(x) \log \frac{Q(x)}{P(x)} \; dx \tag{1.35}$$

where log is the natural logarithm[4]. The KL divergence has the property of being zero if Q is equal to P and positive otherwise. For those with a background in source coding, the KL divergence can be interpreted as the average number of *nats*[5] that would be wasted if we encoded samples from a distribution Q, using a perfect encoder that was optimised for samples from P [Hinton and Zemel 1994].

The KL divergence is not a true distance measure as it is not symmetric: in general, $\mathrm{KL}(Q \,||\, P) \neq \mathrm{KL}(P \,||\, Q)$. This is an important distinction to make when we are minimising the KL divergence between an approximating distribution Q and a distribution P. As illustrated by the examples of Figure 1.6, minimising $\mathrm{KL}(Q \,||\, P)$ will favour settings of Q whose probability mass all lies within regions of high probability under P, but without requiring that all such areas are covered. In contrast, minimising $\mathrm{KL}(P \,||\, Q)$ will favour Q distributions which cover all the areas of high probability under P even if this involves assigning high probability to areas of low probability under P.

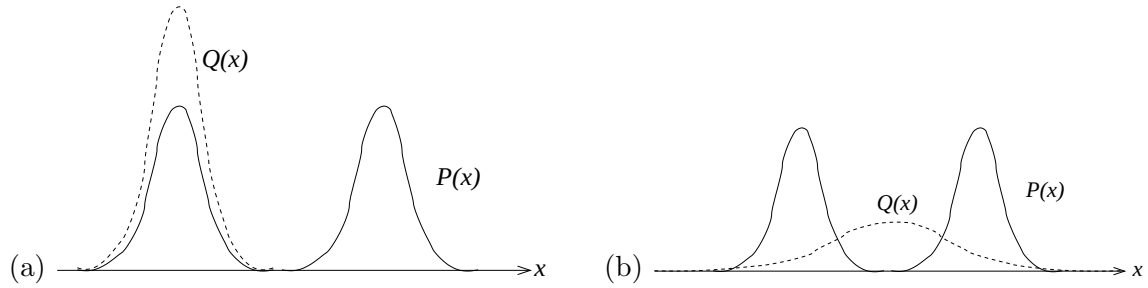


Figure 1.6: Illustration of the asymmetry of the KL divergence. If the distribution P is bimodal and the approximating distribution Q is unimodal then (a) minimising $\mathrm{KL}(Q \,||\, P)$ will give a Q distribution with almost all probability mass in one mode of P and negligible mass in the other mode (b) minimising $\mathrm{KL}(P \,||\, Q)$ will give a Q distribution that covers both modes but which also places high probability mass in the space between them (where $P(x)$ is actually negligible).

[4]Throughout this thesis, $\log x$ will be used to mean the natural logarithm of x, except where the base of the logarithm is specified in a subscript. For example, $\log_2 x$ is the logarithm to base 2 of x.

[5]A nat (natural unit) is a unit of information content. It is similar to a bit (binary unit), but is based on the natural logarithm rather than the logarithm to base 2, so 1 nat = $\log_2 e$ bits.

1.8.2 Minimising the divergence

Choosing the KL divergence as the measure of dissimilarity between the variational approximation $Q(\mathbf{H})$ and the true posterior $P(\mathbf{H} \mid \mathbf{D})$ means that it will be necessary to minimise either

$$\mathrm{KL}(Q \,||\, P) = \int_{\mathbf{H}} Q(\mathbf{H}) \log \frac{Q(\mathbf{H})}{P(\mathbf{H} \mid \mathbf{D})} \, d\mathbf{H} \tag{1.36}$$

or

$$\mathrm{KL}(P \,||\, Q) = \int_{\mathbf{H}} P(\mathbf{H} \mid \mathbf{D}) \log \frac{P(\mathbf{H} \mid \mathbf{D})}{Q(\mathbf{H})} \, d\mathbf{H}. \tag{1.37}$$

Neither of these can be evaluated directly without knowing $P(\mathbf{H} \mid \mathbf{D})$, which is assumed intractable or approximate methods would not be necessary. However, in Equation 1.36 we can make the substitution $P(\mathbf{H} \mid \mathbf{D}) = \mathbf{P}(\mathbf{H}, \mathbf{D})/\mathbf{P}(\mathbf{D})$ and write:

$$\begin{aligned} \mathrm{KL}(Q \,||\, P) &= \int_{\mathbf{H}} Q(\mathbf{H}) \log \frac{Q(\mathbf{H}) P(\mathbf{D})}{P(\mathbf{H}, \mathbf{D})} \, d\mathbf{H} \\ &= \int_{\mathbf{H}} Q(\mathbf{H}) \log \frac{Q(\mathbf{H})}{P(\mathbf{H}, \mathbf{D})} \, d\mathbf{H} + \int_{\mathbf{H}} Q(\mathbf{H}) \log P(\mathbf{D}) \, d\mathbf{H} \\ &= \int_{\mathbf{H}} Q(\mathbf{H}) \log \frac{Q(\mathbf{H})}{P(\mathbf{H}, \mathbf{D})} \, d\mathbf{H} + \log P(\mathbf{D}) \end{aligned} \tag{1.38}$$

In this equation, the last term does not depend on Q, making it necessary only to minimise the first term. Let us define a quantity $\mathcal{L}(Q)$ to be the negative of this first term:

$$\mathcal{L}(Q) \stackrel{\text{def}}{=} \int_{\mathbf{H}} Q(\mathbf{H}) \log P(\mathbf{H}, \mathbf{D}) \, d\mathbf{H} - \int_{\mathbf{H}} Q(\mathbf{H}) \log Q(\mathbf{H}) \, d\mathbf{H} \tag{1.39}$$

$$= \langle \log P(\mathbf{H}, \mathbf{D}) \rangle_{Q(\mathbf{H})} + \mathbb{H}(Q), \tag{1.40}$$

where the notation $\langle . \rangle_Q$ as been introduced to indicate an expectation with respect to the distribution Q and $\mathbb{H}(Q)$ is the entropy of Q. If our joint distribution is a Gibbs distribution (defined in Section 1.4.3), then

$$\log P(\mathbf{H}, \mathbf{D}) = -E(\mathbf{H}, \mathbf{D}) - \log Z \tag{1.41}$$

where E is an energy function and Z is a normalisation constant. We then choose an appropriate form of Q distribution whose entropy is calculable and which also allows the expectation of $E(\mathbf{H}, \mathbf{D})$ to be computed. The value of $\mathcal{L}(Q)$ may then be found and maximised using standard optimisation methods, so as to minimise the KL divergence.

Note that the same trick cannot be used to calculate $\mathrm{KL}(P \,||\, Q)$ as this would require calculating the expectation of $\log Q(\mathbf{H})$ under the true posterior $P(\mathbf{H} \mid \mathbf{D})$, which must be intractable or we would not need to use approximate methods. Being constrained to minimise $\mathrm{KL}(Q \,||\, P)$ instead of $\mathrm{KL}(P \,||\, Q)$ means that the optimal Q distribution will have its mass in some regions where P has high probability but may have low probability in other regions

where P has high probability (such as in the example of Fig. 1.6a). In other words, the optimal Q distribution will generally be more compact than P.

The are other methods for finding an approximate posterior which use the KL divergence. *Expectation propagation* [Minka 2001a,b] is a recently developed algorithm where the approximate posterior $Q(\mathbf{x})$ consists of a normalised product of terms $\tilde{t}_i(\mathbf{x})$ each approximating a corresponding term $t_i(\mathbf{x})$ in the true posterior. This approximation is made by minimising the divergence $\mathrm{KL}(\hat{P}_i \,||\, Q)$ where $\hat{P}_i$ is the normalised product of one term of the true posterior t_i and all the remaining approximate terms $\{\tilde{t}_j\}_{j \neq i}$. As the optimisation process depends on the current overall approximation, the process is iterative and terms can be refined in any order. Hence, refinement of terms proceeds in a similar fashion to the way factors are updated in variational inference. Whilst expectation propagation will not be investigated further as part of this thesis, in Chapter 6 I propose some further work involving expectation propagation and its relation to variational inference.

1.8.3 Variational model selection

The quantity $\mathcal{L}(Q)$ plays a useful role in the task of model selection. Let us reintroduce the conditioning on the model $\mathcal{H}$ into our probability distributions. We can now view the KL divergence as the difference between $\mathcal{L}(Q)$ and the log evidence given the model $\mathcal{H}$:

$$\mathrm{KL}(Q \,||\, P) \quad = \quad \log P(\mathbf{D} \,|\, \mathcal{H}) - \mathcal{L}(Q). \tag{1.42}$$

Given that the KL divergence must be zero or positive, it follows that

$$\mathcal{L}(Q) \quad \leq \quad \log P(\mathbf{D} \,|\, \mathcal{H}) \tag{1.43}$$

and so $\mathcal{L}(Q)$ provides a *lower bound* on the log evidence, with the difference being the KL divergence (as illustrated in Figure 1.7). It follows that if Q is optimised to be a good approximation in terms of KL divergence, then the bound will be tight and $\mathcal{L}(Q)$ will provide a good approximation to the log evidence. From Equation 1.15, the posterior probability of a particular model $\mathcal{H}_i$ given the data is:

$$P(\mathcal{H}_i \,|\, \mathbf{D}) \quad = \quad \frac{P(\mathbf{D} \,|\, \mathcal{H}_i) P(\mathcal{H}_i)}{P(\mathbf{D})} \tag{1.44}$$

$$\approx \quad \frac{\exp(\mathcal{L}_i(Q_i)) P(\mathcal{H}_i)}{P(\mathbf{D})}, \tag{1.45}$$

where $\mathcal{L}_i(Q_i)$ is the lower bound for the model $\mathcal{H}_i$ and Q_i has been optimised to minimise KL divergence. Assuming that $P(\mathcal{H}_i)$ is a uniform prior, we may then make an approximate ranking amongst models by evaluating $\mathcal{L}_i(Q_i)$. In this way, the variational approach provides a method of comparing many different Bayesian models objectively.

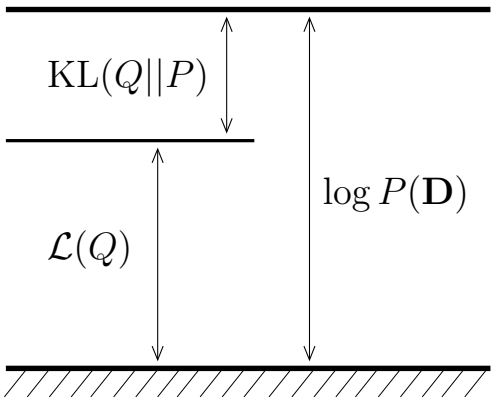


Figure 1.7: The quantity $\mathcal{L}(Q)$ provides a lower bound on the log evidence, $\log P(\mathbf{D})$, with the difference being given by the KL divergence, $\mathrm{KL}(Q \,||\, P)$. By maximising $\mathcal{L}(Q)$, we can minimise the KL divergence since the log evidence is fixed with respect to Q.

1.8.4 Factorised Q distribution

As mentioned previously, we want to choose a variational distribution $Q(\mathbf{H})$ with a simpler dependency structure than that of the model so as to make the calculation of the lower bound $\mathcal{L}(Q)$ tractable. One way to simplify the dependency structure is by choosing a variational distribution where disjoint groups of variables are independent. This is equivalent to choosing Q to have a *factorised* form

$$Q(\mathbf{H}) = \prod_i Q_i(\mathbf{H}_i), \tag{1.46}$$

where $\{\mathbf{H}_i\}$ are the disjoint groups of variables. It is essential to remember that Q is an approximation to the *posterior* distribution conditioned on the observed data. Choosing a Q distribution which is factorised does not mean that the relationships between groups of variables in the original model are in any way lost or ignored. All that it means is that any dependencies between such variables that remain *given the observations*, will not be captured in the approximation. In cases where we expect the posterior distribution to be tightly peaked around a single mode, neglecting such dependencies will still lead to a good approximation.

Substituting this form of Q into our expression for $\mathcal{L}(Q)$ in Equation 1.40 gives

$$\begin{aligned}\mathcal{L}(Q) &= \langle \log P(\mathbf{H}, \mathbf{D}) \rangle_{\prod_i Q_i(\mathbf{H}_i)} - \int_{\mathbf{H}} \prod_i Q_i(\mathbf{H}_i) \sum_j \log Q_j(\mathbf{H}_j)\, d\mathbf{H} \\ &= \int_{\mathbf{H}} \prod_i Q_i(\mathbf{H}_i) \log P(\mathbf{H}, \mathbf{D})\, d\mathbf{H} - \sum_i \int_{\mathbf{H}_i} Q_i(\mathbf{H}_i) \log Q_i(\mathbf{H}_i)\, d\mathbf{H}_i. \end{aligned} \tag{1.47}$$

Now separate out terms in one factor Q_j

$$\mathcal{L}(Q) = \int_{\mathbf{H}_j} Q_j(\mathbf{H}_j) \langle \log P(\mathbf{H}, \mathbf{D}) \rangle_{\prod_{i \neq j} Q_i(\mathbf{H}_i)}\, d\mathbf{H}_j + \mathbb{H}(Q_j) + \sum_{i \neq j} \mathbb{H}(Q_i) \tag{1.48}$$

and define

$$Q_j^*(\mathbf{H}_j) = \frac{1}{Z}\exp\left(\langle \log P(\mathbf{H},\mathbf{D})\rangle_{\prod_{i\neq j} Q_i(\mathbf{H}_i)}\right), \tag{1.49}$$

where Z is the normalisation factor needed to make Q^* a valid probability distribution. The bound may now be written as

$$\mathcal{L}(Q) = \int_{\mathbf{H}_j} Q_j(\mathbf{H}_j)\log Q_j^*(\mathbf{H}_j)\, d\mathbf{H}_j - \log Z + \mathbb{H}(Q_j) + \sum_{i\neq j}\mathbb{H}(Q_i) \tag{1.50}$$

$$= -\mathrm{KL}(Q_j \,||\, Q_j^*) - \log Z + \sum_{i\neq j}\mathbb{H}(Q_i). \tag{1.51}$$

The last two terms do not depend on Q_j and so $\mathcal{L}(Q)$ is maximised with respect to Q_j by minimising $\mathrm{KL}(Q_j \,||\, Q_j^*)$. As described earlier, the KL divergence between two distributions has a minimum value of zero which only occurs where the distributions are equal. Thus the bound can be maximised by setting $Q_j = Q_j^*$. This solution for this optimal distribution Q_j^* can be compactly written in terms of its logarithm:

$$\log Q_j^*(\mathbf{H}_j) = \langle \log P(\mathbf{H},\mathbf{D})\rangle_{\sim Q(\mathbf{H}_j)} + \text{const.} \tag{1.52}$$

which introduces the notation $\langle . \rangle_{\sim Q(\mathbf{H}_j)}$ to mean an expectation with respect to all factors except $Q(\mathbf{H}_j)$. This is an implicit solution as it depends on the settings of the remaining factors $\{Q(\mathbf{H}_i)\}_{i\neq j}$. It follows that $\mathcal{L}(Q)$ can be maximised by picking each factor in turn, updating it using Equation 1.49 and then repeating the entire procedure until convergence. Consequently, $\mathcal{L}(Q)$ will increase monotonically throughout the optimisation. The bound $\mathcal{L}(Q)$ can be evaluated at any point and so convergence can be diagnosed when the increase in the bound is negligible over the course of an iteration across all the factors. The inference process is summarised in Algorithm 1.1.

Algorithm 1.1 Variational Inference with a factorised Q distribution

1. Initialise Q distributions for all factors.
2. For each factor j, update Q_j distribution to maximise lower bound using Equation 1.52.
3. Calculate the new value of the lower bound $\mathcal{L}(Q)$.
4. If increase in bound is negligible, stop. Otherwise repeat from step 2.

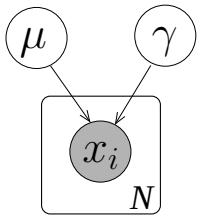


Figure 1.8: The Bayesian network for a probabilistic model that represents a set of N observed data points with a Gaussian distribution of mean μ and precision γ. The rectangle is a *plate*, which indicates that the contained node and its connected edges are duplicated N times.

1.8.5 Example: a univariate Gaussian model

Consider the simple probabilistic model $\mathcal{H}$ that represents a set of observed one-dimensional data $\{x_i\}_{i=1}^N$ with a univariate Gaussian distribution of mean μ and precision γ:

$$P(\mathbf{x} \,|\, \mathcal{H}) = \prod_{i=1}^{N} \mathcal{N}(x_i \,|\, \mu, \gamma^{-1}). \tag{1.53}$$

The Bayesian network for this model is shown in Fig. 1.8. Note the use of a *plate* to indicate N copies of the node x, to save having to draw N individual indexed nodes. The parameters μ and γ are the latent variables in this model and variational inference can be used to learn an approximate posterior distribution over μ and γ given the data. However, it is first necessary to complete the model by defining prior distributions over μ and γ. To make inference tractable, conjugate priors are chosen,[6]

$$P(\mu) = \mathcal{N}(\mu \,|\, m, \beta^{-1}) \tag{1.54}$$

$$P(\gamma) = \text{Gamma}(\gamma \,|\, a, b), \tag{1.55}$$

where the Gamma distribution is parameterised as defined in Appendix A.3. The variational distribution is chosen to be fully factorised:

$$Q(\mu, \gamma) = Q_\mu(\mu) Q_\gamma(\gamma). \tag{1.56}$$

The choice of conjugate priors means that the optimal Q distributions have the same form as the corresponding factors in P,

$$Q_\mu(\mu) = \mathcal{N}(\mu \,|\, m', \beta'^{-1}) \tag{1.57}$$

$$Q_\gamma(\gamma) = \text{Gamma}(\gamma \,|\, a', b'), \tag{1.58}$$

and so inference involves updating the set of variational parameters $\boldsymbol{\theta} = \{m', \beta', a', b'\}$. Ap-

[6] For full conjugacy, we could use a Normal-Gamma prior over both μ and γ. However, this would lead to an exact solution and would not demonstrate the use of a separable variational distribution.

plying Equation 1.49 gives

$$\begin{aligned}
\beta' &= \beta + N\langle\gamma\rangle && (1.59)\\
m' &= \frac{1}{\beta'}\left(\beta m + \langle\gamma\rangle\sum_{i=1}^{N} x_i\right) && (1.60)\\
a' &= a + \frac{N}{2} && (1.61)\\
b' &= b + \frac{1}{2}\sum_{i=1}^{N}(x_i^2 - 2x_i\langle\mu\rangle + \langle\mu^2\rangle), && (1.62)
\end{aligned}$$

where all expectations are with respect to the Q distribution. Inference proceeds by applying Equations 1.59–1.62 until convergence.

At convergence, the variational distribution over the parameters will be the separable distribution closest to the true posterior, in terms of KL divergence. To illustrate this with an example, consider the data set of just four samples from a Gaussian distribution with mean 5 and standard deviation 1. The model parameters are chosen to give broad priors over μ and γ, by setting $m = 0$, $\beta = 0.001$, $a = 0.001$ and $b = 0.001$. Figure 1.9a shows the variational posterior Q over μ and γ, along with the true posterior P. The distributions can be seen to be similar, particularly in areas of high probability under P, but differ in shape where the separable variational solution is unable to capture correlations between the two variables. Note that minimising the $\mathrm{KL}(Q\,||\,P)$ has favoured a Q distribution that lies *inside* P. By this I mean that it has avoided assigning high probability to any areas of low probability under P, at the cost of assigning near zero probability to some relatively probable areas. This effect is due to the constraint of using a separable distribution and minimising $\mathrm{KL}(Q\,||\,P)$ rather than $\mathrm{KL}(P\,||\,Q)$. Figure 1.9b shows the actual distribution the data points were sampled from, together with the four data points and samples from the variational posterior, showing the uncertainty in the inferred mean and precision.

To obtain an estimate of the log evidence, the bound $\mathcal{L}(Q)$ can be calculated using Equation 1.47 and the distribution entropies given in Appendix A:

$$\begin{aligned}
\mathcal{L}(Q) = {} & \frac{N}{2}\left[\langle\log\gamma\rangle - \log 2\pi\right] - \frac{\langle\gamma\rangle}{2}\sum_{i=1}^{N}\left[x_i^2 - 2x_i\langle\mu\rangle + \langle\mu^2\rangle\right]\\
& + \frac{1}{2}\left[\log\beta - \beta(\langle\mu^2\rangle - 2m\langle\mu\rangle + m^2)\right] - \frac{1}{2}\left[\log\beta' - 1\right]\\
& + \left[a\log b + a\langle\log\gamma\rangle - b\langle\gamma\rangle - \log\Gamma(a)\right]\\
& - \left[a'\log b' + a'\langle\log\gamma\rangle - b'\langle\gamma\rangle - \log\Gamma(a')\right] \qquad (1.63)
\end{aligned}$$

where, again, all expectations are with respect to Q.

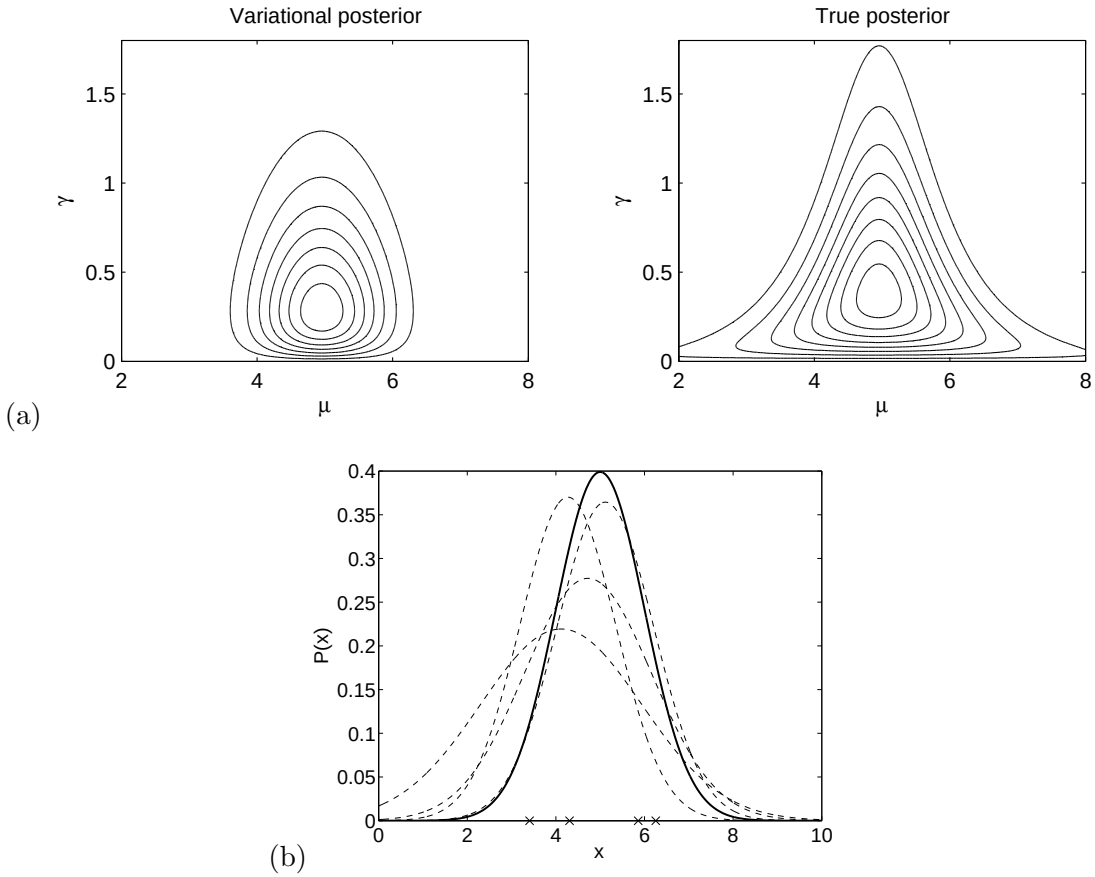


Figure 1.9: (a) Variational and true posterior over the mean μ and precision γ of a Gaussian given four samples from it. The priors over the parameters were $P(\mu) = \mathcal{N}(0, 1000)$ and $P(\gamma) = \text{Gamma}(0.001, 0.001)$. (b) The actual distribution the samples were drawn from (thick line) along with the four samples taken (crosses). The dashed lines show distributions whose parameters are samples from the variational posterior.

1.8.6 Comparison to Maximum A Posteriori

The above method gave a posterior distribution over the parameters μ and γ. It might seem more intuitive to determine optimal single values of μ and γ. This can be achieved using a variational distribution consisting of delta function distributions

$$Q_\mu(\mu) = \delta(\mu \,|\, \mu') \tag{1.64}$$

$$Q_\gamma(\gamma) = \delta(\gamma \,|\, \gamma') \tag{1.65}$$

where the variational parameters μ' and γ' are the single values of the parameters which are to be optimised by maximising $\mathcal{L}(Q)$.

From Equation 1.47, the bound is

$$\mathcal{L}(Q) = \langle \log P(\mathbf{x}, \mu, \gamma) \rangle \tag{1.66}$$

$$= \log P(\mathbf{x}, \mu', \gamma') \tag{1.67}$$

and thus maximising the bound simply corresponds to maximising the posterior probability with respect to the parameters, a method which is known as *Maximum A Posteriori* (MAP). This gives the following update equations, which can again be solved iteratively:

$$\mu' = \frac{\beta m + \gamma' \sum_{i=1}^{N} x_i}{\beta + N\gamma'} \tag{1.68}$$

$$\gamma' = \frac{N + 2(a-1)}{2b + \sum_{i=1}^{N}(x_i - \mu')^2} \tag{1.69}$$

Such maximisation methods have a number of problems. Firstly, the uncertainty in the parameters is no longer being represented, which leads to over-fitting – the situation where a model fits the observed data too tightly, so that it is unable to generalise to new data. Secondly, the optimal values given by the maximisation depend on the parameterisation of the distribution function – it is *basis dependent*. If we make a nonlinear change of basis from some parameter θ to a new parameter $u = f(\theta)$, the probability density is transformed to

$$P(u) = P(\theta)\left|\frac{\partial\theta}{\partial u}\right|. \tag{1.70}$$

The maximum of $P(u)$ will typically not be the same as the maximum of $P(\theta)$. By using variational distributions over the parameters that are not delta distributions, the variational approach is made invariant to reparameterisation of the model, as the variational posterior distributions are transformed with the parameters. In other words, standard variational inference (without delta distributions) is *not* basis dependent.

To illustrate this difference, consider reparameterising the Gaussian model by making the transformation $\nu = \log\gamma$. The prior on γ must be transformed to give the prior on ν

$$P(\nu|a,b) = P(\gamma|a,b)\left|\frac{\partial\gamma}{\partial\nu}\right| \tag{1.71}$$

$$= P(\gamma|a,b)\,\gamma \tag{1.72}$$

The MAP estimate of ν is now given by

$$\nu' = \log\left(\frac{N + 2a}{2b + \sum_{i=1}^{N}(x_i - \mu')^2}\right) \tag{1.73}$$

which gives a different answer from that of Equation 1.69. In contrast, optimising the variational posterior by applying Equation 1.52 to the transformed model leads to the same update equations as for the untransformed model (Equations 1.59–1.62).

It is worth noting that, in many applications, only a single value of a latent variable can be used and so maximisation methods must be applied. An example of this is a decoder in a communications system, which must infer which codeword was sent over a noisy channel, given the received signal. As only a single symbol can be output by the decoder, a maximisation method must be used to select the most probable transmitted codeword.

1.8.7 Example: a Gaussian mixture model

We will now consider another example of variational inference, this time using a *mixture model*. Mixture models [Everitt and Hand 1981; Bishop 1995, Section 2.6] provide one way of obtaining distributions with a richer probability density than simple exponential family distributions. In a mixture model, the distribution over a variable x is a weighted sum of a number of simple distributions

$$P(x \mid \{\pi_k\}, \{\boldsymbol{\theta}_k\}) = \sum_{k=1}^{K} \pi_k P_k(x \mid \boldsymbol{\theta}_k) \tag{1.74}$$

where each P_k is a simple distribution with parameters $\boldsymbol{\theta}_k$ and a corresponding *mixture coefficient* π_k which indicates the weight of the distribution in the weighted sum. The K mixture coefficients must sum to one.

In a Gaussian mixture model, each of the mixture components is a Gaussian distribution with its own mean μ_k and precision γ_k and so the distribution is

$$P(x \mid \{\pi_k\}, \{\mu_k\}, \{\gamma_k\}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \gamma_k^{-1}). \tag{1.75}$$

It is important to note that the logarithm of this distribution (or indeed any mixture distribution) cannot be found analytically, because it contains a summation. It follows that, if we use a mixture distribution as a conditional distribution in our model, variational inference will not be tractable as we will not be able to evaluate $\langle \log P(\mathbf{H}, \mathbf{D}) \rangle_Q$ during the optimisation of the Q distribution.

To solve this problem, we introduce an additional latent variable λ for the data point x which indicates which component distribution the data point was drawn from. Thus, λ is a discrete variable with one of K values, corresponding to one of the K component distributions. The distribution may now be written as

$$P(x \mid \lambda, \{\mu_k\}, \{\gamma_k\}) = \prod_{k=1}^{K} \mathcal{N}(X \mid \mu_k, \gamma_k^{-1})^{\delta(\lambda,k)} \tag{1.76}$$

where the discrete function $\delta(\lambda, k)$ is 1 if $\lambda = k$ and 0 otherwise. Consequently, the value of λ will 'pick out' the corresponding mixture distribution from the product. If we then assign a prior $P(\lambda)$ over λ, the marginal distribution of X is given by

$$P(x \mid \{\mu_k\}, \{\gamma_k\}) = \sum_{\lambda=1}^{K} P(\lambda) \prod_{k=1}^{K} \mathcal{N}(x \mid \mu_k, \gamma_k^{-1})^{\delta(\lambda,k)} \tag{1.77}$$

$$= \sum_{\lambda=1}^{K} P(\lambda) \mathcal{N}(x \mid \mu_k, \gamma_k^{-1}) \tag{1.78}$$

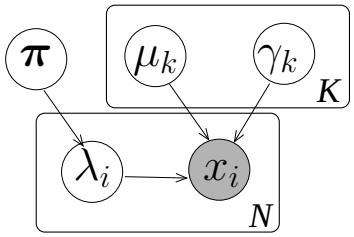


Figure 1.10: The Bayesian network for a probabilistic model which represents a set of N observed data points by a mixture of Gaussian distributions. The top plate contains the K sets of component means and precisions. A discrete variable λ_i has been introduced for each data point to indicate from which mixture component it has been drawn.

which gives us the desired mixture distribution of Equation 1.75, with $P(\lambda = k)$ replacing the mixture coefficients π_k.

We can now define a mixture model $\mathcal{H}$ which represents a set of observed one-dimensional data $\{x_i\}_{i=1}^N$ with a Gaussian mixture distribution

$$P(\mathbf{x} \,|\, \mathcal{H}) = \prod_{i=1}^{N} \prod_{k=1}^{K} \mathcal{N}(x_i \,|\, \mu_k, \gamma_k^{-1})^{\delta(\lambda_i, k)} \tag{1.79}$$

where a separate indicator variable λ_i has been introduced for each data point. These indicator variables are given distributions governed by a hyperparameter $\boldsymbol{\pi}$, such that

$$P(\lambda_i = k \,|\, \boldsymbol{\pi}) = \pi_k \tag{1.80}$$

where π_k is the kth element of the vector $\boldsymbol{\pi}$. We complete the definition of the model by defining conjugate priors over each of the parameters[7]

$$\begin{aligned} P(\mu_k) &= \mathcal{N}(\mu_k \,|\, m, \beta^{-1}) && (1.81)\\ P(\gamma_k) &= \text{Gamma}(\gamma_k \,|\, a, b) && (1.82)\\ P(\boldsymbol{\pi}) &= \text{Dirichlet}(\boldsymbol{\pi} \,|\, \mathbf{u}), && (1.83) \end{aligned}$$

where the Dirichlet distribution is as defined in Appendix A.5. The Bayesian network for the complete model is shown in Figure 1.10.

Once again, the variational distribution is chosen to be fully factorised with respect to all the latent variables

$$Q(\{\mu_k\}, \{\gamma_k\}, \{\lambda_i\}, \boldsymbol{\pi}) = Q_\pi(\boldsymbol{\pi}) \prod_{k=1}^{K} Q_{\mu_k}(\mu_k) Q_{\gamma_k}(\gamma_k) \prod_{i=1}^{N} Q_{\lambda_i}(\lambda_i). \tag{1.84}$$

[7] A separable prior over μ_k and γ_k has be chosen for simplicity. The use of a Normal-Gamma prior would mean that the variational posterior over these two parameters would not be separable, leading to a slightly improved approximation.

When optimised, the factors of Q have the same form as the corresponding factors in P

$$\begin{aligned} Q_{\mu_k}(\mu_k) &= \mathcal{N}(\mu_k \,|\, m'_k, \beta'^{-1}_k) && (1.85)\\ Q_{\gamma_k}(\gamma_k) &= \text{Gamma}(\gamma_k \,|\, a'_k, b'_k) && (1.86)\\ Q_{\pi}(\boldsymbol{\pi}) &= \text{Dirichlet}(\boldsymbol{\pi} \,|\, \mathbf{u}') && (1.87) \end{aligned}$$

and $Q_{\lambda_i}(\lambda_i)$ is a K-valued discrete distribution. Applying Equation 1.49 leads to the following update equations for the variational parameters

$$\begin{aligned} \beta'_k &= \beta + \sum_{i=1}^{N} Q_{\lambda_i}(k)\langle\gamma_k\rangle && (1.88)\\ m'_k &= \frac{1}{\beta'_k}\left(\beta m + \langle\gamma_k\rangle \sum_{i=1}^{N} Q_{\lambda_i}(k) x_i\right) && (1.89)\\ a'_k &= a + \frac{1}{2}\sum_{i=1}^{N} Q_{\lambda_i}(k) && (1.90)\\ b'_k &= b + \frac{1}{2}\sum_{i=1}^{N} Q_{\lambda_i}(k)(x_i^2 - 2x_i\langle\mu_k\rangle + \langle\mu_k^2\rangle) && (1.91)\\ u'_k &= u_k + \sum_{i=1}^{N} Q_{\lambda_i}(k) && (1.92) \end{aligned}$$

and Q_{λ_i} is updated using

$$\begin{aligned} \log Q_{\lambda_i}(k) &= \langle\log\pi_k\rangle + \langle\log\mathcal{N}(x_i \,|\, \mu_k, \gamma_k^{-1})\rangle - \log Z_i \\ &= \langle\log\pi_k\rangle + \tfrac{1}{2}\left[\langle\log\gamma_k\rangle - \langle\gamma_k\rangle(x_i^2 - 2x_i\langle\mu_k\rangle + \langle\mu_k^2\rangle)\right] - \log Z'_i \qquad (1.93) \end{aligned}$$

where Z'_i is a normalisation factor set to ensure that $\sum_{k=1}^{K} Q_{\lambda_i(k)} = 1$ for all i. Variational inference can then be performed in this model by applying Equations 1.88–1.93 iteratively.

Mixture of Gaussians model applied to toy one-dimensional data

We now apply this model to some toy data. The data consists of 150 samples from a mixture of three Gaussian components with means $(0, 0, 6)$, precisions $(50, 1, 0.3)$ and mixture coefficients $(0.2, 0.4, 0.4)$. The model was set to have five mixture components ($K = 5$) which is more than required and so some may be 'switched off' during optimisation. The other parameters were set to give broad priors: $m = 0$, $\beta = 0.001$, $a = 0.001$, $b = 0.001$ and $u_k = 1$ for all k.

The model was trained variationally to find an optimal separable approximation to the true posterior. In the trained model, two of the five mixture components were not used[8], as no data points were assigned to them, showing that the correct number of components had been

[8]During variational optimisation of mixtures, it is possible for more mixture components to be retained in the converged solution than necessary (although that has not happened here). When this occurs, pruning unnecessary components results in an increase in $\mathcal{L}(Q)$.

inferred. The expected values of the component means under the approximate posterior were $(0.03, 0.10, 6.36)$, their expected precisions were $(48.8, 0.74, 0.26)$ and the expected mixture coefficients were $(0.21, 0.42, 0.37)$. Figure 1.11 shows the true source distribution along with distributions whose parameters have been sampled from the approximate separable distribution. This shows that the uncertainty in the source distribution, given the relatively small amount of data, has been captured by the trained model.

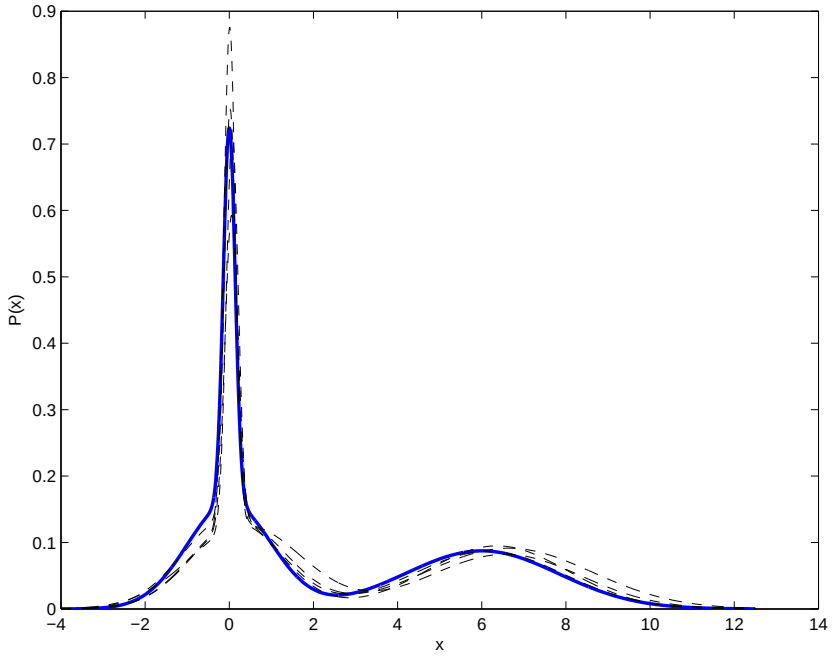


Figure 1.11: Diagram showing the true mixture of Gaussians distribution (thick line) used to generate the 120 data points, along with four distributions whose parameters have been sampled from the separable variational posterior trained on these data (dashed lines).

Mixture of Gaussians model applied to toy two-dimensional data

A straightforward extension of this model allows the use of d-dimensional data $\{\mathbf{x}_i\}$ by using a mixture of multivariate Gaussian distributions with diagonal inverse covariance matrices. In the modified model, each mean is now a d-dimensional vector $\boldsymbol{\mu}_k$ and the entries on the diagonal of each the inverse covariance matrix form a d-dimensional vector $\boldsymbol{\gamma}_k$.

The update equations for the factor distributions over the elements of these vectors are the same as for the corresponding variables in the univariate case. In fact, the only update equation that has changed is the one for Q_{λ_i}, which becomes

$$\log Q_{\lambda_i}(k) = \langle \log \pi_k \rangle + \tfrac{1}{2} \sum_{j=1}^{d} \left[\langle \log \gamma_{k,j} \rangle - \langle \gamma_{k,j} \rangle (x_{i,j}^2 - 2 x_{i,j} \langle \mu_{k,j} \rangle + \langle \mu_{k,j}^2 \rangle) \right] - \log Z_i. \quad (1.94)$$

This multi-dimensional model can be applied to the two-dimensional toy data set of Figure 1.12a. The model was set to have $K = 20$ mixture components with broad priors on

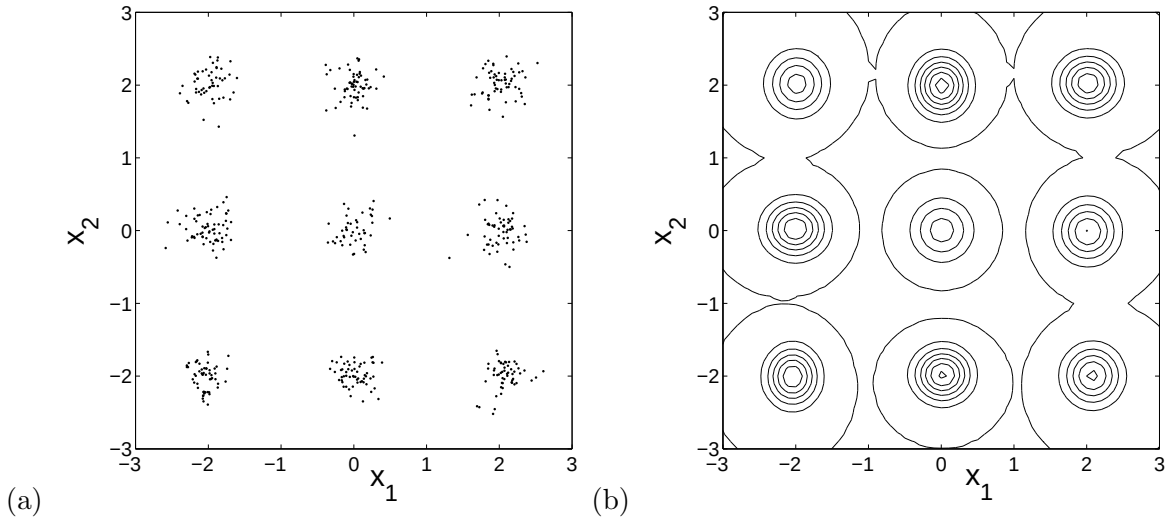


Figure 1.12: (a) 500 two-dimensional data points sampled from a mixture of Gaussians model with nine components of equal weight. (b) Probability contours of the mixture model whose parameters are set to their expected values under the approximate variational distribution.

all other parameters, as in the previous example. In the trained model, only nine components were retained, the rest being assigned zero weight. Figure 1.12b shows the contours of the probability distribution over $\mathbf{x}$ using the expected values of all parameters under the variational posterior.

The bound for this fitted model evaluates to -1019 nats $= -1470$ bits which gives an estimate of the log evidence for the data given the model. In Section 2.4, the log evidence for a number of other models will be estimated using the same data, allowing us to perform Bayesian model selection and so find a better model for this toy data set.

1.9 Overview

In this chapter, I have introduced the concepts of probabilistic models and Bayesian inference. I have shown that message-passing algorithms, like belief propagation, allow for inference to be performed using local computation in graphical models. Given the intractability of exact inference in many models, I have focussed on variational inference as a useful approximate inference method which is deterministic, gives an analytic approximation to the posterior, and allows for model selection using a bound on the evidence for the model. In Chapter 2, I will bring these elements together to provide a general purpose variational inference algorithm which is also based on message-passing in a graph. This algorithm will provide a framework for inference and model selection that can be used in a wide range of applications.

The framework is applied to two applications in Chapters 3 and 4, where I investigate the problems of modelling image subspaces and DNA microarrays respectively. Finally, in Chapter 5, the inference algorithm is extended to provide an improved approximation by using variational distributions that can retain posterior dependencies between variables.

Chapter 2

A Variational Inference Framework

Variational methods have been applied successfully to a wide range of probabilistic models including:

- Principal Component Analysis [Bishop 1999b],
- Independent Component Analysis [Miskin and MacKay 2000; Choudrey et al. 2000],
- Relevance Vector Machines [Bishop and Tipping 2000],
- Hidden Markov Models [MacKay 1997],
- Mixtures of Factor Analysers [Ghahramani and Beal 1999],
- Neural Networks [Barber and Bishop 1998].

In each case, the update equations for the model had to be worked out by hand, along with an expression for the lower bound. Even for very simple models these calculations can be lengthy (for example Equations 1.59-1.63), whilst for more complex models they rapidly become very unwieldy indeed (see Section 3.3.1). Furthermore, these equations then need to be implemented in model-specific code. Each of these steps is both time consuming and error-prone. These difficulties present a significant barrier to the adoption of variational methods and to their use with models of the complexity required for many real world applications.

In this chapter, I describe Variational Message Passing, a general purpose algorithm for variational inference which allows a wide variety of probabilistic models to be implemented and solved variationally without recourse to coding. I will also describe my software implementation of this algorithm, which is called VIBES (for Variational Inference for BayESian networks). This chapter extends the description of the variational inference framework and my work on VIBES originally presented in Bishop, Winn, and Spiegelhalter [2002].

2.1 Variational Message Passing

In Chapter 1, it was demonstrated how the exact inference method of variable elimination can be replaced by a graph algorithm which uses message passing: belief propagation. Belief propagation has the advantage of being a generic process which can be applied to any singly-connected discrete probabilistic model to perform efficient exact Bayesian inference within that model. Similarly, BUGS allows approximate inference using Gibbs Sampling in almost any probabilistic model, again by exploiting an efficient graph algorithm. Each of these algorithms relies on being able to decompose the required computation into calculations that are local to each node in the graph and which require only messages passed along the edges connected to that node.

In order to create a general purpose variational inference algorithm, it is necessary to decompose the maximisation of the lower bound (Equation 1.40) into a set of local computations for each node and a mechanism for passing messages between them. In this section, this procedure will be followed to create the Variational Message Passing algorithm.

2.1.1 A factorised Q distribution leads to local computations

Let us assume that we have a probabilistic model which is a Bayesian network with variables $\mathbf{X}$ divided into observed variables $\mathbf{D}$ and hidden variables $\mathbf{H}$. For the initial derivation, it will be assumed that the variational distribution is factorised over the hidden variables $\{H_j\}$, where each variable corresponds to a node in the Bayesian network:

$$Q(\mathbf{H}) = \prod_j Q_j(H_j). \tag{2.1}$$

This factorisation is an example of the factorised Q distribution of Section 1.8.4 and leads to the same solution for the optimised form of the jth factor

$$\log Q_j^*(H_j) = \langle \log P(\mathbf{H}, \mathbf{D}) \rangle_{\sim Q(H_j)} + \text{const.} \tag{2.2}$$

We now substitute in the form of the joint probability distribution of a Bayesian network, as given in Equation 1.8.

$$\log Q_j^*(H_j) = \left\langle \sum\nolimits_i \log P(X_i \,|\, \mathrm{pa}_i) \right\rangle_{\sim Q(H_j)} + \text{const.} \tag{2.3}$$

Any terms in the sum over i that do not depend on H_j will be constant under the expectation and can be subsumed into the constant term. This leaves only the conditional $P(H_j \,|\, \mathrm{pa}_j)$ together with the conditionals for all the children of H_j, as these have H_j in their parent set.

$$\log Q_j^*(H_j) = \left\langle \log P(H_j \,|\, \mathrm{pa}_j) + \sum\nolimits_{i \in \mathrm{ch}_j} \log P(X_i \,|\, \mathrm{pa}_i) \right\rangle_{\sim Q(H_j)} + \text{const.} \tag{2.4}$$

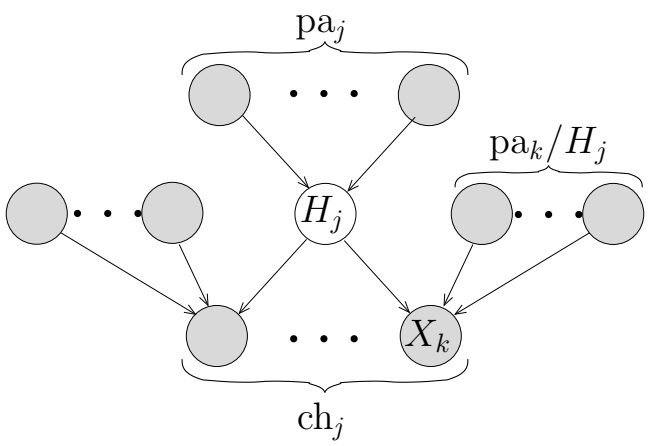


Figure 2.1: A key observation is that the variational update equation for a node H_j depends only on expectations over variables appearing in the *Markov blanket* of that node (shown shaded). This is defined as the set of parents, children and co-parents of a node.

Thus, the expectations that must be found to evaluate Q_j^* involve only those variables lying in the *Markov blanket* of H_j, consisting of its parents, children and co-parents[1] (as illustrated in Figure 2.1). The optimisation of Q_j can therefore be expressed in terms of a local computation at the node H_j. Moving the expectation inside the summation gives

$$\log Q_j^*(H_j) = \langle \log P(H_j \,|\, \mathrm{pa}_j) \rangle_{\sim Q(H_j)} + \sum\nolimits_{k \in \mathrm{ch}_j} \langle \log P(X_k \,|\, \mathrm{pa}_k) \rangle_{\sim Q(H_j)} + \text{const.} \qquad (2.5)$$

which shows that the optimisation can be written as the sum of a message from the parent nodes along with one message from each of the child nodes. We can thus decompose the overall optimisation into a set of local computations that depend only on messages from neighbouring (i.e. parent and child) nodes in the graph. The form that these messages take will depend on the functional form of the conditional distributions.

2.1.2 Message passing in conjugate-exponential models

It has been noted [Attias 2000; Ghahramani and Beal 2001] that important simplifications to the variational update equations occur when the distributions of variables, conditioned on their parameters, are drawn from the exponential family and are conjugate with respect to the prior distributions over these parameters. In this thesis, a somewhat different viewpoint is adopted, in that I make no distinction between latent variables and model parameters. In a Bayesian setting, these both correspond to unobserved stochastic variables and can be treated on an equal footing. This allows conjugacy to be considered not just between variables and their parameters, but hierarchically between all child-parent pairs in the graph. As described in Section 1.4.3, a model where all the conditional distributions are both conjugate and in the exponential family is called a *conjugate-exponential* model.

[1] The co-parents of a node X are all the nodes with at least one child which is also a child of X (excluding X itself).

I will now derive an algorithm for performing variational inference in any conjugate exponential model using message passing on a Bayesian network. In such a model, the conditional probability of one of the latent variables H_j given its parents will be in the exponential family

$$\log P(H_j \,|\, \mathrm{pa}_j) = \boldsymbol{\phi}_j(\mathrm{pa}_j)^{\mathrm{T}} \mathbf{u}_j(H_j) + f_j(H_j) + g_j(\mathrm{pa}_j). \tag{2.6}$$

The subscript j on each of the functions $\boldsymbol{\phi}_j, \mathbf{u}_j, f_j, g_j$ is required as these functions differ for different members of the exponential family and so need to be defined separately for each node.

Consider a node $X_k \in \mathrm{ch}_j$ which is a child of H_j. The conditional probability of X_k given its parents will also be in the exponential family and so can be written in the form

$$\log P(X_k \,|\, H_j, \mathrm{cp}_k^{(j)}) = \boldsymbol{\phi}_k(H_j, \mathrm{cp}_k^{(j)})^{\mathrm{T}} \mathbf{u}_k(X_k) + f_k(X_k) + g_k(H_j, \mathrm{cp}_k^{(j)}) \tag{2.7}$$

where $\mathrm{cp}_k^{(j)}$ are the co-parents of H_j with respect to X_k – in other words, the set of parents of X_k excluding H_j itself. The quantity $P(H_j \,|\, \mathrm{pa}_j)$ of Equation 2.6 can be thought of as a prior over H_j, and $P(X_k \,|\, H_j, \mathrm{cp}_k^{(j)})$ as a (contribution to) the likelihood of H_j. Consider an example where X_k is Gaussian distributed with mean H_j and precision β. It follows that the co-parent set $\mathrm{cp}_k^{(j)}$ contains only β, and the log conditional for X_k is

$$\log P(X_k \,|\, H_j, \beta) = \begin{bmatrix} \beta H_j \\ -\frac{\beta}{2} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} X_k \\ X_k^2 \end{bmatrix} + 0 + \tfrac{1}{2}(\log \beta - \beta H_j^2 - \log 2\pi). \tag{2.8}$$

Conjugacy requires that the conditionals of Equations 2.6 and 2.7 have the same functional form with respect to H_j, and so the latter can be rewritten in terms of $\mathbf{u}_j(H_j)$ by defining functions $\widetilde{\boldsymbol{\phi}}_{k,j}$ and λ as follows

$$\log P(X_k \,|\, H_j, \mathrm{cp}_k^{(j)}) = \widetilde{\boldsymbol{\phi}}_{k,j}(X_k, \mathrm{cp}_k^{(j)})^{\mathrm{T}} \mathbf{u}_j(H_j) + \lambda(X_k, \mathrm{cp}_k^{(j)}). \tag{2.9}$$

It may appear from this expression that the function $\widetilde{\boldsymbol{\phi}}_{k,j}$ depends on the form of $P(H_j \,|\, \mathrm{pa}_j)$ and so cannot be determined locally at X_k. This is not the case, because the conjugacy constraint dictates what $\mathbf{u}_j(H_j)$ must be for any parent H_j of X_k, meaning that $\widetilde{\boldsymbol{\phi}}_{k,j}$ can be found directly from the form of the conditional $P(X_k \,|\, \mathrm{pa}_k)$. In the example, we can find $\widetilde{\boldsymbol{\phi}}_{k,j}$ by rewriting the log conditional of Equation 2.8 in terms of H_j to give

$$\log P(X_k \,|\, H_j, \beta) = \begin{bmatrix} \beta X_k \\ -\frac{\beta}{2} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} H_j \\ H_j^2 \end{bmatrix} + \tfrac{1}{2}(\log \beta - \beta X_k^2 - \log 2\pi), \tag{2.10}$$

which lets us define $\widetilde{\boldsymbol{\phi}}_{k,j}$ and dictate what $\mathbf{u}_j(H_j)$ must be to enforce conjugacy,

$$\widetilde{\boldsymbol{\phi}}_{k,j}(X_k, \beta) \stackrel{\mathrm{def}}{=} \begin{bmatrix} \beta X_k \\ -\frac{\beta}{2} \end{bmatrix}, \qquad \mathbf{u}_j(H_j) = \begin{bmatrix} H_j \\ H_j^2 \end{bmatrix}. \tag{2.11}$$

In order to compute the variational update for H_j, we need to be able to find the expectations of these log conditionals with respect to all factors but $Q_j(H_j)$. The expectation of the natural statistic vector $\mathbf{u}$ under any exponential family distribution can be found from the natural parameter vector of that distribution[2]. The consequence of this is that we can find $\langle \mathbf{u}_i(X_i) \rangle_Q$ for any variable X_i. In the case where X_i is observed, the expectation is irrelevant and we can simply calculate $\mathbf{u}_i(X_i)$ directly. We can therefore calculate the expectation under the variational distribution Q of any function which is linear (or multi-linear[3]) in the set of functions $\{\mathbf{u}_i(X_i)\}$.

From Equations 2.7 and 2.9, it can be seen that $\log P(X_k \,|\, H_j, \mathrm{cp}_k^{(j)})$ is linear in $\mathbf{u}_k(X_k)$ and $\mathbf{u}_j(H_j)$ respectively. Conjugacy also requires that this log conditional will be linear in $\mathbf{u}_l(X_l)$ for each co-parent $X_l \in \mathrm{cp}_k^{(j)}$. Hence, $\log P(X_k \,|\, H_j, \mathrm{cp}_k^{(j)})$ must be a multi-linear function of those members of $\{\mathbf{u}_i(X_i)\}$ corresponding to X_k and its parents. This result is general in that *any* log conditional $\log P(X_i \,|\, \mathrm{pa}_i)$ must be a multi-linear function of $\mathbf{u}_i(X_i)$ and $\{\mathbf{u}_i(X_j)\}_{X_j \in \mathrm{pa}_i}$. For example, the log conditional of Equation 2.8 is multi-linear in $\mathbf{u}_k(X_k) = \begin{bmatrix} X_k \\ X_k^2 \end{bmatrix}$, $\mathbf{u}_j(H_j) = \begin{bmatrix} H_j \\ H_j^2 \end{bmatrix}$ and $\mathbf{u}_\beta(\beta) = \begin{bmatrix} \beta \\ \log \beta \end{bmatrix}$.

Returning to the variational update equation for a node H_j (Equation 2.5), it follows that all the expectations on the right hand side can be calculated in terms of the $\langle \mathbf{u} \rangle$ for each node in the Markov blanket of H_j. Substituting for these expectations, we get

$$\begin{aligned} \log Q_j^*(H_j) &= \langle \boldsymbol{\phi}_j(\mathrm{pa}_j)^{\mathrm{T}} \mathbf{u}_j(H_j) + f_j(H_j) + g_j(\mathrm{pa}_j) \rangle_{\sim Q(H_j)} \\ &\quad + \sum_{k \in \mathrm{ch}_j} \left\langle \widetilde{\boldsymbol{\phi}}_{k,j}(X_k, \mathrm{cp}_k^{(j)})^{\mathrm{T}} \mathbf{u}_j(H_j) + \lambda(X_k, \mathrm{cp}_k^{(j)}) \right\rangle_{\sim Q(H_j)} + \text{const.} \end{aligned}$$

which can be rearranged to give

$$\begin{aligned} \log Q_j^*(H_j) &= \left[\left\langle \boldsymbol{\phi}_j(\mathrm{pa}_j) \right\rangle_{\sim Q(H_j)} + \sum_{k \in \mathrm{ch}_j} \left\langle \widetilde{\boldsymbol{\phi}}_{k,j}(X_k, \mathrm{cp}_k^{(j)}) \right\rangle_{\sim Q(H_j)} \right]^{\mathrm{T}} \mathbf{u}_j(H_j) \\ &\quad + f_j(H_j) + \text{const.} \end{aligned} \tag{2.12}$$

It follows that Q_j^* is an exponential family distribution of the same form as Q_j but with an updated natural parameter vector $\boldsymbol{\phi}_j^*$ such that

$$\boldsymbol{\phi}_j^* = \left\langle \boldsymbol{\phi}_j(\mathrm{pa}_j) \right\rangle + \sum_{k \in \mathrm{ch}_j} \left\langle \widetilde{\boldsymbol{\phi}}_{k,j}(X_k, \mathrm{cp}_k^{(j)}) \right\rangle \tag{2.13}$$

where all expectations are with respect to Q. As explained above, the expectations of $\boldsymbol{\phi}_j$ and $\widetilde{\boldsymbol{\phi}}_{k,j}$ are both multi-linear functions of the expectations of the natural statistic vectors

[2] The natural statistic vector and natural parameter vector of an exponential family distribution were defined in Section 1.4.3 on page 7.

[3] f is a multi-linear function of variables $a, b \ldots$ if it varies linearly with respect to each variable. For example, $y = ab + 3b$ is multi-linear in a and b as we can write $y = (b)a + 3b$ and $y = (3 + a)b$.

corresponding to their dependent variables. It is therefore possible to define these functions explicitly as $\boldsymbol{\theta}_j$ and $\widetilde{\boldsymbol{\theta}}_{k,j}$:

$$\boldsymbol{\theta}_j\left(\{\langle \mathbf{u}_i\rangle\}_{i\in \mathrm{pa}_j}\right) \stackrel{\mathrm{def}}{=} \left\langle \boldsymbol{\phi}_j(\mathrm{pa}_j)\right\rangle \tag{2.14}$$

$$\widetilde{\boldsymbol{\theta}}_{k,j}\left(\langle \mathbf{u}_k\rangle, \{\langle \mathbf{u}_l\rangle\}_{l\in \mathrm{cp}_k^{(j)}}\right) \stackrel{\mathrm{def}}{=} \left\langle \widetilde{\boldsymbol{\phi}}_{k,j}(X_k, \mathrm{cp}_k^{(j)})\right\rangle. \tag{2.15}$$

In our example of Equation 2.11, we had $\widetilde{\boldsymbol{\phi}}_{k,j}(X_k,\beta) = \begin{bmatrix} \beta X_k \\ -\frac{\beta}{2}\end{bmatrix}$. We can now define $\widetilde{\boldsymbol{\theta}}_{k,j}$ as follows

$$\widetilde{\boldsymbol{\theta}}_{k,j}\left(\langle \mathbf{u}_k\rangle, \langle \mathbf{u}_\beta\rangle\right) \stackrel{\mathrm{def}}{=} \begin{bmatrix} \langle \mathbf{u}_\beta\rangle_0\langle \mathbf{u}_k\rangle_0 \\ -\frac{1}{2}\langle \mathbf{u}_\beta\rangle_0\end{bmatrix} \tag{2.16}$$

where $\langle \mathbf{u}_k\rangle_0$ and $\langle \mathbf{u}_\beta\rangle_0$ are the first elements of the vectors $\langle \mathbf{u}_k\rangle$ and $\langle \mathbf{u}_\beta\rangle$ respectively (and so are equal to $\langle X_k\rangle$ and $\langle\beta\rangle$). As required, we have reparameterised $\widetilde{\boldsymbol{\phi}}_{k,j}$ into a function $\widetilde{\boldsymbol{\theta}}_{k,j}$ which is a multi-linear function of natural statistic vectors.

We have now reached the point where we can define exactly what form the messages between nodes must take. The message from a parent node X_i to a child node X_j is just the expectation under Q of the natural statistic vector

$$m_{X_i\to X_j} = \langle \mathbf{u}_i\rangle. \tag{2.17}$$

The message from a child node X_k to a parent node X_j is

$$m_{X_k\to X_j} = \widetilde{\boldsymbol{\theta}}_{k,j}\left(\langle \mathbf{u}_k\rangle, \{m_{X_l\to X_k}\}_{l\in \mathrm{cp}_k^{(j)}}\right) \tag{2.18}$$

which relies on having received messages previously from all the co-parents. If any node X_i is observed then the messages are as defined above but with $\langle \mathbf{u}_i\rangle$ replaced by $\mathbf{u}_i$.

The approximate posterior distribution Q_j^* at a node X_j is defined by the natural parameter vector $\boldsymbol{\phi}_j^*$. This vector $\boldsymbol{\phi}_j^*$ is computed from all the messages received at a node using

$$\boldsymbol{\phi}_j^* = \boldsymbol{\theta}_j\left(\{m_{X_i\to X_j}\}_{i\in \mathrm{pa}_j}\right) + \sum_{k\in \mathrm{ch}_j} m_{X_k\to X_j}. \tag{2.19}$$

The new expectation of the natural statistic vector $\langle \mathbf{u}_j\rangle_{Q_j^*}$ can then be found, as it is a deterministic function of $\boldsymbol{\phi}_j^*$.

2.1.3 Example: the univariate Gaussian model

To illustrate how Variational Message Passing works, let us return to the univariate Gaussian model of Section 1.8.5 (page 23). The conditional distribution of each data point x_i is a univariate Gaussian. This distribution is in the exponential family and so its logarithm can

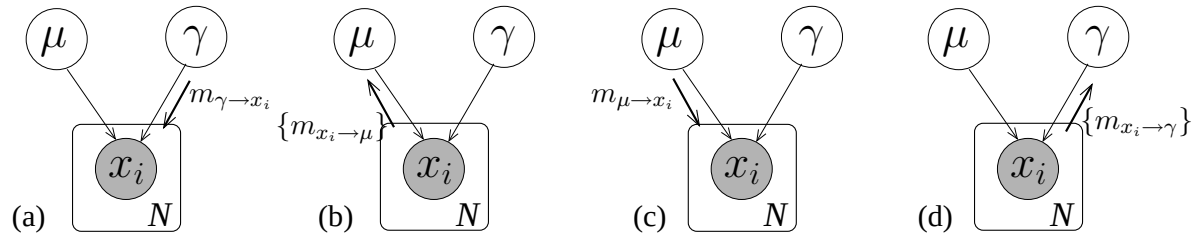


Figure 2.2: (a)-(d) Message passing procedure for variational inference in a univariate Gaussian model. The braces around two of the messages indicate that a set of N distinct messages are being sent.

be expressed in standard form as

$$\log P(x_i \,|\, \mu, \gamma^{-1}) = \begin{bmatrix} \gamma\mu \\ -\frac{\gamma}{2} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} x_i \\ x_i^2 \end{bmatrix} + \frac{1}{2}(\log\gamma - \gamma\mu^2 - \log 2\pi) \tag{2.20}$$

and so $\mathbf{u}_x(x_i) = [x_i, x_i^2]^{\mathrm{T}}$. This conditional can also be written so as to separate out the dependencies on μ and γ

$$\begin{aligned} \log P(x_i \,|\, \mu, \gamma^{-1}) &= \begin{bmatrix} \gamma x_i \\ -\frac{\gamma}{2} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mu \\ \mu^2 \end{bmatrix} + \frac{1}{2}(\log\gamma - \gamma x_i^2 - \log 2\pi) & (2.21) \\ &= \begin{bmatrix} -\frac{1}{2}(x_i - \mu)^2 \\ \frac{1}{2} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \gamma \\ \log\gamma \end{bmatrix} - \log 2\pi & (2.22) \end{aligned}$$

which shows that, for conjugacy, $\mathbf{u}_\mu(\mu)$ must be $[\mu, \mu^2]^{\mathrm{T}}$ and $\mathbf{u}_\gamma(\gamma)$ must be $[\gamma, \log\gamma]^{\mathrm{T}}$ or linear transforms of these[4]. Therefore, μ must have a Gaussian prior and γ a Gamma prior as they are the exponential family distributions with these natural statistic vectors. We introduce the parameters m, β, a and b, so that

$$\begin{aligned} \log P(\mu \,|\, m, \beta) &= \begin{bmatrix} \beta m \\ -\frac{\beta}{2} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mu \\ \mu^2 \end{bmatrix} + \frac{1}{2}(\log\beta - \beta m^2 - \log 2\pi) & (2.23) \\ \log P(\gamma \,|\, a, b) &= \begin{bmatrix} -b \\ a - 1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \gamma \\ \log\gamma \end{bmatrix} + a\log b - \log\Gamma(a). & (2.24) \end{aligned}$$

Message passing in the univariate Gaussian model

We can now apply message passing to infer the distributions over μ and γ variationally. The first message must be sent from a parent to a child and so we have the choice as to whether to start with a message from μ or γ. If we start with a message from γ then variational message passing will proceed as follows (illustrated in Figure 2.2a-d).

[4]To prevent the need for linear transformation of messages, a normalised form of natural statistic vectors will always be used, for example $[\mu, \mu^2]^{\mathrm{T}}$ rather than $[\mu^2 + 3\mu, -4\mu]^{\mathrm{T}}$.

(a) The messages $m_{\gamma \to x_i}$ from γ to each of the observed nodes x_i are the same, and are just equal to $\langle \mathbf{u}_\gamma(\gamma) \rangle = [\langle \gamma \rangle, \langle \log \gamma \rangle]^{\mathrm{T}}$. These expectation are with respect to the initial setting of Q_γ.

(b) Each x_i node has now received messages from all parents but μ, and so can send a message to μ which is the expectation of the natural parameter vector in Equation 2.21,

$$m_{x_i \to \mu} = \begin{bmatrix} \langle \gamma \rangle x_i \\ -\frac{\langle \gamma \rangle}{2} \end{bmatrix}. \tag{2.25}$$

(c) Node μ has now received its full complement of incoming messages and can update its natural parameter vector,

$$\boldsymbol{\phi}^*_\mu = \begin{bmatrix} \beta m \\ -\frac{\beta}{2} \end{bmatrix} + \sum_{i=1}^{N} m_{x_i \to \mu}. \tag{2.26}$$

The new expectation $\langle \mathbf{u}_\mu(\mu) \rangle$ can then be computed under the updated distribution Q^*_μ and sent to each x_i as the message $m_{\mu \to x_i} = [\langle \mu \rangle, \langle \mu^2 \rangle]^{\mathrm{T}}$.

(d) Finally, each x_i node sends a message back to γ which is

$$m_{x_i \to \gamma} = \begin{bmatrix} -\frac{1}{2}(x_i^2 - 2x_i \langle \mu \rangle + \langle \mu^2 \rangle) \\ \frac{1}{2} \end{bmatrix} \tag{2.27}$$

and γ can update its variational posterior

$$\boldsymbol{\phi}^*_\gamma = \begin{bmatrix} -b \\ a - 1 \end{bmatrix} + \sum_{i=1}^{N} m_{x_i \to \gamma}. \tag{2.28}$$

As the expectation of $\mathbf{u}_\gamma(\gamma)$ has changed, we can now go back to step (a) and send an updated message to each x_i node and so on. Hence, in variational message passing, the message passing procedure is repeated again and again until convergence (unlike in belief propagation where the exact posterior is available after a message passing is performed once).

Each round of variational message passing is equivalent to one iteration of the update equations in standard variational inference. To see this, compare the variational posteriors computed by this algorithm to the update equations 1.59–1.62 derived before.

2.1.4 Calculation of the lower bound $\mathcal{L}(\mathbf{Q})$

The lower bound is essential for diagnosing convergence, setting termination criteria and performing model selection. In previous variational approaches, an expression for the lower bound (or an analogous quantity) has been determined by hand and coded separately to the variational update equations. It has been suggested that this calculation can act as a 'double check' on the correctness of the implementation of update equations (as the bound should

only ever increase or stay the same). Whilst this is true to some extent, it is possible for implementation errors to occur in the bound calculation as well and, since the complexity of this calculation is significant (typically equivalent to that of all the update equations together), considerable effort must be expended to check its correctness. In addition, if the update equations are changed due to altering the model, the bound evaluation has to be changed as well. Keeping the two synchronised can be time consuming.

This problem is addressed in this framework by providing a way to calculate the bound automatically. To recap, the lower bound on the log evidence is defined to be

$$\mathcal{L}(Q) = \langle \log P(\mathbf{H}, \mathbf{D}) \rangle - \langle Q(\mathbf{H}) \rangle \,, \tag{2.29}$$

where the expectations are with respect to Q. In a Bayesian network, with a factorised Q distribution, the bound becomes

$$\begin{aligned} \mathcal{L}(Q) &= \sum_i \langle \log P(X_i \,|\, \mathrm{pa}_i) \rangle - \sum_j \langle \log Q_j(H_j) \rangle & (2.30) \\ &\stackrel{\text{def}}{=} \sum_i \mathcal{L}_i & (2.31) \end{aligned}$$

where it has been decomposed into contributions from the individual nodes $\{\mathcal{L}_i\}$. For a particular latent variable node H_j, the contribution is

$$\mathcal{L}_j = \langle \log P(H_j \,|\, \mathrm{pa}_j) \rangle - \langle \log Q_j(H_j) \rangle \,. \tag{2.32}$$

Given that the model is conjugate-exponential, we can substitute in the standard form for the exponential family

$$\begin{aligned} \mathcal{L}_j &= \langle \boldsymbol{\phi}_j(\mathrm{pa}_j)^{\mathrm{T}} \rangle \langle \mathbf{u}_j(H_j) \rangle + \langle f_j(H_j) \rangle + \langle g_j(\mathrm{pa}_j) \rangle \\ &\quad - \left[\boldsymbol{\phi}_j^{*\mathrm{T}} \langle \mathbf{u}_j(H_j) \rangle + \langle f_j(H_j) \rangle + g'_j(\boldsymbol{\phi}_j^*) \right] , \end{aligned} \tag{2.33}$$

where the function g'_j is a reparameterisation of g_j so as to make it a function of the natural parameter vector rather than the parent variables. This expression simplifies to

$$\mathcal{L}_j = \left(\langle \boldsymbol{\phi}_j(\mathrm{pa}_j) \rangle - \boldsymbol{\phi}_j^* \right)^{\mathrm{T}} \langle \mathbf{u}_j(H_j) \rangle + \langle g_j(\mathrm{pa}_j) \rangle - g'_j(\boldsymbol{\phi}_j^*). \tag{2.34}$$

Three of these terms are already calculated during the variational message passing algorithm: $\langle \boldsymbol{\phi}_j(\mathrm{pa}_j) \rangle$ and $\boldsymbol{\phi}_j^*$ when finding the posterior distribution over H_j (see Equation 2.19), and $\langle \mathbf{u}_j(H_j) \rangle$ when calculating outgoing messages from H_j. Thus, considerable saving in computation are made compared to when the bound is calculated separately.

Each observed variable D_k also makes a contribution to the bound

$$\mathcal{L}_k = \langle \log P(D_k \,|\, \mathrm{pa}_k) \rangle \tag{2.35}$$

$$= \langle \boldsymbol{\phi}_j(\mathrm{pa}_j) \rangle^{\mathrm{T}} \mathbf{u}_k(D_k) + f_k(D_k) + g'_k \left(\langle \boldsymbol{\phi}_j(\mathrm{pa}_j) \rangle \right). \tag{2.36}$$

Again, computation can be saved by computing $\mathbf{u}_k(D_k)$ during the initialisation of the message passing algorithm.

Example: Calculation of the lower bound for the univariate Gaussian model

In the univariate Gaussian model, the bound contribution from each observed node x_i is

$$\mathcal{L}_{x_i} = \begin{bmatrix} \langle\gamma\rangle\langle\mu\rangle \\ -\frac{\langle\gamma\rangle}{2} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} x_i \\ x_i^2 \end{bmatrix} + \frac{1}{2} \left(\langle \log\gamma \rangle - \langle\gamma\rangle\langle\mu^2\rangle - \log 2\pi \right) \tag{2.37}$$

and the contributions from the parameter nodes μ and γ are

$$\mathcal{L}_\mu = \begin{bmatrix} \beta' m' - \beta m \\ -\frac{\beta'}{2} + \frac{\beta}{2} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \langle\mu\rangle \\ \langle\mu^2\rangle \end{bmatrix} + \frac{1}{2} \left(\log\beta - \beta m^2 - \log\beta' + \beta' m'^2 \right) \tag{2.38}$$

$$\mathcal{L}_\gamma = \begin{bmatrix} -b' + b \\ a - a' \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \langle\gamma\rangle \\ \langle\log\gamma\rangle \end{bmatrix} + a\log b - \log\Gamma(a) - a'\log b' + \log\Gamma(a'). \tag{2.39}$$

The sum of the contributions from all the nodes may be seen to be equal to expression for the bound given in Equation 1.63, if we note that $\beta'(\langle\mu^2\rangle - 2m'\langle\mu\rangle + m'^2) = 1$.

2.2 Allowable Models

The Variational Message Passing algorithm can be applied to a wide class of models, which will be characterised in this section.

2.2.1 Conjugacy constraints

The main constraint on the model is that each parent–child edge must satisfy the constraint of conjugacy. Conjugacy allows a Gaussian variable to have a Gaussian parent for its mean and we can extend this hierarchy to any number of levels. Each Gaussian node has a Gamma parent as the distribution over its precision. Furthermore, each Gamma distributed variable can have a Gamma distributed scale parameter b, and again this hierarchy can be extended to multiple levels.

A discrete variable can have multiple discrete parents with a Dirichlet prior over the entries in the conditional probability table. This allows for an arbitrary graph of discrete variables. A variable with an Exponential or Poisson distribution can have a Gamma prior over its scale

Distribution	1st parent	Conjugate dist.	2nd parent	Conjugate dist.
Gaussian	mean μ	Gaussian	precision γ	Gamma
Gamma	shape a	None	scale b	Gamma
Discrete	probabilities $\mathbf{p}$	Dirichlet	parents $\{x_i\}$	Discrete
Dirichlet	pseudo-counts $\mathbf{a}$	None		
Exponential	scale a	Gamma		
Poisson	mean λ	Gamma		

Table 2.1: The valid distributions for each parameter of a number of exponential family distributions if the model is to satisfy conjugacy constraints. Conjugacy also holds if the distributions are replaced by their multivariate counterparts e.g. the distribution which is conjugate to the precision matrix of a multivariate Gaussian is a Wishart distribution. Where "None" is specified, no standard distribution satisfies conjugacy.

or mean respectively, although, as these distributions do not lead to hierarchies, they may be of limited interest.

These constraints are listed in Table 2.1. This table is encoded in the variational message passing algorithm and used, during initialisation, to check the conjugacy of the supplied model.

Truncated distributions

The conjugacy constraint does not put any restrictions on the $f_X(X)$ term in the exponential family distribution. If we choose f_X to be a step function

$$f_X(X) = \begin{cases} 0 & : \quad X \geq 0 \\ -\infty & : \quad X < 0 \end{cases} \tag{2.40}$$

then we end up with a *rectified* distribution, so that $P(X \,|\, \boldsymbol{\theta}) = 0$ for $X < 0$. The choice of such a truncated distribution has no effect on the messages that are passed during inference, but will affect how the moments of the distribution are calculated from the updated parameters (which will lead to different outgoing messages). For example, the moments of a rectified Gaussian distribution are expressed in terms of the 'erfc' function (see Appendix A.2). Similarly, we can consider doubly truncated distributions which are non-zero only over some finite interval, as long as the calculation of the moments remains tractable.

2.2.2 Deterministic functions

We can considerably enlarge the class of tractable models if variables are allowed to be defined as *deterministic* functions of the states of their parent variables. This is achieved by adding deterministic nodes into the graph, whose state is defined to be a function of the parent variables. The use of such deterministic nodes greatly extends the applicability of this approach; they have also been used to similar effect in BUGS.

Consider a deterministic node X which has stochastic parents $\{Y_1, \ldots, Y_M\}$ and which has

a stochastic child node Z. The state of X is given by a deterministic function f of the state of its parents, so that $X = f(Y_1, \ldots, Y_M)$. If X were stochastic, the conjugacy constraint with Z would require that $P(X \,|\, Y_1, \ldots, Y_M)$ must have the same functional form, with respect to X, as $P(Z \,|\, X)$. This in turn would dictate the form of the natural statistic vector $\mathbf{u}_X$ of X, whose expectation $\langle \mathbf{u}_X(X) \rangle_Q$ would be the message from X to Z.

Returning to the case where X is deterministic, it is still necessary to provide a message to Z of the form $\langle \mathbf{u}_X(X) \rangle_Q$ where the function $\mathbf{u}_X$ is dictated by the conjugacy constraint. This message can be evaluated only if it can be expressed as a function of the messages from the parent variables, which are the expectations of their natural statistics functions $\{\langle \mathbf{u}_{Y_i}(Y_i) \rangle_Q\}$. In other words, there must exist a vector function $\boldsymbol{\psi}_X$ such that

$$\big\langle \mathbf{u}_X\big(f(Y_1, \ldots, Y_M)\big) \big\rangle_Q = \boldsymbol{\psi}_X\big(\langle \mathbf{u}_{Y_1}(Y_1) \rangle_Q, \ldots, \langle \mathbf{u}_{Y_M}(Y_M) \rangle_Q\big). \tag{2.41}$$

As was explained in Section 2.1.2, this constrains $\mathbf{u}_X\big(f(Y_1, \ldots, Y_M)\big)$ to be a multi-linear function of the set of functions $\{\mathbf{u}_{Y_i}(Y_i)\}$.

A deterministic node can be viewed as a delta distribution, so that $P(X \,|\, Y_1, \ldots, Y_M) = \delta\big(X \,|\, f(Y_1, \ldots, Y_M)\big)$. If X is discrete, this is the distribution that assigns probability one to the state $X = f(Y_1, \ldots, Y_M)$ and zero to all other states. If X is continuous, this is the distribution with the property that $\int g(X)\, \delta\big(X \,|\, f(Y_1, \ldots, Y_M)\big)\, dX = g\big(f(Y_1, \ldots, Y_M)\big)$. The contribution to the lower bound from a deterministic node is zero.

Example: using a deterministic function as the mean of a Gaussian

Consider a model where a deterministic node X is to be used as the mean of a child Gaussian distribution $\mathcal{N}(Z \,|\, X, \beta^{-1})$ and where X equals a function f of Gaussian-distributed variables $Y_1, \ldots, Y_M$. The natural statistic vectors of X (as dictated by conjugacy with Z) and those of $Y_1, \ldots, Y_M$ are

$$\mathbf{u}_X(X) = \begin{bmatrix} X \\ X^2 \end{bmatrix} \tag{2.42}$$

$$\mathbf{u}_{Y_i}(Y_i) = \begin{bmatrix} Y_i \\ Y_i^2 \end{bmatrix} \text{ for } i = 1 \ldots M \tag{2.43}$$

The constraint on f is that $\mathbf{u}_X(f)$ must be multi-linear in $\{\mathbf{u}_{Y_i}(Y_i)\}$. Hence, f can be any multi-linear function of $Y_1, \ldots, Y_M$. In other words, the mean of a Gaussian can be the sum of products of other Gaussian-distributed variables.

Example: using a deterministic function as the precision of a Gaussian

As another example, consider a model where X is to be used as the precision of a child Gaussian distribution $\mathcal{N}(Z \,|\, \mu, X^{-1})$ and where X is a function f of Gamma-distributed

variables $Y_1, \ldots, Y_M$. The natural statistic vectors of X and $Y_1, \ldots, Y_M$ are now

$$\mathbf{u}_X(X) = \begin{bmatrix} X \\ \log X \end{bmatrix} \tag{2.44}$$

$$\mathbf{u}_{Y_i}(Y_i) = \begin{bmatrix} Y_i \\ \log Y_i \end{bmatrix} \text{ for } i = 1 \ldots M. \tag{2.45}$$

The multi-linearity constraint now restricts f to be proportional to a product of the variables $Y_1, \ldots, Y_M$ as the logarithm of a product can be found in terms of the logarithms of terms in that product. Hence $f = c \prod_i Y_i$ where c is a constant. Note that a function containing a summation, such as $f = \sum_i Y_i$, would not be valid as the expectation of the logarithm of the sum cannot be found in terms of the individual expectations of Y_i and $\log Y_i$.

Validating chains of deterministic functions

The validity of a deterministic function for a node X is dependent on the form of the stochastic nodes it is connected to, as these dictate the functions $\mathbf{u}_X$ and $\{\mathbf{u}_{Y_i}(Y_i)\}$. For example, if the function was a summation $f = \sum_i Y_i$, it would be valid as connected in the first of the above examples but not as connected in the second. In addition, it is possible for deterministic functions to be chained together to form more complicated expressions. For example, the expression $X = Y_1 + Y_2 Y_3$ can be achieved by having a deterministic product node A with parents Y_2 and Y_3 and a deterministic sum node X with parents Y_1 and A. In this case, the form of the function $\mathbf{u}_A$ cannot be determined directly. However, if a function node has some of the neighbouring $\mathbf{u}$ functions available it may be able to determine what form the remaining functions must take in order for that node to be valid. In this way, conjugacy constraints may be propagated through deterministic nodes, so as to check the validity of entire subgraphs of deterministic nodes.

Alternatively, this complexity may be avoided by dictating the function $\mathbf{u}_{X_i}$ for each deterministic node X_i and using the existing mechanism for checking conjugacy.

Deterministic node messages

To examine message passing for deterministic nodes, we must consider the general case where the deterministic node X has multiple children $\{Z_j\}$. The message from the node X to any child Z_j is simply

$$m_{X \to Z_j} = \langle \mathbf{u}_X(f(Y_1, \ldots, Y_M)) \rangle_Q \tag{2.46}$$

$$= \boldsymbol{\psi}_X(m_{Y_1 \to X}, \ldots, m_{Y_M \to X}). \tag{2.47}$$

For a particular parent Y_k, the function $\mathbf{u}_X(f(Y_1, \ldots, Y_M))$ is linear with respect to $\mathbf{u}_{Y_k}(Y_k)$ and so it can be written as

$$\mathbf{u}_X(f(Y_1, \ldots, Y_M)) = \boldsymbol{\Psi}_{X,Y_k}\big(\{\mathbf{u}_{Y_i}(Y_i)\}_{i\neq k}\big)\,.\,\mathbf{u}_{Y_k}(Y_k) + \boldsymbol{\lambda}\big(\{\mathbf{u}_{Y_i}(Y_i)\}_{i\neq k}\big) \tag{2.48}$$

where $\boldsymbol{\Psi}_{X,Y_k}$ is a matrix function of the natural statistics vectors of the co-parents of Y_k. The message from a deterministic node to a parent Y_k is then

$$m_{X\to Y_k} = \left[\sum_j m_{Z_j\to X}\right] \boldsymbol{\Psi}_{X,Y_k}\big(\{m_{Y_i\to X}\}_{i\neq k}\big) \tag{2.49}$$

which relies on having received messages from all the child nodes and from all the co-parents. The sum of child messages can be computed and stored locally at the node and used to evaluate all child-to-parent messages. In this sense, it can be viewed as the natural parameter vector of a distribution which acts as a kind of pseudo-posterior over the value of X.

2.2.3 Mixture models

So far, only distributions from the exponential family have been considered within the framework. Often it is desirable to use richer distributions that better capture the true distributions inherent in a data set. Mixture models, such as the Gaussian mixture model of Section 1.8.7, provide one common way of creating richer probability densities.

A mixture distribution is not in the exponential family and cannot therefore be used directly as a conditional distribution within a conjugate-exponential model. As in the Gaussian mixture model example, we introduce an additional variable λ which indicates from which component distribution the data point was drawn and write the distribution as

$$P(X \mid \lambda, \{\boldsymbol{\theta}_k\}) = \prod_{k=1}^{K} P_k(X \mid \boldsymbol{\theta}_k)^{\delta(\lambda,k)}. \tag{2.50}$$

Conditioned on this new variable, the distribution is now in the exponential family provided that all of the component distributions are also in the exponential family. In this case, the log conditional probability of X given all the parents (including λ) can be written as

$$\log P(X \mid \lambda, \{\boldsymbol{\theta}_k\}) = \sum_k \delta(\lambda, k)\left[\boldsymbol{\phi}_k(\boldsymbol{\theta}_k)^{\mathrm{T}}\mathbf{u}_k(X) + f_k(X) + g_k(\boldsymbol{\theta}_k)\right]. \tag{2.51}$$

If X has a child Z, then conjugacy will require that all the component distributions have the same natural statistic vector, which we can then call $\mathbf{u}_X$ so: $\mathbf{u}_1(X) = \mathbf{u}_2(X) = \cdots = \mathbf{u}_K(X) \stackrel{\text{def}}{=} \mathbf{u}_X(X)$. In addition, we may choose to specify, as part of the model, that all these distributions have exactly the same form (that is, $f_1 = f_2 = \cdots = f_K \stackrel{\text{def}}{=} f_X$) although this is not required by conjugacy. In this case, where all the distributions are the same, the log

conditional becomes

$$\begin{aligned}\log P(X \mid \lambda, \{\boldsymbol{\theta}_k\}) &= \left[\sum_k \delta(\lambda, k)\boldsymbol{\phi}_k(\boldsymbol{\theta}_k)\right]^{\mathrm{T}} \mathbf{u}_X(X) + f_X(X) \\ &\quad + \sum_k \delta(\lambda, k) g_k(\boldsymbol{\theta}_k) \qquad (2.52)\\ &= \widetilde{\boldsymbol{\phi}}_X(\lambda, \{\boldsymbol{\theta}_k\})^T \mathbf{u}_X(X) + f_X(X) + g'_X(\widetilde{\boldsymbol{\phi}}_X(\lambda, \{\boldsymbol{\theta}_k\})) \qquad (2.53)\end{aligned}$$

where the function g'_X is a reparameterisation of g_X to make it a function of the natural parameter vector (as in Section 2.1.4). The conditional is therefore of the same exponential family form as each of the components.

We can now apply variational message passing. The message from the node X to any child is $\langle \mathbf{u}_X(X) \rangle$ as calculated from the mixture parameter vector $\widetilde{\boldsymbol{\phi}}_X(\lambda, \{\boldsymbol{\theta}_k\})$. The message from X to a parent $\boldsymbol{\theta}_k$ is the message that would be sent by the corresponding component if it were not in a mixture, scaled by $Q(\lambda = k)$. Finally, the message from X to λ is the vector whose kth element is $\langle \log P_k(X \mid \theta_k) \rangle$.

Use of a deterministic pick function

An alternate approach to mixture models is to use a deterministic 'pick' function to select between a number of latent variables. We introduce K latent variables $Y_1 \ldots Y_K$ each with conditional distribution $P_k(Y_k \mid \boldsymbol{\theta}_k)$. The variable X is then set to be the deterministic function $X = \prod_{k=1}^K Y_k^{\delta(\lambda,k)}$. This deterministic function will be valid for all component distributions as it always satisfies Equation 2.41, because:

$$\langle \mathbf{u}_X(X) \rangle = \sum_k \langle \delta(\lambda, k) \rangle \langle \mathbf{u}_{Y_k}(Y_k) \rangle \qquad (2.54)$$

and we can note that $\langle \delta(\lambda, k) \rangle = Q(\lambda = k)$ which is the kth element of the vector $\langle \mathbf{u}_\lambda(\lambda) \rangle$ for a discrete distribution. This approach specifies a different generative model to the above approach and the approximating distribution takes a different form. The variational distribution is now

$$Q(\{Y_k\}, \{\boldsymbol{\theta}_k\}, \lambda) = Q_\lambda(\lambda) \prod_k Q_{Y_k}(Y_k) \prod_k Q_{\boldsymbol{\theta}_k}(\boldsymbol{\theta}_k) \qquad (2.55)$$

which, in the original approach, is equivalent to

$$\begin{aligned}Q(X, \{\boldsymbol{\theta}_k\}, \lambda) &= Q_\lambda(\lambda) \prod_k Q_{X \mid \lambda}(X \mid \lambda = k) \prod_k Q_{\boldsymbol{\theta}_k}(\boldsymbol{\theta}_k) \qquad (2.56)\\ &= Q_{\lambda,X}(\lambda, X) \prod_k Q_{\boldsymbol{\theta}_k}(\boldsymbol{\theta}_k). \qquad (2.57)\end{aligned}$$

The variational approximation now retains the joint distribution between X and λ rather than being factorised, as in the previous case. In practice, the posterior distribution $Q(\lambda)$

will often converge to give almost all probability mass to one value of λ and so retaining the joint distribution provides practically no advantage. Furthermore, the latter case requires a factor of k times as much storage to hold the moments of X – an unwelcome expense in large models. However, if it is expected that posterior probability mass will be spread amongst more than one value of λ and that the state of λ will have significant impact on the inferred state of X, then retaining the joint distribution will provide an advantage.

2.2.4 Multivariate distributions

Until now, only scalar variables have been considered. It is also possible to handle vector variables in this framework (or to handle scalar variables which have been grouped into a vector). In each case, a multivariate conditional distribution is defined in P and the corresponding factor in Q will also be multivariate, rather than factorised with respect to the elements of the vector. To understand how multivariate distributions are handled, consider the d-dimensional Gaussian distribution with mean $\boldsymbol{\mu}$ and precision matrix[5] $\boldsymbol{\Lambda}$:

$$P(\mathbf{x} \,|\, \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) = \sqrt{\frac{|\boldsymbol{\Lambda}|}{(2\pi)^d}} \exp\left(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Lambda}\,(\mathbf{x}-\boldsymbol{\mu})\right). \tag{2.58}$$

This distribution can be written in exponential family form

$$\log \mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) = \begin{bmatrix} \boldsymbol{\Lambda}\boldsymbol{\mu} \\ -\frac{1}{2}\mathrm{col}(\boldsymbol{\Lambda}) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{x} \\ \mathrm{col}(\mathbf{x}\mathbf{x}^{\mathrm{T}}) \end{bmatrix} + \tfrac{1}{2}(\log|\boldsymbol{\Lambda}| - \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Lambda}\boldsymbol{\mu} - d\log 2\pi) \tag{2.59}$$

where col() is a function that re-arranges the elements of a matrix into a column vector in some consistent fashion, such as by concatenating the columns of the matrix. The natural statistic function for a multivariate distribution therefore depends on both the type of the distribution and its dimensionality d. As a result, the conjugacy constraint between a parent node and a child node will also constrain the dimensionality of the corresponding vector-valued variables to be the same.

Multivariate conditional distributions can therefore be handled in the Variational Message Passing algorithm like any other exponential family distribution, which extends the class of allowed distributions to include multivariate Gaussian and Wishart distributions.

A group of scalar variables can act as a single parent of a vector-valued node. This is achieved using a deterministic *concatenation* function which simply concatenates a number of scalar values into a vector. In order for this to be a valid function, the scalar distributions must still be conjugate to the multivariate distribution. For example, a set of d univariate Gaussian distributed variables can be concatenated to act as the mean of a d-dimensional multivariate Gaussian distribution.

[5] The precision matrix of a multivariate Gaussian is the inverse of its covariance matrix.

2.2.5 Summary of allowable models

In summary, the Variational Message Passing algorithm can handle probabilistic models with the following very general architecture: arbitrary directed acyclic subgraphs of multinomial discrete variables (each having Dirichlet priors) together with arbitrary subgraphs of univariate and multivariate linear Gaussian nodes (having Gamma and Wishart priors), with arbitrary mixture nodes providing connections from the discrete to the continuous subgraphs. In addition, deterministic nodes can be included to allow parameters of child distributions to be deterministic functions of parent variables. Finally, any of the continuous distributions can be singly or doubly truncated to restrict the range of allowable values, provided that the appropriate moments under the truncated distribution can be calculated.

This architecture includes as special cases models such as Hidden Markov Models, Kalman filters, factor analysers, principal component analysers and independent component analysers, as well as mixtures and hierarchical mixtures of these.

2.3 VIBES: A Software Implementation

The framework described above has been implemented in a software package called VIBES (Variational Inference in BayESian networks). Inspired by WinBUGS (a graphical user interface for BUGS by Lunn et al. [2000]), VIBES allows for models to be specified graphically, simply by constructing the Bayesian network for the model. This involves drawing the graph for the network (using operations similar to those in a drawing package) and then assigning properties to each node such as its name, the functional form of the conditional distribution, its dimensionality and its parents. As an example, Figure 2.3 shows the Bayesian network for the univariate Gaussian model along with a screenshot of the same model in VIBES. Models can also be specified in a text file, which contains XML according to a pre-defined model definition schema. VIBES is written in Java and so can be used on Windows, Linux or any operating system with a Java 1.3 virtual machine.

As in WinBUGS, the convention of making deterministic nodes explicit in the graphical representation has been adopted, as this greatly simplifies the specification and interpretation of the model. VIBES also uses the plate notation of a box surrounding one or more nodes to denote that those nodes are replicated some number of times, specified by the parameter in the bottom right hand corner of the box.

Once the model is specified, data can be attached from a separate data file which contains observed values for some of the nodes, along with sizes for some or all of the plates. The model can then be *initialised* which involves: checking that the model is valid by ensuring that conjugacy and dimensionality constraints are satisfied and that all parameters are specified; checking that the observed data is of the correct dimensionality; allocating memory for all moments and messages, and initialisation of the individual distributions Q_i.

Following a successful initialisation, inference can begin immediately. As inference proceeds, the current state of the distribution Q_i for any node can be inspected using a range

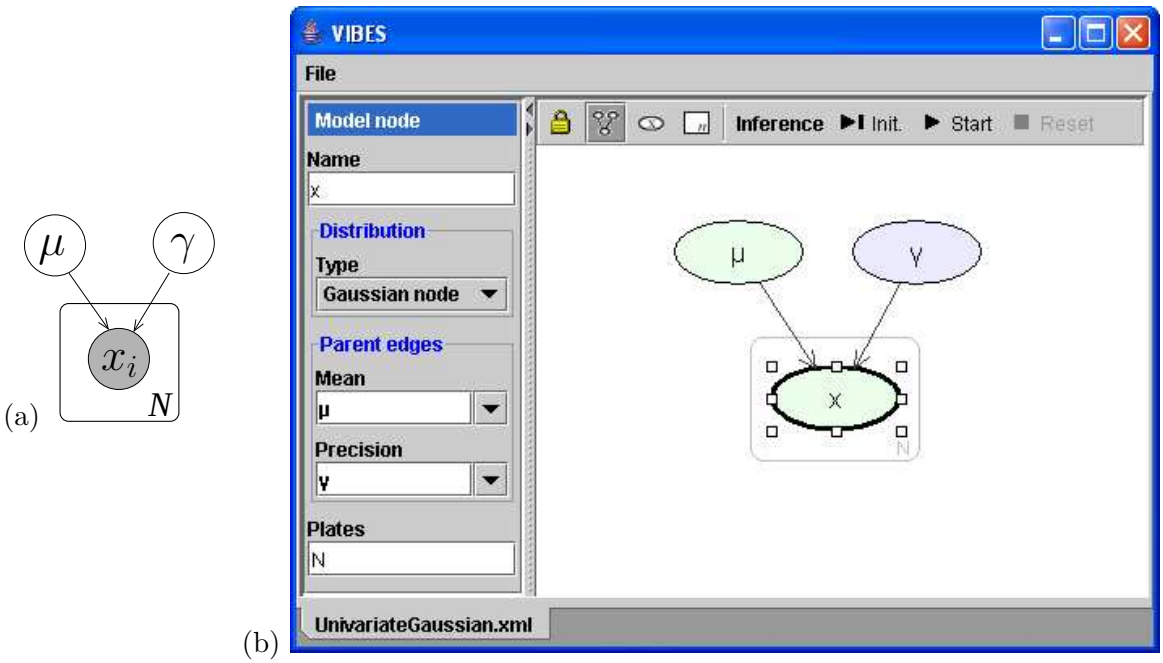


Figure 2.3: (a) Bayesian network for the univariate Gaussian model. (b) Screenshot of VIBES showing how the same model appears as it is being edited. The node x is selected and the panel to the left shows that it has a Gaussian conditional distribution with mean μ and precision γ. The plate surrounding x shows that it is duplicated N times and the heavy border indicates that it is observed (according to the currently attached data file).

of diagnostics including tables of values and Hinton diagrams[6]. If desired, the lower bound $\mathcal{L}(Q)$ can be monitored (at the expense of slightly increased computation), in which case the optimisation can be set to automatically terminate when the change in the bound during one iteration drops below a small value. Alternatively, the optimisation can be stopped after a fixed number of iterations.

The software architecture of VIBES

The Java implementation of VIBES has a modular architecture to provide easy of extensibility and re-use. The software consists of a set of reusable libraries and a fairly small number of VIBES-specific classes whose main purpose is to implement the Variational Message Passing algorithm. The libraries created for VIBES were:

- A graph library: for creating and manipulating graphs and extending them with potentials to form graphical models.
- A dataset library: for handling multi-dimensional data sets of either double-precision floating point numbers or generic objects and iterating over such data sets. The library also allows for saving and loading numerical data sets as Matlab files.

[6]For an example of a Hinton diagram, see Figure 2.9.

- An algorithm library: classes for generic iterative and graph algorithms along with implementations of sampling methods, gradient descent and other algorithms.
- A user interface library: a view/model based framework for automatically creating user interfaces corresponding to particular objects. Within this framework, views were created for graphs, data sets and arbitrary Java objects.
- An XML marshalling library: for saving an arbitrary Java object graph to an XML file and then recovering it.

The modular implementation makes it straightforward to add new inference algorithms, data file types, views or other software components. It also means that the libraries are readily re-usable for other projects. The author intends to make the source code for these libraries (and VIBES itself) available under an open source licence, which will allow free re-use of the libraries for non-commercial projects.

The implementation of Variational Message Passing has been optimised for speed at the expense of slightly increased memory usage. The memory required for the algorithm is dominated by the storage of posterior parameters. For each node, both the natural parameter representation and the moment vector are stored (to prevent having to continually convert between the two forms). This means that memory consumption has been effectively doubled to achieve significant improvements in speed. Nonetheless, the algorithm has been demonstrated to be capable of performing inference on very large data sets. All required memory is allocated during the initialisation of the algorithm, so as to minimise garbage collection overhead during inference.

The Variational Message Passing implementation is itself extensible in that new deterministic functions or exponential family distributions can be added by creating a single Java class corresponding to that function or distribution. The algorithm can be executed without the VIBES front end or it can be invoked programmatically from another Java program.

2.4 Tutorial: the Gaussian Mixture Model

In this section, I will give a complete tutorial for using VIBES to perform inference in a Gaussian Mixture model. I will then demonstrate the flexibility of VIBES by changing the model to fit the data better (using the lower bound as an estimate of the log evidence for each model). The data set used will be the same toy data set as used in the mixture model example of Chapter 1, shown in Figure 1.12a.

The first step is to load the data set into VIBES. This is achieved by creating a node with the name x which corresponds to a matrix x in a Matlab `.mat` file. As the data matrix has dimensions of 500×2, the node is placed inside two plates N and d which are given sizes of 500 and 2 respectively. The data filename (in this case `MixGaussianData2D.mat`) is entered and selecting `File`→`Load data` loads the data into the node. The node is marked as observed

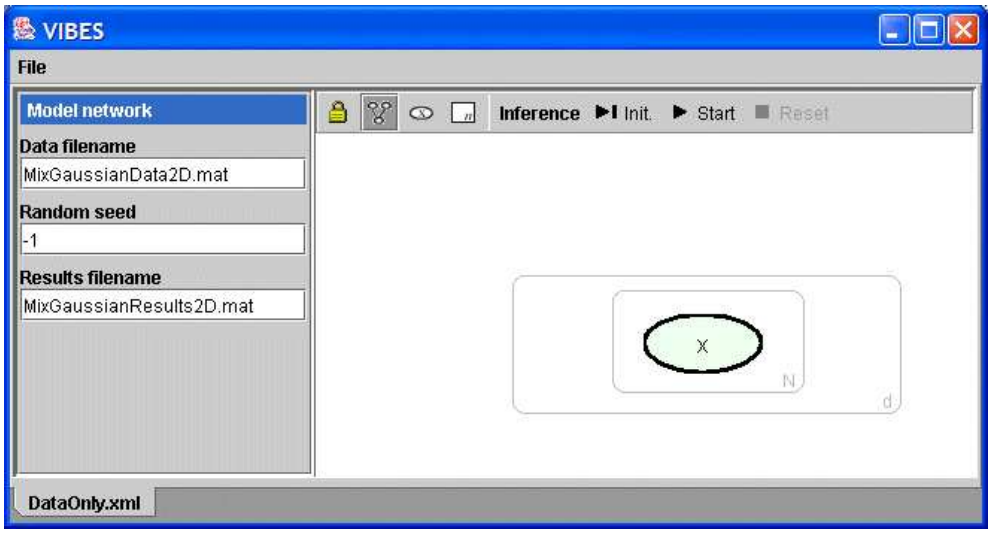


Figure 2.4: A VIBES model with a single observed node x which has attached data.

(shown with a bold edge) and the observed data can be inspected by double-clicking the node with the mouse. At this point, the display is as shown in Figure 2.4.

The node x has been marked as Gaussian by default and so the model is invalid as neither the mean nor the precision of the Gaussian have been set (attempting to initialise the model by pressing the `Init.` button will give an error message to this effect). We can specify latent variables for these parameters by creating a node μ for the mean parameter and a node γ for the precision parameter. These nodes are created within the d plate to give a model which is separable over each data dimension. These are then set as the `Mean` and `Precision` properties of x, as shown in Figure 2.5.

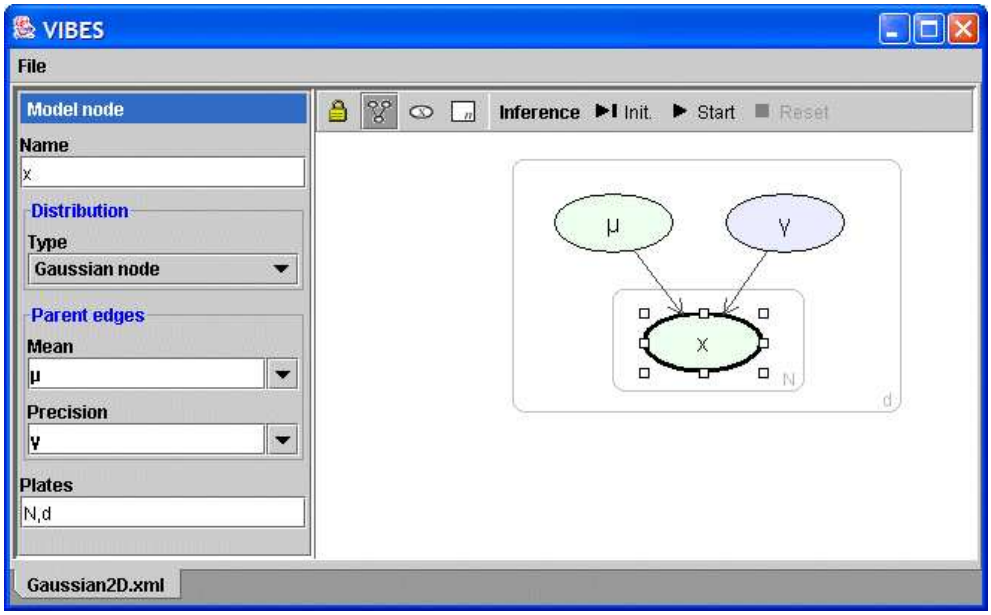


Figure 2.5: A two-dimensional Gaussian model, showing that the latent variables μ and γ are being used as the mean and precision parameters of the conditional distribution over x.

The model is still invalid as the parameters of μ and γ are unspecified. In this case, rather than create further latent variables, these parameters will be set to fixed values to give appropriate priors (for example setting μ to have mean = 0 and precision = 0.01 and γ to have $a = 0.001$ and $b = 0.001$). The model is now a two-dimensional Gaussian equivalent of the model of Section 1.8.5 and variational inference can be performed automatically by pressing the `Start` button (which also performs initialisation). For this data set, inference converges after four iterations and gives a bound of -1984 nats. At this point, the expected values of each latent variable under the fully-factorised Q distribution can be displayed or graphed by double-clicking on the corresponding node.

Extending the Gaussian model to a Gaussian mixture model

Our aim is to create a Gaussians mixture model and so we must extend our simple Gaussian model to be a mixture with K Gaussian components. As there will now be K sets of the latent variables μ and γ, these are placed in a new plate, called K, whose size is set to 20, as before. We modify the conditional distribution for the x node to be a mixture of dimension K, with each component being Gaussian. The display is then as shown in Figure 2.6.

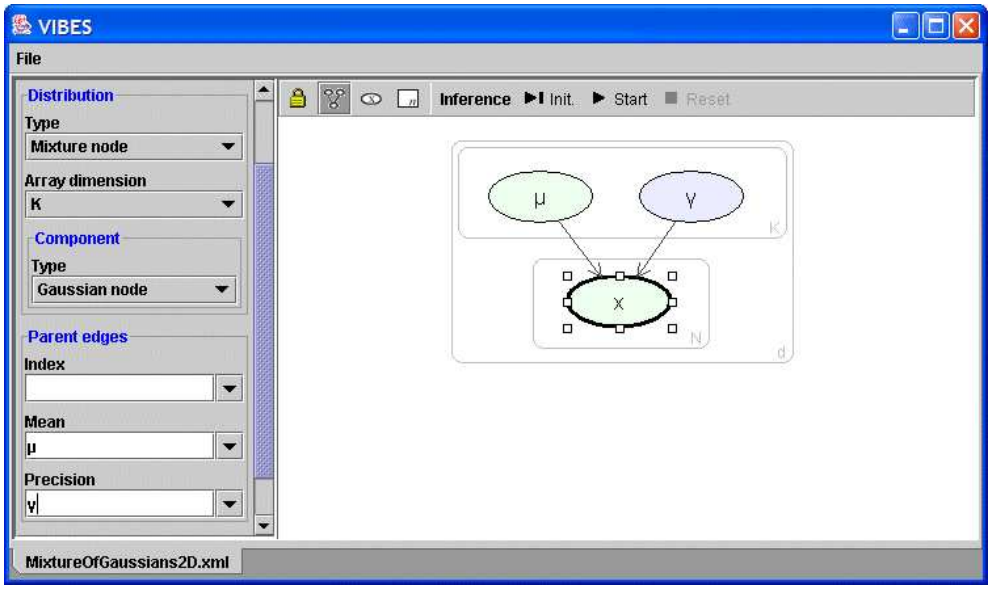


Figure 2.6: An incomplete model which shows that x is now a mixture of K Gaussians. There are now K sets of parameters and so μ and γ have been placed in a plate K. The model is incomplete as the `Index` parent of x has not been specified.

The model is currently incomplete as making x a mixture requires a new discrete `Index` parent to indicate which component distribution each data point was drawn from. We must therefore create a new node λ, sitting in the N plate, to represent this new discrete latent variable. We also create a node π with a Dirichlet distribution which provides a prior over λ. The completed mixture model is shown in Figure 2.7.

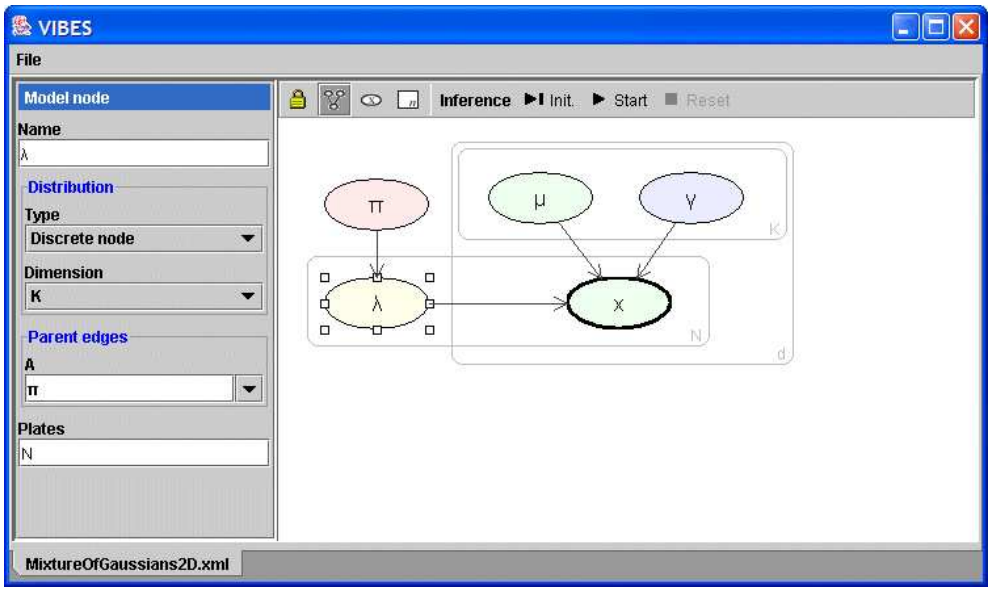


Figure 2.7: The completed Gaussian mixture model showing the discrete indicator node λ.

Inference using the Gaussian mixture model

With the model complete, inference can once again proceed automatically by pressing the `Start` button. A Hinton diagram of the expected value of π can be displayed by double-clicking on the π node, giving the result shown in Figure 2.8. As can be seen, nine of the twenty components have been retained.

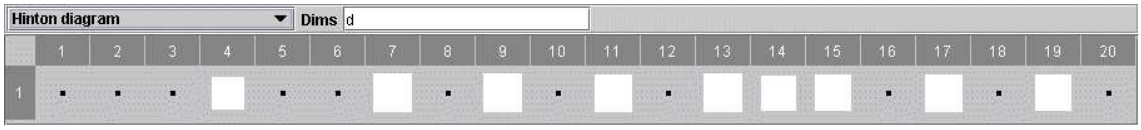


Figure 2.8: A Hinton diagram showing the expected value of π for each mixture component. The learned mixture consists of only nine components.

The means of the retained components can be inspected by double-clicking on the μ node, giving the Hinton diagram of Figure 2.9. These learned means correspond to the centres of each of the data clusters.

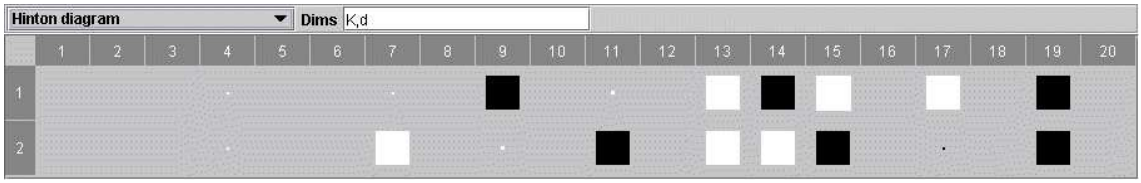


Figure 2.9: A Hinton diagram whose columns give the expected two-dimensional value of the mean μ for each mixture component. The mean of each of the eleven unused components is just the expected value under the prior which, in this case, is $(0, 0)$. Column 4 corresponds to retained component whose mean is roughly $(0, 0)$.

A graph of the evolution of the bound can be displayed by clicking on the bound value and is shown in Figure 2.10. The converged lower bound of this new model is -1019 nats, which is significantly higher than that of the single Gaussian model, showing that there is much greater evidence for this model. As expected, the bound is the same as the one found when this exact model was implemented by hand.

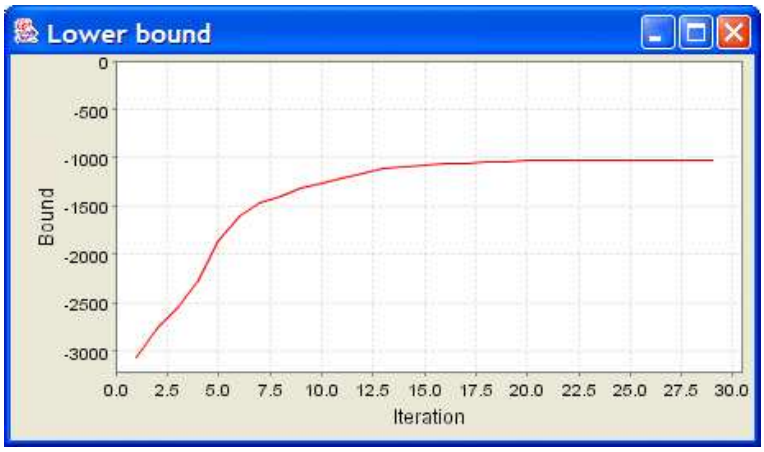


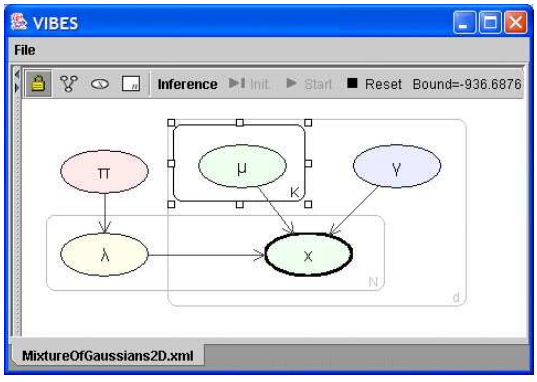
Figure 2.10: A graph of the evolution of the lower bound during inference.

Modifying the mixture model

The rapidity with which models can be constructed using VIBES allows new models to be quickly developed and compared. For example, we can take our existing mixture of Gaussians model and modify it to try and better explain the data.

Firstly, we may hypothesise that the each of the clusters have similar size and so that they may be modelled by a mixture of Gaussian components with a common variance in each dimension. Graphically, this corresponds to shrinking the K plate so that it no longer contains the γ node, as shown in Figure 2.11a. The converged lower bound for this new model

(a)



(b)

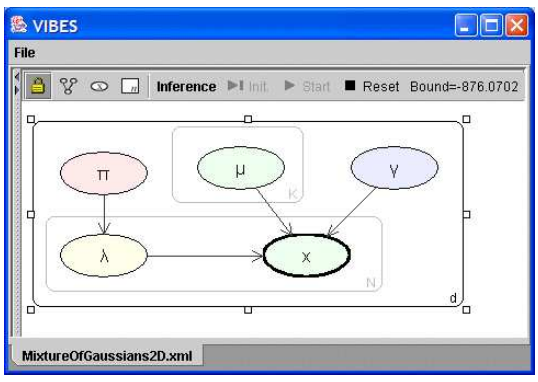


Figure 2.11: (a) Mixture of Gaussians model with shared precision parameter γ (the γ node is no longer inside the K plate). (b) Model with independent data dimensions, each a univariate Gaussian mixture with common variance.

is −937 nats showing that this modified model is better at explaining this data set than the standard mixture of Gaussians model.

We may further hypothesise that the data set is separable with respect to its two dimensions (i.e. the two dimensions are independent). Graphically this consists of moving all nodes inside the d plate (so we effectively have two copies of a one-dimensional mixture of Gaussians model with common variance). A VIBES screenshot of this further modification is shown in Figure 2.11b. Performing variational inference on this separable model leads to each one-dimensional mixture having three retained mixture components and gives an improved bound of -876 nats.

We will consider one final model. In this model both the π and the γ nodes are common to both data dimensions, as shown in Figure 2.12. This change corresponds to the assumption that the mixture coefficients are the same for each of the two mixtures and that the component variances are the same for all components in both mixtures. Inference leads to a final improved bound of −856 nats. Whilst this tutorial has been on a toy data set, the principles of model construction, modification and comparison can be applied just as readily to real data sets.

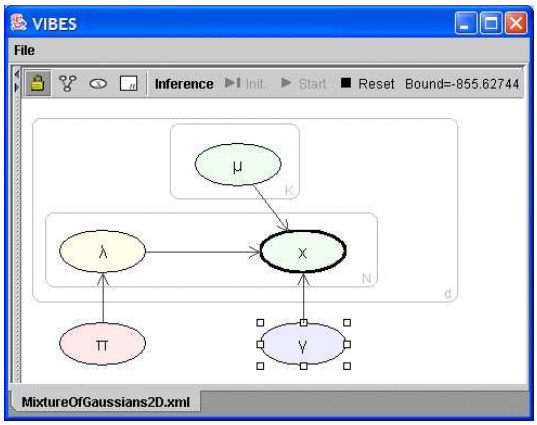


Figure 2.12: Further modified mixture model where the π and γ nodes are now common to all data dimensions.

2.5 Discussion

The definition of the standard variational inference algorithm (as described in Algorithm 1.1) does not specify a number of important details concerning the variational optimisation. For example, there is no standard way to initialise the factors of the Q distribution. In the creation of a general purpose variational framework, it has been necessary to investigate these issues in order to create an approach that will work in as wide a variety of models as possible, as will now be discussed.

2.5.1 Initialisation of the variational distribution

The variational update of a particular factor Q_i depends on the state of all other factors $\{Q_j\}_{j\neq i}$. It follows that, before inference can begin, all but one of the factors must be initialised to suitable values. This initialisation provides the 'starting point' for the optimisation. The local maximum of the bound which the optimisation will find will depend on this initialisation.

In certain optimisations, there may be a number of solutions that are equally close (in terms of KL divergence) to the true posterior. For example, if the model is invariant to permutations of a subset of its parameters, there will be equivalent solutions with those parameters permuted, a situation known as *parameter non-identifiability*. Which of these solutions is reached will depend only on the initialisation of the factors. Hence, there can be no single optimal initialisation.

This argument implies that, whilst the initialisation is extremely important, there can be no one general-purpose initialisation method. In practice, the following methods have been shown to work well:

- initialising to very broad distributions and then adding random perturbations to allow symmetry breaking;
- optimising the Q distribution to be as close to the *prior* distribution as possible and then using that optimised distribution (with random perturbations) as the initial state;
- initialising the mean of each factor distribution to a good ML or MAP solution.

It is straightforward to make the first two methods automatic within the existing framework (the second requires performing several iterations of the message passing procedure but with no observed data). In order to make the third method available in VIBES, a facility has been included to allow custom initialisations to be used.

2.5.2 Interpretation as a factor graph algorithm

The variational message passing algorithm has been defined on a directed graph; it is also possible to define the algorithm on a factor graph. This leads to a simpler definition and allows comparison with other factor graph algorithms.

In the factor graph formulation of VMP, the function nodes are either stochastic (where the function is a conditional probability distribution) or deterministic (where the function is just the deterministic function itself). We can now define generic messages from variable nodes to function node and vice versa. The message from an unobserved variable node X_i to a function node f_j is defined to be

$$m_{X_i \to f_j} = \text{Moments}(Q_i) \tag{2.60}$$

where Moments() is a functional which takes an exponential family distribution and returns a vector of its moments – in other words, it returns $\langle \mathbf{u}_i(X_i) \rangle$. The message from a function node f_i to a variable node X_j is

$$m_{f_i \to X_j} = \text{Natural}(\langle f_i(\text{ne}_i) \rangle_{Q_{\text{ne}_i \setminus X_j}}) \tag{2.61}$$

where Natural() is a functional that takes an exponential family distribution and returns a vector of its natural parameters. Note that whether the function f_i is deterministic or stochastic, conjugacy constraints ensure that the resultant expectation is an exponential family distribution over X_j.

The updated distribution Q^* at a variable node X_i is then updated from the sum of all messages received at that node

$$\text{Natural}(Q^*(X_i)) = \sum_{j \in \text{ne}_i} m_{f_j \to X_i}. \tag{2.62}$$

Comparison to belief propagation

Now that the variational message passing algorithm has been defined on a factor graph, it can be compared to the Sum-Product algorithm, as defined in Section 1.6.2. To aid the comparison, we define a distribution within the Sum-Product algorithm

$$\hat{P}_{i,j}(X_i) = \frac{1}{Z} \prod_{k \in \text{ne}_i \setminus j} m_{f_k \to X_i} \tag{2.63}$$

which can be interpreted as a pseudo-posterior probability over X_i that excludes the influence of the function node j (and all nodes connected to i through j).

In addition, note that all the messages in variational message passing are vectors of moments or vectors of natural parameters, either of which fully defines the corresponding probability distribution. Therefore the messages can be considered as being the corresponding distributions themselves. The messages and posterior distributions for each algorithm can then be written as in the table below, where each Variational Message Passing message is redefined to be an exponential family distribution rather than the moments or natural parameters of that distribution.

Quantity	**Sum-Product**	**Variational Message Passing**
Variable-to-function message	$\hat{P}_{i,j}$	Q_i
Function-to-variable message	$\langle f_i(\text{ne}_i) \rangle_{\prod_{k \in \text{ne}_i \setminus j} \hat{P}_{k,i}}$	$\langle f_i(\text{ne}_i) \rangle_{\prod_{k \in \text{ne}_i \setminus j} Q_k}$
Posterior distribution	$\frac{1}{Z} \prod_{j \in \text{ne}_i} m_{f_j \to X_i}$	$\frac{1}{Z} \prod_{j \in \text{ne}_i} m_{f_j \to X_i}$

The only significant differences between the two algorithms are in the variable-to-function message and when the posterior is calculated. In the Sum-Product algorithm, the message sent from a variable node X_i is different for each neighbour as it excludes the effect of any messages received previously from that neighbour, to prevent double-counting. The exact posterior distribution is then found after one round of message passing. However, in variational message passing, an approximate posterior is maintained throughout and each factor Q_i updated assuming that this approximate posterior is correct. It follows that the message from a variable node to each neighbour is the same and is just Q_i.

2.6 Extensions to the Framework

In this section, a number of small extensions to the variational inference framework will be presented. They are included to give an example of how the framework can be extended to perform different inference calculations and to show how the conjugacy constraints of the framework may be overcome in certain limited circumstances.

2.6.1 Finding a Maximum A Posteriori solution

As discussed in Section 1.8.6, it is possible to obtain a single value for a latent variable by maximising the posterior probability with respect to that variable. Such a Maximum A Posteriori (MAP) solution may, of course, lead to problems as it will not capture the uncertainty in the variable's value. However, if desired, a MAP solution can be obtained using the variational message passing framework as follows. As discussed previously, a MAP solution corresponds to using a variational distribution which is a delta function

$$Q^{\mathrm{MAP}}(\mathbf{H}) = \delta(\mathbf{H} - \mathbf{H}^*) \tag{2.64}$$

where H^* is the MAP solution. This can be written in factorised form as

$$Q^{\mathrm{MAP}}(\mathbf{H}) = \prod_j \delta(H_j - H_j^*). \tag{2.65}$$

and so $Q_j(H_j) = \delta(H_j - H_j^*)$. In section 1.8.4, it was shown that the KL divergence between the approximating distribution and the true posterior is minimised if $\mathrm{KL}(Q_j \,||\, Q_j^*)$ is minimised, where Q_j^* is the standard variational solution given by Equation 1.52. Normally, Q_j is unconstrained so we can simply set it to Q_j^*. However, in this case, Q_j is a delta function and so we have to find the value of H_j^* that minimises $\mathrm{KL}(\delta(H_j - H_j^*) \,||\, Q_j^*)$. Unsurprisingly, this is simply the value of H_j that maximises $Q_j^*(H_j)$.

In the message passing framework, a MAP solution can be obtained for a particular latent variable H_j directly from the updated natural statistic vector ϕ_j^* using

$$(\phi_j^*)^{\mathrm{T}} \frac{d\mathbf{u}_j(H_j)}{dH_j} = 0. \tag{2.66}$$

For example, if Q_j^* is Gaussian with mean μ then $H_j^* = \mu$ or if Q_j^* is Gamma with parameters a, b, then $H_j^* = (a-1)/b$.

Given that the variational posterior is now a delta function, the expectation of any function $f(H_j)$ under the variational posterior is just $f(H_j^*)$. Therefore, in any outgoing messages, $\langle \mathbf{u}_j(H_j) \rangle$ is replaced by $\mathbf{u}_j(H_j^*)$. As all surrounding nodes can process these messages as normal, a MAP solution may be obtained for any chosen subset of variables (such as particular hyper-parameters), whilst a full posterior distribution is retained for all other variables.

2.6.2 Non-conjugate priors

The parameters of some exponential family distributions do not have conjugate prior distributions in the standard exponential family. For example, there is no conjugate distribution for the shape parameter a of the Gamma distribution. This means that there is no way to learn a posterior distribution over a Gamma shape parameter in the existing conjugate-exponential framework.

The purpose of the conjugacy constraint is two-fold. First, it means that the posterior distribution of each variable, conditioned on its neighbours, has the same form as the prior distribution. Hence, the updated variational distribution factor for that variable also has the same form and inference involves just updating the parameters of that distribution. Secondly, conjugacy results in variational distributions being in standard exponential family form allowing their moments to be calculated analytically.

If we ignore the conjugacy constraint, we get non-standard posterior distributions and we must resort to using sampling or other methods to determine the moments of these distributions. The disadvantages of using sampling include computational expense, inability to calculate an analytical lower bound and the fact that inference is no longer deterministic for a given initialisation. The use of sampling methods will now be illustrated by an example showing how to sample from the posterior over the shape parameter of a Gamma distribution.

Example 2.1: Learning a Gamma shape parameter

Let us assume that there is a latent variable a which is to be used as the shape parameter of K gamma distributed variables $\{x_1 \dots x_K\}$. We choose a to have a *non-conjugate* prior of an inverse-Gamma distribution:

$$P(a \,|\, \alpha, \beta) \propto a^{-\alpha-1} \exp\left(\frac{-\beta}{a}\right). \tag{2.67}$$

The form of the gamma distribution means that messages sent to the node a are with respect to a natural statistic vector

$$\mathbf{u}_a = \begin{bmatrix} a \\ \log\Gamma(a) \end{bmatrix} \tag{2.68}$$

which means that the updated factor distribution Q_a^* has the form

$$\log Q_a^*(a) = \left[\sum_{i=1}^{K} m_{x_i \to a}\right]^{\mathrm{T}} \begin{bmatrix} a \\ \log \Gamma(a) \end{bmatrix} + (-\alpha - 1) \log a - \frac{\beta}{a} + \text{const.} \quad (2.69)$$

This density is not of standard form, but it can be shown that $Q^*(\log a)$ is log-concave, so we can generate independent samples from the distribution for $\log a$ using Adaptive Rejection Sampling (ARS) [Gilks and Wild 1992]. These samples are then transformed to get samples of a from $Q_a^*(a)$, which is used to estimate the expectation $\langle \mathbf{u}_a(a) \rangle$. This expectation is then sent as the outgoing message to each of the child nodes.

Each factor distribution is normally updated during every iteration and so, in this case, a number of independent samples from Q_a^* would have to be drawn during every iteration. If this proved too computationally expensive, then the distribution need only be updated intermittently.

It is worth noting that, as in the above example, BUGS also uses ARS for sampling when the posterior distribution is log-concave but non-conjugate whilst also providing techniques for sampling when the posterior is not log-concave. This suggests that non-conjugate parts of a general graphical model could be handled within a BUGS-style framework whilst VIBES is used for the rest of the model. The resulting hybrid variational/sampling framework would, to a certain extent, capture the advantages of both techniques. However, further exploration of such hybrid inference frameworks is beyond the scope of this thesis.

2.7 Summary

In this chapter, I have developed the Variational Message Passing algorithm which allows automatic variational inference in conjugate-exponential models using a fully factorised variational distribution. A software package, VIBES, which implements the Variational Message Passing algorithm, has been described and a short tutorial provided to give an example of its use. By framing Variational Message Passing as an algorithm on a factor graph, it has been compared to the Belief Propagation algorithm and found to have considerable similarities. Finally, a number of small extensions to the algorithm have been discussed.

In the following chapter, I look at the application of Variational Message Passing and VIBES to the problem of modelling image subspaces.

Chapter 3

Application: Non-linear Image Modelling

The variational inference framework can be used to perform Bayesian inference in a very wide variety of domains. In this chapter, it will be used in the domain of machine vision for modelling image subspaces. The approach involves constructing a probabilistically consistent density model which can capture essentially arbitrary non-linearities, and which can discover an appropriate dimensionality for modelling the manifold[1]. A key feature is the use of a fully Bayesian formulation in which the appropriate model complexity, and indeed the dimensionality of the manifold itself, can be discovered *automatically* as part of the inference procedure [Bishop 1999a]. The model is based on a mixture of components each of which is a latent variable model whose dimensionality can be inferred from the data. It avoids a discrete model search over dimensionality, involving instead the use of continuous hyper-parameters to determine an effective dimensionality for the components in the mixture model.

I will start by summarising a number of previous approaches to modelling image subspaces. I will go on to describe the probabilistic model used in my approach and how the variational inference framework was used to fit it to the data. Thereafter, results on synthetic and real data sets will be presented.

3.1 Modelling Image Subspaces

Interest in image subspace modelling has grown considerably in recent years in contexts such as recognition, detection, verification and coding. Although an individual image can be considered as a point in a high-dimensional space described by the pixel values, an ensemble of related images, for example faces, lives on a (noisy) non-linear manifold having a much lower *intrinsic* dimensionality. One of the simplest approaches to modelling such manifolds involves finding the principal components of the ensemble of images, as used for example in 'eigen-faces' [Turk and Pentland 1991].

[1] The work in this chapter is based on a paper written in collaboration with Christopher M. Bishop [Bishop and Winn 2000].

However, simple principal component analysis (PCA) suffers from two key limitations. Firstly, it does not directly define a probability distribution, and so it is difficult to use standard PCA as a component in a probabilistic solution to a computer vision problem. Secondly, the manifold defined by PCA is necessarily linear. Techniques which address the first of these problems by constructing a density model include Gaussians and mixtures of Gaussians [Moghaddam and Pentland 1997]. The second problem has been addressed by considering non-linear projective methods such as principal curves and auto-encoder neural networks [Moghaddam 1999]. Bregler and Omohundro [1995] and Heap and Hogg [1998] use mixture representations to try to capture the non-linearity of the manifold. However, their model fitting is based on simple clustering algorithms (related to K-means) and lacks the fully probabilistic approach, as discussed in this chapter.

A central problem in density modelling in high dimensional spaces concerns model complexity. Models fitted using maximum likelihood are particularly prone to severe over-fitting unless the number of free parameters is restricted to be much less than the number of data points. For example, it is clearly not feasible to fit an unconstrained mixture of Gaussians directly to the data in the original high-dimensional space using maximum likelihood due to the excessive number of parameters in the covariance matrices. Moghaddam and Pentland [1997] therefore project the data onto a PCA sub-space and then perform density estimation within this lower dimensional space using Gaussian mixtures. While this limits the number of free parameters in the model, the non-linearity of the manifold requires the PCA space to have a significantly higher dimensionality than that of the manifold itself, and so again the model is prone to over-parameterisation.

One important aspect of model complexity concerns the dimensionality of the manifold itself, which is typically not known in advance. Moghaddam [1999], for example, arbitrarily fixes the model dimensionality to be 20.

Several authors have explored the use of non-linear warping of the image, for example in the context of face recognition, in order to take account of changes of pose or of interpersonal variation [Black and Yacoob 1995; Cootes et al. 1995; Frey and Jojic 1999]. In so far as such distortions can be accurately represented, these transformations should be of significant benefit in tackling the subspace modelling problem, albeit at increased computational expense. Such approaches can be used to augment virtually any sub-space modelling algorithm, including those discussed in this chapter, and so they will not be considered further.

3.2 Models for Manifolds

This approach to modelling the manifolds of images builds upon recent developments in latent variable models and can be seen as a natural development of PCA and mixture modelling frameworks leading to a highly flexible, fully probabilistic framework. Firstly, I will show how conventional PCA can be reformulated probabilistically and hence used as the component distribution in a mixture model. Then I show how a Bayesian approach allows the

model complexity (including the number of components in the mixture as well as the effective dimensionality of the manifold) to be inferred from the data.

3.2.1 Maximum likelihood PCA

Principal component analysis (PCA) is a widely used technique for data analysis. It can be defined as the linear projection of a data set into a lower-dimensional space under which the retained variance is a maximum, or equivalently under which the sum-of-squares reconstruction cost is minimised.

Consider a data set D of observed d-dimensional vectors $D = \{\mathbf{t}_n\}$ where $n \in \{1, \ldots, N\}$. Conventional PCA is obtained by first computing the sample covariance matrix given by

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{t}_n - \overline{\mathbf{t}})(\mathbf{t}_n - \overline{\mathbf{t}})^{\mathrm{T}} \tag{3.1}$$

where $\overline{\mathbf{t}} = N^{-1} \sum_n \mathbf{t}_n$ is the sample mean. Next the eigenvectors $\mathbf{u}_i$ and eigenvalues λ_i of $\mathbf{S}$ are found, where $\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i$ and $i = 1, \ldots, d$. The eigenvectors corresponding to the q largest eigenvalues (where $q < d$) are retained, and a reduced-dimensionality representation of the data set is defined by $\mathbf{x}_n = \mathbf{U}_q^{\mathrm{T}}(\mathbf{t}_n - \overline{\mathbf{t}})$ where $\mathbf{U}_q = (\mathbf{u}_1, \ldots, \mathbf{u}_q)$.

A significant limitation of conventional PCA is that it does not define a probability distribution. Recently, however, Tipping and Bishop [1999b] and Roweis [1998] have shown how PCA can be reformulated as the maximum likelihood solution of a specific latent variable model, as follows. Firstly, a q-dimensional latent variable $\mathbf{x}$ is introduced whose prior distribution is a zero mean Gaussian $P(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \mathbf{I}_q)$ and $\mathbf{I}_q$ is the q-dimensional unit matrix. The observed variable $\mathbf{t}$ is then defined as a linear transformation of $\mathbf{x}$ with additive Gaussian noise $\mathbf{t} = \mathbf{W}\mathbf{x} + \boldsymbol{\mu} + \boldsymbol{\epsilon}$ where $\mathbf{W}$ is a $d \times q$ matrix, $\boldsymbol{\mu}$ is a d-dimensional vector and $\boldsymbol{\epsilon}$ is a zero-mean Gaussian-distributed vector with covariance $\tau^{-1}\mathbf{I}_d$ (where τ is the precision). Thus, $P(\mathbf{t} \,|\, \mathbf{x}) = \mathcal{N}(\mathbf{W}\mathbf{x} + \boldsymbol{\mu}, \tau^{-1}\mathbf{I}_d)$. The marginal distribution of the observed variable is then given by the convolution of two Gaussians and is itself Gaussian

$$P(\mathbf{t}) = \int P(\mathbf{t} \,|\, \mathbf{x}) P(\mathbf{x}) \, d\mathbf{x} = \mathcal{N}(\boldsymbol{\mu}, \mathbf{C}) \tag{3.2}$$

where the covariance matrix $\mathbf{C} = \mathbf{W}\mathbf{W}^{\mathrm{T}} + \tau^{-1}\mathbf{I}_d$. The model of Equation 3.2 represents a constrained Gaussian distribution governed by the parameters $\boldsymbol{\mu}$, $\mathbf{W}$ and τ.

It was shown by Tipping and Bishop [1999b] that the stationary points of the log likelihood with respect to $\mathbf{W}$ satisfy

$$\mathbf{W}_{\mathrm{ML}} = \mathbf{U}_q (\boldsymbol{\Lambda}_q - \tau^{-1}\mathbf{I}_q)^{1/2} \tag{3.3}$$

where the columns of $\mathbf{U}_q$ are eigenvectors of $\mathbf{S}$, with corresponding eigenvalues in the diagonal matrix $\boldsymbol{\Lambda}_q$. It was also shown that the *maximum* of the likelihood is achieved when the q largest eigenvalues are chosen, so that the columns of $\mathbf{U}_q$ correspond to the *principal* eigenvectors, with all other choices of eigenvalues corresponding to saddle points. The maximum

likelihood solution for τ is then given by

$$\frac{1}{\tau_{\mathrm{ML}}} = \frac{1}{d-q} \sum_{i=q+1}^{d} \lambda_i \tag{3.4}$$

which has a natural interpretation as the average variance lost per discarded dimension. The density model (Equation 3.2) thus represents a probabilistic formulation of PCA. It is easily verified that conventional PCA is recovered by treating τ as a parameter and taking the limit $\tau \to \infty$.

Probabilistic PCA has been successfully applied to problems in data compression, density estimation and data visualisation, and has been extended to mixture and hierarchical mixture models [Bishop and Tipping 1998; Tipping and Bishop 1999a,b]. As with conventional PCA, however, the model itself provides no mechanism for determining the value of the latent-space dimensionality q. For $q = d - 1$ the model is equivalent to a full-covariance Gaussian distribution[2], while for $q < d - 1$ it represents a constrained Gaussian in which the variance in the remaining $d - q$ directions is modelled by the single parameter τ. Thus, the choice of q corresponds to a problem in model complexity optimisation. In principal *cross-validation*[3] to compare all possible values of q offers a possible approach. However, maximum likelihood estimation is highly biased (leading to 'overfitting') and so in practice excessively large data sets would be required and the procedure would become computationally intractable.

3.2.2 Bayesian PCA

The issue of model complexity can be handled naturally within a Bayesian paradigm. Armed with the probabilistic reformulation of PCA defined in Section 3.2.1, a Bayesian treatment of PCA is obtained by first introducing prior distributions over the parameters $\boldsymbol{\mu}$, $\mathbf{W}$ and τ. A key goal is to control the effective dimensionality of the latent space (corresponding to the number of retained principal components). Furthermore, it is desirable to avoid discrete model selection and hence continuous hyper-parameters are introduced to determine automatically an appropriate *effective* dimensionality for the latent space as part of the process of Bayesian inference. This is achieved by introducing a hierarchical prior $P(\mathbf{W} \,|\, \boldsymbol{\alpha})$ over the matrix $\mathbf{W}$, governed by a q-dimensional vector of hyper-parameters $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_q\}$. Each hyper-parameter controls one of the columns of the matrix $\mathbf{W}$ through a conditional Gaussian distribution of the form

$$P(\mathbf{W} \,|\, \boldsymbol{\alpha}) = \prod_{i=1}^{q} \left(\frac{\alpha_i}{2\pi}\right)^{d/2} \exp\left\{-\frac{1}{2}\alpha_i \|\mathbf{w}_i\|^2\right\} \tag{3.5}$$

[2]This follows from the fact that the $q - 1$ linearly independent columns of $\mathbf{W}$ have independent variances along $q - 1$ directions, while the variance along the remaining direction is controlled by τ.

[3]For a definition of cross-validation, see Bishop [1995, pp. 372–375].

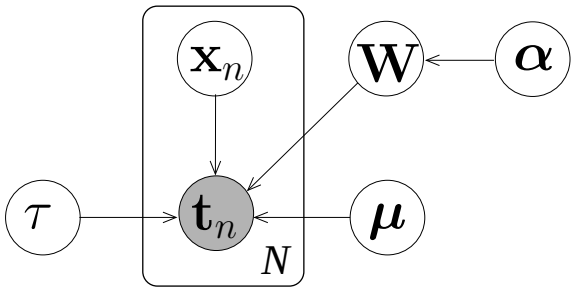


Figure 3.1: The Bayesian network for the Principal Component Analysis model. Each of the N observed data vectors $\mathbf{t}_n$ has a corresponding low-dimensional representation $\mathbf{x}_n$. The columns of the matrix $\mathbf{W}$ correspond to directions in the latent space and are equivalent to the principal components. The hyper-parameter $\boldsymbol{\alpha}$ controls which of the principal components are 'switched off' and so provides a form of automatic relevance determination.

where $\{\mathbf{w}_i\}$ are the columns of $\mathbf{W}$. This form of prior is motivated by the framework of *automatic relevance determination* (ARD) introduced in the context of neural networks by Neal [1994] and MacKay [1995]. Each α_i controls the inverse variance of the corresponding $\mathbf{w}_i$, so that if a particular α_i has a posterior distribution concentrated at large values, the corresponding $\mathbf{w}_i$ will tend to be small, and that direction in latent space will be effectively 'switched off'. The dimensionality of the latent space is set to its maximum possible value $q = d - 1$.

The specification of the Bayesian model is completed by defining the remaining priors to have the form

$$P(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu} \,|\, \mathbf{0}, \beta^{-1}\mathbf{I}) \tag{3.6}$$

$$P(\boldsymbol{\alpha}) = \prod_{i=1}^{q} \Gamma(\alpha_i \,|\, a_\alpha, b_\alpha) \tag{3.7}$$

$$P(\tau) = \Gamma(\tau \,|\, a_\tau, b_\tau). \tag{3.8}$$

Here $\mathcal{N}(\mathbf{x} \,|\, \mathbf{m}, \boldsymbol{\Sigma})$ denotes a multivariate normal distribution over $\mathbf{x}$ with mean $\mathbf{m}$ and covariance matrix $\boldsymbol{\Sigma}$. Similarly, $\Gamma(x \,|\, a, b)$ denotes a Gamma distribution over x as defined in Appendix A.3. Broad priors are obtained by setting $a_\alpha = b_\alpha = a_\tau = b_\tau = 10^{-3}$ and $\beta = 10^{-3}$.

The Bayesian network for the complete model is shown in Figure 3.1. The model is conjugate-exponential and so the variational message passing framework can be applied immediately to train the model for a particular data set; for example, it can be applied to train the model on the toy data set of the following example.

Example 3.1: PCA on 3-dimensional manifold in 10-dimensions

As an illustration of the role of the hyperparameters in determining model complexity, consider a data set consisting of 300 points in 10 dimensions, in which the data is drawn from a Gaussian distribution having standard deviation 1.0 in 3 directions and standard deviation 0.5 in the remaining 7 directions. The result

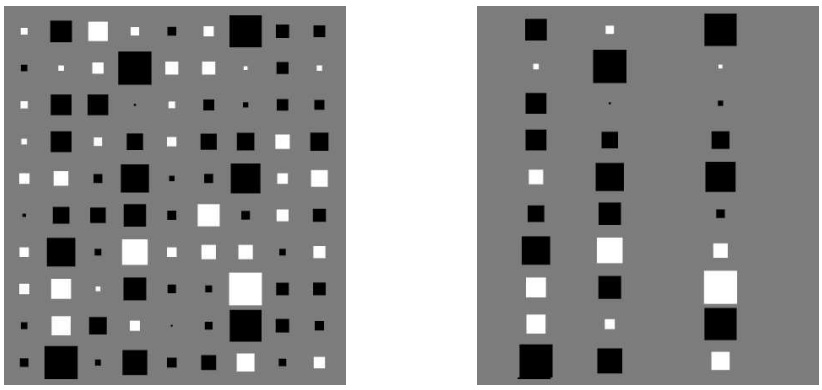


Figure 3.2: Hinton diagrams of the matrix $\mathbf{W}$ for a data set in 10 dimensions having $m = 3$ directions with larger variance than the remaining 7 directions. The area of each square is proportional to the magnitude of the corresponding matrix element, and the squares are white for positive values and black for negative values. The left plot shows $\mathbf{W}_{\mathrm{ML}}$ from maximum likelihood PCA while the right plot shows the posterior mean $\langle \mathbf{W} \rangle$ from the Bayesian approach, showing how the model is able to discover the appropriate dimensionality by suppressing the 6 surplus degrees of freedom.

of fitting both maximum likelihood and Bayesian PCA models is shown in Figure 3.2. In this case the Bayesian model has an effective dimensionality of $q_{\mathrm{eff}} = 3$, as expected, and an inferred noise standard deviation of roughly 0.5.

Effective dimensionality and data set size

Given a latent space of dimensionality q, the effective dimensionality of the space which will be inferred will depend on the number of data points available N. If the number of data points is too small, there may not be sufficient data to justify retaining all the latent space dimensions and some may be switched off. The effective dimensionality will therefore be lower than the actual dimensionality.

To investigate this relationship, consider data sets in 10 dimensions whose data are drawn from a multivariate Gaussian distribution with standard deviations of {1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1} in each of the 10 dimensions. Data set sizes from $N = 40$ to 700 were used with 50 data sets being generated of each size. A Bayesian PCA model was trained on each data set and the effective dimensionality q_{eff} recorded. This effective dimensionality corresponds to the number of dimensions with large α values (where 'large' was defined to be greater than a quarter of the maximum value of any α). The average effective dimensionality over the 50 data sets for each value of N was found and plotted against N. The results are shown in the graph of Figure 3.3. As can be seen, hardly any data points are required to find the first few dimensions of the latent space (which have large variance) but more and more data points are required to support each additional dimension.

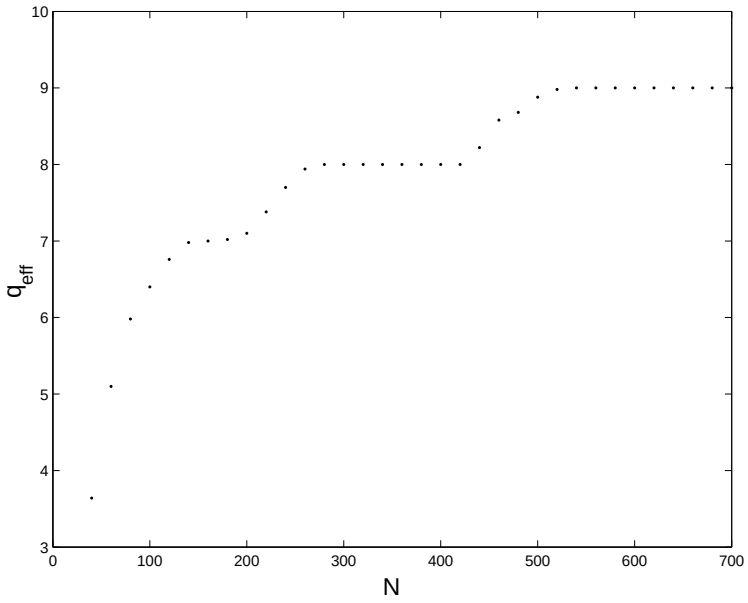


Figure 3.3: Effective dimensionality of a Bayesian PCA model for various sizes of data set. Each point represents the average inferred dimensionality for 50 data sets of size N. As the PCA model assumes that direction of smallest variance is noise, the learned dimensionality of a large data set is nine. However, smaller data sets do not provide sufficient evidence to retain all of these data dimensions; those with smaller variance are instead classified as noise.

3.2.3 Mixtures of Bayesian PCA models

Given a probabilistic formulation of PCA, it is now possible to construct a mixture distribution comprising a linear superposition of principal component analysers. If such a model were to be fitted to data using maximum likelihood we would have to choose both the number M of components and the latent space dimensionality q of the components. For moderate numbers of components and data spaces of several dimensions it quickly becomes computationally costly to use cross-validation.

Here Bayesian PCA offers an advantage in allowing the effective dimensionalities of the models to be determined automatically. Furthermore, we also wish to determine the appropriate number of components in the mixture. We do this by variational model selection (see Section 1.8.3) as an integral part of the learning procedure, as discussed in the next section.

To formulate the probabilistic model we introduce, for each data point $\mathbf{t}_n$, an additional M-dimensional binary latent variable $\mathbf{s}_n$ which has one non-zero element denoting which of the M components in the mixture is responsible for generating $\mathbf{t}_n$. These discrete latent variables have distributions governed by hyperparameters $\boldsymbol{\pi} = \{\pi_m\}$ where $m = 1, \ldots, M$,

$$P(\mathbf{s} = \delta_m \,|\, \boldsymbol{\pi}) = \pi_m \tag{3.9}$$

where δ_m denotes a vector with all elements zero except element m whose value is 1. The parameters $\boldsymbol{\pi}$ are given a Dirichlet distribution, as defined in Appendix A.5.

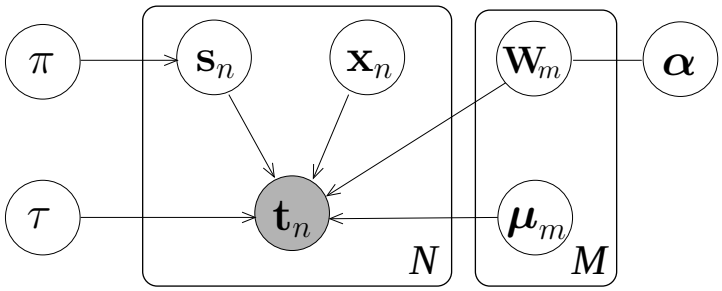


Figure 3.4: The Bayesian network for the mixture of PCA models showing the hierarchical prior over $\mathbf{W}$ governed by the vector of shared hyper-parameters $\boldsymbol{\alpha}$. The left-hand plate contains N independent observations of the data vector $\mathbf{t}_n$ (shown shaded) together with the corresponding hidden variables $\mathbf{x}_n$ and $\mathbf{s}_n$, whilst the right-hand plate contains the M copies of the parameters associated with each component in the mixture.

In a simple mixture of Bayesian PCA models, each component would be free to determine its own dimensionality. A central goal of this work, however, is to model a continuous non-linear manifold. If follows that we may want the components in the mixture to have a common dimensionality whose value is a-priori unknown and which should be inferred from the data. This can be achieved within our framework by using a single set of $\boldsymbol{\alpha}$ hyper-parameters which are *shared* by all of the components in the mixture. The probabilistic structure of the resulting model is displayed diagrammatically in the Bayesian network of Figure 3.4.

3.3 Variational Inference

Having expressed the probabilistic model as a Bayesian network, inference must be performed to determine the posterior distribution over the latent variables in the model, given a data set. As in the case of Bayesian PCA, the mixture of Bayesian PCA model of Figure 3.4 is conjugate-exponential. It follows that Variational Message Passing can be applied to find a variational approximation to the desired posterior distribution. In this initial application of VMP, the variational solution will also be derived by hand to provide both a check on the results given by VMP and an example of the amount of time that can be saved by using the variational inference framework.

3.3.1 Derivation of the variational solution

First, we assume a Q distribution of the form

$$Q(S, X, \boldsymbol{\pi}, \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\mu}, \tau) = Q(S)Q(X \mid S)Q(\boldsymbol{\pi})Q(\mathbf{W})Q(\boldsymbol{\alpha})Q(\boldsymbol{\mu})Q(\tau) \tag{3.10}$$

where $X = \{\mathbf{x}_n\}$. The joint distribution of data and parameters is given by

$$\left[\prod_{n=1}^{N} P(\mathbf{t}_n \mid \mathbf{x}_n, \mathbf{W}, \boldsymbol{\mu}, \tau, S)\right] P(X)P(S \mid \boldsymbol{\pi})P(\boldsymbol{\pi})P(\mathbf{W} \mid \boldsymbol{\alpha})P(\boldsymbol{\alpha})P(\boldsymbol{\mu})P(\tau). \tag{3.11}$$

Using Equations 3.10 and 3.11 in Equation 1.52, and substituting for the various $P(\cdot)$ distributions, we obtain the following results for the component distributions of $Q(\cdot)$:

$$
\begin{aligned}
Q(X \mid S) &= \prod_{n=1}^{N} Q(\mathbf{x}_n \mid \mathbf{s}_n) && (3.12)\\
Q(\mathbf{x}_n \mid \mathbf{s}_n = \delta_m) &= \mathcal{N}(\mathbf{x}_n \mid \mathbf{m}_{\mathbf{x}}^{(nm)}, \boldsymbol{\Sigma}_{\mathbf{x}}^{(m)}) && (3.13)\\
Q(\boldsymbol{\mu}) &= \prod_{m=1}^{M} \mathcal{N}(\boldsymbol{\mu}_m \mid \mathbf{m}_{\boldsymbol{\mu}}^{(m)}, \boldsymbol{\Sigma}_{\boldsymbol{\mu}}^{(m)}) && (3.14)\\
Q(\mathbf{W}) &= \prod_{m=1}^{M} \prod_{k=1}^{d} \mathcal{N}(\widetilde{\mathbf{w}}_{km} \mid \mathbf{m}_{\mathbf{w}}^{(km)}, \boldsymbol{\Sigma}_{\mathbf{w}}^{(m)}) && (3.15)\\
Q(\boldsymbol{\alpha}) &= \prod_{m=1}^{M} \prod_{i=1}^{q} \Gamma(\alpha_{mi} \mid \widetilde{a}_\alpha, \widetilde{b}_\alpha^{(mi)}) && (3.16)\\
Q(\tau) &= \Gamma(\tau \mid \widetilde{a}_\tau, \widetilde{b}_\tau) && (3.17)\\
Q(\Pi) &= \prod_{m=1}^{M} \mathrm{Dir}(\pi_m \mid \widetilde{u}^{(m)}) && (3.18)\\
Q(S) &= \prod_{n=1}^{N} Q(\mathbf{s}_n) && (3.19)
\end{aligned}
$$

where $\widetilde{\mathbf{w}}_k$ denotes a column vector corresponding to the kth row of $\mathbf{W}$. Here I have defined:

$$
\begin{aligned}
\mathbf{m}_{\mathbf{x}}^{(nm)} &= \langle\tau\rangle \boldsymbol{\Sigma}_{\mathbf{x}}^{(m)} \langle\mathbf{W}_m^{\mathrm{T}}\rangle (\mathbf{t}_n - \langle\boldsymbol{\mu}_m\rangle) && (3.20)\\
\boldsymbol{\Sigma}_{\mathbf{x}}^{(m)} &= \left(\mathbf{I}_q + \langle\tau\rangle \langle\mathbf{W}_m^{\mathrm{T}} \mathbf{W}_m\rangle\right)^{-1} && (3.21)\\
\mathbf{m}_{\boldsymbol{\mu}}^{(m)} &= \boldsymbol{\Sigma}_{\boldsymbol{\mu}}^{(m)} \langle\tau\rangle \sum_{n=1}^{N} \langle s_{nm}\rangle \left(\mathbf{t}_n - \langle\mathbf{W}_m\rangle \langle\mathbf{x}_n \mid m\rangle\right) && (3.22)\\
\boldsymbol{\Sigma}_{\boldsymbol{\mu}}^{(m)} &= \left(\beta + \langle\tau\rangle \sum_{n=1}^{N} \langle s_{nm}\rangle\right)^{-1} \mathbf{I}_d && (3.23)\\
\mathbf{m}_{\mathbf{w}}^{(km)} &= \boldsymbol{\Sigma}_{\mathbf{w}} \langle\tau\rangle \sum_{n=1}^{N} \langle s_{nm}\rangle \langle\mathbf{x}_n \mid m\rangle (t_{nk} - \langle\mu_k\rangle) && (3.24)\\
\boldsymbol{\Sigma}_{\mathbf{w}}^{(m)} &= \left(\mathrm{diag}\langle\boldsymbol{\alpha}_m\rangle + \langle\tau\rangle \sum_{n=1}^{N} \langle s_{nm}\rangle \langle\mathbf{x}_n \mathbf{x}_n^{\mathrm{T}} \mid m\rangle\right)^{-1} && (3.25)\\
\widetilde{a}_\alpha &= a_\alpha + \frac{d}{2} \qquad \widetilde{b}_\alpha^{(mj)} = b_\alpha + \frac{\langle\|\mathbf{w}_{mj}\|^2\rangle}{2} \qquad \widetilde{a}_\tau = a_\tau + \frac{Nd}{2} && (3.26)\\
\widetilde{b}_\tau &= b_\tau + \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} \langle s_{nm}\rangle \left\{\|\mathbf{t}_n\|^2 + \langle\|\boldsymbol{\mu}_m\|^2\rangle + \mathrm{Tr}(\langle\mathbf{W}_m^{\mathrm{T}} \mathbf{W}_m\rangle \langle\mathbf{x}_n \mathbf{x}_n^{\mathrm{T}} \mid m\rangle)\right.\\
&\quad \left. + 2\langle\boldsymbol{\mu}_m^{\mathrm{T}}\rangle \langle\mathbf{W}_m\rangle \langle\mathbf{x}_n \mid m\rangle - 2\mathbf{t}_n^{\mathrm{T}} \langle\mathbf{W}_m\rangle \langle\mathbf{x}_n \mid m\rangle - 2\mathbf{t}_n^{\mathrm{T}} \langle\boldsymbol{\mu}_m\rangle\right\} && (3.27)
\end{aligned}
$$

$$\widetilde{u}^{(m)} = u_m + \sum_{n=1}^{N} \langle s_{nm} \rangle \tag{3.28}$$

$$\begin{aligned} \log Q(\mathbf{s}_n = \delta_m) = {} & \langle \log \pi_m \rangle - \frac{1}{2}\langle \mathbf{x}_n^{\mathrm{T}} \mathbf{x}_n \,|\, m \rangle - \frac{1}{2}\langle \tau \rangle \left\{ \|\mathbf{t}_n\|^2 + \langle \|\boldsymbol{\mu}_m\|^2 \rangle \right. \\ & + \mathrm{Tr}(\langle \mathbf{W}_m^{\mathrm{T}} \mathbf{W}_m \rangle \langle \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}} \,|\, m \rangle) + 2\langle \boldsymbol{\mu}_m^{\mathrm{T}} \rangle \langle \mathbf{W}_m \rangle \langle \mathbf{x}_n \,|\, m \rangle \qquad (3.29) \\ & \left. - 2\mathbf{t}_n^{\mathrm{T}} \langle \mathbf{W}_m \rangle \langle \mathbf{x}_n \,|\, m \rangle - 2\mathbf{t}_n^{\mathrm{T}} \langle \boldsymbol{\mu}_m \rangle \right\} + \frac{1}{2} \log |\boldsymbol{\Sigma}_{\mathbf{x}}^{(m)}| + \text{const.} \qquad (3.30) \end{aligned}$$

where $\mathrm{diag}\langle \boldsymbol{\alpha} \rangle$ denotes a diagonal matrix whose diagonal elements are given by $\langle \alpha_i \rangle$. The constant in $\log Q(\mathbf{s}_n = \delta_n)$ is found simply by summing and normalising. Note also that $\langle \mathbf{x}_n \,|\, m \rangle$ denotes an average with respect to $Q(\mathbf{x}_n \,|\, \mathbf{s}_n = \delta_m)$.

The framework also permits a direct evaluation of the posterior distribution over the number M of components in the mixture (assuming a suitable prior distribution, for example a uniform distribution up to some maximum value). However, in order to reduce the computational complexity of the inference problem we adopt an alternative approach based on model comparison using the numerically evaluated lower bound $\mathcal{L}(Q)$ which approximates the log model probability $\log P(\mathbf{t})$. Our optimisation mechanism dynamically adapts the value of M through a scheme involving the addition and deletion of components (as in Ueda et al. [1999]; Ghahramani and Beal [1999]).

One of the limitations of fitting conventional Gaussian mixture models by maximum likelihood is that there are singularities in the likelihood function in which a component's mean coincides with one of the data points while its covariance shrinks to zero. Such singularities do not arise in the Bayesian framework due to the use of priors over model parameters.

3.4 Results

In order to demonstrate the operation of the algorithm, its behaviour has been explored using synthetic data before being applied to real data sets.

3.4.1 Synthetic data: noisy sinusoid

Figure 3.5 shows a non-linear one dimensional manifold embedded in two dimensions, together with the result of fitting a Bayesian PCA mixture model. The lines represent the non-zero principal directions of each component in the mixture. At convergence the model had eight components, having a common effective dimensionality of one.

3.4.2 Synthetic data: noisy sphere

Figure 3.6 shows synthetic data from a noisy two-dimensional sphere in three dimensions together with the converged model, which has 12 components with effective dimensionality of two. Similar results with synthetic data are robustly obtained when embedding low-dimensional non-linear manifolds in spaces of higher dimensionality.

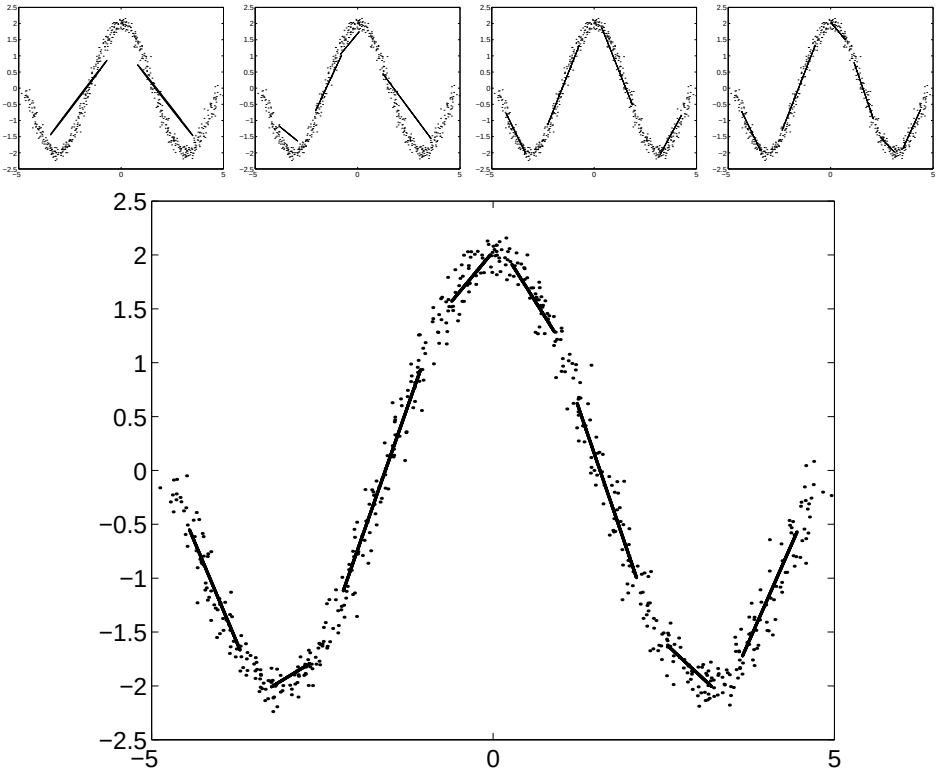


Figure 3.5: Example of a Bayesian PCA mixture model fitted to a highly non-linear one dimensional manifold. The top four graphs are snapshots of the model taken during the inference process. The bottom graph is the converged result.

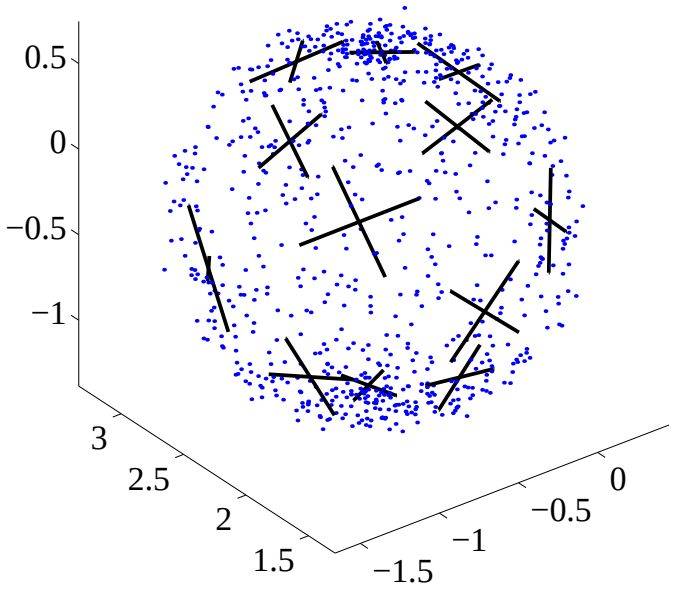


Figure 3.6: Example of a Bayesian PCA mixture model fitted to a two-dimensional manifold corresponding the (noisy) surface of a sphere. The model has been fitted using twelve PCA components, each with an effective dimensionality of two, showing that the dimensionality of the manifold has been correctly determined.

3.4.3 Faces data set

Let us now apply the framework to the problem of modelling the manifold of a data set of face images. The data set used is a combination of images from the Yale face database and the University of Stirling database. The training set comprises 276 training images, which have been cropped, sub-sampled to 26×15, and normalised pixel-wise to zero mean and unit variance. The test set consists of a further 100 face images, together with 200 non-face images, taken from the Corel database, all of which were pre-processed in the same way as the training data.

The converged Bayesian PCA mixture model has four components, having a common dimensionality of five, as emphasised by the Hinton diagram of the shared $\boldsymbol{\alpha}$ hyper-parameters shown in Figure 3.7.

Figure 3.7: Hinton diagram showing the inverses of the $\boldsymbol{\alpha}$ hyper-parameters (corresponding to the variances of the principal components) indicating a manifold with an intrinsic dimensionality of five.

The small number of components and low dimensionality of the resultant model suggests that the small size of the data set has restricted the complexity of the learned model. In order to see how well the model has captured the manifold, we first run the model generatively to give some sample synthetic images, as shown in Figure 3.8. The extent to which the model has succeeded in modelling the manifold of faces can be quantified by using the density model to classify the images in the test set as faces versus non-faces. To do this, the density under the model was evaluated for each test image and if this density exceeded some threshold the image was classified as a face. The threshold value determines the trade-off between false negatives and false positives, leading to a Receiver Operating Curve (ROC), as shown in Figure 3.9. For comparison the corresponding ROC curves for a single maximum likelihood PCA model for a range of different q values are also shown. It can be seen that moving from a linear model (PCA) to a non-linear model (a Bayesian PCA mixture) gives a significant improvement in

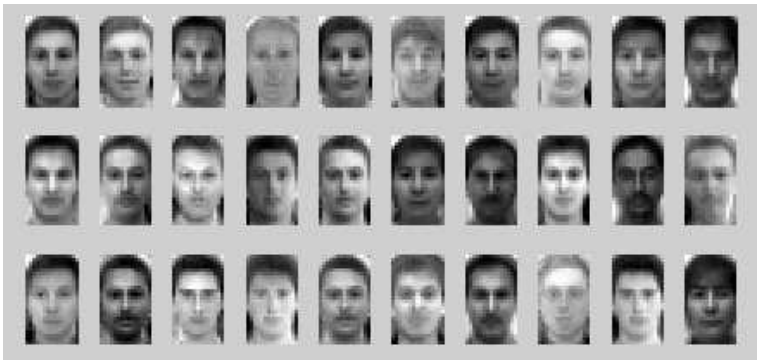


Figure 3.8: Synthetic faces obtained by running the learned mixture distribution generatively.

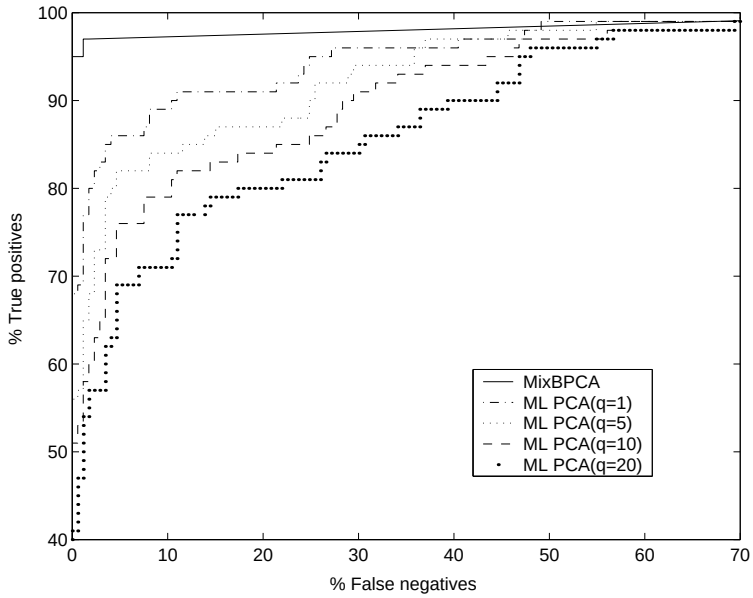


Figure 3.9: ROC curves for classifying images as faces versus non-faces for the Bayesian PCA mixture model (MixBPCA), together with the corresponding results for a maximum likelihood PCA model (ML PCA) with various values for q the number of retained principal components. This highlights the significant improvement in classification performance in going from a linear PCA model to a non-linear mixture model. In the maximum likelihood model it is necessary to repeat inference with a range of values of q; the ability of the Bayesian approach to determine latent space dimensionality avoids this problem.

classification performance. This result also highlights the fact that the Bayesian approach avoids the need to set parameters such as q by exhaustive exploration.

3.4.4 Handwritten digits data set

As a second application of the framework, a model was constructed of the manifolds of images of hand-written digits. A data set taken from the CEDAR U.S. Postal Service database was used, comprising 11,000 images (equally distributed over the ten digits) each of which is 8×8 greyscale, together with a similar independent test set of 2711 images. Figure 3.10 shows synthetic images generated from a Bayesian PCA mixture model fitted to the training set.

The learned model achieved 4.83% error rate on the test set. For comparison we note that Tipping and Bishop [1999a] used the same training and test sets with a maximum likelihood mixture of probabilistic principal component analysers. The training set in this case was itself subdivided into training plus validation sets. For each of the ten digit models considerable computational effort was expended in finding the optimum values of M (the number of components in the mixture) and q (the dimensionality of the latent spaces) by evaluation of performance on the validation set. This approach achieved 4.61% error rate on the test set, which is comparable with the result obtained from the single run of the Bayesian PCA mixture model.

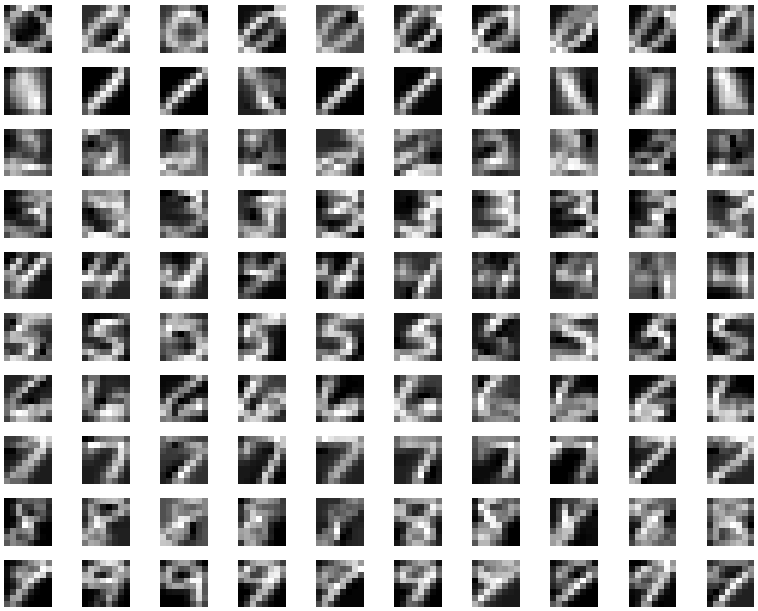


Figure 3.10: Digits synthesised from each of the ten trained Bayesian PCA mixture model by running the models generatively.

3.4.5 Image compression

As a final application, the model was used to perform image compression by block transform image coding. A 480×640 test image, shown in Figure 3.11a, was divided into 4800 8×8 blocks. The blocks were formatted into 64-dimensional vectors. Half of these vectors, corresponding to the left hand side of the image, were used to train the model. The remaining vectors were used to test the trained model. For comparison, a standard PCA model and a vector quantised PCA (VQPCA) model were also tested. Vector quantised PCA is described in Kambhatla and Leen [1997] and is summarised in Algorithm 3.1.

Algorithm 3.1 Vector Quantised PCA (VQPCA)

1. Select cluster centres at random from points in the data set and assign all data points to the nearest cluster centre.
2. Set matrices W_i to the principal axes of the covariance matrix of cluster i.
3. Assign data points to the cluster that best reconstructs them and set new cluster centres to the mean of points assigned to that cluster.
4. Repeat from step 2 to until the cluster allocations are constant.

In each method, the reduced dimensionality representation of each block was quantised to give a final bit rate of 0.5 bits per pixel (which gives a compression ratio of 16 to 1). This includes the cost of coding the component label in the mixture model. The bits were allocated equally to each transform variable.

Figure 3.11b shows an enlarged part of original image and the corresponding reconstructions for standard PCA, mixture of Bayesian PCA and VQPCA along with the reconstruction errors of each. The mixture of Bayesian PCA model gives the smallest reconstruction error and has the fewest reconstruction artefacts (for example, examine the curve of the underside of the bridge in the reconstructed images).

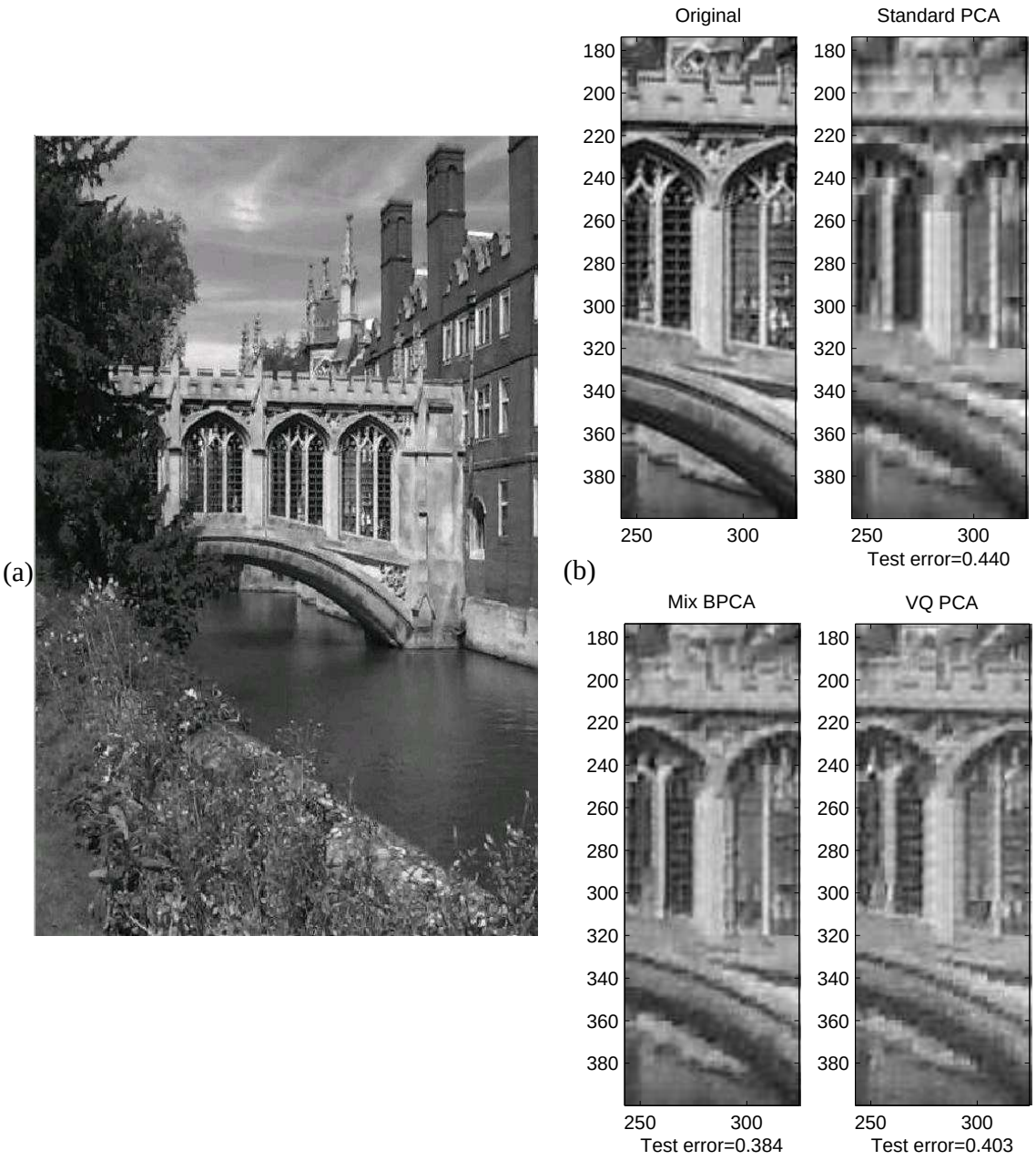


Figure 3.11: (a) The original image used to test the compression algorithm. (b) Detail of the original and reconstructed images for various compression methods. The normalised test errors shown are the reconstruction errors for the whole image.

3.5 Discussion

In this chapter, I have introduced a fully probabilistic approach to modelling manifolds of images, in which an appropriate model complexity, as well as the manifold's intrinsic dimensionality, can be inferred from the data. The use of Variational Message Passing has allowed inference in this model to be performed automatically. As a comparison, the variational update equations have also been derived by hand, showing the considerable additional time and effort required to do this. Results on data sets of images of faces and handwritten digits and on an image compression example demonstrate the practical feasibility of the system as well as improved performance compared to previous approaches.

An important advantage of the system is that there are no significant adjustable parameters in the model which need to be set by the user. The model complexity is inferred from the data and, since no model optimisation is required, the model can be trained just once on the data set, avoiding the need for computationally expensive cross-validation.

Chapter 4

Application: Microarray Image Analysis

In image analysis problems, there is uncertainty about the state of the system being imaged due to the inherent ambiguities of the imaging process. We can use a probabilistic model to represent the imaging process, giving a joint probability distribution over the image and the state of the system being imaged. Determining this hidden state from the image is therefore another example of Bayesian inference. Once again, the complexity of the model tends to render exact inference intractable and therefore approximate solutions can be obtained, for example, using variational inference.

In this chapter, a particular image analysis problem, the problem of analysing scanned images of DNA microarrays, is investigated using this Bayesian approach. Variational inference is carried out using an extended form of Variational Message Passing which uses importance sampling to handle a conditional distribution that is not in the exponential family. Whilst such *variational importance sampling* was developed by Lawrence et al. [2002], its inclusion within the message passing framework is my own work.

4.1 DNA Microarrays

DNA microarray technology allows rapid identification of the level of expression of thousands of genes in a tissue or organism, all on a single slide.

Gene expression microarrays are typically produced by transferring cDNA[1] or oligonucleotides[2] in high salt solutions onto chemically modified glass microscope slides using a contact-printing instrument [Eisen and Brown 1999; Hegde et al. 2000]. These cDNA *probes* are exposed to target cDNA which has been reverse-transcribed and labelled with a fluorescent dye. The target cDNA then binds with just those probes that have complementary

[1] Complementary DNA (cDNA) has a base sequence which is the *complement* of an original DNA sequence. The complement of a sequence is one with each base replaced by its complementary base: A by T, C by G, and vice versa.

[2] An oligonucleotide is a short stretch (usually 2-50 bases) of single-stranded DNA.

base sequences, in a process known as *hybridisation*. The resultant hybridisation patterns are detected by fluorescent imaging of the slide. The image must then be processed to identify the presence and levels of gene expression in the target.

There are a number of existing software tools for analysing microarray images and extracting the gene expression data. For example ScanAlyze[3] allows a user to mark by hand the size and shape of each spot in the image. It would be desirable to automate this process, as this would both reduce the time taken to analyse the images and improve the reliability of the resultant gene expression data. Although there have been several attempts at automating this process using semi-empirical approaches, such as Dapple [Buhler et al. 2000] and Spot [Dudoit et al. 2000], these tools tend to be tuned for a particular type of image. Adopting any one such tool is often a long process of trial and error as the interplay of effects of algorithm parameters is difficult to anticipate.

In this chapter, I present a system for automatic analysis of microarray images using Bayesian methodology and variational inference. Whilst the system does have a small number of parameters, these directly model our knowledge of the images to be analysed and so any necessary initialisation is straightforward. Most image parameters are automatically inferred from the image or taken from the configuration file of the contact-printing instrument that was used to print the microarray.

In Section 4.2, I describe the experimental setup used to obtain test images. A probabilistic model of microarray images is developed in Section 4.3 and a method of performing inference in this model using an extended form of Variational Message Passing is described, with results on real images, in Sections 4.4 and 4.5. A solution to the problem of locating grids of spots is given in Section 4.6 and the entire system discussed in Section 4.7. Finally, in Section 4.8, the question of how to analyse the resultant gene expression data is addressed and a brief example given which uses Variational Message Passing.

4.2 Microarray Images

A typical microarray slide consists of a rectangular array of sub-grids, each sub-grid printed by one pin of the contact-printer. A sub-grid consists of an array of spots, each spot containing a single cDNA probe. The hybridised arrays are imaged using a *scanner*, such as a laser scanning confocal microscope, and the output stored as 16-bit image files. Where a number of dyes are used, one image is produced for each.

Microarray images typically have significant background noise and can also have other noise artefacts, some of which are introduced during the scanning process. For example, when using a wide-field CCD scanner, dust particles on the slide can cause scatter flares (large, bright circular artefacts) which may obscure one or more of the spots. Alternatively, when using a laser scanner, reflections within the optical subsystem of the scanner can result in the introduction of additional false spot images. In addition, the spots themselves vary in

[3]ScanAlyze software is available from `http://rana.lbl.gov/EisenSoftware.htm`.

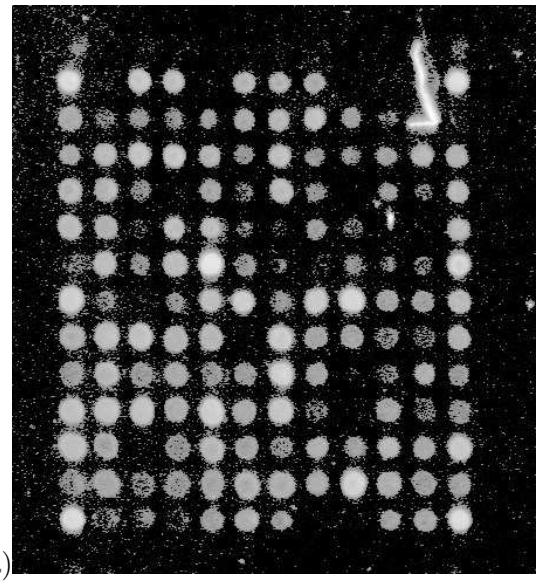
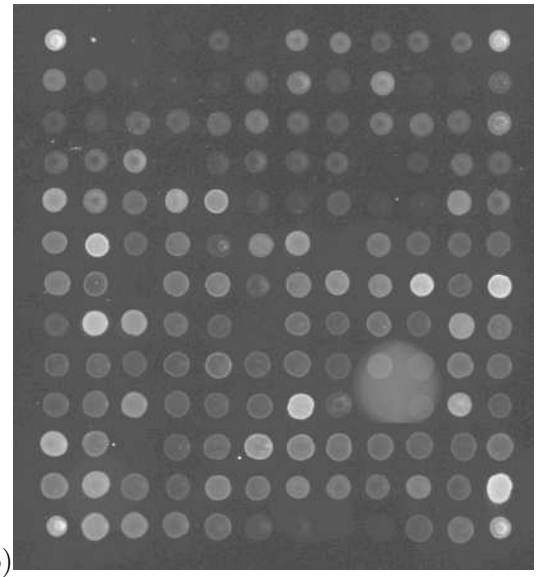
(a)  (b) 

Figure 4.1: (a) Section of an image from a laser scanner showing false spots (the two very faint spots above each top corner of the grid), noise artefacts and a high level of background noise. (b) Section of an image from a wide-field scanner showing the reduced background noise which is an advantage of this scanner. Unfortunately, this type of scanner also causes scatter flares – the one seen here obscures four of the spots. Note the variation in spot shape and size in each image. In both images, the intensities have been mapped so as to make dim spots visible.

size and shape, even within a single sub-grid. When there are sets of images for different dyes on the same slide, some noise artefacts will be common to all of the dye images, whilst there are often systematic variations in background noise from dye to dye.

Figure 4.1 shows two sub-grids extracted from actual microarray images which include examples of many of these noise artefacts.

4.2.1 Experimental methodology

The microarray images used throughout this chapter were created in the Ashburner Laboratory at Cambridge University Genetics Department by Gos Micklem, David Kreil et al., who have kindly made them available for this research. The variations between microarray images are partly due to the different experimental methodologies and equipment used in different laboratories. For this reason, the methodology used to create the test images for this system will now be described in some detail.

The microarrays used were printed using the BioRobotics MicroGrid II Total Array System[4] contact-printing instrument and 48 BioRobotics MicroSpot 2500 split-pins. Drosophila Gene Collection PCR-amplified cDNA inserts from the Berkeley Drosophila Genome Project[5] were printed on in-house coated Poly-L-Lysine slides. To minimise variations due to environ-

[4] `http://www.biorobotics.co.uk/`
[5] `http://www.fruitfly.org/`

mental effects, a BioRobotics Humidity Control Unit was used and printing took place in a temperature controlled room.

Printed slides were heated, and cross-linked using UV light. Unspecific binding of DNA to the slides was blocked using a solution of succinic anhydride in 1-methyl-2-pyrrolidinone and boric acid. Double stranded DNA was denatured by further heat treatment.

Tissue from the fruit fly Drosphila was homogenised in TRIzol, and RNA was extracted and precipitated with chloroform and isopropanol. Samples were then directly labelled by reverse transcription incorporating nucleotides with a covalently bound dye (either Cy3-dCTP or Cy5-dCTP). Samples labelled with different dyes were then jointly hybridised on a microarray slide using a Genomic Solutions GeneTAC hybridisation station.

To provide a variety of test images, the slides were scanned with one of two different scanners: a GenomicSolutions GeneTAC LS-IV confocal laser scanner or an Applied Precision ArrayWoRx wide-field CCD scanner.

4.3 A Probabilistic Model for Microarray Images

As in any inference problem, we start by defining our probabilistic model. The observed variables within this model are the grey levels of the image pixels. The model must also include latent variables representing the information that we are trying to extract from the image: the shape and location of the individual spots. Any assumptions that we make about the imaging process will be explicitly encoded in the model. The model therefore defines, by its assumptions, what types of images are suitable for analysis using this system (i.e. those where these assumptions hold).

Rather than working with the entire slide image, we assume that we have extracted a section of the image which contains a single sub-grid, like those of Figure 4.1. The number of rows and columns in the sub-grid can be found from the configuration file for the array printer. This file also tells us the approximate size of each spot and their approximate separations. Finally, we assume, at this stage, that we have a rough estimate of the location of each spot. This could be provided through user input (such as by specifying the location of three corner spots and interpolating using a regular grid) or by automatic means, as will be discussed later.

4.3.1 Latent variables and their prior distributions

We now define our latent variables. The actual location of each spot will be represented by a two-dimensional vector variable $\mathbf{c} = (c_x, c_y)$, which is the location in pixels specified *relative to* the initial estimated location. The spot is assumed to be an axis-aligned ellipse and so the shape is encoded by $\mathbf{r} = (r_x, r_y)$ where r_x is the radius in the x-direction and r_y is the radius in the y-direction. The assumption that spots are axis-aligned ellipses is a good assumption in the vast majority of cases where the spots are nearly circular and distortions are due to slight differences in scanning resolution on the x and y axes. However, extending the model

to allow for rotated ellipses or other shapes is also possible provided one is willing to accept the additional computation required to learn the extra parameters.

The prior distribution over the position vector $\mathbf{c}$ is defined to be a Gaussian distribution

$$P(\mathbf{c} \,|\, \boldsymbol{\mu}_c, \gamma_c) = \mathcal{N}(\mathbf{c} \,|\, \boldsymbol{\mu}_c, \boldsymbol{\gamma}_c^{-1}), \tag{4.1}$$

where $\boldsymbol{\gamma}_c$ is a diagonal inverse covariance matrix. The parameters $\boldsymbol{\mu}_c$ and $\boldsymbol{\gamma}_c$ are governed by conjugate hyper-priors

$$\begin{aligned} P(\boldsymbol{\mu}_c) &= \mathcal{N}(\boldsymbol{\mu}_c \,|\, \mathbf{m}_c, \beta_c^{-1}\mathbf{I}) & (4.2)\\ P(\boldsymbol{\gamma}_c) &= \text{Gamma}(\gamma_{c00} \,|\, a_c, b_c)\text{Gamma}(\gamma_{c11} \,|\, a_c, b_c). & (4.3) \end{aligned}$$

The parameter $\mathbf{m}_c$ is set to the supplied rough location of the spot and the precision β_c is set to give a corresponding standard deviation of one quarter of the distance between the centres of adjacent spots. The parameters a_c and b_c were set to 0.05 and 0.1 respectively.

We define a similar prior distribution over the size vector $\mathbf{r}$ with parameters $\{\boldsymbol{\mu}_r, \boldsymbol{\gamma}_r\}$ and hyper-parameters $\{\mathbf{m}_r, \beta_r, a_r, b_r\}$:

$$\begin{aligned} P(\boldsymbol{\mu}_r) &= \mathcal{N}(\boldsymbol{\mu}_r \,|\, \mathbf{m}_r, \beta_r^{-1}\mathbf{I}) & (4.4)\\ P(\boldsymbol{\gamma}_r) &= \text{Gamma}(\gamma_{r00} \,|\, a_r, b_r)\text{Gamma}(\gamma_{r11} \,|\, a_r, b_r). & (4.5) \end{aligned}$$

In this case, $\mathbf{m}_r$ is set to the expected radius determined from the configuration file; all other parameters are the same as for the centre prior. To distinguish between the location and size variable for different spots, we define the jth spot to have location $\mathbf{c}_j$ and size $\mathbf{r}_j$.

4.3.2 The likelihood function

The likelihood function defines the probability of a particular image given a particular setting of all the latent variables $\{\mathbf{c}_j, \mathbf{r}_j\}_{j=1}^{J}$. In order to simplify the inference problem, we separate this likelihood function into a product of functions each corresponding to a small area of the image containing a spot. Thus, the likelihood function for the jth spot gives the probability of the rectangular subimage I_j centred on the approximate location of the jth spot given the jth set of parameters $\{\mathbf{c}_j, \mathbf{r}_j\}$. This independence assumption is valid provided that the amount of 'wobble' on the array printer is not so great that the spots actually overlap – if this is not the case and there are overlapping spots, there would be great difficulty determining their individual intensities anyway. In practice, this assumption holds as the spots are typically well separated. In the future, higher density arrayers may try to fit more spots on a single slide and image analysis may then require a model which does not make this independence assumption.

It follows that, within any given subimage I_j, we expect to find a single spot. A setting of the parameters $\{\mathbf{c}_j, \mathbf{r}_j\}$ partitions the pixels of I_j into two disjoint sets: the *spot* pixels S

which lie inside an ellipse with centre $\mathbf{c}_j$ and radii $\mathbf{r}_j$ and the remaining *background* pixels B which are outside the ellipse. In defining our likelihood function, we now make the further assumption that the probability distribution over the intensity of a particular pixel depends only on whether it is in S or B. This pixel independence assumption allows us to write the likelihood function as

$$P(I_j \mid \mathbf{c}_j, \mathbf{r}_j) = \prod_{b \in B} P_B(I_b) \prod_{s \in S} P_S(I_s), \tag{4.6}$$

where $P_B(I_b)$ is the likelihood function for background pixel intensity and $P_S(I_s)$ is the likelihood function for spot pixel intensity.

This raises the question of how to define P_B and P_S. One approach would be to use the rough spot positions to divide the entire sub-grid image into (approximately) spot and background pixels and to use the statistics of these two sets of pixels to define P_B and P_S. The difficulty with this approach is that the distribution over pixel intensities varies from spot to spot and, in most images, varying background noise means that the distribution over background pixel intensities also changes significantly, even within a single sub-grid. Hence, if we were to fix P_B and P_S for the entire sub-grid, we would suffer from problems like background noise in one part of the image masking dim spots in other areas of the image, even if there were little background noise there.

We can avoid these problems by inferring P_B and P_S separately for each spot. To achieve this, we quantise the pixel intensities into one of K bins. Each of P_B and P_S is then a discrete distribution which defines the probability of a pixel intensity being in each bin. The parameters of P_S are the K probabilities $\{p_1, p_2, \ldots, p_K\}$ where each p_i is the probability that a spot pixel will lie in the ith intensity bin. Similarly, the parameters of P_B are $\{q_1, q_2, \ldots, q_K\}$. The likelihood function may now be rewritten as

$$P(I_j \mid \mathbf{c}_j, \mathbf{r}_j, \{p_k\}, \{q_k\}) = \prod_{k=1}^{K} p_k^{n_k} \prod_{j=1}^{K} q_k^{m_k}, \tag{4.7}$$

where n_k is the number of pixels in S that lie in the kth bin and m_k is the number of pixels in B that lie in the kth bin. We then define a Dirichlet prior over these parameters so that

$$P\left(\{p_k\}_{k=1}^K\right) = \text{Dirichlet}\left(\{p_k\}_{k=1}^K \mid \{u_k\}_{k=1}^K\right) \tag{4.8}$$

$$P\left(\{q_k\}_{k=1}^K\right) = \text{Dirichlet}\left(\{q_k\}_{k=1}^K \mid \{v_k\}_{k=1}^K\right). \tag{4.9}$$

Consider just the spot pixels S. We can now marginalise out $\{p_k\}$ and write the likelihood

in terms of the Dirichlet parameters $\{u_k\}$ only,

$$P(S \mid \{u_k\}) = \int \left(\prod_{k=1}^{K} p_k^{n_k} \right) \mathrm{Dir}\left(\{p_k\} \mid \{u_k\}\right) dp_1 \ldots p_K \tag{4.10}$$

$$= \int \mathrm{Dir}\left(\{p_k\} \mid \{u_k + n_k\}\right) dp_1 \ldots p_K \; \frac{\prod_k \Gamma(u_k + n_k)}{\Gamma(\sum_k u_k + n_k)} \; \frac{\Gamma(\sum_k u_k)}{\prod_k \Gamma(u_k)}$$

$$= \frac{\prod_k \Gamma(u_k + n_k)}{\Gamma(\sum_k u_k + n_k)} \; \frac{\Gamma(\sum_k u_k)}{\prod_k \Gamma(u_k)}, \tag{4.11}$$

where $\Gamma()$ is the gamma function. A similar marginalisation for the background pixels B gives us our final likelihood function

$$P(I_j \mid \mathbf{c}_j, \mathbf{r}_j, \{u_k\}, \{v_k\}) = \left[\frac{\prod_k \Gamma(u_k + n_k)}{\Gamma(\sum_k u_k + n_k)} \; \frac{\Gamma(\sum_k u_k)}{\prod_k \Gamma(u_k)} \right] \times \left[\frac{\prod_k \Gamma(v_k + m_k)}{\Gamma(\sum_k v_k + m_k)} \; \frac{\Gamma(\sum_k v_k)}{\prod_k \Gamma(v_k)} \right]. \tag{4.12}$$

The prior parameters $\{u_k\}$ and $\{v_k\}$ can be thought of as pseudo-counts and can be set to be proportional to histograms of spot and background pixels over the entire image plus a constant value of 1 (to allow for previously unobserved intensities). The sums of these pseudo-counts dictate the strength of the Dirichlet priors. Good results were achieved when the sum of pseudo-counts was set to be equal to the number of pixels in each subimage and K was set to 300. Our entire probabilistic model can now be expressed as a Bayesian network, as shown in Figure 4.2.

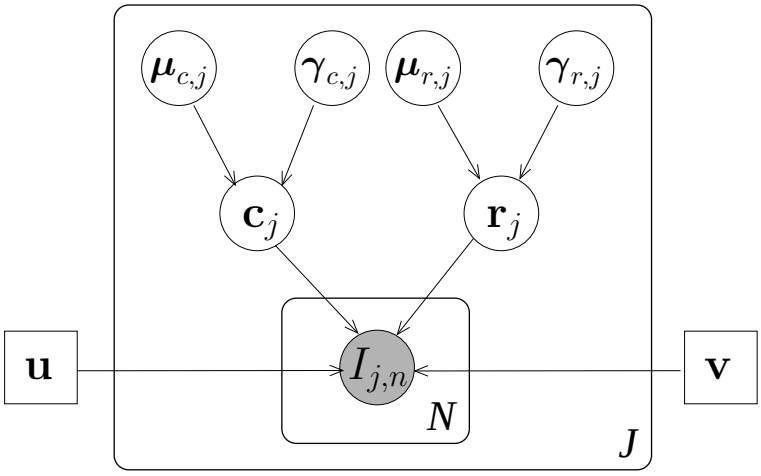


Figure 4.2: The Bayesian network for a probabilistic model of microarray sub-grid images. The sub-grid contains J spots each of which has a centre $\mathbf{c}_j$ and radii $\mathbf{r}_j$ which we wish to infer. The subimage I_j contains the jth spot and consists of N pixel intensity values, quantised to one of K states. Given a spot location and size, these pixels are divided into two disjoint sets: spot pixels and background pixels. The discrete distributions over pixel intensities for each set have been marginalised out and so are not shown, but are instead governed by Dirichlet priors, whose parameters $\mathbf{u}$, $\mathbf{v}$ are common to all spots. In this model, these parameters are fixed to constant values, indicated by the use of square nodes in the graph.

4.4 Variational Message Passing with Importance Sampling

The Bayesian network defined in the previous section does not allow for the direct application of Variational Message Passing in order to find the posterior over spot sizes and positions. The problem arises due to the form of the conditional $P(I_{j,n} \mid \mathbf{c}_j, \mathbf{r}_j)$. This function is nonlinear and not an exponential family distribution. When using variational message passing, this prevents us from finding an analytical form for the child-to-parent messages from $I_{j,n}$ to $\mathbf{c}_j$ and to $\mathbf{r}_j$, which in turn prevents us from finding the updated variational posteriors $Q(\mathbf{c}_j)$ and $Q(\mathbf{r}_j)$. Instead, we turn to sampling methods to approximate the posterior variational distribution of $Q(\mathbf{c}_j)$ and $Q(\mathbf{r}_j)$, whilst continuing to use standard VMP for the rest of the graph. Effectively, the sampling method will be used as a subroutine within the VMP algorithm. A range of sampling methods are available; for simplicity, we follow Lawrence et al. [2002] and use *importance sampling*.

Importance sampling is a technique which allows the calculation of approximate expectations under a posterior distribution $P(\mathbf{x})$. For example, suppose we wish to find the expectation of a function $f(\mathbf{x})$, we would aim to evaluate,

$$\langle f(\mathbf{x}) \rangle_P = \int f(\mathbf{x}) P(\mathbf{x}) \, d\mathbf{x}. \tag{4.13}$$

This integral is intractable and so we introduce a *proposal distribution* $q(\mathbf{x})$ (not to be confused with a variational distribution $Q(\mathbf{x})$),

$$\langle f(\mathbf{x}) \rangle_P = \int f(\mathbf{x}) \frac{P(\mathbf{x})}{q(\mathbf{x})} q(\mathbf{x}) \, d\mathbf{x}. \tag{4.14}$$

The proposal distribution is selected so that it is easy to sample from and we use L samples from it to approximate our desired expectation:

$$\langle f(\mathbf{x}) \rangle_P \approx \frac{1}{L} \sum_{i=1}^{L} f(\mathbf{x}_i) \frac{P(\mathbf{x}_i)}{q(\mathbf{x}_i)} \tag{4.15}$$

where the ratios $P(\mathbf{x}_i)/q(\mathbf{x}_i)$ used to weight each sample are known as *importance weights*.

Importance sampling only allows us to estimate the expectations of functions under a posterior P, rather than generate samples from $P(\mathbf{x})$. The calculation of expectations is sufficient in this case because we only need to compute expectations of natural statistic vectors (which are just functions of individual variables).

When used in conjunction with variational message passing, it makes sense to perform all importance sampling within one node Z: the node whose conditional distribution is non-exponential family (in this case, Z corresponds to the $I_{j,n}$ node). The posterior distribution we aim to approximate is therefore the joint variational distribution over all parents of Z (which in this case is $Q(\mathbf{c}_j)Q(\mathbf{r}_j)$). We make the assumption that Z is observed, as is the case in the microarray model.

Now we must choose a proposal distribution which is as similar as possible to our posterior. Consider if we ignore the effect of the Z node from the calculation of the variational posterior of one of its parents X_j. The parameters of such an (incorrect) posterior can then be found analytically using

$$\boldsymbol{\phi}_j^* = \boldsymbol{\theta}_j \left(\{ m_{X_i \to X_j} \}_{i \in \mathrm{pa}_j} \right) + \sum_{k \in \mathrm{ch}_j \setminus Z} m_{X_k \to X_j} \tag{4.16}$$

where we are ignoring the message from Z. We shall call this posterior $R_j(X_j)$. Whilst this posterior is clearly not equal to Q_j^*, it will be similar to it and so usable as a proposal distribution for importance sampling. We therefore define the message from the parent X_j of an importance sampling node Z to be

$$m_{X_j \to Z} = \langle \mathbf{u}_j \rangle_{R_j} \tag{4.17}$$

where we are sending a natural statistic vector rather than a parameter vector for consistency with other parent-to-child messages only (as either vector is sufficient to parameterise R_j).

At Z, we define a proposal distribution over the parents which is the product of the R distributions for each parent. We then draw S samples $\{\mathrm{pa}_Z{}^{(s)}\}_{s=1}^S$ from this proposal distribution, which is a straightforward operation as we can sample for each parent variable independently. Following the importance sampling methodology, we find the importance weight of each sample from the ratio of the variational distribution to the proposal distribution evaluated for that sample

$$w_s = \frac{1}{K} \frac{Q(\mathrm{pa}_Z{}^{(s)})}{\prod_{j \in \mathrm{pa}_Z} R_j(X_j^{(s)})} \tag{4.18}$$

$$= \frac{1}{K} P(Z \,|\, \mathrm{pa}_Z{}^{(s)}) \tag{4.19}$$

where the normalising constant K is chosen to be $\sum_{s=1}^S w_s$, so that the sum of all the importance weights is one. The fact that these weights are calculated from Z and the samples means that the calculation can be performed locally. All that remains is to use these weights to estimate the required expectations of natural statistic vectors for each parent

$$\langle \mathbf{u}_j(X_j) \rangle_Q \approx \sum_{s=1}^S w_s \mathbf{u}(X_j^{(s)}) \tag{4.20}$$

and to send these as the message from Z to that parent. The parent then adopts this message as the new expectation of its natural statistic vector. The corresponding distribution can be thought of as an exponential family approximation to the variational posterior.

One problem with importance sampling is that the sampling estimate can be dominated by a few samples with very high weights. This occurs when there is a mismatch between the proposal distribution R and the distribution of interest (in this case, the variational posterior

Q). The claim here is that the proposal distribution R is adaptive, adjusting in line with observed data, thereby improving the match between R and Q.

The quality of the samples obtained during importance sampling can be summarised by $S_{\text{eff}} = \frac{1}{\sum_{s=1}^{S} w_s^2}$ where $1 \leq S_{\text{eff}} \leq S$ which is known as the *effective number* of samples. This quantity is used to determine the quality of the sampling approximation and also as a convergence criterion. The contribution of Z to the lower bound $\mathcal{L}$ can also be estimated using

$$\mathcal{L}_Z \approx \sum_{s=1}^{S} w_s P(Z \,|\, \text{pa}_Z{}^{(s)}) = \frac{K}{S_{\text{eff}}}. \tag{4.21}$$

4.5 Inference in the Microarray Image Model

The hybrid variational/sampling algorithm described above was applied to the microarray image model, using $S = 100$ samples. The order of the updates for each spot was that the sampling node $I_{j,n}$ was updated first, followed by the remaining nodes. Due to the high computational expense of updating the sampling node, this node was only updated one iteration in ten.

As only a (noisy) estimate of the lower bound was available, it could not be used in the normal way as a convergence criterion. Instead, the algorithm was deemed to have converged when S_{eff} became greater than $S/4$ or a fixed maximum number of iterations was reached.

4.5.1 Handling missing and obscured spots

Microarray images frequently have gaps where no spots appear, corresponding to cDNA probes where little or no hybridisation has occurred. There are also occasions where noise artefacts are sufficient to obscure or heavily mask the spot. The sub-grids of Figure 4.1 showed examples of each of these situations. In both cases, the image model used above provides a poor model of the resultant spot image; its assumption of an elliptical boundary between two areas with differing intensity distributions simply does not hold. In the case of missing spots, the model assumes a spot exists with the same intensity as the background which leads to significant uncertainty in the inferred spot location and size. In the case of obscured spots, the inferred spot size and position can be incorrect; indeed, it may not be possible to determine the actual position of such spots from the image.

For an image analysis algorithm to be useful, it must identify these two special cases and flag the spots so as to avoid outputting false or inaccurate data. The identification of these cases can be achieved by introducing new image models for each case and performing model comparison. The image model for a missing spot is simply an image whose pixels are all background pixels. As there are no latent variables in this model, we can write the image probability directly as

$$P(I_j \,|\, \mathcal{H}_1) = \left[\frac{\prod_k \Gamma(v_k + m_k)}{\Gamma(\sum_k v_k + m_k)} \; \frac{\Gamma(\sum_k v_k)}{\prod_k \Gamma(v_k)}\right] \tag{4.22}$$

where the Dirichlet parameters $\{v_k\}$ are as defined for the standard model and m_k is the number of pixels in the subimage I_j whose intensities lie in the kth intensity bin.

We can define a similar model for obscured spots. When a spot is badly obscured by noise, then the image will contain non-background pixels due to this noise as well as due to the spot. These pixels will not lie in an elliptical region. In fact, the shape of the region will be unpredictable as we cannot make assumptions about what form the noise may take. Instead, we assume that any pixel is equally likely to be background or non-background and so its intensity distribution is an equal mixture of the background and foreground intensity distributions. The image probability under this model is therefore similar to that of the missing spot model except that the Dirichlet parameters $\{w_k\}$ are set to be the average of $\{u_k\}$ and $\{v_k\}$,

$$P(I_j \,|\, \mathcal{H}_2) = \left[\frac{\prod_k \Gamma(w_k + m_k)}{\Gamma(\sum_k w_k + m_k)} \, \frac{\Gamma(\sum_k w_k)}{\prod_k \Gamma(w_k)}\right]. \tag{4.23}$$

If we refer to the image model described in Section 4.3 as $\mathcal{H}_0$, the approximate evidence for this model can be written as

$$P(I_j \,|\, \mathcal{H}_0) \approx \exp(\mathcal{L}(Q)). \tag{4.24}$$

If we assume that each subimage I_j was generated from one of these three models, then the posterior probability for the ith model is

$$P(\mathcal{H}_i \,|\, I_j) = \frac{P(I_j \,|\, \mathcal{H}_i)P(\mathcal{H}_i)}{\sum_{k=0}^{2} P(I_j \,|\, \mathcal{H}_k)P(\mathcal{H}_k)}. \tag{4.25}$$

For simplicity, $P(\mathcal{H}_i)$ was chosen to be uniform and each spot was flagged as NORMAL, MISSING or BAD (i.e. obscured) based on the model that had the highest posterior probability. The uncertainty in this flag state is not currently maintained as it is difficult for further processing stages to make use of it. Certainly, no existing tools are capable of maintaining many hypotheses about spot states during further processing.

4.5.2 Updating the prior parameters of the model

Because each sub-grid is printed by one pin of the arrayer, it is reasonable to assume that all the spots in a particular sub-grid are of a similar size and have similar deviations in position from a regular array. This assumption could be encoded in our model by the addition of shared hyper-hyper-priors over the parameters $\mathbf{m}_c$, β_c, $\mathbf{m}_r$, β_r and suchlike. Posterior distributions over these parameters could then be inferred using variational message passing. However, the addition of these latent variable nodes in the graph would prevent inference being carried out separately for each spot. It was decided, for the sake of simplicity, just to update the parameters $\mathbf{m}_c$, β_c, $\mathbf{m}_r$, β_r from the results of one pass of the algorithm and then reapply using these new parameter settings. These new prior parameters provide a much stronger learned prior and give an algorithm that is more robust to noise than one based on any fixed setting of these parameters.

4.5.3 Determining the spot intensities

The purpose of image analysis is not to find spot locations and sizes but to determine their intensities. The intensity E_j of the jth spot will be a function of the spot parameters $\boldsymbol{\theta}_j = \{\mathbf{c}_j, \mathbf{r}_j\}$ and the subimage I_j

$$E_j = f(I_j, \boldsymbol{\theta}_j). \tag{4.26}$$

For each spot, there is uncertainty in the parameters $\boldsymbol{\theta}_j$. We cannot therefore solve Equation 4.26 directly, but can only compute the expectation of E_j under the approximate posterior distribution over $\boldsymbol{\theta}_j$,

$$\langle E_j \rangle_{Q(\boldsymbol{\theta}_j \,|\, I_j)} = \int f(I_j, \boldsymbol{\theta}_j) Q(\boldsymbol{\theta}_j \,|\, I_j)\, d\boldsymbol{\theta}_j. \tag{4.27}$$

Our estimate of the posterior distribution over $\boldsymbol{\theta}_j$ is available as a set of samples from this distribution $\{\boldsymbol{\theta}_j^{(1)}, \boldsymbol{\theta}_j^{(2)}, \ldots, \boldsymbol{\theta}_j^{(S)}\}$ with corresponding importance weights $\{w_1, w_2, \ldots, w_S\}$. Importance sampling dictates that the above expectation is approximated by

$$\langle E \rangle_{P(\theta_j \,|\, I)} \approx \sum_{i=1}^{S} w_i\, f(I_j, \theta_j^{(i)}). \tag{4.28}$$

The function f is typically chosen to be the mean or median intensity of all the spot pixels. To give an indication of the accuracy of this intensity value, its variance can be found using $\mathrm{var}(E_j) = \langle E_j^2 \rangle - \langle E_j \rangle^2$.

4.5.4 Spot-finding results

The results of the microarray image analysis algorithm on two test sub-grid images are shown in Figure 4.3. The ellipses drawn over the image show the expected spot size and shape under the approximate posterior distribution given by the inference algorithm. The ellipses are coloured according to the spot states: NORMAL spots are green, BAD spots are red and MISSING spots are yellow. Spots which were found to be missing are marked using ellipses which are the average shape and size of all non-missing spots in the sub-grid.

The two images have very different noise characteristics. In particular, the right hand sub-grid has a high level of background noise including scatter flares. Nonetheless, the algorithm has located the spots with good accuracy given the level of noise.

4.6 Automatic Sub-grid Location

The spot-finding algorithm described above requires a set of approximate spot positions as a starting point. These can be obtained, for example, by requiring the user to locate a regular array of circles over each entire sub-grid. Whilst this is clearly much quicker than locating each spot individually, it is still time-consuming given that each slide image typically contains tens of sub-grids. If our goal is to automate the analysis of these images, then we should certainly aim to be able to find this approximate initialisation automatically.

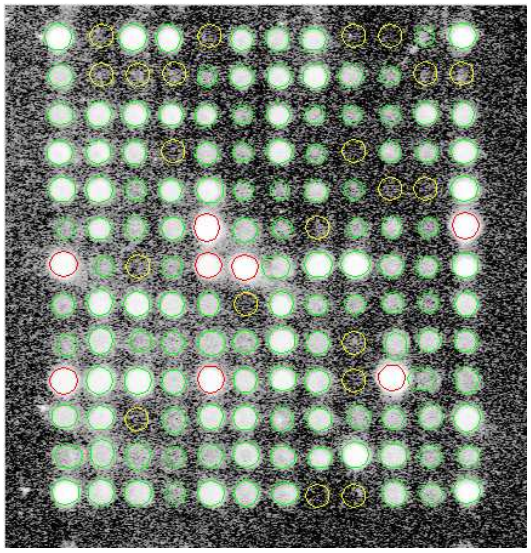
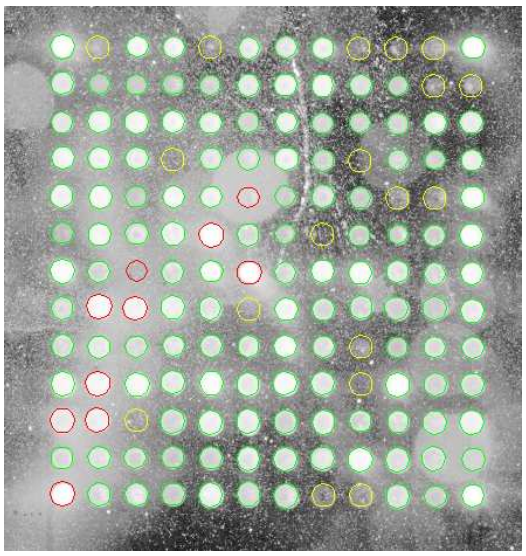
(a)  (b) 

Figure 4.3: Results of the microarray image analysis algorithm on two test sub-images with different noise characteristics. The ellipses show the expected spot size and position under the approximate posterior distribution. The ellipses are coloured green for NORMAL spots, red for BAD (obscured) spots and yellow for MISSING spots. The results show that the algorithm is robust even to high levels of background noise.

As before, let as assume that we have an image I that contains only one entire sub-grid of spots (there may be other partial sub-grids). The printing and scanning process will introduce distortions in the image so that each sub-grid is not an exactly axis-aligned, rectangular array of spots. Indeed, the array may be translated, scaled, rotated, sheared or distorted in a non-linear fashion. However, we shall ignore non-linear effects and assume that the distortion can be modelled by an affine (linear) transform well enough to give a good approximation of spot locations. This assumptions holds in the test images used because the non-linear distortions are not significant over the scale of individual sub-grids. The aim of automatic sub-grid location will therefore be to learn the affine transform which gives the best approximation of spot locations.

4.6.1 The sub-grid transform and its prior

The affine transform $\mathbf{T}$ gives a mapping from the physical slide co-ordinates (x, y) in millimetres to image co-ordinates (u, v) in pixels, defined as follows:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} m_{00} & m_{01} & c_x \\ m_{10} & m_{11} & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \stackrel{\text{def}}{=} \mathbf{T} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{4.29}$$

The vector $\mathbf{c} = (c_x, c_y)$ contains the image co-ordinates of the centre of the sub-grid. The matrix $\mathbf{M} = (m_{00}\ m_{01}; m_{10}\ m_{11})$ represents any rotation, scale or skew introduced by the printing and scanning process.

The inference task is to find the posterior distribution over the latent variable $\mathbf{T}$, so we must define a prior over $\mathbf{T}$. We assume independence between all the parameters

$$\begin{aligned} P(\mathbf{T} \,|\, \mathcal{H}) &= P(\mathbf{c} \,|\, \mathcal{H}) P(\mathbf{M} \,|\, \mathcal{H}) && (4.30) \\ P(\mathbf{c} \,|\, \mathcal{H}) &= \mathcal{N}(c_x \,|\, 0, \sigma_x^2) \mathcal{N}(c_y \,|\, 0, \sigma_y^2) && (4.31) \\ P(\mathbf{M} \,|\, \mathcal{H}) &= \text{Gamma}(m_{00} \,|\, a_0, b_0) \text{Gamma}(m_{11} \,|\, a_1, b_1) \times \\ &\quad \mathcal{N}(m_{01} \,|\, 0, \sigma_0^2) \mathcal{N}(m_{10} \,|\, 0, \sigma_1^2) && (4.32) \end{aligned}$$

where the standard deviations σ_x and σ_y in the sub-grid centre co-ordinates were set to be equal to half the distance between the sub-grids in the x and y directions. Suitable values for the other parameters were found to be $a_0 = b_0 = a_1 = b_1 = 10$, $\sigma_0 = \sigma_1 = 0.02$.

4.6.2 Inferring the sub-grid transform

For any sensible choice of likelihood function $P(I \,|\, \mathbf{T})$, the posterior distribution over $\mathbf{T}$ will have a number of local maxima corresponding to translations of the sub-grid by one or more rows or columns from the true position. This means that gradient-based or local inference methods cannot be used initially as they would almost certainly get caught in one of these local maxima. Instead, we must use a procedure that explores all of the posterior modes sufficiently well to find the one containing the global maximum. Once this has been achieved, we can return to a gradient-based or local inference method to find the maximum of this mode and so find a MAP solution for $\mathbf{T}$. The posterior distribution is fairly tightly peaked around each local maximum and hence there is little point in retaining the uncertainty in $\mathbf{T}$ for a given mode, especially given that we are only looking for a rough initialisation. Note that as we do not require a posterior distribution over $\mathbf{T}$ and we cannot use local inference methods, Variational Message Passing will not be used in this case. Instead, the approach presented here is to start by exhaustively searching the space of $\mathbf{T}$ using a likelihood function that is extremely rapid to compute. This gives a solution which should be in the same region as the MAP solution. Then, conjugate gradient ascent with a more computationally expensive (but higher quality) likelihood function is used to find the MAP solution.

In each case, the transform $\mathbf{T}$ is used to divide the image into spot pixels S and background pixels B. As in spot-finding, we assume independence between pixels and therefore the likelihood function for the entire image can be written as the product of each pixel's likelihood

$$P(I \,|\, \mathbf{T}, \mathcal{H}) = \prod_{b \in B} P_B(I_b) \prod_{s \in S} P_S(I_s), \tag{4.33}$$

where $P_B(I_b)$ is the likelihood function for background pixel intensity and $P_S(I_s)$ is the likelihood function for spot pixel intensity. It is convenient to work with the log-likelihood as

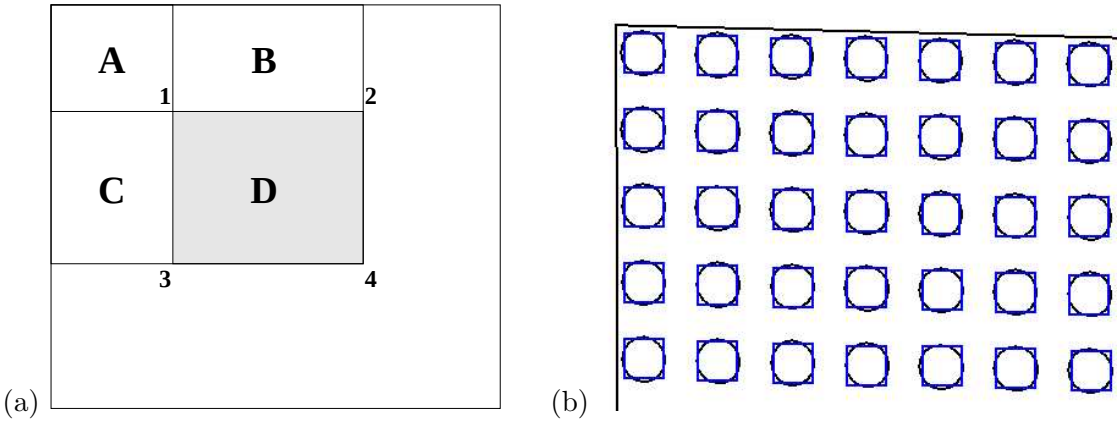


Figure 4.4: (a) The sum of pixel intensities within the shaded rectangle **D** can be found from the values of the integral image at each corner. The value at corner 1 is **A**, at corner 2 it is $\mathbf{A}+\mathbf{B}$, at corner 3 it is $\mathbf{A}+\mathbf{C}$ and at corner 4 it is $\mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D}$. The sum within **D** is thus $4+1-2-3$. (b) Enlargement of part of a transformed sub-grid showing how the spot ellipses can be reasonably well approximated by rectangular regions. Note that the sub-grid has been both rotated and sheared.

then the pixel log-likelihoods can simply be added together

$$\log P(I \mid \mathbf{T}, \mathcal{H}) = \sum_{b \in B} \log P_B(I_b) + \sum_{s \in S} \log P_S(I_s). \tag{4.34}$$

4.6.3 Searching through transform space

The first step of our inference procedure requires us to search through the space of possible transforms. To evaluate a likelihood function directly on the individual pixel intensities would involve at least as many operations as the number of pixels in the image (typically in excess of 200,000 operations). Even if we limited the operation to a simple addition, this would not be an efficient function to evaluate and so not suitable for use with a search.

Instead, inspired by the work of Viola and Jones [2001], the image is first transformed into an intermediate representation known as an *integral image*. The value of the integral image at (x, y) is equal to the sum of all pixel intensities above and to the left of that location in the original image:

$$i(x, y) = \sum_{x' \leq x} \sum_{y' \leq y} I(x', y'). \tag{4.35}$$

The sum of pixel intensities in any axis-aligned rectangular region with corners (x_1, y_1) and (x_2, y_2) can then be found using just four values of the integral image (see Figure 4.4a)

$$i_{\text{rect}}(x_1, y_1, x_2, y_2) = i(x_2, y_2) + i(x_1, y_1) - i(x_1, y_2) - i(x_2, y_1). \tag{4.36}$$

A likelihood function using the integral image

We can exploit the speed of calculating these rectangular area sums by choosing a likelihood function based on intensity sums and approximating our transformed spots with rectangles. Despite the fact that transformed spots are ellipses, their approximation by rectangles can be quite good if the transformation is not too extreme, as shown in Figure 4.4b. Each rectangle is centred on the spot centre and set to have the same area and aspect ratio as the transformed spot. The (approximate) sum of all spot pixels $\sum_S I_s$ is then found by adding the intensity sums for each spot rectangle, given by the integral image. The sum of all background pixel intensities $\sum_B I_b$ is found by subtracting this from the sum of all pixel intensities (which is the value in the lower-right corner of the intensity image):

$$\sum_{b\in B} I_b = i(x_{\max}, y_{\max}) - \sum_{j=1}^{J} i_{\text{rect}}(x_{j,1}, y_{j,1}, x_{j,2}, y_{j,2}), \tag{4.37}$$

where the corners of the rectangle approximating the jth spot are $(x_{j,1}, y_{j,1})$ and $(x_{j,2}, y_{j,2})$. Now we assume that the background pixels have an intensity distribution with a peak close to zero and which decreases monotonically with intensity. This sort of distribution can be modelled using a *truncated exponential distribution*

$$P_B(I_b) = \frac{\frac{1}{\beta}\exp(-\frac{I_b}{\beta})}{1-\exp(-\frac{I_{\max}}{\beta})} \qquad 0 \le I_b \le I_{\max} \tag{4.38}$$

where β is a scale parameter and $I_{\max}$ is the maximum intensity value. The log-likelihood of all background pixels is then

$$\log P(B \,|\, \mathcal{H}) = -\frac{1}{\beta}\sum_{b\in B} I_b - |B|[\log\beta - \log(1-\exp(-I_{\max}/\beta))]. \tag{4.39}$$

In Equation 4.39, the only dependence on the image is the sum of the background pixel intensities and the likelihood can be readily calculated from the integral image.

As spot pixels can have any intensity, they are modelled by a uniform distribution between 0 and $I_{\max}$, giving $P_S(I_s) = 1/I_{\max}$.

Search by regular sampling

Bayes's Theorem gives the posterior distribution over $\mathbf{T}$ to be

$$P(\mathbf{T} \,|\, I, \mathcal{H}) \;\propto\; P(I \,|\, \mathbf{T}, \mathcal{H}) P(\mathbf{T} \,|\, \mathcal{H}), \tag{4.40}$$

where the proportionality has been introduced as we are ignoring the evidence term $P(I \,|\, \mathcal{H})$ which does not depend on $\mathbf{T}$. We now need to search through the space of $\mathbf{T}$ to find a transform $\hat{\mathbf{T}}$ that maximises this posterior probability. Even though we have an extremely efficient likelihood function, this multi-dimensional search is only possible because we need

to search through only those transforms which are close to the identity (as image distortions are relatively small). As only small shears and rotations occur, just four dimensions were considered: the two location and two scale parameters. This four-dimensional space is divided into a regular grid and the posterior evaluated at each point in the grid, a procedure known as regular sampling. In all, the posterior is evaluated at ∼64,000 values of $\mathbf{T}$ and the one that gives the maximum value is taken to be the approximate solution $\hat{\mathbf{T}}$.

4.6.4 Finding the MAP solution

The second step of the inference procedure involves finding a MAP solution by refining $\hat{\mathbf{T}}$ using conjugate gradient ascent.[6] This method will find the local maximum of the posterior in the region of the approximate solution $\hat{\mathbf{T}}$, which should be the overall Maximum A Posteriori solution. To do this, new background and spot intensity distributions are used whose gradient can be computed

$$P_B(I_b) = \mathcal{N}(I_b \,|\, 0, \sigma^2) \tag{4.41}$$

$$P_S(I_s) = \mathcal{N}(I_s \,|\, I_{\max}, \sigma^2). \tag{4.42}$$

If we consider the pixel at image location $\mathbf{x}$, the corresponding point on the physical slide is $\mathbf{u} = \mathbf{T}^{-1}\mathbf{x}$, for a particular transform $\mathbf{T}$. We define a function $\mu(\mathbf{u})$ to be equal to $I_{\max}$ if $\mathbf{u}$ is inside a spot (according to the original configuration of the printing device) and zero elsewhere. At this stage we no longer approximate the spot ellipse by a rectangle. The log likelihood for any pixel is then written as

$$\log P(I(\mathbf{x}) \,|\, \mathbf{T}) = \log \mathcal{N}(I(\mathbf{x}) \,|\, \mu(\mathbf{u}), \sigma^2) \tag{4.43}$$

$$= -\frac{[I(\mathbf{x}) - \mu(\mathbf{u})]^2}{2\sigma^2} + \text{const.} \tag{4.44}$$

The gradient of this pixel log likelihood function w.r.t. $\mathbf{T}$ is

$$\frac{d}{d\mathbf{T}} \log P(I(\mathbf{x}) \,|\, \mathbf{T}) = -\frac{[I(\mathbf{x}) - \mu(\mathbf{u})]}{\sigma^2} \nabla I(\mathbf{x}) \, \mathbf{u}^T. \tag{4.45}$$

The function $\nabla I(\mathbf{x})$ is the two-dimensional gradient of the image intensity at $\mathbf{x}$ which can be approximated using vertical and horizontal Sobel filters [Nalwa 1993]. The gradient of the entire log likelihood function is found by simply summing over all pixels,

$$\frac{d}{d\mathbf{T}} \log P(I \,|\, \mathbf{T}) = \sum_{\mathbf{x}} \frac{d}{d\mathbf{T}} \log P(I(\mathbf{x}) \,|\, \mathbf{T}). \tag{4.46}$$

This gradient can then be used with a conjugate gradient method to find the MAP solution $\mathbf{T}_{\text{MAP}}$. Conjugate gradient methods are efficient in that they only require a few evaluations of the gradient. This efficiency means that, although calculating the gradient requires a

[6] Conjugate gradient descent is described in Bishop [1995].

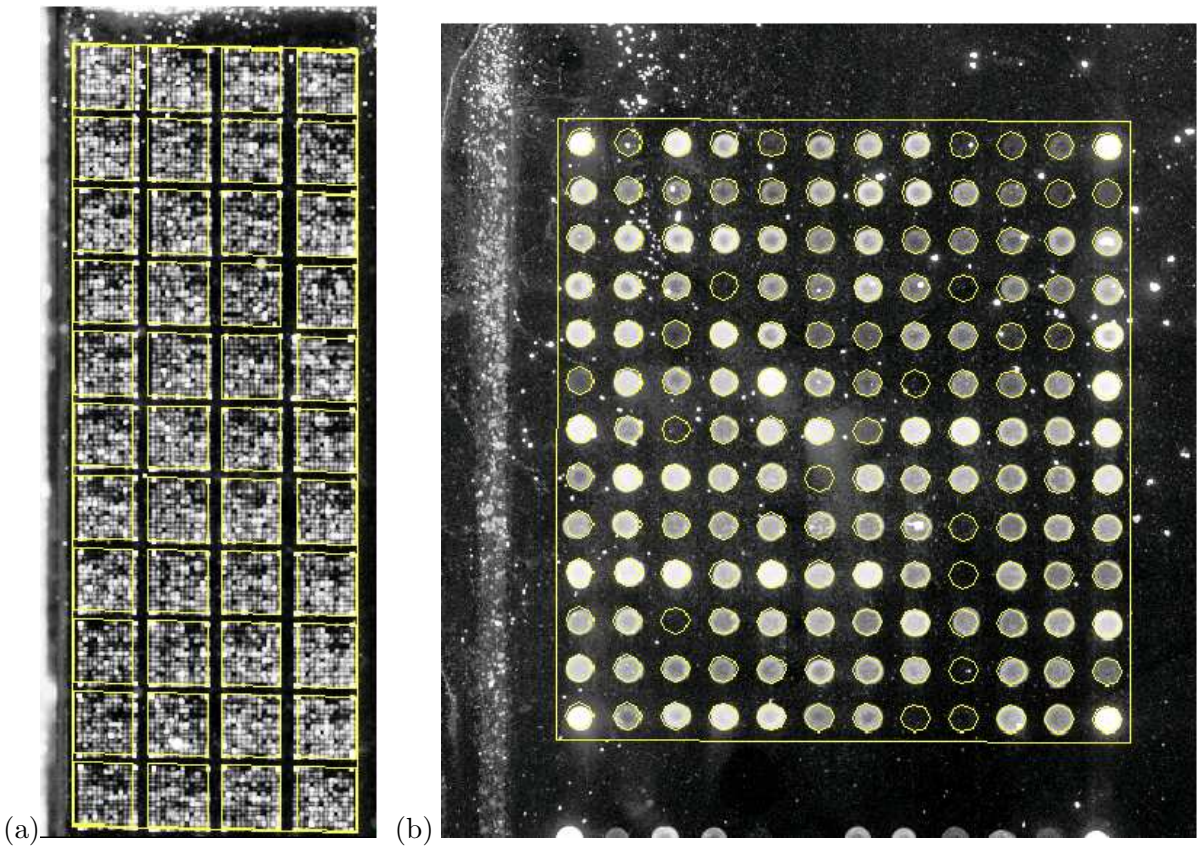


Figure 4.5: (a) The result of using the sub-grid location algorithm on a scaled version of a test slide image. Individual sub-grids are found, which allows initial sub-grid images to be extracted (such as the image on the right). Unfortunately, the assumption of an affine imaging transform combined with extreme noise effects at the edge of many slide images prevents this algorithm from working well in general and it can be necessary to perform this initial step manually. (b) A section of a scanned image which contains a single sub-grid and the sub-grid outline corresponding to the MAP transform found by the inference algorithm. The algorithm has correctly located the sub-grid within the image. The algorithm can be applied to image sections which have been extracted automatically, as in (a), or manually.

calculation for each pixel and is not particularly rapid, the overall process is still very quick.

When extremely noisy images are used, an additional step can be added to ensure that the MAP solution has been found and to avoid off-by-one-row/column errors. The four transforms corresponding to shifting $\mathbf{T}_{\mathrm{MAP}}$ one column to the left or right or one row up or down are used as initial points for the conjugate gradient algorithm. If any of these leads to a solution with higher posterior probability than $\mathbf{T}_{\mathrm{MAP}}$ then it is chosen to be the new $\mathbf{T}_{\mathrm{MAP}}$ and the procedure is repeated. Otherwise, the existing $\mathbf{T}_{\mathrm{MAP}}$ is used.

4.6.5 Results for sub-grid finding

Figure 4.5b shows the outline of a transformed sub-grid found using this method. Unfortunately, it is difficult to assess this algorithm's performance quantitively as I have been unable

to find any competing algorithms that perform the identical task. In addition, as the output is only approximate, it is difficult to compare the results of two algorithms, except qualitatively (i.e. whether the result coincided with the sub-grid or was incorrectly placed). In practice, the above algorithm was able to locate sub-grids correctly in the vast majority of images. The only images it failed on were test images where entire edge rows or columns were extremely faint. The presence of calibration spots in standard sub-grid images would normally prevent this from occurring.

4.6.6 Overall sub-grid location

The sub-grid location method described above relies on having an image with only one entire sub-grid in it. To extract such an image automatically requires an initial step which determines the approximate location of all the sub-grids within the entire slide image.

Once again, we would like to automate this process. One possibility is to re-apply the method of finding sub-grids to an image of the entire slide. We assume that the sub-grids themselves are arranged in a rectangular array which has been transformed by an unknown affine transform. Based on this assumption, the above algorithm can simply be applied to a scaled version of the slide image, so that the algorithm finds an array of sub-grids rather than spots. Figure 4.5a shows the output of the algorithm on an example slide image.

This method only gives an approximate sub-grid position (largely due to the assumption of an affine transform) and thus the extracted image used is the rectangle that contains the sub-grid enlarged in all directions by a small margin. This margin is currently set to twice the distance from spot centre to spot centre.

This method has been found to be effective on relatively undistorted slides with low background noise. Unfortunately, on other slide images it does not perform well. On an entire slide image, the assumption that the imaging transform is affine can be a poor one when there are large scale non-linear distortions. Additionally, there can be extreme noise artefacts around the edge of the slide image, which are not modelled well by the simple background model described earlier. To provide a reliable automatic solution for this step would involve learning a non-linear transform (i.e. a warp) and having a more complex intensity model. The alternative is to require the user to specify a linear transform and use a larger extracted image. As this requires just three mouse clicks per entire slide image, automating this step is not critical.

4.7 Discussion

In Sections 4.1–4.6, I have presented an algorithm for microarray image analysis that is based on an extension of Variational Message Passing which incorporates importance sampling to handle non-exponential family distributions. The algorithm is capable of identifying whether a spot is missing or obscured by noise artefacts and has been shown to perform well even in microarray images with a high level of background noise. In addition, I have shown that the

VMP algorithm can be initialised with a set of rough spot locations using a procedure that is either automatic for fairly clean slide images or requires minimal user interaction (three mouse clicks) for noisy, distorted slide images. This provides considerable time savings compared to using a procedure which requires locating each sub-grid by hand.

4.8 Gene Expression Data Analysis

The analysis of microarray images leads to a large quantity of gene expression data, along with appropriate measures of the certainty of those data (such as the standard deviation of the error in each measurement). The next step is to organise, analyse and visualise this data to reach conclusions about the biological processes being studied. The methods which can be used to achieve this are as open ended as the range of biological processes available to study. To date, a variety of methods have been used including:

- clustering by correlation/mutual information [Eisen et al. 1998; Spellman et al. 1998; Michaels et al. 1998]
- graph-based/hierarchical clustering [Ben-Dor et al. 1999; Bar-Joseph et al. 2001]
- Gaussian mixture model clustering [Yeung et al. 2001]
- self-organising maps [Tamayo et al. 1999]
- dimensionality reduction (PCA, ICA) [Raychaudhuri et al. 2000; Hori et al. 2001]
- latent variable modelling [Martoglio et al. 2002].

The majority of existing approaches do not take into account the uncertainty in the expression level data and do not provide a rigorous way of comparing different models for the data. It is an ongoing theme of this thesis that data analysis should be carried out by proposing probabilistic models, performing Bayesian inference to learn model parameters and comparing models using Bayesian model selection. It follows that analysis of gene expression data should also proceed along these lines and, indeed, this approach is starting to be used by some researchers. Hartemink et al. [2001] discuss the use of Bayesian networks as models of biological function which allow handling of uncertain expression data and rigourous comparison of different models whilst also permitting the introduction of latent variables (such as protein levels). Friedman et al. [2000] have used Bayesian networks to model the *S. cerevisiae* cell-cycle measurements of Spellman et al. [1998] and were able to capture much richer structure from the data than clustering methods, despite using models with no latent variables.

If probabilistic models in general, and Bayesian networks in particular, are to be used for gene expression data analysis, then it follows that the Variational Message Passing algorithm can be applied to rapidly perform approximate inference and model selection on novel models, provided they are conjugate-exponential or can be made so. To demonstrate the ease of use of this algorithm, I now present an example of using Variational Message Passing to perform

Independent Component Analysis on a small gene expression data set. The aim of this example is to provide a short illustration of how VMP allows complex models to be quickly constructed and applied to real data sets, rather than to break new ground in gene expression data analysis.

4.8.1 ICA of gene expression data using VMP

When applying Independent Component Analysis (ICA) to gene expression data, the pattern of gene expression for each tissue is represented as a linear superposition of a small number of underlying patterns or *signatures*. Unlike when using Principal Component Analysis, these signatures are not constrained to be orthogonal but are instead assumed to have amplitudes that are statistically independent of each other.

The core assumption of ICA, therefore, is that our gene expression data $\mathbf{X} = (\mathbf{x}_1 \dots \mathbf{x}_N)^{\mathrm{T}}$ can be modelled as a linear combination of signatures $\mathbf{S} = (\mathbf{s}_1 \dots \mathbf{s}_M)^{\mathrm{T}}$ plus some Gaussian noise τ, so that

$$\mathbf{X} = \mathbf{W}^{\mathrm{T}}\mathbf{S} + \tau \tag{4.47}$$

where each column of $\mathbf{W}$ gives the amounts of each signature present in the corresponding tissue sample. The aim is to infer the signatures $\mathbf{S}$, the amplitude matrix $\mathbf{W}$ and the number of signatures M.

Following Miskin [2000], the rows of $\mathbf{W}$ are modelled using M Gaussian mixture models with C components, each of which has the form described in Section 1.8.7. The number of signatures is found by using an Automatic Relevance Determination prior α_m on each signature (each row of $\mathbf{S}$) which allows signatures to be switched off if their presence is not supported by the data. The Bayesian network for this ICA model is shown in Figure 4.6.

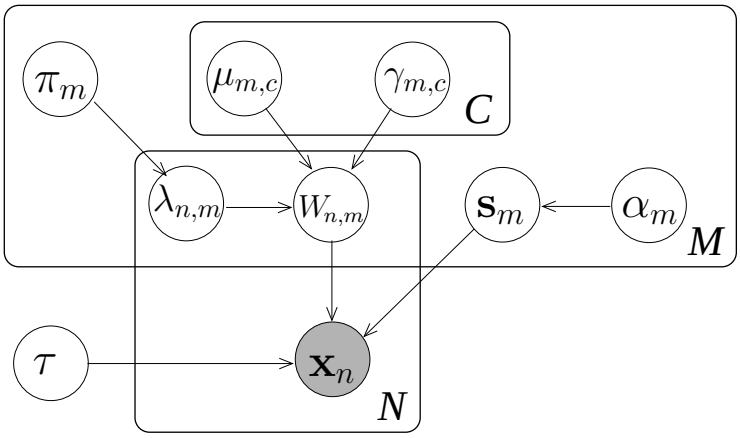


Figure 4.6: The Bayesian network for the Independent Component Analysis model. Each gene expression vector $\mathbf{x}_n$ is viewed as a linear superposition of signatures $\mathbf{s}_m$. The hyper-parameter α_m controls which of the signatures are switched off and so allows the number of signatures to be determined. The elements in each row of the amplitude matrix $\mathbf{W}$ are modelled using a mixture of C Gaussians with parameters $\{\mu_{m,c}, \gamma_{m,c}\}_{c=1}^{C}$, where the means are set to be zero. The reconstruction error is modelled as being Gaussian with precision τ.

ICA model applied to ovarian tissue samples data

Now that the Bayesian network which we are using to model the data has been specified, Variational Message Passing allows inference to proceed automatically for any supplied data set. This will now be demonstrated on a small data set consisting of the gene expression levels of 175 genes in 17 tissue samples from Martoglio et al. [2000]. The tissue set consists of ovarian samples, some of which are tumourous, as described in Table 4.1.

Tissue Number	Description
1	Normal (pre-menopausal)
2-5	Normal (post-menopausal)
6-10	Serous Papillary Adenocarcinoma (SPA)
11-14	Poorly Differentiated SPA (PD-SPA)
15	Benign Serous Carcinoma (BSC)
16-17	Benign Mucinous Carcinoma (BMC)

Table 4.1: Descriptions of the tissue samples in the ovarian tissue data set

The ICA model converged in about 200 iterations. A Hinton diagram showing the expected value of $\mathbf{W}$ under the optimised variational posterior is shown in Figure 4.7. As can be seen from this diagram, only 7 out of a possible 17 signatures have been retained.

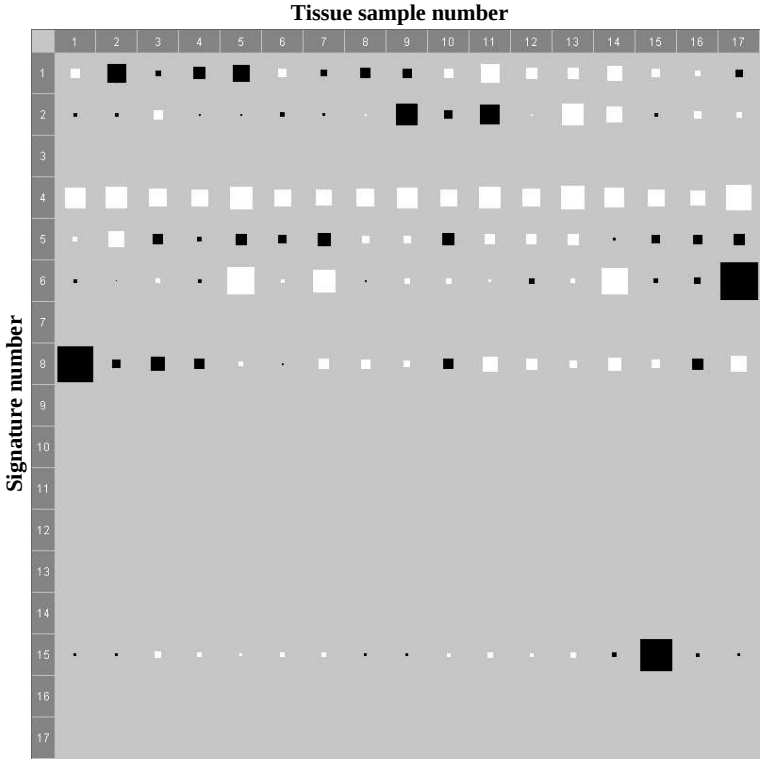


Figure 4.7: Hinton diagram of the expected $\mathbf{W}$ amplitude matrix under the variational posterior when the ICA model is trained on a data set of ovarian tissue samples. The rows correspond to the 17 possible signatures – of which only 7 have been used. The columns show how much of the signatures are present in the each of the 17 tissue samples.

Biological interpretation of the inferred gene signatures

The ICA model assumes that the overall gene expression profile of each tissue is due to the superposition of the gene expressions of a number of independent biological processes. It follows that the amplitude matrix **W** represents the level of activity of these processes in each tissue sample. By comparing the activity of a signature to the known characteristics of each tissue, it is possible to infer broadly which biological process the signature represents and therefore what genes are associated with that process.

Firstly, consider the fourth signature whose activity and gene expression levels are shown in Figure 4.8. This signature is present at a near-constant level in all of the samples. In addition, the signature contains only positive expression levels for all genes. This signature can therefore be interpreted as representing the genes which are expressed in all ovarian tissues at any time. Such genes are referred to as *housekeeping genes* as they are responsible for essential cell function such as the maintenance of cell cycle, metabolism and so on.

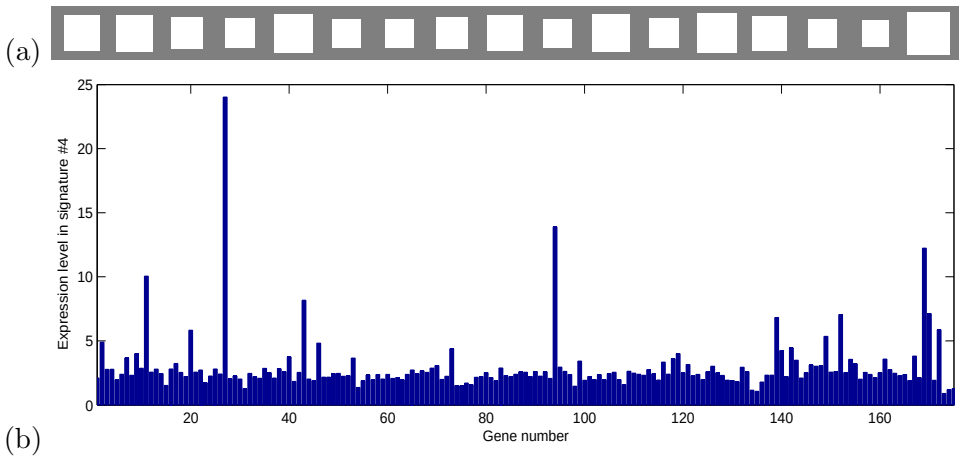


Figure 4.8: (a) Hinton diagram showing the level of activity of the 4th signature in each of the 17 tissue samples. As can be seen, this signature has almost constant activity in all the tissue samples. (b) Bar chart showing the expression levels (in arbitrary units) of each of the 175 genes for the 4th signature. This signature expresses all of the genes and so can be regarded as representing the housekeeping genes for all ovarian tissue samples.

Secondly, consider the 8th signature (Figure 4.9) which is only strongly present in the first tissue sample. This sample is the only pre-menopausal sample in the data set and it seems likely, therefore, that this signature differentiates pre-menopausal from post-menopausal gene expression.

Finally, consider the 15th signature (Figure 4.10) which is only strongly present in the 15th tissue sample. This sample is the one sample in the data set from a Benign Serous Carcinoma and hence this signature may be indicative of the presence of such a tumour. Clearly, a larger data set would be required before any strong conclusions could be drawn concerning the biological interpretation of particular signatures.

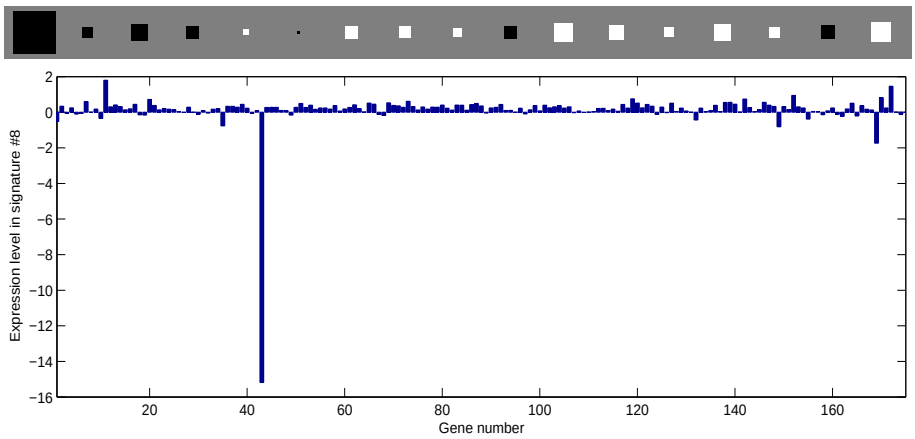


Figure 4.9: (a) Hinton diagram showing the level of activity of the 8th signature in each of the 17 tissue samples. The signature is only strongly present in the first sample. (b) Bar chart of the gene expression levels for the 8th signature. This signature is dominated by the expression of the 43rd gene (which codes for an Endothelin-1 receptor).

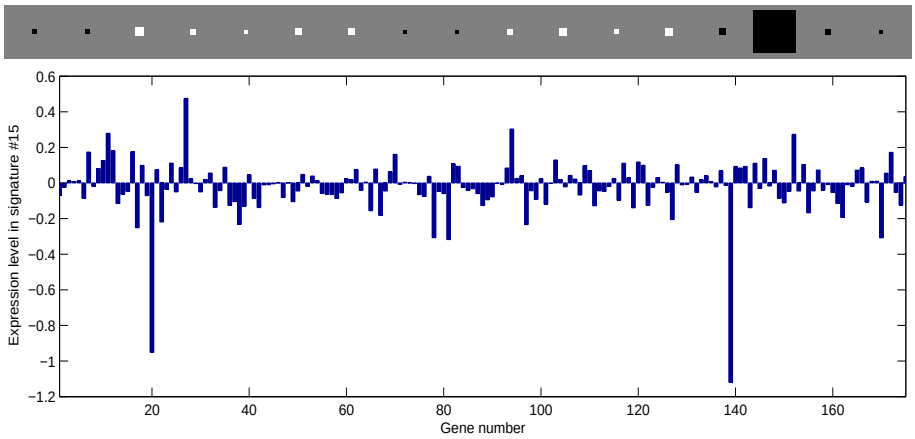


Figure 4.10: (a) Hinton diagram showing the level of activity of the 15th signature in each of the 17 tissue samples. The signature is only strongly present in the 15th sample which was from a Benign Serous Carcinoma. (b) Bar chart of the gene expression levels for the 15th signature.

4.8.2 Conclusion

This brief example has shown that Variational Message Passing allows rapid application of a plausible probabilistic model of gene expression to a small data set. The resultant set of gene expression signatures has allowed some tentative interpretation of the independent biological processes involved.

Overall, it has been shown that Variational Message Passing can be applied successfully both to analyse scanned images of microarrays and to interpret the resultant gene expression levels to reach conclusions about the underlying biological systems.

Chapter 5

Structured Variational Distributions

The fully factorised variational approximation has been widely used with great success in many applications; its use for image analysis and biological modelling has already been illustrated in this thesis. However, the fully factorised variational distribution does represent a somewhat restrictive approximation. It cannot, for instance, capture the posterior correlation between variables, since the approximation is fully separable.

The limitations associated with the fully factorised approximation can be overcome to a large extent by allowing a broader class of variational distributions which retain some of the structure of the original model. These will, in general, give a closer approximation to the true posterior distribution. However, it is essential that inference using this richer family of distributions remains computationally tractable.

In this chapter, the variational inference framework of Chapter 2 will be extended to allow the use of variational distributions in this richer family of structured distributions[1]. This extension will enable the posterior dependencies between particular variables to be captured, improving the quality of the approximation and extending the number of applications where the framework can be applied.

5.1 Inference Using Structured Variational Distributions

The Variational Message Passing algorithm uses a variational distribution which is fully factorised, giving an approximate posterior which is separable with respect to individual variables. In general, an improved approximation will be achieved if a posterior distribution is used which retains some dependency structure. Whilst Wiegerinck [2000] has present a general framework for such structured variational inference, he does not provide a general-purpose algorithm for applying this framework. In this chapter, I extend VMP to allow structured variational distributions and so provide such an algorithm.

[1] The work presented in this chapter builds on a paper written in collaboration with Christopher M. Bishop [Bishop and Winn 2003].

Since this work was carried out, Xing et al. [2003] have presented a Generalised Mean Field (GMF) algorithm for structured variational inference. The GMF algorithm allows for the variational distribution to be factorised into clusters, giving an improved approximation over standard VMP. However, the GMF algorithm requires that the clusters do not overlap and so posterior dependencies cannot be captured between variables in different clusters. As will be described, Structured VMP does not have this constraint and allows for overlapping clusters, hence giving an improved approximation over that of the GMF algorithm.

5.1.1 Optimising structured variational distributions

To allow dependency structure to be retained, our variational distribution must now be defined to have a more general product-of-factors form (previously discussed in Section 1.4.2),

$$Q(\mathbf{H}) = \frac{1}{Z_Q} \prod_i \Phi_i(c_i) \tag{5.1}$$

where each c_i is a *cluster* (the small subset of variables that appear in a particular factor) and Φ_i the corresponding non-negative factor function, known as a *cluster potential*. The clusters are, in general, non-disjoint and their union contains all the latent variables, so $\bigcup_i c_i = \mathbf{H}$. Note that the joint probability distribution of any Bayesian network can be written in this form. The normalisation constant Z_Q is set to ensure that Q is a valid probability distribution:

$$Z_Q = \int \prod_i \Phi_i(c_i)\, d\mathbf{H}. \tag{5.2}$$

As in the fully factorised case, the goal is to maximise the lower bound $\mathcal{L}(Q)$, defined in Equation 1.40 as

$$\mathcal{L}(Q) = \langle \log P(\mathbf{H}, \mathbf{D}) - \log Q(\mathbf{H}) \rangle_Q \,. \tag{5.3}$$

Substituting our new form of Q and omitting the arguments of the cluster potentials gives

$$\mathcal{L}(Q) = \frac{1}{Z_Q} \int \prod_i \Phi_i \left\{ \log P(\mathbf{H}, \mathbf{D}) - \sum_i \log \Phi_i + \log Z_Q \right\} d\mathbf{H}. \tag{5.4}$$

The terms containing one cluster potential Φ_j can be separated out

$$\mathcal{L}(Q) = \frac{1}{Z_Q} \int_{c_j} \Phi_j \int_{\mathbf{H} \setminus c_j} \prod_{i \neq j} \Phi_i \left\{ \log P(\mathbf{H}, \mathbf{D}) - \sum_i \log \Phi_i + \log Z_Q \right\} d\mathbf{H}. \tag{5.5}$$

The aim is to maximise $\mathcal{L}(Q)$ subject to the normalisation constraint. This can be achieved by means of a Lagrange multiplier λ, so we seek instead to maximise

$$\overline{\mathcal{L}}(Q) = \mathcal{L}(Q) + \lambda \left(\frac{1}{Z_Q} \int \prod_i \Phi_i(c_i)\, d\mathbf{H} - 1 \right) \tag{5.6}$$

and so obtain the following stationarity condition

$$0 = \int \prod_{i \neq j} \Phi_i \left\{ \log P(\mathbf{H}, \mathbf{D}) - \sum_{i \neq j} \log \Phi_i - \log \Phi_j + \log Z_Q + 1 \right\} d\mathbf{H} \backslash c_j. \tag{5.7}$$

We can now solve to find the setting of the potential Φ_j^* that maximises $\mathcal{L}(Q)$ whilst all other potentials are held constant

$$\log \Phi_j^* = \left\langle \log P(\mathbf{H}, \mathbf{D}) - \sum_{i \neq j} \log \Phi_i \right\rangle_{\mathbf{H} \backslash c_j} + \text{const.} \tag{5.8}$$

As the original P distribution is defined as a Bayesian network, the joint probability can be written as

$$P(\mathbf{X}) = \prod_i P(X_k \,|\, \text{pa}_k) \tag{5.9}$$

where $\mathbf{X} = (\mathbf{H}, \mathbf{D})$ as before. This can be viewed as a product of potentials of the form $P(X_k \,|\, \text{pa}_k)$ with corresponding clusters $d_k = \{X_k, \text{pa}_k\}$. Substituting into Equation 5.8 and removing constant terms gives the final result (see also Wiegerinck [2000])

$$\log \Phi_j^* = \left\langle \sum_{k \in D_j} \log P(X_k \,|\, \text{pa}_k) - \sum_{i \in C_j} \log \Phi_i \right\rangle_{\mathbf{H} \backslash c_j} - z \tag{5.10}$$

where the summations have been restricted to be over only the sets of log potentials D_j and C_j, defined as follows. The set D_j contains all clusters d_k that have a non-zero intersection with the cluster c_j. Similarly, the set C_j contains all clusters c_i that intersect with c_j, excluding c_j itself. The constant z can be inferred from the global normalisation constraint of Equation 5.2. Thus, the update for a cluster depends only on overlapping clusters and can therefore be performed locally in the cluster graph, provided that the normalisation constant can also be found using local operations.

The expectation in Equation 5.10 is taken with respect to the conditional distribution defined by

$$Q(\mathbf{H} \backslash c_j \,|\, c_j) = \frac{\prod_{i \neq j} \Phi_i}{\int_{\mathbf{H} \backslash c_j} \prod_{k \neq j} \Phi_k}. \tag{5.11}$$

The new update equation is slightly more complex than the one derived for the fully factorised case (Equation 1.49) due to the addition of a second term in the expectation, relating to overlapping clusters in Q. If Q is fully factorised, all the clusters c_j are disjoint and so C_j is empty for all j. The second term then vanishes and we recover the original update equation for the fully factorised Q distribution.

5.1.2 Using a Bayesian network as a variational distribution

Consider the special case of Equation 5.1 where the Q distribution corresponds to a Bayesian network. In this case, the potentials have the form of conditional distributions $\Phi_i(c_i) = Q(X_i \mid \mathrm{pa}_i)$ where $c_i = \{X_i, \mathrm{pa}_i\}$. To ensure the potentials are valid conditional probability distributions, an additional local normalisation constraint must be introduced for each potential

$$\int \Phi_i(c_i)\, dX_i = 1. \tag{5.12}$$

Taking these additional constraints into account leads to a slightly different update equation

$$\log Q(X_j \mid \mathrm{pa}_j)^* = \left\langle \sum_{k \in D_{X_j}} \log P(X_k \mid \mathrm{pa}_k) - \sum_{i \in C_{X_j}} \log Q_i(X_i \mid \mathrm{pa}_i) \right\rangle_{\mathbf{H} \setminus c_j} - z(\mathrm{pa}_i) \tag{5.13}$$

where D_{X_j} is the set of all clusters d_k that depend on X_j and similarly C_{X_j} is the set of all clusters c_i that depend on X_j, excluding c_j itself. The local normalising factor $z(\mathrm{pa}_i)$ can be inferred using Equation 5.12. In this case, the update for the jth cluster depends only on clusters in the Markov blanket of X_j in P and Q.

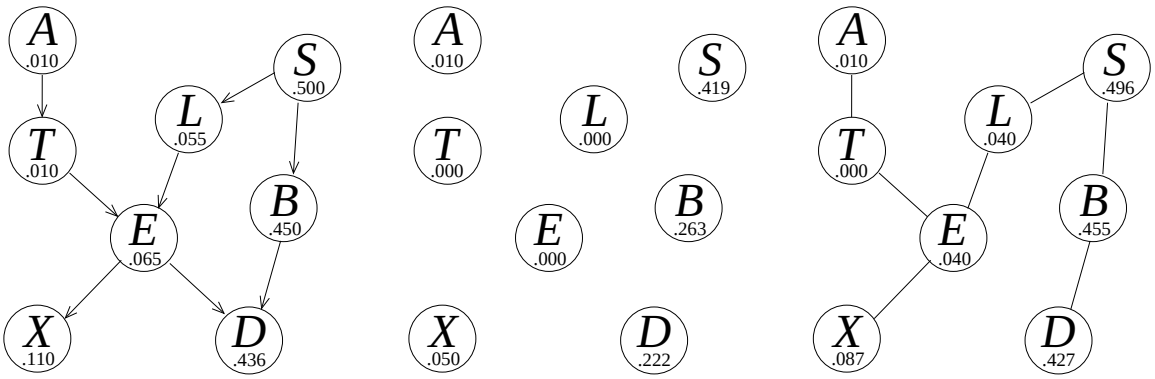
We must now consider how to choose a variational distribution that is tractable (in that it allows cluster potential updates to be computed analytically) whilst providing a good approximation to the true posterior distribution.

5.2 Choice of Structured Variational Distribution

There is a large range of possible structured variational distributions and so we will need to focus on tractable distributions that capture important dependencies in the P distribution. The approach that will be used here is to choose Q distributions whose dependency structure corresponds to sub-graphs of the original graphical model. It is anticipated that the dependency structure of the prior model will provide a good indication of the dependency structure of the posterior [Wiegerinck 2000; Saul and Jordan 1996]. The strategy, therefore, will be to take the original graphical model and to remove edges as required, in order to achieve tractability of inference.

To illustrate this approach, Figure 5.1 shows an example using the ASIA chest clinic Bayesian network from Lauritzen and Spiegelhalter [1988]. This example shows the improved approximation (in terms of KL divergence) given by using a structured Q distribution instead of a fully factorised one. The dependency structure of the original model has been partially retained by using a structured distribution of a tree whose edges are all present in the original graph.

In general, if we do not remove any of the edges of the original graph, then Q will simply equal P and inference will be intractable by assumption (we would not be using an approximate inference method if inference were tractable in P). Alternatively, removing all the edges



(a) Exact (b) KL = 0.43 (c) KL = 0.03

Figure 5.1: Example taken from Wiegerinck [2000] showing how a structured variational distribution gives a better approximation to the exact posterior than a fully factorised distribution. The model used is the ASIA chest clinic Bayesian network from Lauritzen and Spiegelhalter [1988]. (a) The Bayesian network for the model showing marginal probabilities under the exact posterior P. (b) The fully factorised approximation, with marginal probabilities under the variational distribution. (c) A structured approximation where Q is a tree, with marginal probabilities. For each approximation, the KL divergence between Q and P is given, showing the significant improvement in accuracy given by using the structured variational distribution.

of the original graph results in a fully-factorised Q distribution, which has been shown to lead to tractable inference for all conjugate-exponential models. At some point between these two extremes lies a variational distribution that retains the most structure possible whilst still remaining tractable. This distribution will give the best possible approximation to the true posterior distribution. This begs the question "which edges can be retained in Q without loss of tractability?" which will be addressed shortly. It is worth noting that, in practice, it may not always be preferable to use the distribution which gives the absolute best approximation. Retaining additional structure in Q comes with an associated penalty in terms of computational cost and memory requirements and so it makes more sense to use the Q distribution with the simplest structure that provides good results for a particular model and domain.

Given that we wish to use a Q distribution defined by a subgraph of a Bayesian network, it would seem natural to choose Q to be a Bayesian network also. In this case, the cluster potentials would be conditional probabilities $Q(X_i \mid \mathrm{pa}_i)$, as described in Section 5.1.2. Unfortunately, in order to implement the variational update equations, it is necessary to compute appropriate expectations, which in turn requires that marginal distributions over small subsets of variables are available. In a Bayesian Network, message passing would be required to find such marginal distributions. What would be preferable is a form of Q where the cluster potentials themselves are marginal distributions. This can be achieved by using a Q distribution in the form of a *junction tree*.

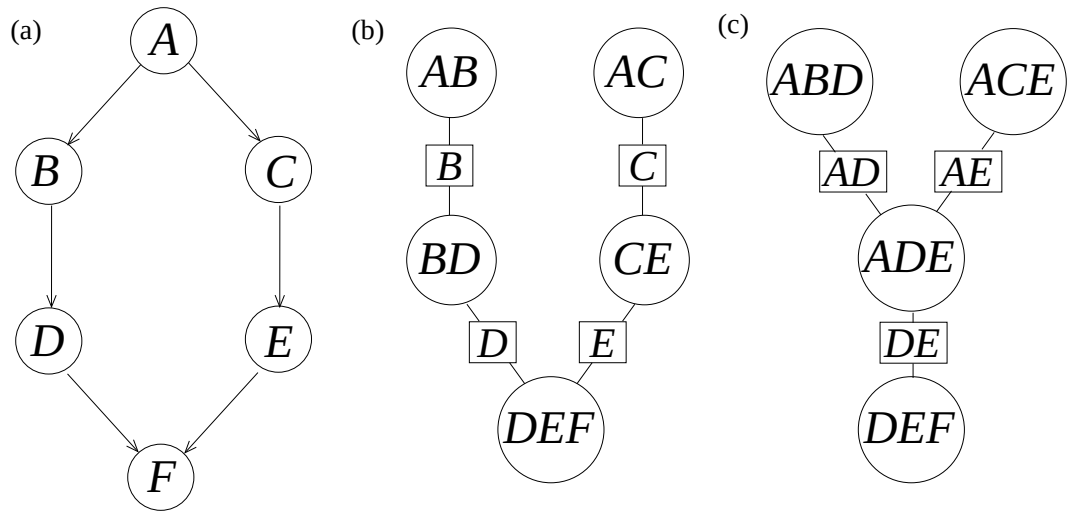


Figure 5.2: (a) A Bayesian network. (b) A cluster graph that captures the dependencies in the Bayesian network. The square nodes are separators, which contain the variables common to the two adjacent clusters. This cluster tree does not satisfy the running intersection property as the nodes on the path from AB to AC do not all contain the intersection variable A. (c) An alternative cluster tree that does obey the running intersection property and hence is a junction tree for the Bayesian network.

5.3 Variational Junction Trees

A junction tree [Jensen 1996; Cowell et al. 1999] is a tree-structured cluster graph that satisfies the *running intersection property*. This property states that for any pair of clusters c_i and c_j, all clusters on the path between c_i and c_j contain the intersection $c_i \cap c_j$. To illustrate the running intersection property, Figure 5.2b shows a cluster tree that does not satisfy this property, whilst Figure 5.2c shows one that does and which is therefore a junction tree. The clusters in a junction tree are connected via *separators* each of which contains the variables common to the two neighbouring clusters. The separator between two connected clusters c_i and c_j is referred to as s_{ij}.

A junction tree can be constructed from a Bayesian network by:

- *moralising* the graph by connecting all pairs of parents for every node and dropping the arrows on the edges (giving an undirected graph);
- *triangulating* the undirected graph by adding additional edges to ensure that every cycle of length four or more has a chord;
- forming a junction tree by using the cliques of the undirected graph as clusters.

As there are typically many possible ways of triangulating a graph, there can be many possible junction trees corresponding to a single Bayesian network. Figure 5.2a shows a Bayesian network and Figure 5.2c shows one possible corresponding junction tree.

A *consistent* junction tree is one where the potentials Φ_i and Φ_j of any two clusters c_i and c_j, which have a non-zero intersection $A = c_i \cap c_j$, satisfy

$$\int \Phi_i(c_i)\ dc_i \backslash A = \int \Phi_j(c_j)\ dc_j \backslash A. \tag{5.14}$$

This constraint means that the marginal potential over any subset of variables is the same no matter which cluster potential it is derived from. The running intersection property means that all nodes containing a particular set of variables must form a connected subgraph and so consistency can be enforced globally using local operations on the graph (i.e. message passing).

When using a junction tree as a variational distribution, Q has the form

$$Q(\mathbf{H}) = \frac{\prod_i \Phi_i(c_i)}{\prod_{j,k} \Phi_{jk}(s_{jk})} \tag{5.15}$$

in which the product in the numerator is over clusters and the product in the denominator is over separators. The separator potentials $\Phi_{ij}(s_{ij})$ are defined by (either) neighbouring cluster potential

$$\Phi_{ij}(s_{ij}) = \int \Phi_i(c_i)\ dc_i \backslash s_{ij}. \tag{5.16}$$

The end result of this form of Q and the consistency constraints is that the cluster potentials can be interpreted directly as marginal probability distributions:

$$\Phi_i(c_i) = Q(c_i). \tag{5.17}$$

This property of junction trees allows expectations to be calculated directly, provided that the variables involved can be found in a single cluster of the tree.

5.3.1 Inference using variational junction trees

It follows from Equation 5.10 that the cluster potential Φ_j^* can be optimised using

$$\log \Phi_j^* = \left\langle \sum_{k \in D_j} \log P(X_k \,|\, \mathrm{pa}_k) - \sum_{i \in C_j} \log \Phi_i + \sum_{(l,m) \in S_j} \log \Phi_{lm} \right\rangle_{\mathbf{H} \backslash c_j} - z \tag{5.18}$$

where C_j and D_j are as defined previously and S_j is the set of separators which depend on c_j. The constant z is set to ensure that Φ_j^* is normalised. It is important to note, however, that having modified Φ_j^*, the junction tree is no longer consistent. To recover consistency, a `DistributeEvidence` procedure must be used with c_j as the root. In the junction tree, this involves sending messages from c_j to each neighbour by first updating the intervening separator

$$\Phi^*_{ij(s_{ij})} = \int \Phi_j^*(c_j)\ dc_j \backslash s_{ij} \tag{5.19}$$

and then the neighbouring cluster using

$$\Phi_i^*(c_i) = \Phi_i(c_i)\frac{\Phi^*_{ij(s_{ij})}}{\Phi_{ij(s_{ij})}}. \tag{5.20}$$

Recursively, each neighbour then sends out messages to all its neighbours except the one from which the message came. When all clusters in the tree have received a message, the tree is consistent again.

The above analysis also applies in the case where Q consists of several disconnected junction trees. In this case, the Q distribution consists of a product of factors, one for each tree. Let $\mathbf{T}_\alpha$ be the set of nodes associated with a particular tree and $Q(\mathbf{T}_\alpha)$ the corresponding factor in the Q distribution so that

$$Q(\mathbf{X}) = \prod_\alpha Q(\mathbf{T}_\alpha). \tag{5.21}$$

Suppose that a cluster c_j is contained in $\mathbf{T}_\alpha$. Its potential can be updated using

$$\log \Phi_j^* = \left\langle \sum_{k\in D_j} \langle \log P(X_k \mid \mathrm{pa}_k)\rangle_{\beta\neq\alpha} - \sum_{i\in C_j} \log \Phi_i + \sum_{(l,m)\in S_j} \log \Phi_{lm} \right\rangle_{\mathbf{T}_\alpha \setminus c_j} - z \tag{5.22}$$

and the `DistributeEvidence` procedure need only be applied to clusters in $\mathbf{T}_\alpha$. The expectation in the first term does not depend on $Q(\mathbf{T}_\alpha)$ and therefore this term will remain unchanged throughout the update of this tree.

It is, of course, also possible to update the separator potentials ϕ_{lm} using Equation 5.10. Consider that the set of separator variables s_{lm} is always a subset of each neighbouring cluster c_l and c_m. The optimisation of $Q(c_l)$ or $Q(c_m)$ is guaranteed to lead to at least as good an increase in $\mathcal{L}$ as optimising $Q(s_{lm})$. This can be seen by writing $Q(c_l) = Q(s_{lm})Q(c_l \setminus s_{lm} \mid s_{lm})$ and hence optimisation of $Q(c_l)$ includes optimising just $Q(s_{lm})$ as a special case. Therefore any optimisation possible by updating separator potentials can be equalled or improved by updating neighbouring clusters and so we choose to update only cluster potentials. Furthermore, following changing a separator potential, it would be necessary to define a new algorithm for making the junction tree consistent.

5.4 Reducing Computation for Inference with VJTs

Optimising a potential using Equation 5.22 and performing `DistributeEvidence` throughout the entire tree after each update leads to a computationally expensive inference algorithm. For many common structures of P and Q, the resultant junction trees satisfy certain constraints which allow considerable savings to be made on the required computation. Where Q has been formed by deleting edges from P, two possible cases arise: where all deleted edges are between nodes in different junction trees of Q (Case I) and where at least one edge is between

nodes contained in the same junction tree (Case II). The modified update procedure and the resultant reduction in computation will now be discussed for these two cases.

5.4.1 Case I: Q junction tree with no internal deleted edges

I will now consider a special case of optimising $Q(\mathbf{T}_\alpha)$ for a junction tree $\mathbf{T}_\alpha$. I start by defining $\hat{\psi}_k(d_k \cap \mathbf{T}_\alpha) = \langle \log P(X_k \mid \mathrm{pa}_k) \rangle_{\beta \neq \alpha}$ for each d_k that intersects with $\mathbf{T}_\alpha$. Now suppose that we have the case where, for each $\hat{\psi}_k$, there is at least one cluster c_j that contains the intersection $d_k \cap \mathbf{T}_\alpha$. When Q is formed by deleting edges of P, this is equivalent to stating that no edges have been deleted between nodes in $\mathbf{T}_\alpha$. I then choose to assign each $\hat{\psi}_k$ potential to a cluster that contains $d_k \cap \mathbf{T}_\alpha$ and define A_i to be the set of such potentials assigned to the ith cluster. Thus, each $\hat{\psi}_k$ is assigned to exactly one A_i. Prior to the optimisation, the cluster and separator potentials are initialised using

$$\phi_i(c_i) \stackrel{\text{def}}{=} \log \Phi_i(c_i) = \sum_{k \in A_i} \hat{\psi}_k \tag{5.23}$$

$$\phi_{jk}(s_{jk}) \stackrel{\text{def}}{=} \log \Phi_{jk}(s_{jk}) = 0 \tag{5.24}$$

where the ϕ notation has been introduced as shorthand for the corresponding $\log \Phi$. The above initialisation means that the junction tree is not consistent. This can be rectified by performing `CollectEvidence` and then `DistributeEvidence` with respect to an arbitrarily chosen root cluster c_k. Following these two message passing procedures, the junction tree will be consistent. It is important to note, however, that neither procedure changes the overall distribution $Q(\mathbf{T}_\alpha)$ and so the potentials are related by

$$\sum_i \phi_i - \sum_{l,m} \phi_{lm} = \sum_k \hat{\psi}_k \tag{5.25}$$

where the first sum is over clusters in $\mathbf{T}_\alpha$ and the second is over separators in $\mathbf{T}_\alpha$. Now consider applying Equation 5.22 to update the potential for the jth cluster

$$\phi_j^* = \left\langle \sum_{k \in D_j} \hat{\psi}_k - \sum_{i \in C_j} \phi_i + \sum_{(l,m) \in S_j} \phi_{lm} \right\rangle_{\mathbf{T}_\alpha \setminus c_j} - z, \tag{5.26}$$

and then extending the summations to include all potentials which intersect with $\mathbf{T}_\alpha$, except ϕ_j itself. The additional potentials are all constant with respect to c_j and so only the normalisation constant z needs to be changed:

$$\phi_j^* = \left\langle \sum_k \hat{\psi}_k - \left(\sum_{i \neq j} \phi_i - \sum_{l,m} \phi_{lm} \right) \right\rangle_{\mathbf{T}_\alpha \setminus c_j} - z'. \tag{5.27}$$

Substituting in from Equation 5.25 gives

$$\phi_j^* = \left\langle \sum_k \hat{\psi}_k - \left(\sum_k \hat{\psi}_k - \phi_j \right) \right\rangle_{\mathbf{T}_\alpha \setminus c_j} - z' = \phi_j \tag{5.28}$$

and hence the update has left the potential ϕ_j unchanged.

It follows that, in the situation where there are no deleted edges in $\mathbf{T}_\alpha$, the factor $Q(\mathbf{T}_\alpha)$ can be optimised using just the initialisation defined above and the `CollectEvidence` and `DistributeEvidence` message-passing procedures. It is therefore true that if Q consists of a single tree $\mathbf{T}$ then the condition that there are no deleted edges is equivalent to stating that Q is a junction tree for P. The above initialisation and message-passing then corresponds to the procedure used for belief propagation. In short, belief propagation is a special case of this more general algorithm.

5.4.2 Case II: Q junction tree with internal deleted edges

The alternate case to the one considered above is when edges *are* deleted between nodes in a tree $\mathbf{T}_\alpha$. This means that there is at least one potential $\hat{\psi}_k(d_k \cap \mathbf{T}_\alpha)$ where the intersection $d_k \cap \mathbf{T}_\alpha$ is not contained in any cluster. The set of such potentials shall be denoted B_α and the set of all other potentials A_α, so that each potential lies in either B_α or A_α.

We assign each potential $\hat{\psi}_k$ in A_α to a cluster that contains $d_k \cap \mathbf{T}_\alpha$, as before. Again, we define A_i to be the set of potentials in A_α assigned to cluster c_i. We also define a set B_i to be the set of potentials in B_α whose variables intersect with c_i. Note that each potential in A_α appears in only one A_i whilst each potential in B_α may appear in one or more B_i.

We initialise the cluster and separator potentials as before

$$\phi_i(c_i) = \sum_{k \in A_i} \hat{\psi}_k \tag{5.29}$$

$$\phi_{jk}(s_{jk}) = 0. \tag{5.30}$$

This initialisation leads to an inconsistent junction tree and so we perform `CollectEvidence` and then `DistributeEvidence` with respect to an arbitrarily chosen root cluster. At this stage, the tree is consistent and the overall distribution $Q(\mathbf{T}_\alpha)$ and the $\hat{\psi}$ potentials are related by

$$\sum_i \phi_i - \sum_{l,m} \phi_{lm} = \sum_{k \in A_\alpha} \hat{\psi}_k. \tag{5.31}$$

Now we consider updating a potential ϕ_j which has no intersection with any of the potentials in B_α (i.e. $B_j = \emptyset$). We apply Equation 5.22 and extend the summations as in the previous section except that the first summation is extended to be only over A_α.

$$\phi_j^* = \left\langle \sum_{k \in A_\alpha} \hat{\psi}_k - \sum_{i \neq j} \phi_i + \sum_{l,m} \phi_{lm} \right\rangle_{\mathbf{T}_\alpha \setminus c_j} - z'. \tag{5.32}$$

Substituting in Equation 5.31 gives

$$\phi_j^* = \left\langle \sum_{k \in A_\alpha} \hat{\psi}_k - \left(\sum_{k \in A_\alpha} \hat{\psi}_k - \phi_j \right) \right\rangle_{\mathbf{T}_\alpha \backslash c_j} - z' = \phi_j \tag{5.33}$$

and so, once again, the update has left the potential ϕ_j unchanged.

Unfortunately, updating a potential which does intersect with at least one of the potentials in B_α is not as straightforward. Suppose ϕ_j is now such a potential (and so $B_j \neq \emptyset$), the update equation becomes

$$\phi_j^* = \left\langle \sum_{k \in A_\alpha} \hat{\psi}_k + \sum_{k \in B_j} \hat{\psi}_k - \sum_{i \neq j} \phi_i + \sum_{l,m} \phi_{lm} \right\rangle_{\mathbf{T}_\alpha \backslash c_j} - z'. \tag{5.34}$$

Now let us apply this to a particular potential ϕ_m which we wish to update first. As we have just initialised all potentials, Equation 5.34 becomes

$$\phi_m^* = \phi_m + \sum_{k \in B_m} \left\langle \hat{\psi}_k \right\rangle_{\mathbf{T}_\alpha \backslash c_m} - z \tag{5.35}$$

where the expectation is with respect to the conditional distribution defined by

$$Q(\mathbf{T}_\alpha \backslash c_m \,|\, c_m) = \frac{Q(\mathbf{T}_\alpha)}{Q(c_m)} = \frac{\prod_{i \neq m} e^{\phi_i}}{\prod_{k,l} e^{\phi_{kl}}}. \tag{5.36}$$

As a potential has been changed, the junction tree is now inconsistent and we must perform `DistributeEvidence`(c_m). This procedure will modify all the other cluster potentials and all separator potentials, whose new states will be denoted ϕ_i^* and ϕ_{kl}^*. It is important to note that the conditional distribution of Equation 5.36 refers to the potentials *prior* to performing `DistributeEvidence`. The expectations in Equation 5.34 will therefore no longer be available after the tree has been made consistent and so we define

$$\lambda_{mk}(c_m \cap d_k) = \left\langle \hat{\psi}_k \right\rangle_{\mathbf{T}_\alpha \backslash c_m} \tag{5.37}$$

and store each λ_{mk} before `DistributeEvidence` is performed. Each λ_{mk} can be thought of as a pseudo-separator potential between the cluster c_m and each d_k.

Now suppose we want to update a general potential ϕ_j^* which has $B_j \neq \emptyset$. We use

$$\phi_j^{**} = \phi_j^* + \sum_{k \in B_j} \left\langle \hat{\psi}_k - \sum_{i \in \mathcal{C}_k} \lambda_{ik} \right\rangle^*_{\mathbf{T}_\alpha \backslash c_j} - z \tag{5.38}$$

where the * above the expectation reminds us that it refers to the new $Q^*(\mathbf{T}_\alpha)$ defined by the updated potentials ϕ_i^* and ϕ_{kl}^*. The set $\mathcal{C}_k$ is the set of clusters that intersect with d_k, excluding c_j itself. The potentials λ_{ik} are set to zero if the cluster c_i has not yet been updated.

Following this update, the potential λ_{jk} is set to be

$$\lambda_{jk}(c_j \cap d_k) = \left\langle \hat{\psi}_k - \sum_{i \in \mathcal{C}_k} \lambda_{ik} \right\rangle^*_{\mathbf{T}_\alpha \setminus c_j.} \tag{5.39}$$

As the potential for cluster c_j has been modified, we must perform `DistributeEvidence`(c_j) to regain a consistent junction tree. As we go on to update the next potential, the required expectations must be with reference to the newly calculated $Q^{**}(\mathbf{T}_\alpha)$ and so on for all remaining potentials.

It follows that $Q(\mathbf{T}_\alpha)$ can be optimised following a consistent initialisation by updating only the potentials for clusters c_j where $B_j \neq \emptyset$. It is, however, necessary to perform `DistributeEvidence` after updating each of these potentials. Depending on the graphs of P and Q, this may well provide significant saving in computation over the naive approach. A further saving can be made by noting that when updating all but the last potential, a reduced `DistributeEvidence` scheme can be used which stops once messages have been received by all clusters c_j where $B_j \neq \emptyset$.

5.5 An Algorithm for Structured Variational Inference

I will now present Structured Variational Message Passing (SVMP) – a general-purpose algorithm which allows variational inference using a structured Q distribution. To keep the explanation of SVMP as straightforward as possible, I will start by dealing only with Case I and then extend the algorithm to handle Case II as well.

5.5.1 Allowable models

In standard Variational Message Passing (Chapter 2), it was required that all conditional probabilities $P(X_k \,|\, \mathrm{pa}_k)$ were members of the exponential family, so that natural parameter vectors (or equivalent) could be used as messages. In order to be able to use a similar form of messages for SVMP, this constraint is extended to require that all junction tree potentials can be expressed in (multivariate) exponential family form as well. The result of this restriction is that we can only retain edges of P in our Q distribution if they lie between nodes whose conditional probabilities have the same functional form. Thus, a subgraph of Q will be tractable if it contains either (1) only discrete nodes in which the conditional distribution of a node given its parents is given by a pick function over the states of the parents, or (2) it comprises Gaussian nodes each of whose mean is a multi-linear function of its parents. In fact, a subgraph of mixed discrete and Gaussian nodes will also be tractable provided there are no edges representing a Gaussian parent with a discrete child – however, I do not consider this more complex case in this analysis.

I shall therefore define our Q distribution by deleting edges of P that do not lie between two discrete or linear-Gaussian nodes. Whilst the user may choose to delete further edges to

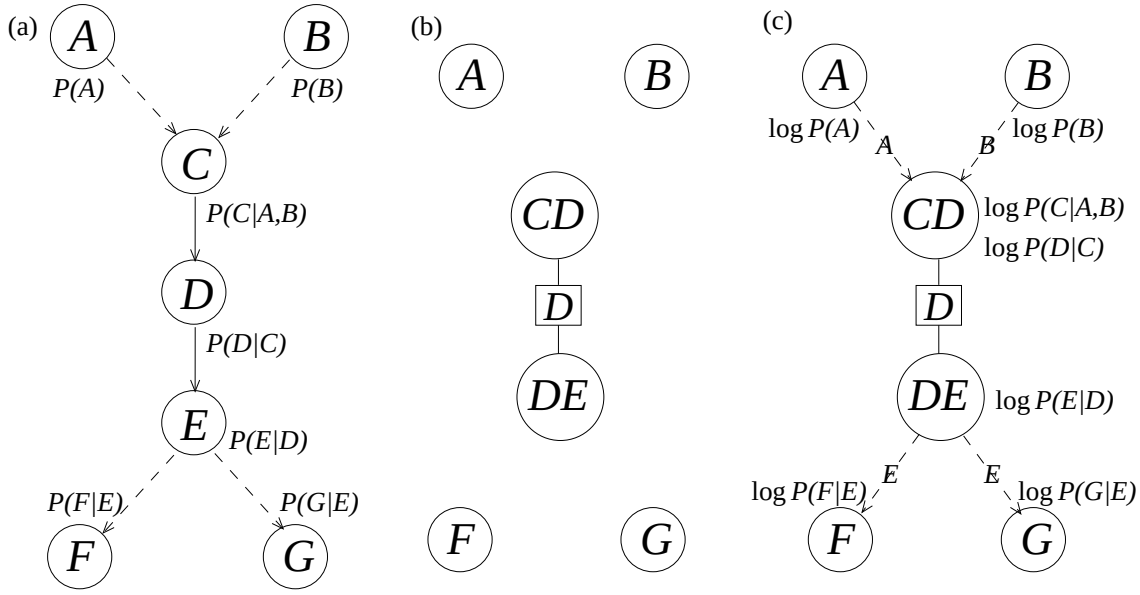


Figure 5.3: (a) The Bayesian network for P showing conditional probabilities. The dashed lines indicate which edges will be deleted in Q. (b) The set of five disjoint junction trees, four of which consist of single nodes, which are created when constructing the junction tree representation of Q. (c) The modified cluster graph for SVMP showing how the log conditionals have been assigned to clusters. The dashed arrows show where messages will be passed between the trees and what variable those messages will depend on.

improve the speed of inference at the expense of some further restriction on the form of the Q distribution, such deletions are not necessary for tractability. To help choose which further edges to delete (if any), it would be straightforward for an implementation of this algorithm to assist the user by providing guidance on the expected computation time of each iteration based on resultant clique sizes.

5.5.2 Structured Variational Message Passing algorithm

The first stage of the Structured VMP algorithm is to construct the junction tree (or set of trees) for the Q distribution. Starting with the directed graph for the Q distribution, we follow the three steps (moralisation, triangulation and construction) outlined in Section 5.3 which results in a set of one or more disjoint junction trees. For example, suppose P takes the form shown in Figure 5.3a where the dashed edges are those which will be deleted in Q. The initial step of SVMP will lead to the set of junction trees shown in Figure 5.3b. Note that all but one of these trees consist of a single node.

The second stage of SVMP involves associating each log conditional $\log P(X_k \,|\, \mathrm{pa}_k)$ with a cluster c_j that contains both X_k and all of the parents pa_k which lie in the same tree as c_j. As Case I applies, this procedure is guaranteed to be possible. I then define A_j to be the set of log conditionals associated with cluster c_j. The only variables in each log conditional that are not in c_j must lie in other junction trees and this is marked graphically using a dashed

arrow pointing to c_j from the cluster c_i containing the log conditional for each such 'external' variable. The arrow is labelled with the names of the external variable or variables and this set of variables is denoted d_{ij}. These directed edges will be used to allow messages (which are functions of d_{ij}) to be passed between junction trees. The result of applying this process to our example is shown in Figure 5.3c. Note that the dashed arrows have no separators. Separators are only required when two cluster potentials are being made jointly consistent – which is not the case here as, being in different junction trees, the connected clusters have no variables in common.

The next stage is to initialise the cluster potentials to give a consistent Q distribution. This can be achieved in a simple fashion by setting each cluster potential to a suitably broad setting (e.g. uniform for discrete potentials) and then applying `CollectEvidence` and `DistributeEvidence` in each tree.

Given this consistent starting point, optimisation of the Q distribution factor for an individual junction tree $\mathbf{T}_\alpha$ can begin. Following the procedure described in Section 5.4.1, the separators are initialised to zero and each cluster potential c_i is initialised using

$$\phi_i(c_i) = \sum_{k \in A_i} \langle \log P(X_k \,|\, \mathrm{pa}_k) \rangle_{\mathrm{pa}_k \setminus c_i} + \sum_{j \in \mathrm{ch}_i} \sum_{k \in A_j} \langle \log P(X_k \,|\, \mathrm{pa}_k) \rangle_{c_j \setminus c_i} \tag{5.40}$$

where ch_i is the set of clusters which are connected by dashed edges pointing away from c_i (which will *not* include any clusters in $\mathbf{T}_\alpha$). Thus, in Case I, we need only define an inter-tree parent-to-child message from parent cluster c_m to c_i

$$m_{c_m \to c_i} = \mathrm{Moments}(Q(d_{mi})) \tag{5.41}$$

and an inter-tree child-to-parent message from child c_j to c_i

$$m_{c_j \to c_i} = \sum_{k \in A_j} \mathrm{Natural}(\langle \log P(X_k \,|\, \mathrm{pa}_k) \rangle_{Q(c_j \setminus d_{ji})}) \tag{5.42}$$

where the functionals Moments() and Natural() are as defined in Section 2.5.2. These messages allow the initialisation of each cluster potential using

$$\mathrm{Natural}(\phi_i(c_i)) = \sum_{k \in A_i} \mathrm{Natural}(\langle \log P(X_k \,|\, \mathrm{pa}_k) \rangle_{Q(\mathrm{pa}_k \setminus c_i)}) + \sum_{j \in \mathrm{ch}_i} m_{c_j \to c_i}. \tag{5.43}$$

Following this initialisation, the optimisation of $Q(\mathbf{T}_\alpha)$ can be completed by performing `CollectEvidence` and `DistributeEvidence`.

To illustrate this process, consider initialising the centre tree in the graph of Figure 5.3c. For cluster DE, the incoming messages from F and G are

$$m_{F \to DE} = \mathrm{Natural}(\langle \log P(F \,|\, E) \rangle_F) \tag{5.44}$$

$$m_{G \to DE} = \mathrm{Natural}(\langle \log P(G \,|\, E) \rangle_G) \tag{5.45}$$

and so ϕ_{DE} is initialised to

$$\text{Natural}(\phi_{DE}) = \text{Natural}(\log P(E \mid D)) + m_{F \to DE} + m_{G \to DE}. \tag{5.46}$$

For cluster CD, the incoming messages are:

$$m_{A \to CD} = \text{Moments}(Q(A)) \tag{5.47}$$

$$m_{B \to CD} = \text{Moments}(Q(B)) \tag{5.48}$$

and therefore ϕ_{CD} is initialised to

$$\text{Natural}(\phi_{CD}) = \text{Natural}(\log P(D \mid C)) + \text{Natural}(\langle \log P(C \mid A, B) \rangle_{Q(A)Q(B)}). \tag{5.49}$$

Applying `CollectEvidence` and `DistributeEvidence` will result in the optimal distribution for $Q(C, D, E)$, given that all other factors of Q are held fixed. A summary of the entire SVMP algorithm is given in Algorithm 5.1.

Algorithm 5.1 The Structured Variational Message Passing (SVMP) Algorithm

1. Moralise the graph of Q, triangulate and construct junction tree(s).
2. Associate each log conditional $\log P(X_k \mid \text{pa}_k)$ with a cluster c_j in a tree $\mathbf{T}_\alpha$ that contains $\mathbf{T}_\alpha \cap (X_k \cup \text{pa}_k)$ and create dashed edges to clusters containing variables in $(X_k \cup \text{pa}_k) \backslash c_j$.
3. Initialise each junction tree to a consistent distribution.
4. For each junction tree in turn:
   (a) Initialise all separator potentials to zero;
   (b) For each cluster, set its potential using Equation 5.43 and messages received from all parent and children clusters (which will lie in other junction trees);
   (c) Perform `CollectEvidence` and `DistributeEvidence` on the tree.
5. Repeat from step 4.

The SVMP algorithm is guaranteed to optimise one factor $Q(\mathbf{T}_\alpha)$ given the remaining factors, except in the (avoidable) circumstances which will now be described. Consider the graph shown in Figure 5.4a and the corresponding Q junction trees of Figure 5.4b. In updating the factor $Q(Y)$ we need to compute the expectation of $P(Y \mid A, E)$ with respect to the variational posterior distribution $Q(A, E)$. As it stands, this will be represented by a product of marginals $Q(A)Q(E)$ given by the two message to Y from A and E. We can extend the formalism to capture correctly the correlation between A and E by adding an edge connecting these two nodes, thereby ensuring that they are in the same cluster of the junction tree. This can be achieved in general by moralising nodes such as Y before removing the edges.

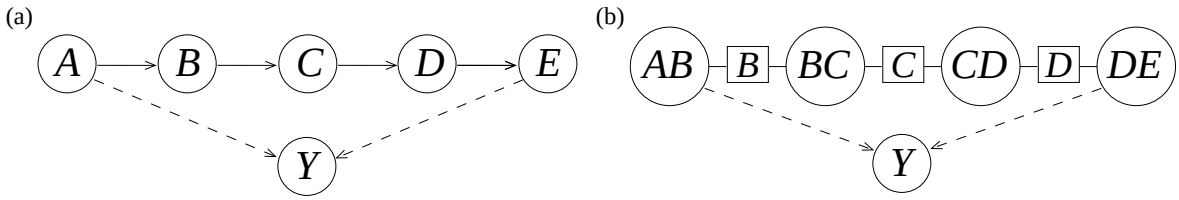


Figure 5.4: (a) Example graph showing the need for additional moralisation in order to find optimal variational marginals. The Q distribution structure for this example is defined by the subgraph comprising the Markov chain at the top, with the dashed links removed. (b) The junction trees of the Q distribution, with dashed arrows showing where messages will pass to (and from) node Y.

5.5.3 Extending the algorithm to allow internal deleted edges (Case II)

As stated earlier, Case II is the situation where there is (at least) one pair of variables in a junction tree $\mathbf{T}_\alpha$ of Q that are connected by an edge in P, but do not lie in the same cluster in Q (for example nodes C and E in Figure 5.5a). It follows that there is at least one conditional distribution $P(X_j \,|\, \mathrm{pa}_j)$ where the intersection $\mathbf{T}_\alpha \cap (X_j \cup \mathrm{pa}_j)$ cannot be found in any cluster of $\mathbf{T}_\alpha$.

The existing Algorithm 5.1 will fail at step 2 when it tries to associate these conditionals with clusters, because no cluster will contain the required variables. We can extend the algorithm to handle these 'uncontained' conditionals $\{\hat{\psi}_j\}$, by creating a new cluster Ω_j

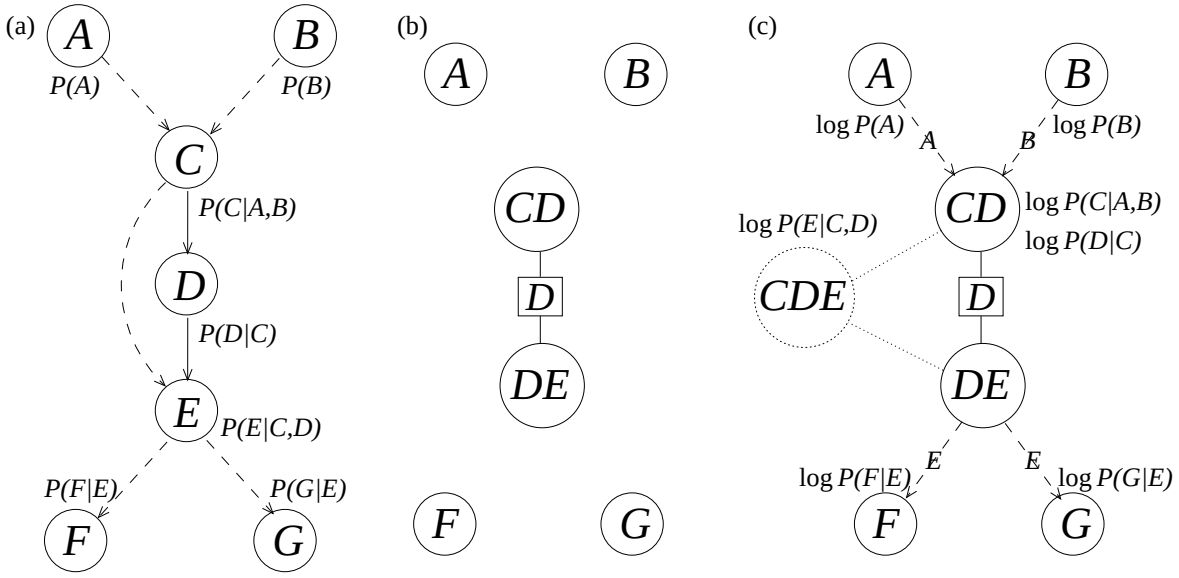


Figure 5.5: (a) The Bayesian network for P where dashed lines indicate which edges will be deleted in Q. Note that the addition of dashed edge between C and E, compared to the model of 5.3a. (b) The set of junction trees of Q. These are identical to 5.3b as the structure of Q is unchanged. (c) The modified cluster graph for SVMP showing how a P-cluster (shown dotted) has been created for the conditional $P(E \,|\, C, D)$ which is not contained in any cluster of Q. This P-cluster is connected to clusters of Q with which it has any variables in common.

containing the variables $\mathbf{T}_\alpha \cap (X_j \cup \mathrm{pa}_j)$ for each such conditional (see for example, the CDE cluster shown dotted in Figure 5.5c). These new clusters will be denoted P-clusters (as they correspond to factors of P). Each P-cluster has a potential ω_j and is connected to all clusters in $\mathbf{T}_\alpha$ which have any variables in common with the P-cluster. We define B_i to be the set of P-clusters connected to a standard cluster c_i.

The algorithm then proceeds as before, ignoring the P-clusters and their corresponding conditionals until after the `CollectEvidence` and `DistributeEvidence` procedures at the end of step 4. At this point, the extended algorithm applies the changes required to take into account the effect of the ignored conditionals. Firstly, each ω_j potential is initialised to the corresponding $\hat{\psi}_j$ potential. Secondly, we take each standard cluster c_i which is connected to at least one P-cluster (so $B_i \neq \emptyset$) and update its potential using,

$$\phi_i^* = \phi_i + \sum_{j \in B_i} \langle \omega_j \rangle_{\Omega_j \setminus c_i} - z, \tag{5.50}$$

and immediate modify each connected P-cluster potential using

$$\omega_j \leftarrow \omega_j - \langle \omega_j \rangle_{\Omega_j \setminus c_i} . \tag{5.51}$$

This can be achieved using message passing by sending a message from the P-cluster Ω_j to the cluster being updated c_i of the form

$$m_{\Omega_j \to c_i} = \mathrm{Natural}(\langle \omega_j \rangle_{\Omega_j \setminus c_i}) \tag{5.52}$$

and subtracting it locally from ω_j. This message can be computed in the P-cluster Ω_j only if it has first received messages from all other clusters $c_k, k \neq i$ of the form

$$m_{c_k \to \Omega_j} = \mathrm{Moments}(Q(c_k \cap \Omega_j)). \tag{5.53}$$

The cluster potential is then updated using

$$\mathrm{Natural}(\phi_i^*) = \mathrm{Natural}(\phi_i) + \sum_{j \in B_i} m_{\Omega_j \to c_i}. \tag{5.54}$$

Following this update, the variational junction tree will no longer be consistent and so `DistributeEvidence`(c_i) must be performed. This update procedure is then repeated for all other clusters where $B_i \neq \emptyset$. Algorithm 5.2 on the next page gives a summary of this extended SVMP algorithm.

Algorithm 5.2 Extended SVMP Algorithm

1. Moralise the graph of Q, triangulate and construct junction tree(s).
2. Associate each log conditional $\log P(X_k \mid \mathrm{pa}_k)$ with a cluster c_j in a tree $\mathbf{T}_\alpha$ that contains $\mathbf{T}_\alpha \cap (X_k \cup \mathrm{pa}_k)$ and create dashed edges to clusters containing variables in $(X_k \cup \mathrm{pa}_k) \backslash c_j$. Where a conditional cannot be associated a cluster, create a new P-cluster and connect it to all clusters with any variables in common.
3. Initialise each junction tree to a consistent distribution.
4. For each junction tree in turn:
   (a) Initialise all separator potentials to zero;
   (b) For each cluster, set its potential using Equation 5.43 and messages received from all parent and children clusters (which will lie in other junction trees);
   (c) Perform `CollectEvidence` and `DistributeEvidence` on the tree;
   (d) Initialise each ω_j to $\hat{\psi}_j$;
   (e) For each cluster c_i where $B_i \neq \emptyset$, update its potential and all connected ω_j potentials using Equations 5.52–5.54. Then perform `DistributeEvidence`(c_i).
5. Repeat from step 4.

5.6 Structured VIBES: A Partial Implementation of SVMP

A full implementation of the Structured Variational Message Passing algorithm would be a significant undertaking. Instead, the VIBES software described in Section 2.3 has been extended to implement a partial form of SVMP with the following constraints:

- retained edges in Q had to be between discrete nodes (leading to discrete multivariate potentials). Nodes with no connections in Q could have any distribution supported by standard VIBES;
- the retained structure of Q could not have cycles (i.e. had to be a tree), thus removing the need for moralisation and triangulation steps;
- edges between nodes connected by another path in Q could not be deleted (so only Case I was allowed).

5.6.1 Example: Hidden Markov Model

The extended VIBES software will be illustrated using a Bayesian hidden Markov model in which prior distributions are defined over the probabilities for the initial state of the hidden variables as well as over the transition and emission matrices. This model was described, and also solved variationally, in MacKay [1997]. In order to highlight the comparison against the structured framework we have allowed all of the variables to be unobserved. The screen shot

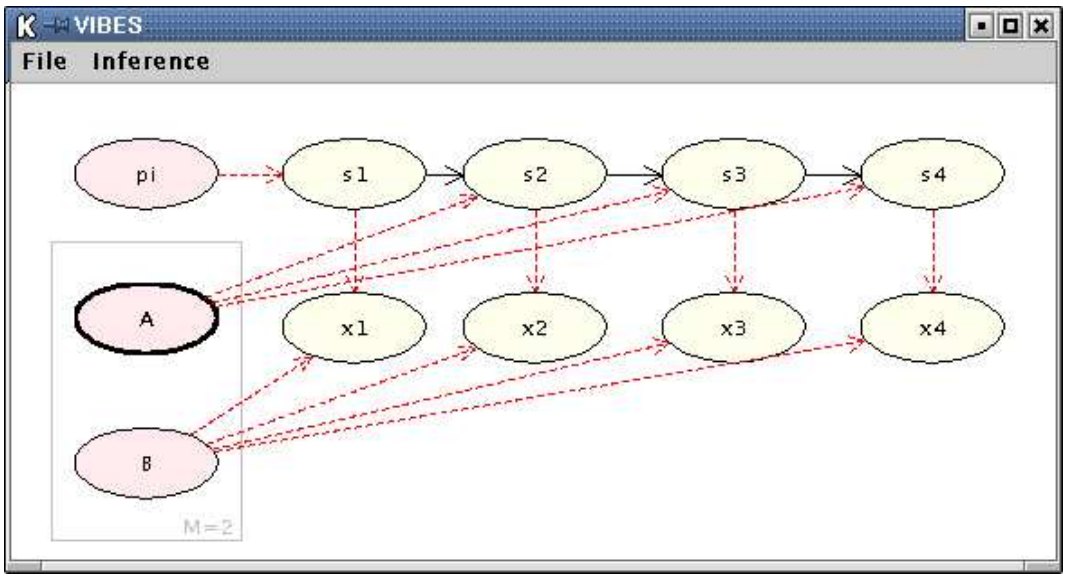


Figure 5.6: VIBES screenshot showing the graphical model defining the distribution for a Bayesian hidden Markov model. Links shown in dashed red are those that will be removed in defining the structured Q distribution, while those shown in black will remain.

from VIBES for the directed graph defining the $P(X)$ distribution is shown in Figure 5.6. As a point of comparison I first solve this model using the fully factorised variational approximation using standard VIBES. Next I apply a structured variational approximation as shown in Figure 5.7 using SVMP. Here the links along the hidden Markov chain are retained, leading to a more flexible class of Q distributions. On this tiny model, the converged value of the lower bound $\mathcal{L}$ for the structured distribution of Figure 5.7 is 3.873 nats compared to 3.631

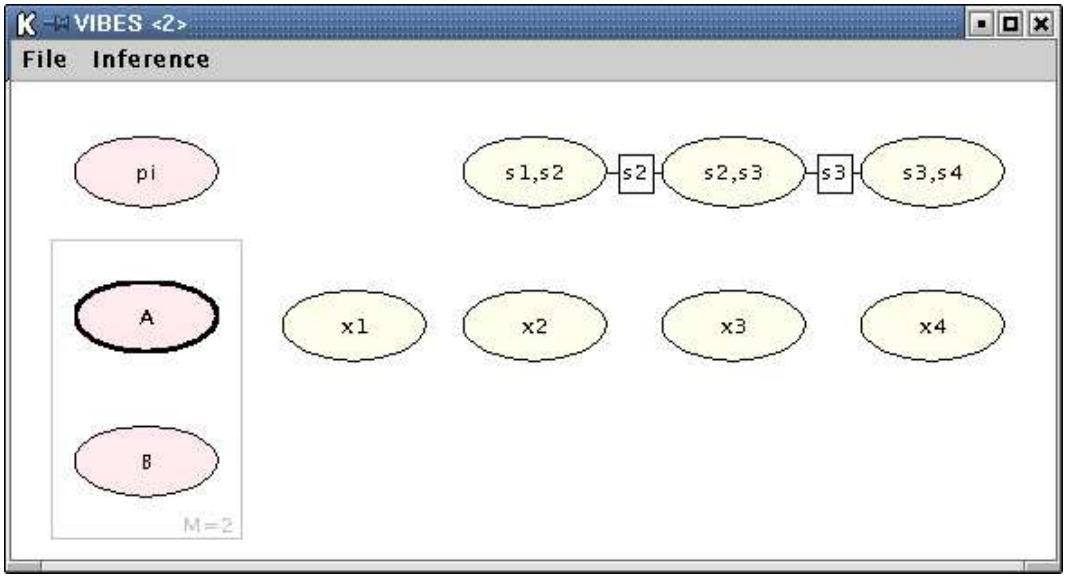


Figure 5.7: VIBES screenshot of the HMM structured variational distribution, showing the junction tree.

nats for the fully factorised distribution, showing that the use of a structured Q distribution leads to an improved approximation.

5.7 Discussion

I have demonstrated an algorithm, Structured Variational Message Passing, which allows variational inference to be performed automatically using a Q distribution that retains some of the structure of the original P distribution. In addition, a partial implementation of this algorithm has been shown to give an improved approximation over standard VMP for a Hidden Markov model. As belief propagation is a special case of SVMP, a full implementation of SVMP would be a powerful tool, able to:

- perform exact inference (on pure discrete or linear Gaussian graphs) where clique sizes would make this practicable;
- perform approximate inference on all other conjugate-exponential graphs (including mixed continuous and discrete graphs) whilst allowing the approximating Q distribution to retain some of the structure of the P distribution;
- allow the accuracy of the approximation to be adjusted downwards to reduce computation time, until the limiting case of a fully-factorised Q is reached.

It is also possible to consider Q distributions represented by a graph that is not a sub-graph of the original P distribution. In fact, my assumption that the Q distribution is created by deleting edges of P is not required by SVMP, which allows any structure to be used in the Q distribution, provided that all resultant Q potentials are tractable.

Chapter 6

Conclusions and Future Work

In the introduction, I expressed the hope that the work in this thesis could be a 'first step' towards a general purpose inference system. In this final chapter, I will conclude by describing the progress made towards this goal in terms of my development of a variational inference framework and its application to problems in a range of domains. I will also suggest some future research directions that could provide the next steps along the path to a practical and widely applicable inference system.

6.1 Conclusions

The aim of this thesis has been to develop a framework for automatically performing variational inference. In Chapter 1, I explained that such automatic inference systems will meet the increasing need for machine-based reasoning in a range of applications. I demonstrated the importance of modelling uncertainty in such systems and showed how probabilistic models and Bayesian inference provide both the representation and the method for a powerful approach to automated learning. Belief propagation, an algorithm for performing exact Bayesian inference, was used as an example of how message passing in a graph allows the use of general-purpose inference algorithms. Unfortunately, exact inference is intractable in many models, leading to the requirement for approximate inference methods. I focussed on one such technique, variational inference, as a deterministic method which gives an analytic approximation to the posterior and has been shown to be successful in a wide range of models. In variational inference, an approximating distribution Q is selected which has a simpler form than the true posterior P. This variational distribution is then recursively adapted to be as close to the true posterior as possible, in terms of KL divergence. On convergence, Q provides a functional approximation to P, whose quality depends on the data set and initialisation.

One problem with variational inference is that implementing it for new models is extremely time consuming and prone to errors. In Chapter 2, I described a general purpose framework which can apply variational inference automatically to a large class of probabilistic models and so avoid these implementation problems. The framework uses message-passing in a graphical model, leading to the algorithm being called Variational Message Passing (VMP). In VMP,

the Q distribution is fully-factorised and hence the approximate posterior is separable with respect to all variables. I showed that the class of models that VMP supports includes all conjugate-exponential models (discrete, continuous and mixed) and it allows for both mixtures and deterministic relationships between variables. I also investigated some extensions to VMP that allow non-conjugate hyper-priors and the option of finding Maximum A Posteriori values for some variables.

The VMP framework has been implemented in a software package called VIBES (Variational Inference in BayESian networks). This Java software allows the model to be specified graphically, by simply drawing the Bayesian network and assigning conditional probabilities through a graphical user interface. Variational inference can then be applied automatically. During inference, individual variables can be monitored by a range of methods and the lower bound on the evidence automatically calculated and displayed. Experience with VIBES has shown that it dramatically simplifies the construction and testing of new variational models and readily allows a range of alternative models to be evaluated on a given problem. This has been illustrated by applying VIBES to problems in the domains of machine vision and of DNA microarrays.

In Chapter 3, the problem of modelling nonlinear image subspaces was addressed using the VMP framework and VIBES. A mixture of Bayesian Principal Component Analysis models was developed and shown to be able to model the nonlinear manifolds of sets of digit and face images, as well as segments of a natural image. By also implementing variational inference by hand, the significant time and effort saving of using VIBES was demonstrated. The resultant system was shown to be able to determine manifold shape and dimensionality automatically from the data, avoiding the need for cross-validation.

The new technology of DNA microarrays has raised a range of inference problems, two of which were addressed in Chapter 4. The main problem investigated was that of analysing the scanned images of DNA microarrays in order to determine the corresponding gene expression levels. A model of the image process was developed with a latent variable for the intensity distribution of each spot (corresponding to the expression of a single gene) which, combined with the spot locations and shapes, led to the observed image. As this model was not conjugate-exponential, the VMP algorithm was extended to allow importance sampling in the non-exponential parts of the graph. I showed that this extended algorithm allowed the model to be used to analyse images successfully. I also introduced models for missing and obscured spots which allowed these to be identified and handled separately. The second problem of how to analyse the resultant gene expression data was investigated and VMP was used to perform independent component analysis on a small gene expression data set. The aim of this experiment was to motivate the development of complex models for gene expression data (for example, based on knowledge of the latent biological processes) by using automatic inference systems, such as VIBES, to construct and evaluate each model.

Finally, in Chapter 5, I investigated the use of Q distributions which are not fully factorised and hence retain some dependencies between variables in the posterior approximation.

The use of such a structured Q distribution guarantees at least as good an approximation as a fully factorised distribution and, in general, a better one as it can capture posterior correlations between variables. The particular form of Q distribution investigated was a junction tree (or set of trees) because it allows local computation of marginals – essential for the development of local, message passing algorithms. I demonstrated how inference can be performed efficiently using such variational junction trees by creating an extended form of VMP called Structured VMP (SVMP). VIBES was extended to include a partial implementation of SVMP and applied to show the advantages of SVMP over VMP in a small Hidden Markov Model.

The use of inference frameworks, such as VIBES, separates the design of a probabilistic model from the process of performing inference within the model. This separation permits the construction and comparison of domain models to be left to the domain expert without their needing to understand the learning and inference process in detail. Equally, it lets the machine learning researcher concentrate on improving inference methods, since the comparative performance of different methods can be readily assessed using not only the same data set but also the same probabilistic model.

6.2 Suggestions for Future Work

A number of open problems must be solved to allow the development of a truly general purpose learning and inference system. These problems suggest a variety of research directions that need to be pursued to make such a system feasible.

One such direction would be to investigate allowing automatic learning of the structure of the probabilistic model. The current framework requires that the model be specified explicitly. It would be preferable that an initial model be suggested and the framework allowed to adapt or extend it so as to best fit the data. Bayesian model selection provides a mechanism for choosing between models and so learning one for a particular data set or domain.

Another possibility would be to create hybrid sampling/variational inference systems which would allow models outside the conjugate-exponential family. This idea was touched upon in Chapter 2 and led to the use of VMP with Importance Sampling in Chapter 4, and is one that would certainly merit further investigation. It would also be very valuable to perform a comparison of sampling methods (such as Markov chain Monte Carlo) and variational methods (with structured and unstructured variational distributions). The aim would be to compare speed and accuracy of approximation for identical models on real and toy data sets.

Expectation propagation provides an alternative to variational methods for Bayesian inference. As there are many similarities between the two, it seems likely that VIBES could be modified to use expectation propagation instead of, or in combination with, VMP. The resulting framework would allow comparison of the two methods and hence give an understanding of the strengths and weaknesses of each. The extended framework would then allow the appropriate method to be applied automatically to novel models, providing a more general

purpose inference tool.

Finally, in terms of applications of the variational inference framework, there are a plethora of possible areas in which it can be applied, from social sciences to cosmology. My particular interest would be to use the framework to learn more complex models of natural image formation, as a means of performing the more general image analysis required for machine vision applications.

6.3 Summary

In summary, I have created a framework which allows variational inference to be performed automatically in a wide range of probabilistic models. The framework has been used to solve difficult problems in the real world domains of machine vision and DNA microarray analysis. I have also demonstrated that inference with structured variational distributions is both tractable and capable of being performed automatically. The use of such structured distributions both improves the quality of the approximation and extends the applicability of the original framework.

Appendix A

Exponential Family Distributions

This appendix lists the exponential family distributions referred to in this thesis. Each distribution is written in its ordinary form and then in the standard exponential family form, defined as,

$$P(x \,|\, \mathbf{y}) = \exp[\boldsymbol{\phi}(\mathbf{y})^{\mathrm{T}} \mathbf{u}(x) + f(x) + g(\mathbf{y})] \tag{A.1}$$

where $\boldsymbol{\phi}(\mathbf{y})$ is the natural parameter vector and $\mathbf{u}(x)$ is the natural statistic vector. For each distribution P, the expectation of the natural statistic vector under P is also given.

A.1 Gaussian distribution

The Gaussian distribution, also known as the Normal distribution, is defined as

$$\begin{aligned} P(x) &= \mathcal{N}(x \,|\, \mu, \gamma^{-1}) \\ &= \sqrt{\frac{\gamma}{2\pi}} \exp\left[-\frac{\gamma}{2}(x-\mu)^2\right] \quad \gamma > 0 \end{aligned} \tag{A.2}$$

where μ is the mean and γ is the precision or inverse variance. Being a member of the exponential family, this distribution can then be written in standard form as

$$\mathcal{N}(x \,|\, \mu, \gamma^{-1}) = \exp\left(\begin{bmatrix} \gamma\mu \\ -\frac{\gamma}{2} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} x \\ x^2 \end{bmatrix} + \tfrac{1}{2}(\log\gamma - \gamma\mu^2 - \log 2\pi) \right). \tag{A.3}$$

The expectation under P of the natural statistic vector $\mathbf{u}_x$ is given by

$$\langle \mathbf{u}(x) \rangle_P = \left\langle \begin{bmatrix} x \\ x^2 \end{bmatrix} \right\rangle_P = \begin{bmatrix} \mu \\ \mu^2 + \gamma^{-1} \end{bmatrix}. \tag{A.4}$$

The entropy of the Gaussian distribution can be found by replacing $\mathbf{u}(x)$ by $\langle \mathbf{u}(x) \rangle_P$ in the negative log of (A.3), giving

$$\mathbb{H}(P) = \tfrac{1}{2}(1 + \log 2\pi - \log\gamma). \tag{A.5}$$

A.2 Rectified Gaussian distribution

The rectified Gaussian distribution is equal to the Gaussian except that it is zero for negative x, so

$$\begin{aligned} P(x) &= \mathcal{N}^R(x \,|\, \mu, \gamma^{-1}) \\ &= \begin{cases} \frac{\sqrt{2\gamma}}{\sqrt{\pi}\,\mathrm{erfc}(-\mu\sqrt{\gamma/2})} \exp\left[-\frac{\gamma}{2}(x-\mu)^2\right] & : \quad x \geq 0 \\ 0 & : \quad x < 0 \end{cases} \end{aligned} \tag{A.6}$$

where erfc is the complementary error function

$$\mathrm{erfc(x)} = \frac{2}{\sqrt{\pi}} \int_x^\infty \exp(-t^2)\, dt. \tag{A.7}$$

The rectified Gaussian distribution has the same natural parameter and statistic vectors as the standard Gaussian. It differs only in f and g, which are now defined as

$$f(x) = \begin{cases} 0 & : \quad x \geq 0 \\ -\infty & : \quad x < 0 \end{cases} \tag{A.8}$$

$$g(\mu, \gamma) = \tfrac{1}{2}(\log 2\gamma - \log \pi - \gamma\mu^2) - \log \mathrm{erfc}(-\mu\sqrt{\gamma/2}). \tag{A.9}$$

The expectation of the natural statistic vector is

$$\left\langle \begin{bmatrix} x \\ x^2 \end{bmatrix} \right\rangle_P = \begin{bmatrix} \mu + \sqrt{\frac{2}{\pi\gamma}} \frac{1}{\mathrm{erfcx}(-\mu\sqrt{\gamma/2})} \\ \mu^2 + \gamma^{-1} + \sqrt{\frac{1}{\pi\gamma}} \frac{\mu}{\mathrm{erfcx}(-\mu\sqrt{\gamma/2})} \end{bmatrix} \tag{A.10}$$

where the scaled complementary error function $\mathrm{erfcx(x)} = \exp(x^2) \times \mathrm{erfc}(x)$. For notes on avoiding numerical errors when using Equation A.10, see Miskin [2000].

A.3 Gamma distribution

The Gamma distribution is defined over nonnegative x to be

$$\begin{aligned} P(x) &= \mathrm{Gamma}(x \,|\, a, b) \\ &= \frac{b^a x^{a-1} \exp(-bx)}{\Gamma(a)} \quad x \geq 0;\ a, b > 0 \end{aligned} \tag{A.11}$$

where a is the shape and b is the inverse scale. Rewriting in exponential family standard form gives

$$\mathrm{Gamma}(x \,|\, a, b) = \exp\left(\begin{bmatrix} -b \\ a-1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} x \\ \log x \end{bmatrix} + a \log b - \log \Gamma(a) \right) \tag{A.12}$$

and the expectation of the natural statistic vector is

$$\left\langle \begin{bmatrix} x \\ \log x \end{bmatrix} \right\rangle_P = \begin{bmatrix} a/b \\ \Psi(a) - \log b \end{bmatrix} \tag{A.13}$$

where we define the digamma function $\Psi(a) = \frac{\delta}{\delta a} \log \Gamma(a)$. It follows that the entropy of the Gamma distribution is

$$\mathbb{H}(P) = a + (1-a)\Psi(a) - \log b + \log \Gamma(a). \tag{A.14}$$

A.4 Discrete distribution

A discrete, or multinomial, distribution is a probability distribution over a variable which can take on only discrete values. If x can only take one of K discrete values, the distribution is defined as

$$\begin{aligned} P(x) &= \text{Discrete}(x \,|\, p_1 \dots p_K) \\ &= \sum_{i=1}^{K} p_i \, \delta(x - i) \qquad x \in \{1 \dots K\}; \sum_{i=1}^{K} p_i = 1 \qquad \text{(A.15)} \\ &= p_x. \qquad \text{(A.16)} \end{aligned}$$

The discrete distribution is a member of the exponential family and can be written in standard form

$$\text{Discrete}(x \,|\, p_1 \dots p_K) = \exp\left(\begin{bmatrix} \log p_1 \\ \vdots \\ \log p_K \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \delta(x-1) \\ \vdots \\ \delta(x-K) \end{bmatrix} \right) \tag{A.17}$$

and the expectation of the natural statistic vector is simply

$$\left\langle \begin{bmatrix} \delta(x-1) \\ \vdots \\ \delta(x-K) \end{bmatrix} \right\rangle_P = \begin{bmatrix} p_1 \\ \vdots \\ p_K \end{bmatrix}. \tag{A.18}$$

The entropy of the discrete distribution is $\mathbb{H}(P) = -\sum_i p_i \log p_i$.

A.5 Dirichlet distribution

The Dirichlet distribution is the conjugate prior for the parameters $p_1 \dots p_K$ of a discrete distribution

$$\begin{aligned} P(p_1 \dots p_K) &= \mathrm{Dir}(p_1 \dots p_K \,|\, \mathbf{u}) \\ &= \frac{1}{Z(\mathbf{u})} \prod_{i=1}^{K} p_i^{u_i - 1} \qquad p_i > 0, \sum_i p_i = 1;\ u_i > 0 \end{aligned} \tag{A.19}$$

where u_i is the ith element of the parameter vector $\mathbf{u}$ and the normalisation constant $Z(\mathbf{u})$ is

$$Z(\mathbf{u}) = \frac{\prod_{i=1}^{K} \Gamma(u_i)}{\Gamma\left(\sum_{i=1}^{K} u_i\right)}. \tag{A.20}$$

The Dirichlet distribution can be written in exponential family form as

$$\mathrm{Dir}(p_1 \dots p_K \,|\, bfu) = \exp\left(\begin{bmatrix} u_1 - 1 \\ \vdots \\ u_K - 1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \log p_1 \\ \vdots \\ \log p_K \end{bmatrix} + \log \Gamma(U) - \sum_{i=1}^{K} \Gamma(u_i) \right) \tag{A.21}$$

where $U = \sum_{i=1}^{K} u_i$. The expectation under P of the natural statistic vector is

$$\left\langle \begin{bmatrix} \log p_1 \\ \vdots \\ \log p_K \end{bmatrix} \right\rangle_P = \begin{bmatrix} \psi(u_1) - \psi(U) \\ \vdots \\ \psi(u_K) - \psi(U) \end{bmatrix} \tag{A.22}$$

where the digamma function $\psi()$ is as defined previously.

Bibliography

D. Ackley, G. Hinton, and T. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

H. Attias. A variational Bayesian framework for graphical models. In S. Solla, T. K. Leen, and K-L Muller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 209–215, Cambridge MA, 2000. MIT Press.

Z. Bar-Joseph, D. Gifford, and T. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17:S22–29, 2001.

D. Barber and C. M. Bishop. Variational learning in Bayesian neural networks. In C. M. Bishop, editor, *Generalization in Neural Networks and Machine Learning*. Springer Verlag, 1998.

K. J. Bathe. *Finite Element Procedures*. Prentice-Hall, Englewood Cliffs, NJ, 1996.

Rev. T. Bayes. An essay towards solving a problem in the doctrine of chances. In *Philosophical Transactions of the Royal Society*, volume 53, pages 370–418, 1763.

A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

J. M. Bernardo and A.F.M. Smith. *Bayesian Theory*. John Wiley and Sons, New York, 1994.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

C. M. Bishop. Bayesian PCA. In S. A. Solla M. S. Kearns and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 382–388. MIT Press, 1999a.

C. M. Bishop. Variational principal components. In *Proceedings Ninth International Conference on Artificial Neural Networks, ICANN'99*, volume 1, pages 509–514. IEE, 1999b.

C. M. Bishop and M. E. Tipping. A hierarchical latent variable model for data visualization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):281–293, 1998.

C. M. Bishop and M. E. Tipping. Variational Relevance Vector Machines. In *Proceedings of 16th Conference in Uncertainty in Artificial Intelligence*, pages 46–53. Morgan Kaufmann, 2000.

C. M. Bishop and J. M. Winn. Non-linear Bayesian image modelling. In *Proceedings Sixth European Conference on Computer Vision*, volume 1, pages 3–17. Springer-Verlag, 2000.

C. M. Bishop and J. M. Winn. Structured variational distributions in VIBES. In *Proceedings Artificial Intelligence and Statistics*, Key West, Florida, 2003. Society for Artificial Intelligence and Statistics.

C. M. Bishop, J. M. Winn, and D. Spiegelhalter. VIBES: A variational inference engine for Bayesian networks. In *Advances in Neural Information Processing Systems*, volume 15, 2002.

M. J. Black and Y. Yacoob. Recognizing facial expressions under rigid and non-rigid facial motions. In *International Workshop on Automatic Face and Gesture Recognition, Zurich*, pages 12–17, 1995.

C. Bregler and S.M. Omohundro. Nonlinear manifold learning for visual speech recognition. In *Fifth International Conference on Computer Vision*, pages 494–499, Boston, Jun 1995.

J. Buhler, T. Ideker, and D. Haynor. Dapple: Improved techniques for finding spots on DNA microarrays. Technical report, University of Washington, 2000.

R. Choudrey, W. Penny, and S. Roberts. An ensemble learning approach to independent component analysis. *In IEEE International Workshop on Neural Networks for Signal Processing*, 2000.

G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models — their training and application. In *Computer vision, graphics and image understanding*, volume 61, pages 38–59, 1995.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer-Verlag, 1999.

R. T. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14(1):1–13, 1946.

P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60:141–153, 1993.

A. Darwiche. Conditioning methods for exact and approximate inference in causal networks. In *Eleventh Annual Conference on Uncertainty in Artificial Intelligence*. Morgan Kauffmann, August 1995.

S. Dudoit, Y. H. Yang, Matthew J. Callow, and T. P. Speed. Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. Technical report, Department of Biochemistry, Stanford University School of Medicine, 2000.

M. Eisen, P. Spellman, D. Botstein, and P. Brown. Cluster analysis and display of genome-wide expression patterns. In *Proceedings of National Academy of Science*, volume 95, pages 14863–14867, 1998.

M.B. Eisen and P.O. Brown. DNA arrays for analysis of gene expression. *Methods in Enzymology*, 303:179–205, 1999.

B.S. Everitt and D.J. Hand. *Finite Mixture Distributions*. Chapman and Hall, London, 1981.

R.P. Feynman. *Statistical Mechanics*. W. A. Benjamin, Inc., MA, 1972.

B. Frey. *Graphical models for machine learning and digital communications*. MIT Press, Cambridge, MA, 1998.

B. Frey and N. Jojic. Transformed component analysis: joint estimation of spatial transformations and image components. In *Seventh International Conference on Computer Vision*, pages 1190–1196, 1999.

B. Frey, F. Kschischang, H. Loeliger, and N. Wiberg. Factor graphs and algorithms. In *Proceedings of the 35th Allerton Conference on Communication, Control and Computing 1997*, 1998.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. In *RECOMB*, pages 127–135, 2000.

R.G. Gallager. Low density parity check codes. *IRE Trans. Info. Theory*, IT-8:21–28, Jan 1962.

R.G. Gallager. *Low density parity check codes*. Number 21 in Research monograph series. MIT Press, Cambridge, MA, 1963.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1): 721–741, 1984.

Z. Ghahramani and M. J. Beal. Variational inference for Bayesian mixture of factor analysers. In *Advances in Neural Information Processing Systems*, volume 12, 1999.

Z. Ghahramani and M. J. Beal. Propagation algorithms for variational Bayesian learning. In T. K. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, Cambridge MA, 2001. MIT Press.

W. R. Gilks and P. Wild. Adaptive rejection sampling for Gibbs sampling. *Applied Statistics*, 41(2):337–348, 1992.

A. Hartemink, D. Gifford, T. Jaakkola, and R. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *Pacific Symposium on Biocomputing*, volume 6, pages 422–433, 2001.

T. Heap and D. Hogg. Wormholes in shape space: Tracking through discontinuous changes in shape. In *Sixth International Conference on Computer Vision*, pages 344–349, 1998.

P. Hegde, R. Qi, R. Abernathy, C. Gay, S. Dharap, R. Gaspard R, J. Earle-Hughes, E. Snesrud, N. H. Lee, and J. Quackenbush. A concise guide to cDNA microarray analysis. *Biotechniques*, 29(3):548–562, 2000.

T. Heskes. Stable fixed points of loopy belief propagation are minima of the Bethe free energy, 2002.

G. E. Hinton and D. van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 5–13, 1993.

G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length, and Helmholtz free energy. In *Advances in Neural Information Processing Systems*, volume 6, 1994.

G. Hori, M. Inoue, S. Nishimura, and H. Nakahara. Blind gene classification on ICA of microarray data. In *ICA 2001*, pages 332–336, 2001.

T. Jaakkola. *Variational Methods for Inference and Estimation in Graphical Models*. PhD thesis, MIT, 1997.

F. Jensen. *An introduction to Bayesian networks*. UCL Press, 1996.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 105–162. Kluwer, 1998.

N. Kambhatla and T.K. Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9(7):1493–1516, 1997.

R. Kinderman and J. L. Snell. Markov random fields and their applications. *American Mathematical Society*, 1:1–142, 1980.

F. R. Kschischang, B. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory*, 47(2):498–519, 2001.

S. Kullback. *Information Theory and Statistics*. Dover Publications, New York, 1959.

S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

S. L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50:157–224, 1988.

S. L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57, 1989.

N. Lawrence, M. Milo, M. Niranjan, P. Rashbass, and S. Soullier. Reducing the variability in microarray image processing by Bayesian inference. Technical report, Department of Computer Science, University of Sheffield, 2002.

D. B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

D. J. Lunn, A. Thomas, N. G. Best, and D. J. Spiegelhalter. WinBUGS – a Bayesian modelling framework: concepts, structure and extensibility. *Statistics and Computing*, 10:321–333, 2000. http://www.mrc-bsu.cam.ac.uk/bugs/.

D. J. C. MacKay. Probable networks and plausible predictions – a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3): 469–505, 1995.

D. J. C. MacKay. Ensemble learning for hidden Markov models, 1997. Unpublished manuscript, Department of Physics, University of Cambridge.

D. J. C. MacKay. Introduction to Monte Carlo methods. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

D. J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, Cambridge, UK, 2003.

D. J .C. MacKay and R. M. Neal. Good codes based on very sparse matrices. In *IMA: IMA Conference on Cryptography and Coding, LNCS lately (earlier: Cryptography and Coding II, Edited by Chris Mitchell, Clarendon Press, 1992)*, 1995.

A. Martoglio, J. W. Miskin, S. K. Smith, and D. J. C. MacKay. A decomposition model to track gene expression signatures: preview on observer-independent classification of ovarian cancer. *Bioinformatics*, 18:1617–1624, 2002.

A. Martoglio, B. D. Tom, M. Starkey, A. N. Corps, S. Charnock-Jones, and S. K. Smith. Changes in tumorigenesis- and angiogenesis-related gene transcript abundance proles in ovarian cancer detected by tailored high density cDNA arrays. *Molecular Medicine*, 6(9): 750–765, 2000.

R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng. Turbo decoding as an instance of Pearl's Belief Propagation algorithm. *IEEE Journal on selected areas in communication*, 1997.

G. S. Michaels, D. B. Carr, M. Askenazi, S. Fuhrman, X. Wen, and R. Somogyi. Cluster analysis and data visualization of large-scale gene expression data. In *Pacific Symposium on Biocomputing*, volume 3, pages 42–53, 1998.

T. P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence*, pages 362–369. Morgan Kauffmann, 2001a.

T. P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT, 2001b.

J. W. Miskin. *Ensemble Learning for Independent Component Analysis*. PhD thesis, University of Cambridge, 2000.

J. W. Miskin and D. J. C. MacKay. Ensemble learning for blind source separation. In S. J. Roberts and R. M. Everson, editors, *ICA: Principles and Practice*. Cambridge University Press, 2000.

B. Moghaddam. Principal manifolds and Bayesian subspaces for visual recognition. In *Seventh International Conference on Computer Vision*, pages 1131–1136, 1999.

B. Moghaddam and A. Pentland. Probabilistic visual learning for object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, 1997.

V.S. Nalwa. *A Guided Tour of Computer Vision*. Addison-Wesley, 1993.

R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.

R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, Cananda, 1993.

R. M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, Canada, 1994.

R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer, 1998.

J. Pearl. Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29: 241–288, 1986.

J. Pearl. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32:245–257, 1987.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, 1988.

S. Raychaudhuri, J. Stuart, and R. Altman. Principal Components Analysis to summarize microarray experiments: application to sporulation time series. In *Pacific Symposium on Biocomputing*, volume 5, 2000.

S. Roweis. EM algorithms for PCA and SPCA. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998.

J. Rustagi. *Variational Methods in Statistics*. Academic Press, New York, 1976.

J. Sakurai. *Modern Quantum Mechanics*. Addison-Wesley, Redwood City ,CA, 1985.

L. K. Saul and M. I. Jordan. Exploiting tractable substructures in intractable networks. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 486–492. MIT Press, 1996.

E. H. Shortcliffe. *Computer-Based Medical Consultations: MYCIN*. Elsevier Science, New York, 1976.

P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization. In *Molecular Biology of the Cell*, volume 9, pages 3273–3297, 1998.

D. J. Spiegelhalter. Probabilistic reasoning in predictive expert systems. In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 47–68, Amsterdam, 1986. North Holland.

P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. Lander, and T. Golub. Interpreting patterns of gene expression with self-organizing maps: Methods and application to homatopoietic differentiation. In *Proceedings of the National Academy of Science*, volume 96, pages 2907–2912, 1999.

A. Thomas, D. J. Spiegelhalter, and W. R. Gilks. BUGS: A program to perform Bayesian inference using Gibbs sampling. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics*, Oxford: Clarendon Press, 1992.

M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999a.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 21(3):611–622, 1999b.

M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3 (1):71–86, 1991.

N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton. SMEM algorithm for mixture models. In *Advances in Neural Information Processing Systems*, volume 11, 1999.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 2001.

Niclas Wiberg. *Codes and Decoding on General Graphs.* PhD thesis, Linköping University, 1996.

W. Wiegerinck. Variational approximations between mean field theory and the junction tree algorithm. In *Uncertainty in Artificial Intelligence*. Morgan Kauffmann, 2000.

C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 514–520. MIT Press, 1996.

P. H. Winston. *Artificial Intelligence.* Addison-Wesley, third edition, 1992.

E. P. Xing, M. I. Jordan, and S. Russell. A generalized mean field algorithm for variational inference in exponential families. In *Uncertainty in Artificial Intelligence*. Morgan Kauffmann, 2003.

J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

K. Yeung, C. Fraley, A. Murua, A. Raftery, and W. Ruzzo. Model-based clustering and data transformations for gene expression data. *Bioinformatics*, 17(10):977–987, 2001.