

## 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* [?]. 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 [?] 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 [?]. 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. ? 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

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

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

$$P(x | y)P(y) = P(x, y) = P(y | x)P(x). \quad (1.4)$$

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

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

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  $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$ . 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  $P(X_1, \dots, X_n)$  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.

$$P(G, L, R, S, B) = P(G)P(L|G)P(R|G, L)P(S|G, L, R)P(B|G, L, R, S). \quad (1.6)$$

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

$$P(G, L, R, S, B) = P(G)P(L|G)P(R|G, L)P(S|L)P(B|S). \quad (1.7)$$

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 has 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 [?], factor graphs [?], Markov random fields [?] and chain graphs [?].

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  $P(X | Y_1, Y_2, \dots)$ . 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.

The edges in a Bayesian network represent conditional independence relationships in that

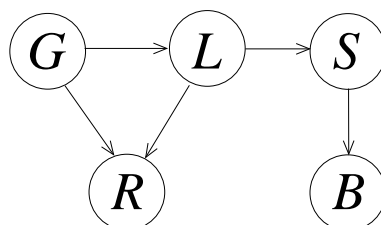


Figure 1.1: A Bayesian network corresponding to the probabilistic model used in the lie detector example on page 4. The variables are:  $G$  the guilt of the suspect,  $L$  whether the suspect is lying,  $R$  the subject's response,  $S$  whether the subject is stressed and  $B$  the subject's bio-physical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

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:

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i | \text{pa}_i). \quad (1.8)$$

### 1.4.2 Factor graphs

A *factor graph* [?] 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:

$$P(X_1, \dots, X_n) = \frac{1}{Z} \prod_i \Psi_i(c_i). \quad (1.9)$$

In this equation, each  $c_i$  is the small subset of variables that the corresponding factor function  $\Psi_i$  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 ?). 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  $\Psi_i$  associated with the function node is then the conditional probability  $P(X_i | \text{pa}_i)$ . A factor graph for the lie detector example is shown in Figure 1.2a.

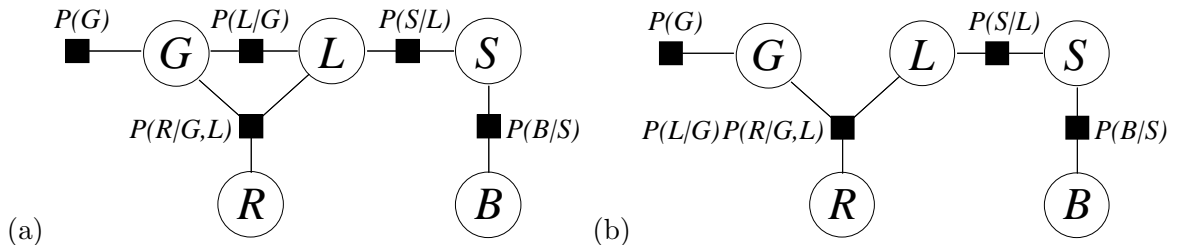


Figure 1.2: (a) A factor graph corresponding to the probabilistic model used in the lie detector. The Bayesian network of Figure 1.1 can be converted to this factor graph by creating a function node for each variable node and connecting it to that node and all its parents. (b) An equivalent factor graph 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  $P(L|G)$  and  $P(R|G, L)$ .

### 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  $P(X_i | \text{pa}_i)$ . Frequently, probability distributions are chosen to be Gibbs distributions, defined as

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

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* [?, pp.197–202]. The form of a distribution in the exponential family is

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

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<sup>1</sup> as

$$\mathcal{N}(x | \mu, \gamma^{-1}) = \exp \left( \left[ \begin{array}{c} \gamma\mu \\ -\frac{\gamma}{2} \end{array} \right]^T \left[ \begin{array}{c} x \\ x^2 \end{array} \right] + 0 + \frac{1}{2}(\log \gamma - \gamma\mu^2 - \log 2\pi) \right). \quad (1.12)$$

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

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

---

<sup>1</sup>In this equation for the Gaussian distribution,  $\gamma$  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.

for some definition of the functions  $\psi(W)$  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

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

which allows us to update our *prior* belief about the model parameters  $P(\boldsymbol{\theta} | \mathcal{H})$  to a *posterior* belief  $P(\boldsymbol{\theta} | \mathbf{D}, \mathcal{H})$  given the data  $\mathbf{D}$ . In this way, the parameters of the model have been learnt from the data. The *likelihood* function  $P(\mathbf{D} | \boldsymbol{\theta}, \mathcal{H})$  simply defines how likely the observed data are given a particular setting of the model parameters  $\boldsymbol{\theta}$ . The *evidence*  $P(\mathbf{D} | \mathcal{H})$  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  $\mathcal{H}_1 \dots \mathcal{H}_N$  is most plausible given the data. Again, we apply Bayes's theorem:

$$P(\mathcal{H}_i | \mathbf{D}) = \frac{P(\mathbf{D} | \mathcal{H}_i)P(\mathcal{H}_i)}{P(\mathbf{D})} \quad (1.15)$$

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  $P(\mathbf{D} | \mathcal{H}_i)$ .

We can therefore learn a probabilistic model by proposing a number of models  $\mathcal{H}_1 \dots \mathcal{H}_N$ , 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  $P(\mathbf{Z} | \mathbf{D} = \mathbf{d})$ , 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 ? 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:

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

so we can compute  $P(\mathbf{Z}, \mathbf{D} = \mathbf{d})$  for all possible values of  $\mathbf{Z}$  and then normalise to get  $P(\mathbf{Z} | \mathbf{D} = \mathbf{d})$ . 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  $\mathbf{Z}$ ):

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

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  $P(G | R = r, B = b)$ . Following the method above, we marginalise out  $L$  and  $S$  from the joint distribution in Equation 1.7:

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

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

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

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

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<sup>2</sup>.

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 ? and ? [??].

### 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 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* [??]), which leads to simpler definitions for the messages. In the Sum-

<sup>2</sup>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.

Product algorithm, the message from an unobserved variable node  $X_i$  to a function node  $f_j$  is defined as:

$$m_{X_i \rightarrow f_j} = \prod_{k \in \text{ne}_i \setminus j} m_{f_k \rightarrow X_i} \quad (1.21)$$

where  $\text{ne}_i$  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

$$m_{X_i \rightarrow f_j} = \delta(X_i | x_i), \quad (1.22)$$

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

$$m_{f_i \rightarrow X_j} = \sum_{\text{ne}_i \setminus j} f_i(\text{ne}_i) \prod_{k \in \text{ne}_i \setminus j} m_{X_k \rightarrow f_i}, \quad (1.23)$$

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,

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

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.

- (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  $P(G)$ .

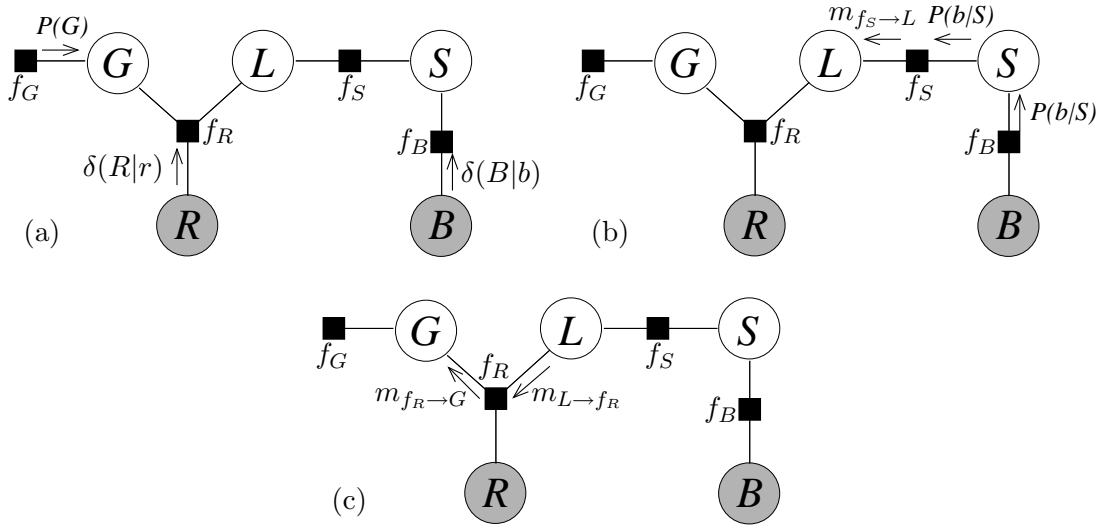


Figure 1.3: (a)-(c) Messages passed during the `CollectEvidence` phase of belief propagation as applied to the lie detector factor graph, with  $G$  as the root node. The nodes corresponding to the observed variables  $R$  and  $B$  are shown shaded.

- (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

$$m_{f_S \rightarrow L} = \sum_S P(S|L)P(B = b|S). \quad (1.25)$$

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

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

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 product of these messages

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

This is the same answer as given by variable elimination (Equation 1.20) but found using

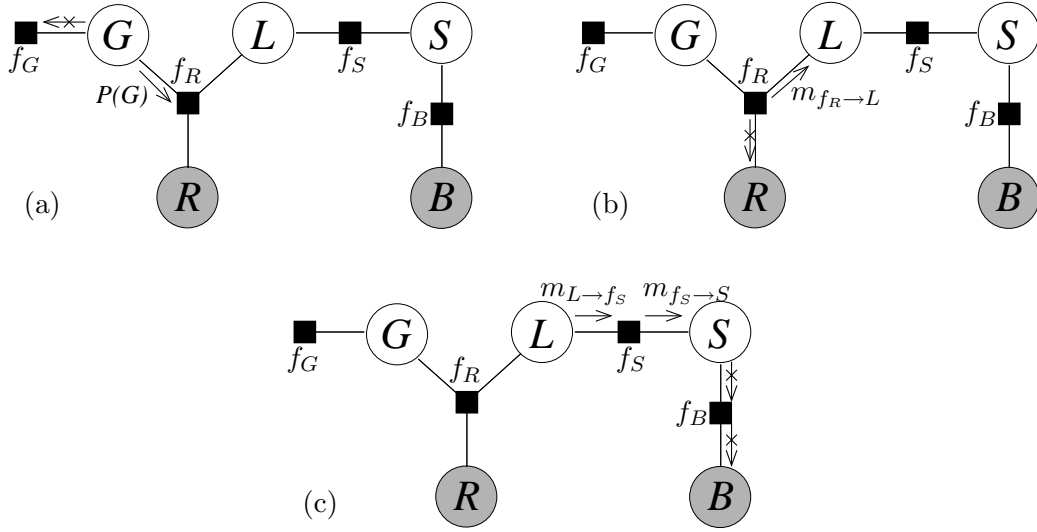


Figure 1.4: (a)-(c) Messages passed during `DistributeEvidence`. Messages marked with crosses are not computed as they are not needed to find the posterior over the latent variables.

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

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

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

$$m_{f_S \rightarrow S} = \sum_L P(S|L)m_{L \rightarrow f_S}. \quad (1.29)$$

The remaining messages  $m_{S \rightarrow f_B}$  and  $m_{f_B \rightarrow B}$  are not computed since  $B$  is observed. The

posterior distributions over  $L$  and  $S$  can be found using Equation 1.24,

$$\begin{aligned} P(L | R = r, B = b) &= \frac{1}{Z} m_{f_R \rightarrow L} m_{f_S \rightarrow L} \\ &\propto \sum_G P(G) P(L | G) P(R = r | G, L) \sum_S P(S | L) P(B = b | S) \end{aligned} \quad (1.30)$$

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

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 `CollectEvidence` and `DistributeEvidence` provide just one method of performing belief propagation. It is possible to use more flexible message passing schemes (see also ?, 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<sup>3</sup>). 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 [????]. Recently, machine learning researchers have made some progress by considering LBP as finding fixed points in the Bethe free energy [??].

<sup>3</sup>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.

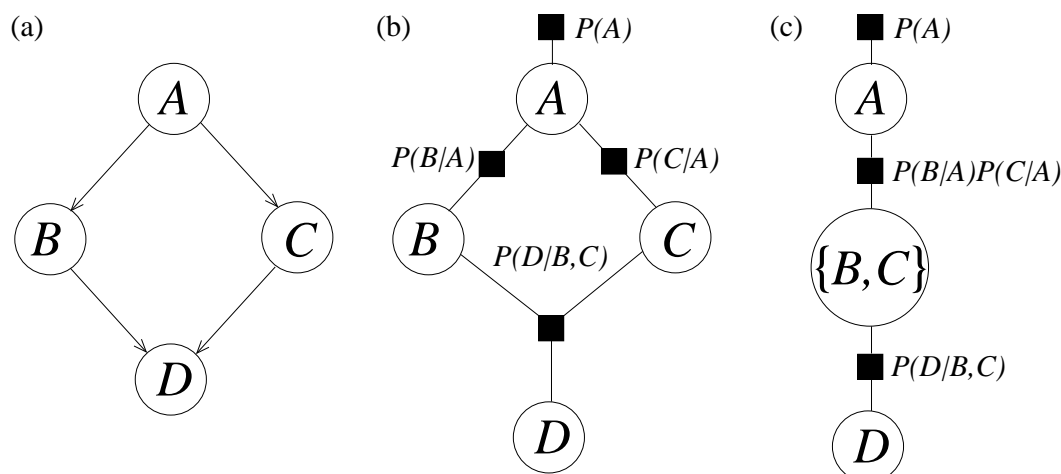


Figure 1.5: (a) A Bayesian network. (b) The corresponding cyclic factor graph. (c) The singly-connected factor graph created by combining the variables  $B$  and  $C$  into a two-variable cluster corresponding to a new compound variable.

### Cutset conditioning

Cutset conditioning [?] 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, ?) 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, ? and ? 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,

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

and similar integrals in the messages of belief propagation algorithms. The integrals are typically high-dimensional, non-linear and, with some exceptions (for example ?), 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 ? 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:

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

In the limit  $S \rightarrow \infty$ , 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 [?] 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 [??] as

well as other graphical models [??]. 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 ??: 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 [????] 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 [?], statistics [?], quantum mechanics [?], and finite element analysis [?]. 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 [??] 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

$$\text{KL}(Q \parallel P) = \int_{\mathcal{X}} Q(x) \log \frac{Q(x)}{P(x)} dx \quad (1.35)$$

where  $\log$  is the natural logarithm<sup>4</sup>. 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*<sup>5</sup> that would be wasted if we encoded samples from a distribution  $Q$ , using a perfect encoder that was optimised for samples from  $P$  [?].

The KL divergence is not a true distance measure as it is not symmetric: in general,  $\text{KL}(Q \parallel P) \neq \text{KL}(P \parallel Q)$ . 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  $\text{KL}(Q \parallel P)$  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  $\text{KL}(P \parallel Q)$  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  $P$ .

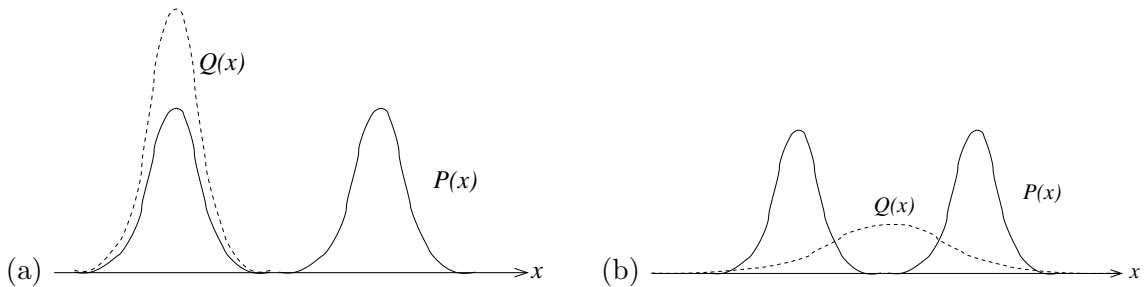


Figure 1.6: Illustration of the asymmetry of the KL divergence. If the distribution  $P$  is bimodal and the approximating distribution  $Q$  is unimodal then (a) minimising  $\text{KL}(Q \parallel P)$  will give a  $Q$  distribution with almost all probability mass in one mode of  $P$  and negligible mass in the other mode (b) minimising  $\text{KL}(P \parallel Q)$  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).

<sup>4</sup>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  $x$ .

<sup>5</sup>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  $1 \text{ nat} = \log_2 e$  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  $P(\mathbf{H} | \mathbf{D})$  means that it will be necessary to minimise either

$$\text{KL}(Q || P) = \int_{\mathbf{H}} Q(\mathbf{H}) \log \frac{Q(\mathbf{H})}{P(\mathbf{H} | \mathbf{D})} d\mathbf{H} \quad (1.36)$$

or

$$\text{KL}(P || Q) = \int_{\mathbf{H}} P(\mathbf{H} | \mathbf{D}) \log \frac{P(\mathbf{H} | \mathbf{D})}{Q(\mathbf{H})} d\mathbf{H}. \quad (1.37)$$

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

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

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:

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

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

where the notation  $\langle \cdot \rangle_Q$  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

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

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  $\text{KL}(P || Q)$  as this would require calculating the expectation of  $\log Q(\mathbf{H})$  under the true posterior  $P(\mathbf{H} | \mathbf{D})$ , which must be intractable or we would not need to use approximate methods. Being constrained to minimise  $\text{KL}(Q || P)$  instead of  $\text{KL}(P || Q)$  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  $P$ .

There are other methods for finding an approximate posterior which use the KL divergence. *Expectation propagation* [??] 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  $\text{KL}(\hat{P}_i || Q)$  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  $\mathcal{H}$ :

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

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

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

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:

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

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

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.

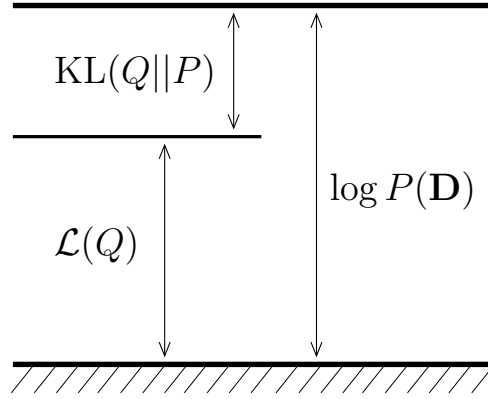


Figure 1.7: The quantity  $\mathcal{L}(Q)$  provides a lower bound on the log evidence,  $\log P(\mathbf{D})$ , with the difference being given by the KL divergence,  $\text{KL}(Q||P)$ . By maximising  $\mathcal{L}(Q)$ , we can minimise the KL divergence since the log evidence is fixed with respect to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

#### 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

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

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

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

Now separate out terms in one factor  $Q_j$

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

and define

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

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

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

$$= -\text{KL}(Q_j \parallel Q_j^*) - \log Z + \sum_{i \neq j} \mathbb{H}(Q_i). \quad (1.51)$$

The last two terms do not depend on  $Q_j$  and so  $\mathcal{L}(Q)$  is maximised with respect to  $Q_j$  by minimising  $\text{KL}(Q_j \parallel Q_j^*)$ . 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:

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

which introduces the notation  $\langle \cdot \rangle_{\sim Q(\mathbf{H}_j)}$  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.
-

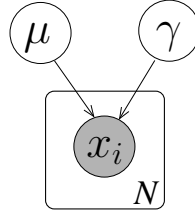


Figure 1.8: The Bayesian network for a probabilistic model that represents a set of  $N$  observed data points with a Gaussian distribution of mean  $\mu$  and precision  $\gamma$ . The rectangle is a *plate*, which indicates that the contained node and its connected edges are duplicated  $N$  times.

### 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  $\mu$  and precision  $\gamma$ :

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

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  $\mu$  and  $\gamma$  are the latent variables in this model and variational inference can be used to learn an approximate posterior distribution over  $\mu$  and  $\gamma$  given the data. However, it is first necessary to complete the model by defining prior distributions over  $\mu$  and  $\gamma$ . To make inference tractable, conjugate priors are chosen,<sup>6</sup>

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

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

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

$$Q(\mu, \gamma) = Q_\mu(\mu)Q_\gamma(\gamma). \quad (1.56)$$

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

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

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

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

<sup>6</sup>For full conjugacy, we could use a Normal-Gamma prior over both  $\mu$  and  $\gamma$ . However, this would lead to an exact solution and would not demonstrate the use of a separable variational distribution.

plying Equation 1.49 gives

$$\beta' = \beta + N\langle\gamma\rangle \quad (1.59)$$

$$m' = \frac{1}{\beta'} \left( \beta m + \langle\gamma\rangle \sum_{i=1}^N x_i \right) \quad (1.60)$$

$$a' = a + \frac{N}{2} \quad (1.61)$$

$$b' = b + \frac{1}{2} \sum_{i=1}^N (x_i^2 - 2x_i\langle\mu\rangle + \langle\mu^2\rangle), \quad (1.62)$$

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  $\mu$  and  $\gamma$ , by setting  $m = 0$ ,  $\beta = 0.001$ ,  $a = 0.001$  and  $b = 0.001$ . Figure 1.9a shows the variational posterior  $Q$  over  $\mu$  and  $\gamma$ , 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  $\text{KL}(Q \parallel P)$  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  $\text{KL}(Q \parallel P)$  rather than  $\text{KL}(P \parallel Q)$ . 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:

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

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

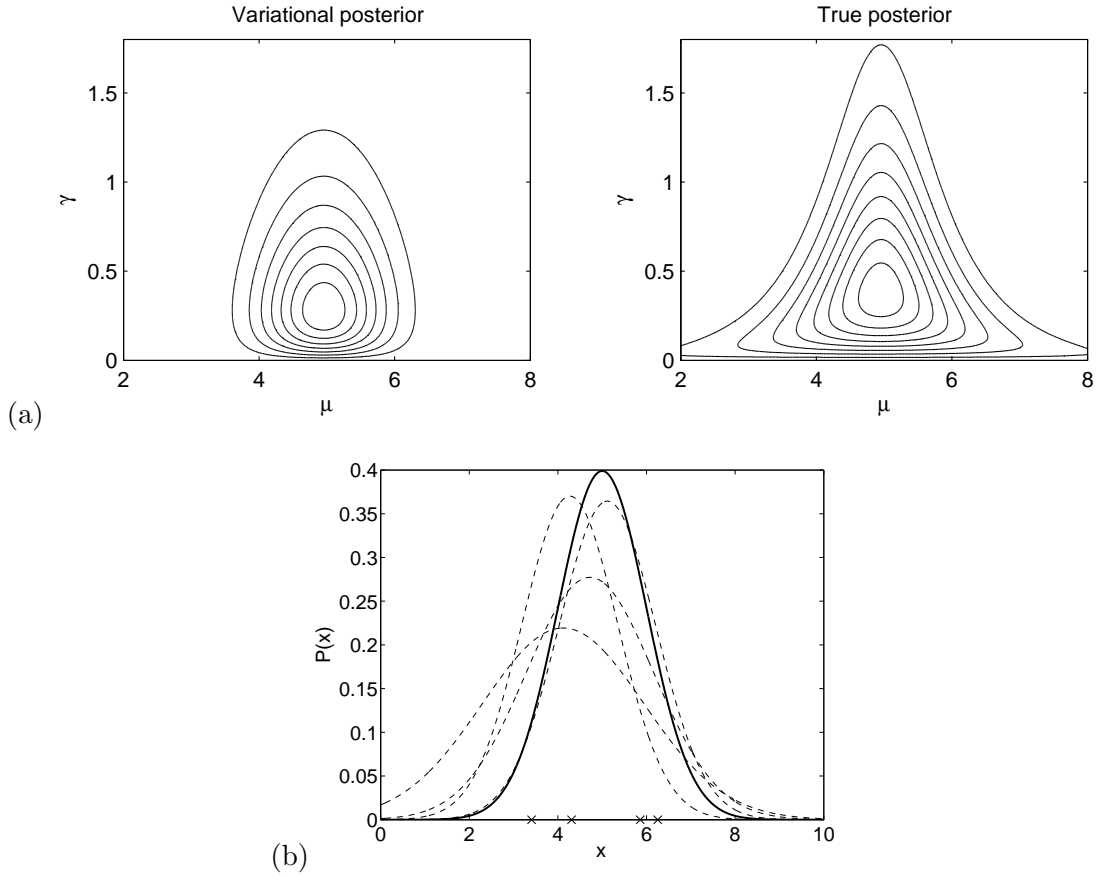


Figure 1.9: (a) Variational and true posterior over the mean  $\mu$  and precision  $\gamma$  of a Gaussian given four samples from it. The priors over the parameters were  $P(\mu) = \mathcal{N}(0, 1000)$  and  $P(\gamma) = \text{Gamma}(0.001, 0.001)$ . (b) The actual distribution the samples were drawn 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  $\mu$  and  $\gamma$ . It might seem more intuitive to determine optimal single values of  $\mu$  and  $\gamma$ . This can be achieved using a variational distribution consisting of delta function distributions

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

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

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

From Equation 1.47, the bound is

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

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

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:

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

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

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  $\theta$  to a new parameter  $u = f(\theta)$ , the probability density is transformed to

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

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  $\gamma$  must be transformed to give the prior on  $\nu$

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

$$= P(\gamma|a, b) \gamma \quad (1.72)$$

The MAP estimate of  $\nu$  is now given by

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

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 [??, 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

$$P(x | \{\pi_k\}, \{\theta_k\}) = \sum_{k=1}^K \pi_k P_k(x | \theta_k) \quad (1.74)$$

where each  $P_k$  is a simple distribution with parameters  $\theta_k$  and a corresponding *mixture coefficient*  $\pi_k$  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  $\mu_k$  and precision  $\gamma_k$  and so the distribution is

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

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  $\lambda$  for the data point  $x$  which indicates which component distribution the data point was drawn from. Thus,  $\lambda$  is a discrete variable with one of  $K$  values, corresponding to one of the  $K$  component distributions. The distribution may now be written as

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

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

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

$$= \sum_{\lambda=1}^K P(\lambda) \mathcal{N}(x | \mu_\lambda, \gamma_\lambda^{-1}) \quad (1.78)$$

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

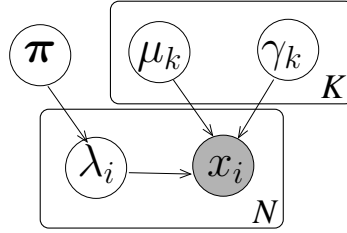


Figure 1.10: The Bayesian network for a probabilistic model which represents a set of  $N$  observed data points by a mixture of Gaussian distributions. The top plate contains the  $K$  sets of component means and precisions. A discrete variable  $\lambda_i$  has been introduced for each data point to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

the mixture coefficients  $\pi_k$ .

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

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

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

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

where  $\pi_k$  is the  $k$ th element of the vector  $\boldsymbol{\pi}$ . We complete the definition of the model by defining conjugate priors over each of the parameters<sup>7</sup>

$$P(\mu_k) = \mathcal{N}(\mu_k | m, \beta^{-1}) \quad (1.81)$$

$$P(\gamma_k) = \text{Gamma}(\gamma_k | a, b) \quad (1.82)$$

$$P(\boldsymbol{\pi}) = \text{Dirichlet}(\boldsymbol{\pi} | \mathbf{u}), \quad (1.83)$$

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

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

<sup>7</sup>A separable prior over  $\mu_k$  and  $\gamma_k$  has been 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  $P$

$$Q_{\mu_k}(\mu_k) = \mathcal{N}(\mu_k | m'_k, \beta_k'^{-1}) \quad (1.85)$$

$$Q_{\gamma_k}(\gamma_k) = \text{Gamma}(\gamma_k | a'_k, b'_k) \quad (1.86)$$

$$Q_{\pi}(\boldsymbol{\pi}) = \text{Dirichlet}(\boldsymbol{\pi} | \mathbf{u}') \quad (1.87)$$

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

$$\beta'_k = \beta + \sum_{i=1}^N Q_{\lambda_i}(k) \langle \gamma_k \rangle \quad (1.88)$$

$$m'_k = \frac{1}{\beta'_k} \left( \beta m + \langle \gamma_k \rangle \sum_{i=1}^N Q_{\lambda_i}(k) x_i \right) \quad (1.89)$$

$$a'_k = a + \frac{1}{2} \sum_{i=1}^N Q_{\lambda_i}(k) \quad (1.90)$$

$$b'_k = b + \frac{1}{2} \sum_{i=1}^N Q_{\lambda_i}(k) (x_i^2 - 2x_i \langle \mu_k \rangle + \langle \mu_k^2 \rangle) \quad (1.91)$$

$$u'_k = u_k + \sum_{i=1}^N Q_{\lambda_i}(k) \quad (1.92)$$

and  $Q_{\lambda_i}$  is updated using

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

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<sup>8</sup>, as no data points were assigned to them, showing that the correct number of components had been

<sup>8</sup>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.

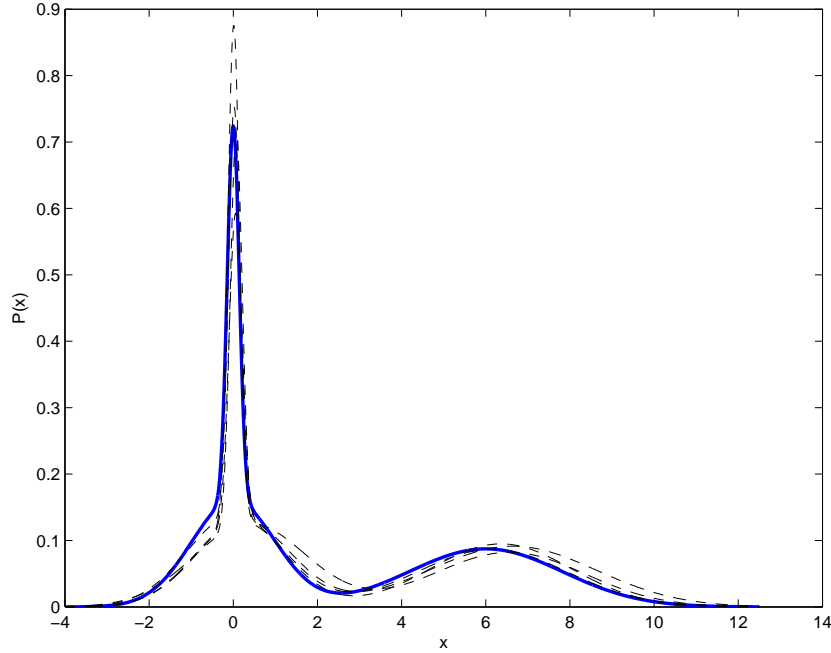


Figure 1.11: Diagram showing the true mixture of Gaussians distribution (thick line) used to generate the 120 data points, along with four distributions whose parameters have been sampled from the separable variational posterior trained on these data (dashed lines).

### 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  $Q_{\lambda_i}$ , which becomes

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

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

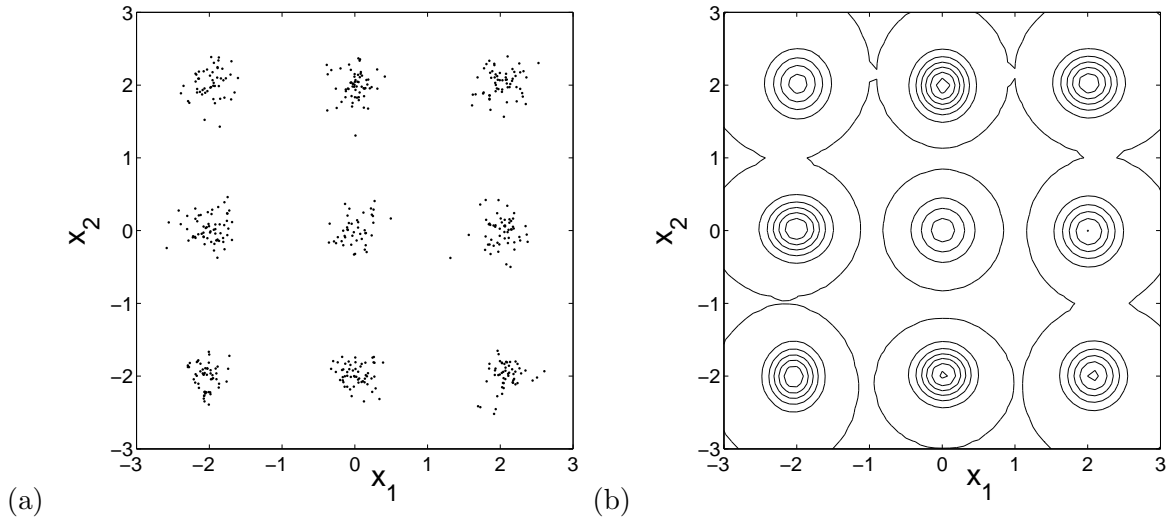


Figure 1.12: (a) 500 two-dimensional data points sampled from a mixture of Gaussians model with nine components of equal weight. (b) Probability contours of the mixture model whose parameters are set to their expected values under the approximate variational distribution.

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.

# 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 Kaufmann, 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. Niranjana, 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 profiles 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 Kaufmann, 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, Canada, 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 hematopoietic 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. Wiegnerinck. Variational approximations between mean field theory and the junction tree algorithm. In *Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 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 Kaufmann, 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 Millennium*. 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.