

Introduction to Learning Bayesian Networks from Data

Dirk Husmeier

Biomathematics and Statistics Scotland (BioSS)
JCMB, The King's Buildings, Edinburgh EH9 3JZ, UK
`dirk@bioass.ac.uk`

Summary. Bayesian networks are a combination of probability theory and graph theory. Graph theory provides a framework to represent complex structures of highly-interacting sets of variables. Probability theory provides a method to infer these structures from observations or measurements in the presence of noise and uncertainty. Many problems in computational molecular biology and bioinformatics, like sequence alignment, molecular evolution, and genetic networks, can be treated as particular instances of the general problem of learning Bayesian networks from data. This chapter provides a brief introduction, in preparation for later chapters of this book.

2.1 Introduction to Bayesian Networks

Bayesian networks (BNs) are interpretable and flexible models for representing probabilistic relationships between multiple interacting entities. At a *qualitative* level, the structure of a Bayesian network describes the relationships between these entities in the form of conditional independence relations. At a *quantitative level*, (local) relationships between the interacting entities are described by (conditional) probability distributions. Formally, a BN is defined by a graphical structure, \mathcal{M} , a family of (conditional) probability distributions, \mathcal{F} , and their parameters, \mathbf{q} , which together specify a joint distribution over a set of random variables of interest. These three components are discussed in the following two subsections.

2.1.1 The Structure of a Bayesian Network

The graphical structure \mathcal{M} of a BN consists of a set of *nodes* or *vertices*, \mathcal{V} , and a set of *directed edges* or *arcs*, \mathcal{E} : $\mathcal{M} = (\mathcal{V}, \mathcal{E})$. The nodes represent random variables, while the edges indicate conditional dependence relations. If we have a directed edge from node A to node B , then A is called the *parent* of B , and B is called the *child* of A . Take, as an example, Figure 2.1, where

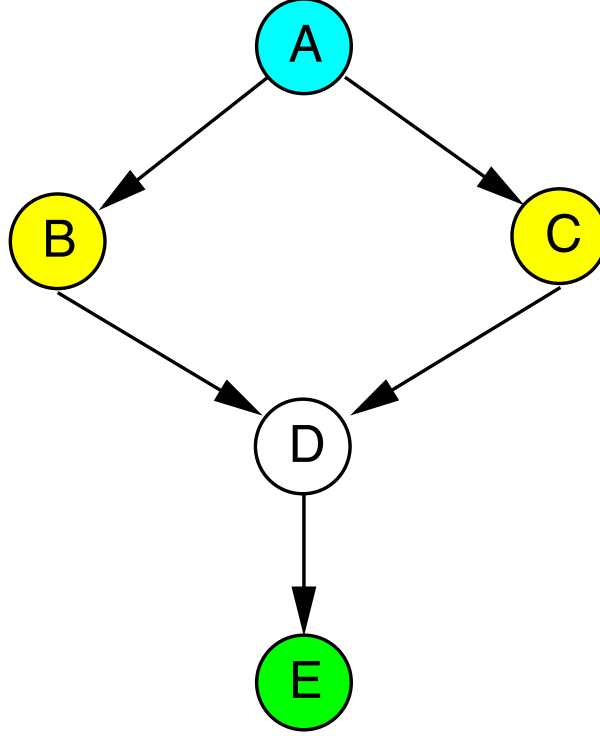


Fig. 2.1. Example of a Bayesian network. Nodes represent random variables, edges indicate conditional dependence relations. The joint probability $P(A, B, C, D, E)$ factorizes into the product $P(A)P(B|A)P(C|A)P(D|B, C)P(E|D)$. Reprinted from [23], by permission of Cambridge University Press.

we have the set of vertices $\mathcal{V} = \{A, B, C, D, E\}$, and the set of edges $\mathcal{E} = \{(A, B), (A, C), (B, D), (C, D), (D, E)\}$. Node A does not have any parents. Nodes B and C are the children of node A , and the parents of node D . Node D itself has one child: node E . The graphical structure has to take the form of a *directed acyclic graph* or DAG, which is characterized by the absence of directed cycles, that is, cycles where all the arcs point in the same direction. A BN is characterized by a simple and unique rule for expanding the joint probability in terms of simpler conditional probabilities. Let X_1, X_2, \dots, X_n be a set of random variables represented by the nodes $i \in \{1, \dots, n\}$ in the graph, define $pa[i]$ to be the parents of node i , and let $\mathcal{X}_{pa[i]}$ represent the set of random variables associated with $pa[i]$. Then

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i | \mathcal{X}_{pa[i]}) \quad (2.1)$$

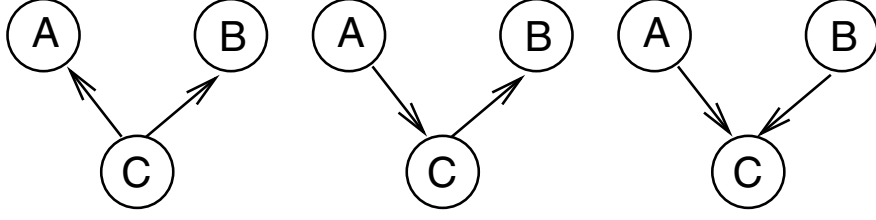


Fig. 2.2. Three elementary BNs. The BNs on the left and in the middle have equivalent structures: A and B are conditionally independent given C . The BN on the right belongs to a different equivalence class in that conditioning on C causes, in general, a dependence between A and B .

As an example, applying (2.1) to the BN of Figure 2.1, we obtain the factorization

$$P(A, B, C, D, E) = P(A)P(B|A)P(C|A)P(D|B, C)P(E|D) \quad (2.2)$$

An equivalent way of expressing these independence relations is based on the concept of the *Markov blanket*, which is the set of children, parents, and coparents (that is, other parents of the children) of a given node. This set shields the selected node from the remaining nodes in the graph. So, if $MB[i]$ is the Markov blanket of node i , and $\mathcal{X}_{MB[i]}$ is the set of random variables associated with $MB[i]$, then

$$P(X_k|X_1, \dots, X_{k-1}, X_{k+1}, \dots, X_n) = P(X_k|\mathcal{X}_{MB[i]}) \quad (2.3)$$

Applying (2.3) to Figure 2.1 gives:

$$P(A|B, C, D, E) = P(A|B, C) \quad (2.4)$$

$$P(B|A, C, D, E) = P(B|A, C, D) \quad (2.5)$$

$$P(C|A, B, D, E) = P(C|A, B, D) \quad (2.6)$$

$$P(D|A, B, C, E) = P(D|B, C, E) \quad (2.7)$$

$$P(E|A, B, C, D) = P(E|D) \quad (2.8)$$

To illustrate the equivalence of the factorization rule (2.1) and the notion of the Markov blanket (2.3), let us derive, for instance, (2.6) from (2.2):

$$\begin{aligned} P(C|A, B, D, E) &= \frac{P(A, B, C, D, E)}{P(A, B, D, E)} \\ &= \frac{P(A)P(B|A)P(C|A)P(D|B, C)P(E|D)}{\sum_C P(A)P(B|A)P(C|A)P(D|B, C)P(E|D)} \\ &= \frac{P(A)P(E|D)P(B|A)P(C|A)P(D|B, C)}{P(A)P(E|D)P(B|A) \sum_C P(C|A)P(D|B, C)} \\ &= \frac{P(C|A)P(D|B, C)}{\sum_C P(C|A)P(D|B, C)} \end{aligned}$$

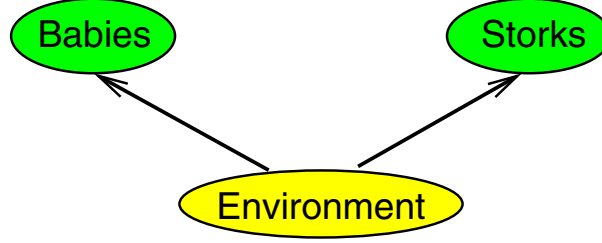


Fig. 2.3. Storks and babies. The numbers of stork sightings and new-born babies depend on common environmental factors. Without the knowledge of these environmental factors, the number of new-born babies seems to depend on the number of stork sightings, but conditional on the environmental factors, both events are independent.

where we have applied (2.2) for the factorization of the joint probability $P(A, B, C, D, E)$. Note that the last term does not depend on E , which proves (2.6) true. For a general proof of the equivalence of (2.1) and (2.3), see [24] and [34].

Consider the BN on the left of Figure 2.2. Expanding the joint probability according to (2.1) gives

$$P(A, B, C) = P(A|C)P(B|C)P(C) \quad (2.9)$$

For the conditional probability $P(A, B|C)$ we thus obtain:

$$P(A, B|C) = \frac{P(A, B, C)}{P(C)} = P(A|C)P(B|C) \quad (2.10)$$

Hence, A and B are *conditionally* independent given C . Note, however, that this independence does not carry over to the marginal probabilities, and that in general

$$P(A, B) \neq P(A)P(B) \quad (2.11)$$

As an example, consider the BN in Figure 2.3. The number of new-born babies has been found to depend on the number of stork sightings [39], which, in former times, even led to the erroneous conclusion that storks deliver babies. In fact, both events depend on several environmental factors. In an urban environment, families tend to be smaller as a consequence of changed living conditions, while storks are rarer due to the destruction of their natural habitat. The introduction of contraceptives has led to a decrease of the number of new-born babies, but their release into the environment also adversely affected the fecundity of storks. So while, without the knowledge of these environmental factors, the number of new-born babies depends on the number of stork sightings, conditionally on the environmental factors both events are independent.

The situation is similar for the BN in the middle of Figure 2.2. Expanding the joint probability by application of the factorization rule (2.1) gives:



Fig. 2.4. Clouds and rain. When no information on the rain is available, the wetness of the grass depends on the clouds: the more clouds are in the sky, the more likely the grass is found to be wet. When information on the rain is available, information on the clouds is no longer relevant for predicting the state of wetness of the grass: conditional on the rain, the wetness of the grass is independent of the clouds.

$$P(A, B, C) = P(B|C)P(C|A)P(A) \quad (2.12)$$

For the conditional probability we thus obtain:

$$P(A, B|C) = \frac{P(A, B, C)}{P(C)} = P(B|C) \frac{P(C|A)P(A)}{P(C)} = P(B|C)P(A|C) \quad (2.13)$$

where we have used Bayes' rule, (1.4). So again we find that A and B are conditionally independent given C , while, in general, this does not hold for the marginal probabilities; see (2.11).

An example is shown in Figure 2.4. Clouds may cause rain, and rain makes grass wet. So if information on precipitation is unavailable, that is, if the node “rain” in Figure 2.4 is hidden, the state of wetness of the grass depends on the clouds: an increased cloudiness, obviously, increases the likelihood for the grass to be wet. However, if information on precipitation is available, meaning that the node “rain” in Figure 2.4 is observed, the wetness of the grass becomes independent of the clouds. If it rains, the grass gets wet no matter how cloudy it is. Conversely, if it does not rain, the grass stays dry irrespective of the state of cloudiness.

The situation is different for the BN on the right of Figure 2.2. Expanding the joint probability $P(A, B, C)$ according to (2.1) gives:

$$P(A, B, C) = P(C|A, B)P(A)P(B) \quad (2.14)$$

Marginalizing over C leads to

$$P(A, B) = \sum_C P(A, B, C) = P(A)P(B) \quad (2.15)$$

where we have used the fact that a probability function is normalized: $\sum_C P(C|A, B) = 1$. We thus see that, as opposed to the previous two examples, A and B are *marginally* independent. However, it can *not* be shown, in general, that the same holds for the conditional probabilities, that is, different from the previous examples we have

$$P(A, B|C) \neq P(A|C)P(B|C) \quad (2.16)$$

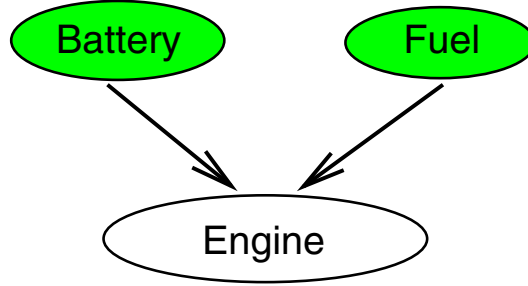


Fig. 2.5. Fuel and battery. Nationwide, the unfortunate events of having a flat car battery and running out of fuel are independent. This independence no longer holds when an engine failure is observed in a particular car, since establishing one event as the cause of this failure explains away the other alternative.

An illustration is given in Figure 2.5. Suppose you cannot start your car engine in the morning. Two possible reasons for this failure are: (1) a flat battery, B , or (2) an empty fuel tank, F . Nationwide, these two unfortunate events can be assumed to be independent: $P(B, F) = P(B)P(F)$. However, this independence no longer holds when you observe an engine failure, E , in your particular car: $P(B, F|E) \neq P(B|E)P(F|E)$. Obviously, on finding the fuel tank empty, there is little need to check the voltage of the battery: the empty tank already accounts for the engine failure and thus *explains away* any problems associated with the battery.

Figure 2.6 gives an overview of the independence relations we have encountered in the previous examples. The power of Bayesian networks is that we can deduce, in much more complicated situations, these independence relations between random variables from the network structure without having to resort to algebraic computations. This is based on the concept of *d-separation*, which is formally defined as follows (see [34], and references in [23]):

- Let A and B be two nodes, and let \mathcal{Z} be a set of nodes.
- A *path* from A to B is *blocked* with respect to \mathcal{Z}
 - if there is a node $C \in \mathcal{Z}$ without converging edges, that is, which is *head-to-tail* or *tail-to-tail* with respect to the path, or
 - if two edges on the path converge on a node C , that is, the configuration of edges is *head-to-head*, and neither C nor any of its descendents are in \mathcal{Z} .
- A and B are *d-separated* by \mathcal{Z} if and only if all possible paths between them are blocked.
- If A and B are d-separated by \mathcal{Z} , then A is conditionally independent of B given \mathcal{Z} , symbolically written as $A \perp B | \mathcal{Z}$.

An illustration is given in Figure 2.7. As a first example, consider the elementary BNs of Figure 2.8. Similar to the preceding examples, we want to decide whether A is independent of B conditional on those other nodes

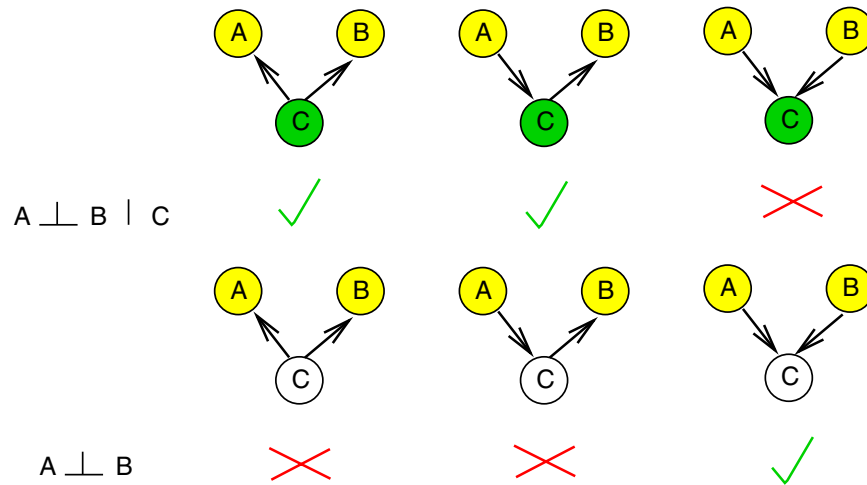


Fig. 2.6. Overview of elementary BN independence relations. $A \perp B$ means that A and B are marginally independent: $P(A, B) = P(A)P(B)$. $A \perp B \mid C$ means that A and B are conditionally independent: $P(A, B \mid C) = P(A \mid C)P(B \mid C)$. The figure summarizes the independence relations of Figures 2.3–2.5, which can easily be derived with the method of d-separation, illustrated in Figure 2.7. A tick indicates that an independence relation holds true, whereas a cross indicates that it is violated.

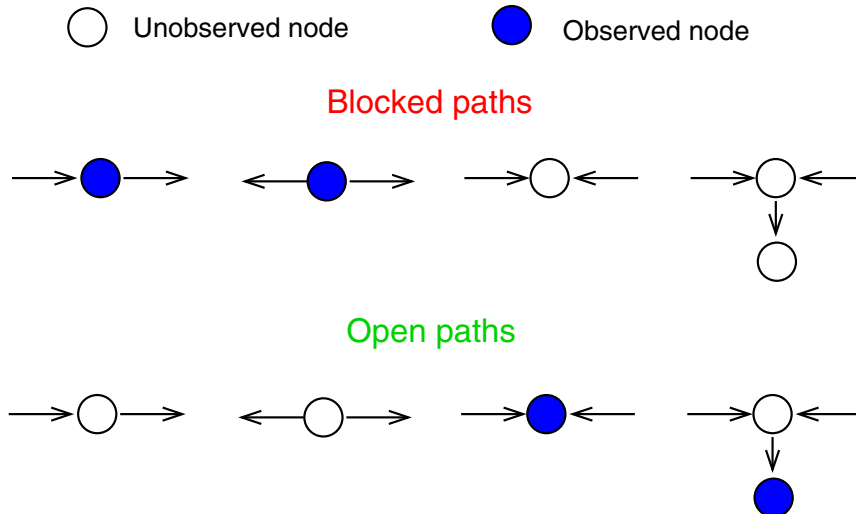


Fig. 2.7. Illustration of d-separation when the separating set \mathcal{Z} is the set of observed nodes. Filled circles represent observed nodes, empty circles indicate hidden states (for which no data are available).

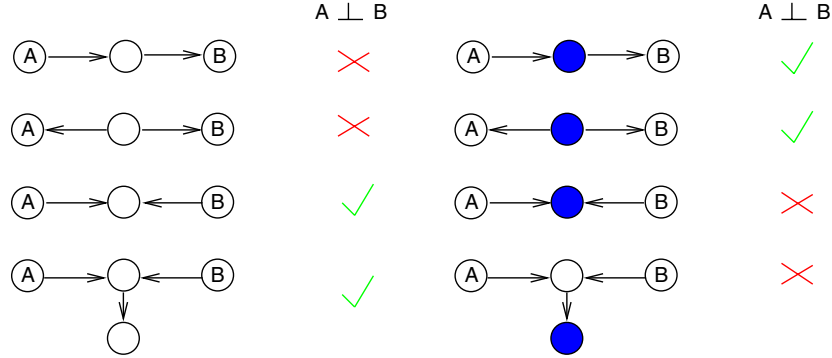


Fig. 2.8. Illustration of d-separation in elementary BN structures. Filled circles represent observed nodes, empty circles represent hidden nodes (for which no data are available). The column on the right of each subfigure indicates whether A and B are independent given \mathcal{Z} , where \mathcal{Z} is the set of observed nodes. Compare with Figure 2.6.

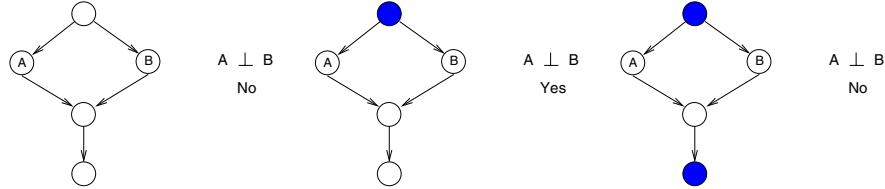


Fig. 2.9. Illustration of d-separation in a Bayesian network. Filled circles represent observed nodes, empty circles represent hidden nodes (for which no data are available). The legend on the right of each network indicates whether nodes A and B are independent given \mathcal{Z} , where \mathcal{Z} is the set of observed nodes. Adapted from [23], by permission of Cambridge University Press.

in the graph that have been observed. It can easily be seen that testing for *d-separation* leads to the same results as before, but without having to go through the (albeit very simple) algebra of equations (2.10), (2.13), and (2.15). This is useful in more complex networks, where the algebra is more involved.

Take, as a second example, Figure 2.9. Again, we are interested in whether A is independent of B given the set of observed nodes, \mathcal{Z} . There are two paths connecting nodes A and B . If no other node is observed (Figure 2.9, left), the upper path is *not* blocked, because the edges do not converge and the separating node is not observed (that is, it is not in \mathcal{Z}). Consequently, A and B are *not* d-separated, and A and B are *not* conditionally independent given \mathcal{Z} , symbolically written as $A \not\perp B | \mathcal{Z}$. On observing the top node (Figure 2.9, middle), the upper path gets blocked. The lower path is also blocked, because the edges converge on the separating node, and neither the separating node itself nor its descendant is observed. Consequently, A and B are *d-separated*, and $A \perp B | \mathcal{Z}$. This changes when the descendant of the separating node is

observed (Figure 2.9, right), which opens the lower path and thus destroys the d-separation between A and B , implying that $A \not\perp B | \mathcal{Z}$.

Here, we have used the observation of nodes as an obvious criterion for membership in the separating set \mathcal{Z} . However, other membership criteria can also be employed. For instance, when trying to infer genetic networks from microarray experiments, described in Chapters 8 and 9, we are particularly interested in the up- and down-regulation of gene expression levels. So rather than asking whether two genes A and B are independent given a set \mathcal{Z} of measured mediating genes, we could ask whether A and B are independent conditional on a set \mathcal{Z}' of up- or down-regulated genes. We will return to this issue later, in Chapter 8.

2.1.2 The Parameters of a Bayesian Network

Recall that a BN is defined by a graphical structure, \mathcal{M} , a family of (conditional) probability distributions, \mathcal{F} , and their parameters, \mathbf{q} . The structure \mathcal{M} defines the independence relations between the interacting random variables, as discussed in the previous subsection and expressed in the factorization rule (2.1). The family \mathcal{F} defines the functional form of the (conditional) probabilities in the expansion (2.1) and defines, for instance, whether these probabilities are Gaussian, multinomial, etc. To fully specify the conditional probabilities associated with the edges we need certain parameters, for instance, the mean and variance of a Gaussian, etc. In what follows, the function family \mathcal{F} will be assumed to be fixed and known, chosen according to some criteria discussed in Section 8.2. Consequently, the probability distribution (2.1) is completely defined by the network structure, \mathcal{M} , henceforth also referred to as the *model*, and the vector of network *parameters*, \mathbf{q} . An illustration is given in Figure 2.10. Note that a parameter vector \mathbf{q} is associated with its respective network structure \mathcal{M} , with structures of different degrees of connectivity having associated parameter vectors of different dimension. Consequently, it would be more accurate to write $\mathbf{q}_{\mathcal{M}}$ instead of \mathbf{q} . For the sake of simplicity of the notation, however, the subscript is dropped.

2.2 Learning Bayesian Networks from Complete Data

2.2.1 The Basic Learning Paradigm

Our next goal is to learn a Bayesian network from a set of training data, \mathcal{D} . These data are assumed to be complete, meaning that observations or measurements are available on *all nodes* in the network. The case of *incomplete data* will be treated later, in Section 2.3.

Recall from the discussion in Sections 1.2 and 1.3 that there are two principled inference paradigms in machine learning and statistics. *Bayesian networks* are not necessarily related to the concept of *Bayesian learning*, and

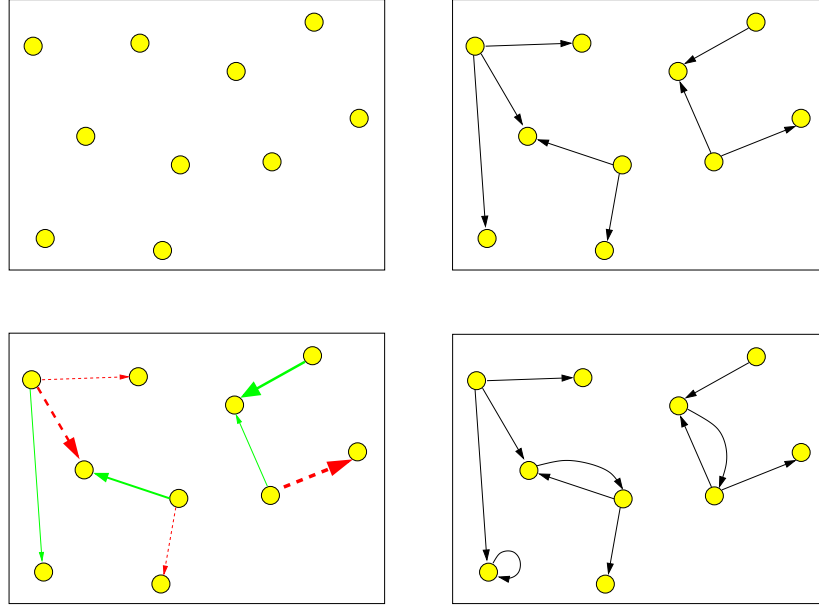


Fig. 2.10. Structure and parameters of a Bayesian network. *Top left:* A set of random variables for which we want to learn the Bayesian network, which is defined by its *structure* and its *parameters*. *Top right:* The structure \mathcal{M} defines the set of edges between the nodes, which indicate the interactions between the entities of interest. *Bottom left:* The parameters \mathbf{q} specify the functional form of the conditional probabilities associated with the edges, that is, they determine the nature of the interactions between the objects and indicate, for instance, whether an influence is strong (thick arrow) or weak (thin arrow) and whether an interaction is of an activating (solid line) or inhibitory (dashed line) nature. Note that the structure of the graph must satisfy the acyclicity constraint. A network with feedback loops, as shown in the *bottom right*, is *not* a Bayesian network.

learning Bayesian networks from data can, in fact, follow either of the two approaches discussed in Chapter 1. However, as pointed out in Section 1.3, and discussed in more detail in Section 4.4.7, a proper frequentist approach is often prohibitively computationally expensive. This chapter will therefore focus on the Bayesian approach.

Note from the discussion in the previous subsection that learning is a two-stage process, as illustrated in Figure 2.10. Given the data, we first want to find the posterior distribution of network *structures* \mathcal{M} and, from this distribution, the structure \mathcal{M}^* that is most supported by the data:

$$\mathcal{M}^* = \operatorname{argmax}_{\mathcal{M}} \left\{ P(\mathcal{M}|\mathcal{D}) \right\} \quad (2.17)$$

Then, given the best structure \mathcal{M}^* and the data, we want to find the posterior distribution of the parameters \mathbf{q} , and the best parameters:

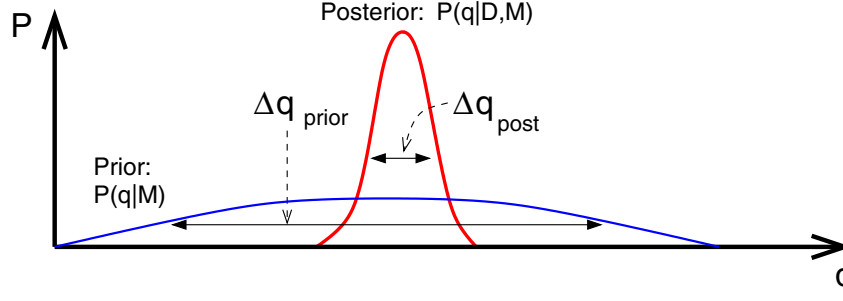


Fig. 2.11. Marginal likelihood and Occam factor. The figure shows a one-dimensional illustration of equation (2.20) to demonstrate the regularization effect of integrating out the network parameters. See text for details. Adapted from [3], by permission of Oxford University Press.

$$\mathbf{q}^* = \operatorname{argmax}_{\mathbf{q}} \left\{ P(\mathbf{q}|\mathcal{M}^*, \mathcal{D}) \right\} \quad (2.18)$$

Applying Bayes' rule (1.4) to (2.17) gives:

$$P(\mathcal{M}|\mathcal{D}) \propto P(\mathcal{D}|\mathcal{M})P(\mathcal{M}) \quad (2.19)$$

where the marginal likelihood $P(\mathcal{D}|\mathcal{M})$ implies an integration over the whole parameter space:

$$P(\mathcal{D}|\mathcal{M}) = \int P(\mathcal{D}|\mathbf{q}, \mathcal{M})P(\mathbf{q}|\mathcal{M})d\mathbf{q} \quad (2.20)$$

Note that this marginal likelihood includes an inherent penalty for unnecessary complexity. To see this simply, consider the one-dimensional case of Figure 2.11. Assume that the posterior probability $P(\mathbf{q}|\mathcal{D}, \mathcal{M})$ is unimodal and peaked around its mode, $\hat{\mathbf{q}}$, with width $\Delta\mathbf{q}_{post}$. Also, assume that the prior can be approximated by a distribution that is uniform over some large interval $\Delta\mathbf{q}_{prior}$. With this approximation, (2.20) becomes:

$$P(\mathcal{D}|\mathcal{M}) = P(\mathcal{D}|\hat{\mathbf{q}}, \mathcal{M}) \frac{\Delta\mathbf{q}_{post}}{\Delta\mathbf{q}_{prior}} \quad (2.21)$$

The first term on the right-hand side, $P(\mathcal{D}|\hat{\mathbf{q}}, \mathcal{M})$, is the likelihood, evaluated at the maximum likelihood parameters (note that for a uniform prior $P(\mathbf{q}|\mathcal{M})$, the MAP estimate $\hat{\mathbf{q}}$ is equal to the maximum likelihood estimate). Obviously, the likelihood is maximized for a complex structure with many edges, which is bound to over-fit to the observed data \mathcal{D} . The second term on the right-hand side, $\Delta\mathbf{q}_{post} / \Delta\mathbf{q}_{prior}$, referred to as the Occam factor in [3] and [26], measures the ratio of the posterior and prior accessible volumes in parameter space. For an over-complex model, this ratio will be small. The structure \mathcal{M} with the largest marginal likelihood $P(\mathcal{D}|\mathcal{M})$ will be determined by the

Number of nodes	2	4	6	8	10
Number of topologies	3	543	3.7×10^6	7.8×10^{11}	4.2×10^{18}

Table 2.1. Number of Bayesian network topologies as a function of the number of nodes in the graph. From [32].

trade-off between having to fit the data well, so as to get a large likelihood $P(\mathcal{D}|\hat{\mathbf{q}}, \mathcal{M})$, and the need to do so with a low model complexity, so as to get a large Occam factor. Consequently, even for a flat prior $P(\mathcal{M})$, the posterior probability (2.19) includes a penalty for unnecessary complexity, which guards against over-fitting.

This chapter will not discuss the choice of prior. It is just noted that under certain regularity conditions, the parameter priors $P(\mathbf{q}|\mathcal{M})$ for all structures \mathcal{M} can be specified using a single prior network, together with a “virtual” data count that describes the confidence in that prior [17].

Now, it can be shown that when certain regularity conditions for the prior $P(\mathbf{q}|\mathcal{M})$ and the likelihood $P(\mathcal{D}|\mathbf{q}, \mathcal{M})$ are satisfied and when the data are *complete*, then the integral in (2.20) becomes analytically tractable [16]. Unfortunately, this closed-form solution to (2.20) does not imply a straightforward solution to (2.17): the number of network structures increases super-exponentially with the number of nodes, as demonstrated in Table 2.1, and the optimization problem is known to be NP-hard [6]. We therefore have to resort to heuristic optimization methods, like hill-climbing or simulated annealing [22]. A heuristic acceleration of these procedures, which restricts the search in structure space to the most “relevant” regions, is discussed in [12].

2.2.2 Markov Chain Monte Carlo (MCMC)

In many situations, there is reason to question the appropriateness of the learning paradigm based on (2.17) altogether. For example, when inferring genetic network from microarray data, as discussed in Chapters 8 and 9, the data \mathcal{D} are usually sparse, which implies that the posterior distribution over structures, $P(\mathcal{M}|\mathcal{D})$, is likely to be diffuse. Consequently, $P(\mathcal{M}|\mathcal{D})$ will not be adequately represented by a single structure \mathcal{M}^* , as illustrated in Figure 8.5 on page 245, and it is more appropriate to sample networks from the posterior probability

$$P(\mathcal{M}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{M})P(\mathcal{M})}{P(\mathcal{D})} = \frac{P(\mathcal{D}|\mathcal{M})P(\mathcal{M})}{\sum_{\mathcal{M}'} P(\mathcal{D}|\mathcal{M}')P(\mathcal{M}')} \quad (2.22)$$

so as to obtain a representative sample of high-scoring network structures, that is, structures that offer a good explanation of the data. Again, a direct approach is impossible due to the denominator in (2.22), which in itself is a sum over the whole model space and, consequently, intractable. A solution to this problem, proposed by Metropolis et al. [30] and Hastings [15], reviewed,

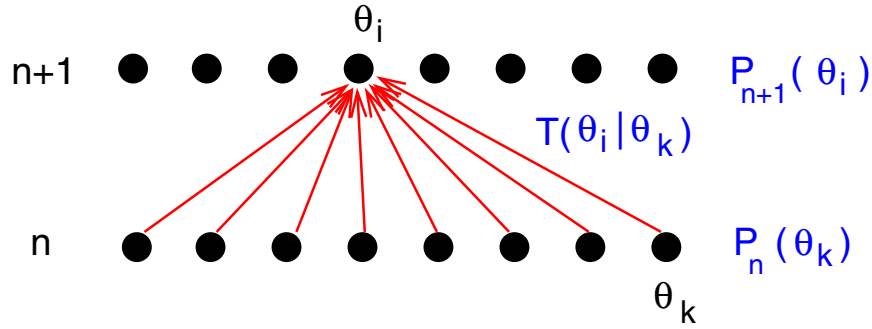


Fig. 2.12. Illustration of a Markov chain. $P_n(\theta_k)$ is the probability distribution in the n th step of the algorithm, where θ_k , in general, can be a structure, \mathcal{M}_k , a parameter vector, \mathbf{q}_k , or a combination of both, $(\mathcal{M}_k, \mathbf{q}_k)$. In the current application, we are only interested in the former, but applications of the other two cases will be discussed in later chapters of this book. The probability $P_n(\theta_k)$ evolves in time by application of the Markov transition matrix \mathbf{T} , which, in a single application, transforms $P_n(\theta_k)$ into $P_{n+1}(\theta_k)$. If the Markov chain is ergodic, $P_n(\theta_k)$ will converge to the equilibrium distribution $P_\infty(\theta_k)$, which is unique and independent of the initial conditions.

for instance, in [4], [13] and [28], and first applied to Bayesian networks by Madigan and York [29], is to devise a Markov chain

$$P_{n+1}(\mathcal{M}_i) = \sum_k T(\mathcal{M}_i|\mathcal{M}_k)P_n(\mathcal{M}_k) \quad (2.23)$$

that converges in distribution to the posterior probability $P(\mathcal{M}|\mathcal{D})$ of (2.22):

$$P_n(\mathcal{M}) \xrightarrow{n \rightarrow \infty} P(\mathcal{M}|\mathcal{D}) \quad (2.24)$$

The Markov matrix \mathbf{T} in (2.23) is a matrix of transition probabilities, with $T(\mathcal{M}_i|\mathcal{M}_k)$ denoting the probability of a transition from model \mathcal{M}_k into model \mathcal{M}_i : $\mathcal{M}_k \rightarrow \mathcal{M}_i$. An illustration of (2.23) is given in Figure 2.12.

The important feature of a Markov chain is that, under the fairly weak condition of *ergodicity*,¹ the distribution $P_n(\mathcal{M}_k)$ converges to a *stationary distribution* $P_\infty(\mathcal{M}_k)$:

$$P_n(\mathcal{M}_k) \xrightarrow{n \rightarrow \infty} P_\infty(\mathcal{M}_k) \quad (2.25)$$

This stationary distribution is independent of the initialization of the Markov chain and uniquely determined by the Markov transition matrix \mathbf{T} :

¹ A Markov chain is called *ergodic* if it is *aperiodic* and *irreducible*. An *irreducible* Markov chain is one in which all states are reachable from all other states. A sufficient test for *aperiodicity* is that each state has a “self-loop,” meaning that the probability that the next state is the same as the current state is non-zero. In general it is difficult to prove that a Markov chain is ergodic. However, ergodicity can be assumed to hold in most real-world applications.

$$P_\infty(\mathcal{M}_i) = \sum_k T(\mathcal{M}_i|\mathcal{M}_k)P_\infty(\mathcal{M}_k) \quad (2.26)$$

The idea, therefore, is to construct the transition matrix \mathbf{T} in such a way that the resulting Markov chain has the desired posterior probability $P(\mathcal{M}|\mathcal{D})$ of (2.22) as its stationary distribution: $P(\mathcal{M}|\mathcal{D}) = P_\infty(\mathcal{M})$. A sufficient condition for this to hold is the equation of *detailed balance*:

$$\frac{T(\mathcal{M}_k|\mathcal{M}_i)}{T(\mathcal{M}_i|\mathcal{M}_k)} = \frac{P(\mathcal{M}_k|\mathcal{D})}{P(\mathcal{M}_i|\mathcal{D})} = \frac{P(D|\mathcal{M}_k)P(\mathcal{M}_k)}{P(D|\mathcal{M}_i)P(\mathcal{M}_i)} \quad (2.27)$$

To prove that this holds true, we have to show that for a transition matrix \mathbf{T} satisfying (2.27), the posterior probability $P(\mathcal{M}|\mathcal{D})$ of (2.22) is the stationary distribution and therefore obeys (2.26):

$$\sum_k T(\mathcal{M}_i|\mathcal{M}_k)P(\mathcal{M}_k|\mathcal{D}) = P(\mathcal{M}_i|\mathcal{D}) \quad (2.28)$$

Now, from (2.27) we have

$$T(\mathcal{M}_k|\mathcal{M}_i)P(\mathcal{M}_i|\mathcal{D}) = T(\mathcal{M}_i|\mathcal{M}_k)P(\mathcal{M}_k|\mathcal{D}) \quad (2.29)$$

and consequently

$$\begin{aligned} \sum_k T(\mathcal{M}_i|\mathcal{M}_k)P(\mathcal{M}_k|\mathcal{D}) &= \sum_k T(\mathcal{M}_k|\mathcal{M}_i)P(\mathcal{M}_i|\mathcal{D}) \\ &= P(\mathcal{M}_i|\mathcal{D}) \sum_k T(\mathcal{M}_k|\mathcal{M}_i) \\ &= P(\mathcal{M}_i|\mathcal{D}) \end{aligned} \quad (2.30)$$

which is identical to (2.28) and thus completes the proof. Note that the last step in (2.30) follows from the fact that $T(\mathcal{M}_k|\mathcal{M}_i)$ is a conditional probability, which is normalized.

In practically setting up a Markov chain, note that a transition into another structure, $\mathcal{M}_k \rightarrow \mathcal{M}_i$, consists of two parts. First, given \mathcal{M}_k , a new structure is proposed with a proposal probability $Q(\mathcal{M}_i|\mathcal{M}_k)$. In a second step, this new structure is then accepted with an acceptance probability $A(\mathcal{M}_i|\mathcal{M}_k)$. A transition probability is therefore given by the product of a proposal and an acceptance probability and can be written as

$$T(\mathcal{M}_k|\mathcal{M}_i) = Q(\mathcal{M}_k|\mathcal{M}_i)A(\mathcal{M}_k|\mathcal{M}_i) \quad (2.31)$$

The proposal probabilities $Q(\mathcal{M}_k|\mathcal{M}_i)$ are defined by the way we design our moves in the model space (see, for instance, Figure 2.15). From (2.27) and (2.31), we then obtain the following condition for the acceptance probabilities:

$$\frac{A(\mathcal{M}_k|\mathcal{M}_i)}{A(\mathcal{M}_i|\mathcal{M}_k)} = \frac{P(D|\mathcal{M}_k)P(\mathcal{M}_k)Q(\mathcal{M}_i|\mathcal{M}_k)}{P(D|\mathcal{M}_i)P(\mathcal{M}_i)Q(\mathcal{M}_k|\mathcal{M}_i)} \quad (2.32)$$

for which a sufficient condition is

$$A(\mathcal{M}_k|\mathcal{M}_i) = \min \left\{ \frac{P(D|\mathcal{M}_k)P(\mathcal{M}_k)Q(\mathcal{M}_i|\mathcal{M}_k)}{P(D|\mathcal{M}_i)P(\mathcal{M}_i)Q(\mathcal{M}_k|\mathcal{M}_i)}, 1 \right\} \quad (2.33)$$

To summarize: Accepting new configurations \mathcal{M}_k with the probability (2.33) is a sufficient condition for satisfying the equation of detailed balance (2.27) which itself (assuming ergodicity) is a sufficient condition for the convergence of the Markov chain to the desired posterior distribution (2.22). While a direct computation of the posterior probability (2.22) is intractable due to the sum in the denominator, the equation of detailed balance (2.27) and the acceptance criterion (2.33) only depend on the ratio of the posterior probabilities. Consequently, the intractable denominator cancels out. The algorithm, thus, can be summarized as follows:

Metropolis–Hastings algorithm

- Start from an initial structure $\mathcal{M}^{(0)}$
- Iterate for $n = 1 \dots N$
 1. Obtain a new structure $\mathcal{M}^{(n)}$ from the proposal distribution $Q(\mathcal{M}^{(n)}|\mathcal{M}^{(n-1)})$
 2. Accept the new model with probability $A(\mathcal{M}^{(n)}|\mathcal{M}^{(n-1)})$, given by (2.33), otherwise leave the model unchanged: $\mathcal{M}^{(n)} = \mathcal{M}^{(n-1)}$
- Discard an initial equilibration or burn-in period to allow the Markov chain to reach stationarity. For example, discard $\mathcal{M}_1, \dots, \mathcal{M}_{N/2}$
- Compute expectation values from the MCMC sample $\{\mathcal{M}_{N/2+1}, \dots, \mathcal{M}_N\}$:

$$\langle f \rangle = \sum_{\mathcal{M}} f(\mathcal{M})P(\mathcal{M}|\mathcal{D}) \simeq \frac{2}{N} \sum_{n=N/2+1}^N f(\mathcal{M}_n)$$

An illustration is given in Figure 2.13. Note that this algorithm is not restricted to discrete (cardinal) entities, like topologies \mathcal{M} , but that it can equally be applied to continuous entities, like network parameters \mathbf{q} . In this case expectation values are typically given by integrals of the form $\langle f \rangle = \int f(\mathbf{q})P(\mathbf{q}|\mathcal{D})d\mathbf{q}$, which are approximated by discrete sums over the parameters $\{\mathbf{q}_1, \dots, \mathbf{q}_N\}$ sampled along the MCMC trajectory:

$$\langle f \rangle = \int f(\mathbf{q})P(\mathbf{q}|\mathcal{D})d\mathbf{q} \simeq \frac{2}{N} \sum_{n=N/2+1}^N f(\mathbf{q}_n)$$

The MCMC approximation is exact in the limit of an infinitely long Markov chain. In theory, the initialization of the Markov chain and the details of the proposal distribution are unimportant: if the condition of detailed balance (2.27) is satisfied, an ergodic Markov chain will converge to its stationary distribution (2.28) irrespective of these details. In practice, however, extreme starting values and unskillfully chosen proposal distributions may slow down

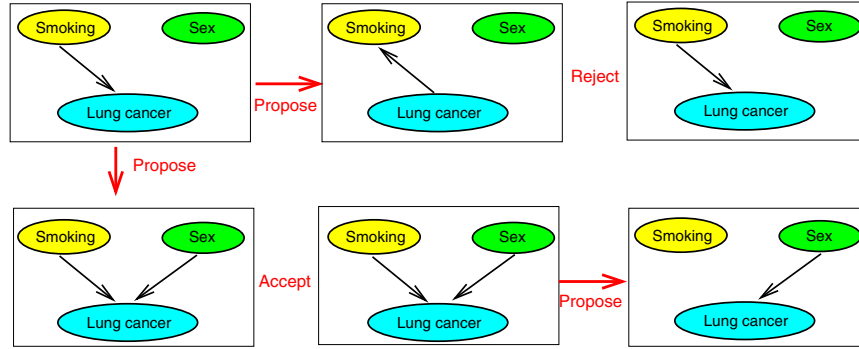


Fig. 2.13. Illustration of MCMC. Starting from the initial network shown in the top left, a new network is proposed (top middle) and accepted with a probability given by (2.33). In this example, the proposed network is rejected (top right). A new network is proposed (bottom left) and again accepted with a probability given by (2.33), which in this example leads to an acceptance of the proposed structure (bottom middle). This structure is now taken as the initial configuration, from which a new network is proposed in the next move (bottom right).

Step size λ	0.5	2.0	10.0
Relative error	79%	17%	58%

Table 2.2. Dependence of the relative prediction error on the steps size of the proposal scheme for the MCMC simulation of Figure 2.14.

the mixing and convergence of the chain and result in a very long burn-in, in which case the MCMC sampler may fail to converge towards the main support of the stationary distribution in the available simulation time. An example is given in Figure 2.14, where we want to infer the mean of a univariate Normal distribution from an MCMC sample of size 200, discarding a burn-in phase of the first 100 MCMC steps. (This is just an illustration. In practice we would not resort to an MCMC simulation to solve this simple problem.) Our entity of interest, in this case, is a single continuous random variable q , and we choose the proposal distribution $Q(q^{(n+1)}|q^{(n)})$ to be a uniform distribution over an interval of length λ (the *step size*), centred on the current value in the Markov chain, $q^{(n)}$. Figure 2.14 shows a trace plot of $q^{(n)}$ for three values of the step size λ . In the left subfigure, λ has been chosen too small, and the convergence of the Markov chain is slow. This is indicated by a high acceptance ratio of about 80% of the moves, which suggests that the step size should be increased. In the right subfigure, λ has been chosen too large, leading to a Markov chain that is too sticky and has a very low acceptance ratio of only 17%, wasting a lot of computer time by rejecting most of the moves. The subfigure in the middle shows an appropriate choice of λ , where the acceptance ratio is about 50%. This optimal choice results in the fastest convergence of the Markov chain and is reflected by the smallest prediction

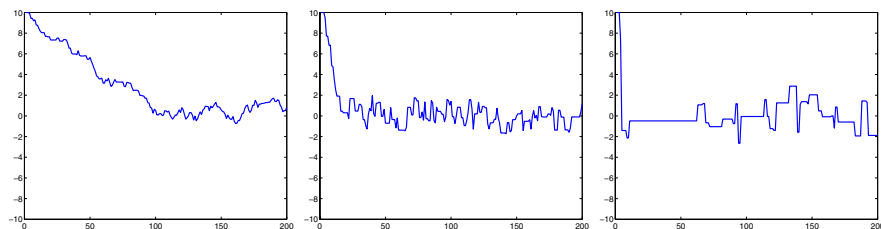


Fig. 2.14. Dependence of the MCMC convergence on the proposal distribution. The figures show MCMC trace plots of a univariate continuous random variable q with a (known) normal $N(0, 1)$ distribution; hence the true mean is $\langle q \rangle = 0$. The proposal distribution $Q(q^{(n+1)}|q^{(n)})$ was chosen to be a uniform distribution over an interval of length λ , centred on the current value $q^{(n)}$. The figures were obtained for three different values of the step size λ . Left: $\lambda = 0.5$; middle: $\lambda = 2.0$; right: $\lambda = 10.0$.

error, as seen from Table 2.2. In this particular example, the choice of the proposal distribution (determined by λ) is not particularly critical because one can easily continue the MCMC simulation over about 10,000 or 100,000 steps, in which case all three alternatives give practically identical results. For more complex problems, however, where computer time is a critical issue, the optimization of the proposal distribution can become most important. The previous example suggests that, starting from a random choice of λ , one should adjust this parameter until an acceptance ratio of about 50% is reached. In general, the parameters of more complex proposal distributions should be tuned in a similar way. Note, however, that this tuning has to be restricted to the burn-in phase of the algorithm, and it must not be continued in the sampling phase. The reason is that optimizing the parameters of the proposal distribution on the basis of a history of past configurations violates the condition of detailed balance (2.27) and may therefore lead to a biased distribution that may not be representative of the true stationary distribution (2.28).

When the proposal distribution is symmetric, $Q(\mathcal{M}_k|\mathcal{M}_i) = Q(\mathcal{M}_i|\mathcal{M}_k)$, it cancels out in (2.33), and the algorithm reduces to the *Metropolis algorithm* [30]. For asymmetric proposal distributions, the scheme is called the *Metropolis–Hastings algorithm* [15], and the ratio of the proposal probabilities is usually referred to as the *Hastings ratio*. While in some cases the asymmetry of the proposal distribution is introduced deliberately as a means of accelerating the convergence of the Markov chain, it is often inherent in the nature of the proposal mechanism and needs to be considered carefully by the user in order to avoid biased results. Take, for example, the proposal mechanism for generating new DAGs, as illustrated in Figure 2.15. One might, naively, assume that the proposal distribution is symmetric. After all, there are only three elementary operations – edge creation, edge reversal, and edge deletion – all of which can be chosen with the same probability for the forward and the

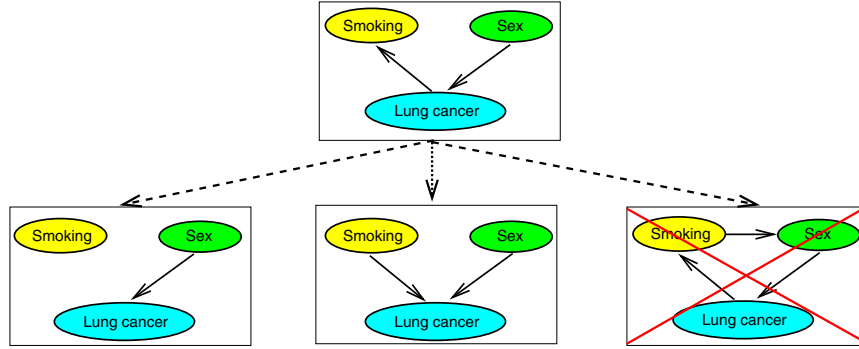


Fig. 2.15. Elementary MCMC moves for DAGs. The figure shows three typical elementary proposal moves: (1) deletion of an edge (left), (2) reversal of an edge (middle), and (3) creation of a new edge (right). Note that the last two operations may lead to graphs that violate the acyclicity constraint and therefore have to be discarded. An example is shown on the right.

backward move and, therefore, should cancel out when computing the Hastings ratio. A more careful consideration, however, reveals that this assumption is false. The reason for this fallacy is that, as a consequence of the acyclicity constraint, certain proposal moves will lead to invalid DAGs that have to be discarded (as illustrated in Figure 2.15). Figure 2.16 demonstrates how the Hastings ratio is computed properly. The figure shows the neighbourhoods of two DAGs, where the neighbourhood is the set of all valid DAGs that can be reached from the given DAG with one of the elementary operations of Figure 2.15. As a consequence of the acyclicity constraint, the neighbourhoods of two neighbouring DAGs are not necessarily of the same size. Consequently, the proposal probability of an MCMC move, which is given by the inverse of the neighborhood size, is not equal to that of the opposite move, leading to a Hastings ratio that is different from 1. For complex networks with large neighbourhoods, the computation of the Hastings ratio is therefore not trivial and requires the determination of the number of all valid (acyclic) graphs in the neighbourhoods of the two DAGs involved in the proposal move.

Recall that in theory an ergodic Markov chain converges to the true posterior distribution irrespective of the choice of the proposal distribution and the initialization. In practice, however, it is difficult to decide whether an MCMC simulation has sufficiently converged. A simple heuristic convergence test is shown in Figure 2.17. Note, however, that passing the indicated test is only a necessary rather than a sufficient condition for convergence as it may not distinguish between meta-stable disequilibrium and true equilibrium.

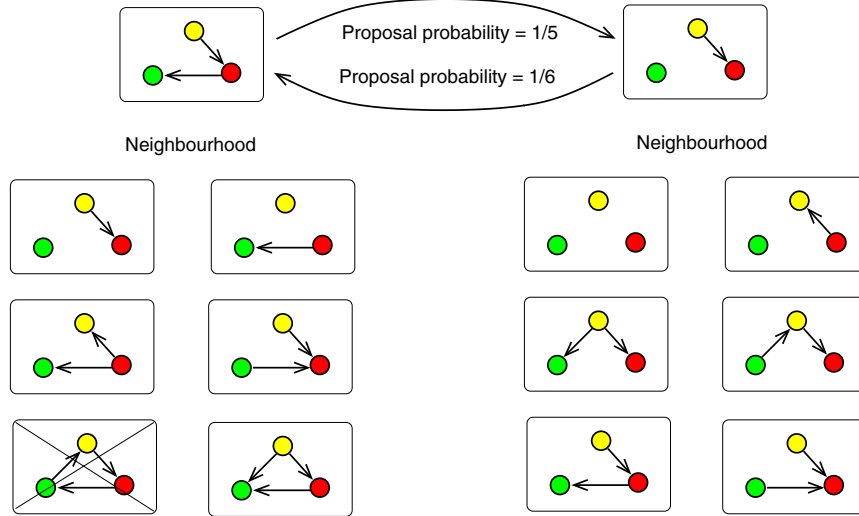


Fig. 2.16. DAG neighbourhoods and Hastings ratio. The figure shows the neighbourhoods of two DAGs, where the neighbourhood of a DAG is the set of all valid DAGs that can be reached from the specified DAG with one of the elementary operations of Figure 2.15. The neighbourhoods of two neighbouring DAGs are not necessarily of the same size: while the DAG on the right has six neighbours, the DAG on the left has only five because one of the graphs in its neighbourhood violates the acyclicity constraint. The Hastings ratio is given by the ratio of the neighbourhood sizes of the two networks involved in the proposal move. The Hastings ratio for an MCMC proposal move from the left to the right DAG is thus given by $5/6$, while the Hastings ratio for the opposite move is $6/5$.

2.2.3 Equivalence Classes

Figure 2.18 shows the four elementary Bayesian networks we have already encountered several times in this chapter. All networks have the same skeleton, which is the configuration of edges without their direction, but they differ with respect to the edge directions. However, expanding the joint probability $P(A, B, C)$ according to (2.1) gives the same factorization for three of the networks irrespective of the edge directions. These networks are therefore *equivalent*, that is, they show alternative ways of describing the same set of independence relations. In general, it can be shown that networks are equivalent if and only if they have the same skeleton and the same *v-structure*, where the latter denotes a configuration of two directed edges converging on the same node without an edge between the parents [5]. An equivalence class can be represented by a *partially directed acyclic graph* (PDAG), which is a graph that contains both directed and undirected edges. An example is given in the bottom of Figure 2.18, and in Figure 2.19, where the subfigure on the right shows the equivalence class that corresponds to the Bayesian network on

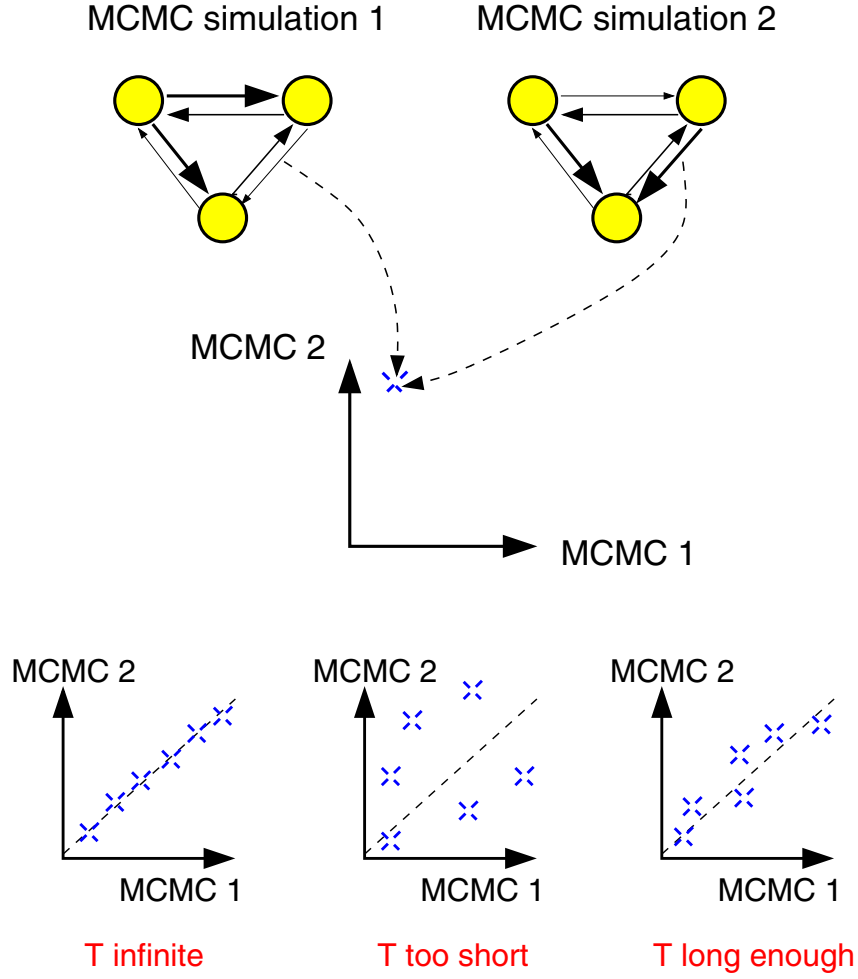


Fig. 2.17. Convergence test for MCMC simulations. *Top:* MCMC simulations are started from different initializations and/or different random number generator seeds. Corresponding posterior probabilities of the edges, obtained from different simulations, are plotted against each other. *Bottom left:* *Infinite simulation time T .* For an infinitely long simulation time, all MCMC simulations give the same results: the estimated posterior probabilities of the edges are equal to the true posterior probabilities irrespective of the initialization of the Markov chain, and the scatter plot has the form of a straight line. *Bottom middle:* *Simulation time T too short.* Insufficient convergence or mixing of the Markov chain is indicated by a scatter plot that strongly deviates from the straight line. This deviation indicates a strong dependence of the results on the initialization, resulting from insufficient convergence or mixing. *Bottom right:* *Simulation time T long enough.* A necessary condition for sufficient convergence and mixing is a scatter plot that does not deviate markedly from the diagonal line.

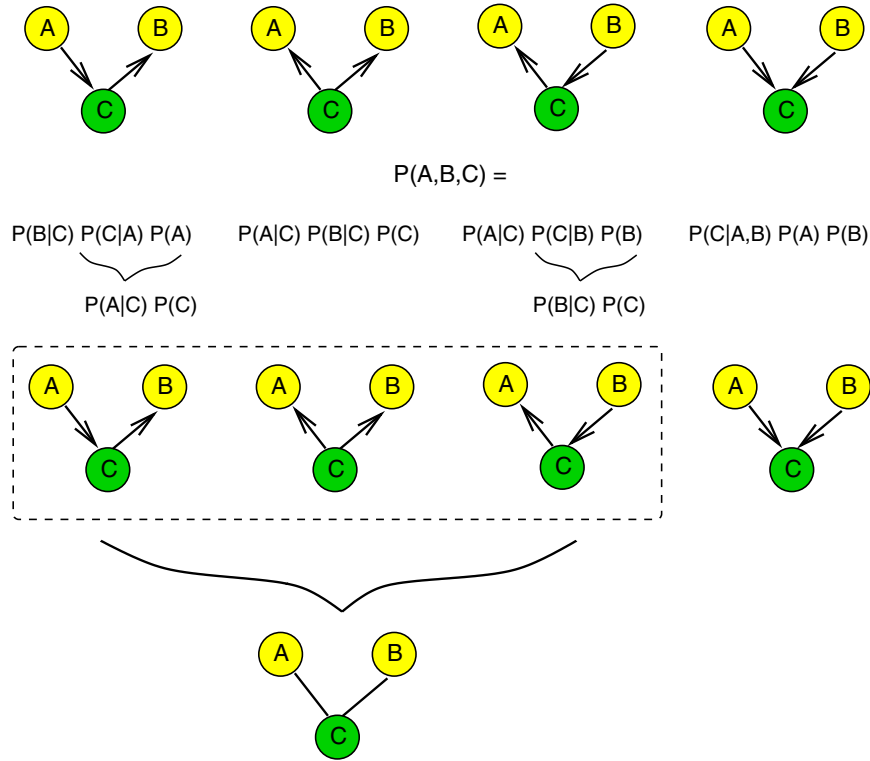


Fig. 2.18. Equivalent Bayesian networks. The top subfigure shows four BNs with their respective expansion of the joint probability distribution. The three BNs on the left are equivalent and lead to the same expansion. These BNs thus belong to the same equivalence class, which can be represented by the undirected graph at the bottom.

the left. Note that an undirected edge indicates that its direction is not un-equivocal among the DAGs in the equivalence class represented by the PDAG. Conversely, a directed edge indicates that all DAGs in the equivalence class concur about its direction.

Under fairly general conditions, equivalent BNs have the same likelihood score [16].² Unless this symmetry is broken by the prior,³ the posterior probabilities are the same, that is, equivalent BNs can *not* be distinguished on the basis of the data. This implies that, in general, we can only learn PDAGs

² Heckerman [16] distinguishes between *structure equivalence*, identical to the notion of equivalence used in the present chapter, and *distribution equivalence*. The latter equivalence concept is defined with respect to a distribution family \mathcal{F} and implies invariance with respect to the likelihood.

³ Heckerman et al. [17] discuss the choice of priors that do not break the symmetry of equivalence classes.

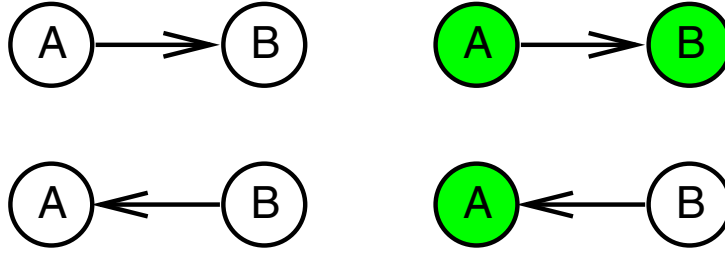


Fig. 2.20. Effect of intervention. The two DAGs on the left are in the same equivalence class, and the edge direction can *not* be inferred from observations alone. To infer the causal direction, the value for A has to be set externally. If A is a causal ancestor of B , this intervention is likely to lead to a changed value of B (top right). If, however, B is a causal ancestor of A , this intervention will have no effect on B (bottom right).

tence of equivalence classes. Recall from the discussion of the previous section that we can only learn equivalence classes of DAGs (represented by PDAGs) rather than DAGs themselves. This implies that effectively a lot of edge directions get lost, obstructing the inference of causality. In what follows we have to distinguish between *observations* and *interventions*. An *observation* is a passive measurement of variables in the domain of interest, for instance, the simultaneous measurement of gene expression levels in a standard microarray experiment.⁴ In an *intervention*, the values of some variables are set from outside the system, for instance, by knocking out or over-expressing a particular gene. An example is given in Figure 2.20. More formally, recall that the likelihood scores of two BNs with equivalent DAGs, \mathcal{M} and \mathcal{M}' , are the same, where the likelihood is computed from (2.1). When setting the value of node k externally to some value x_k^* , then $P(X_k = x_k^* | \mathcal{X}_{pa[k]}) = 1$. This is because setting a value by external force means that the respective node takes on this particular value with probability 1 irrespective of the values of the other nodes in the network. Consequently, the contributions of all those nodes that are subject to intervention effectively disappear from (2.1):

$$P(X_1, X_2, \dots, X_n) = \prod_{i \notin I} P(X_i | \mathcal{X}_{pa[i]}) \quad (2.34)$$

where I is the set of intervened nodes. This modification can destroy the symmetry within an equivalence class, that is, the likelihood scores for \mathcal{M} and \mathcal{M}' might no longer be the same, which may resolve the ambiguity about certain edge directions.

A second and potentially more serious difficulty in trying to learn causal structures from data is the possible presence of hidden, unobserved variables. Figure 2.20, for instance, shows two possible DAG structures that explain

⁴ See Chapter 7 for an introduction to microarray experiments.

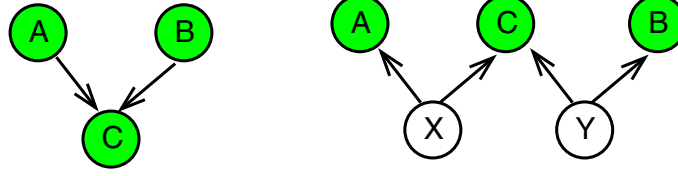


Fig. 2.21. Equivalence of networks with hidden variables. The network on the left without hidden variables is equivalent to the network on the right, which contains two additional hidden variables. Filled circles represent observed, empty circles represent hidden nodes.

a conditional dependence between two random variables. However, a third possibility is that both observed random variables depend on a third, hidden variable, as in Figure 2.3. Another example is given in Figure 2.21, where the network on the left, which only includes observed nodes, is equivalent to the network on the right, which contains two extra hidden nodes. This equivalence can easily be shown. Applying the factorization rule (2.1) to the graph on the right gives:

$$\begin{aligned} P(A, B, C, X, Y) &= P(A|X)P(X)P(C|X, Y)P(B|Y)P(Y) \\ &= P(X|A)P(A)P(C|X, Y)P(Y|B)P(B) \end{aligned}$$

where in the second step (1.3) has been used. Then, marginalizing over the unobserved variables X and Y yields

$$\begin{aligned} P(A, B, C) &= \sum_X \sum_Y P(A, B, C, X, Y) \\ &= P(A)P(B) \sum_X \sum_Y P(C|X, Y)P(X|A)P(Y|B) \\ &= P(A)P(B)P(C|A, B) \end{aligned}$$

This result is identical to the factorization one obtains from the structure on the left of Figure 2.21. Consequently, the two network structures in Figure 2.21 are equivalent, and we can *not* decide whether A and B are causal ancestors of C , or whether all three variables are controlled by some hidden causal ancestors. A more complex analysis [40] reveals that it is possible to characterize all networks with latent variables that can result in the same set of independence relations over the observed variables. However, it is not clear how to score such an equivalence class, which consists of many models with different numbers of latent variables, and this lack of a score defies any inference procedure.

2.3 Learning Bayesian Networks from Incomplete Data

2.3.1 Introduction

The previous section was based on the assumption that we can find a closed-form solution to the integral in (2.20). This is only possible if we have complete observation without any missing or *hidden* variables. In many applications this cannot be assumed, since observations may be missing either systematically or at random. Take, for example, a microarray experiment, where we measure the expression levels of hundreds or thousands of genes simultaneously. As described in Chapter 7, some genes get occasionally *flagged*, meaning that the data quality is so poor that these measurements are at best ignored. It is also known that certain interactions in genetic regulatory networks are mediated by transcription factors, whose activation is often undetectable at the level of gene expression. In phylogenetics, covered in Chapters 4–6, DNA or RNA sequences are only available for contemporary or extant species, while those for extinct species are systematically missing. Further examples will be given in later chapters of this book. In all these applications, the assumption of complete observation is violated, and the integration in (2.20) becomes intractable.

2.3.2 Evidence Approximation and Bayesian Information Criterion

A simple approximation to (2.20) is as follows. First, consider a focused model in which $P(\mathbf{q}|\mathcal{D}, \mathcal{M})$ is assumed to be dominated by the likelihood, either because of assuming a uniform prior on the parameters, or by increasing the sample size. That is, we set (assuming prior parameter independence)

$$P(\mathbf{q}|\mathcal{M}) = \prod_{i=1}^{\nu} P(q_i|\mathcal{M}) = c^{\nu} \quad (2.35)$$

where $\nu = \dim(\mathbf{q})$ is the dimension of the parameter space, and c is a constant. Next, define

$$E(\mathbf{q}) = -\log P(\mathcal{D}|\mathbf{q}, \mathcal{M}) \quad (2.36)$$

$$\hat{\mathbf{q}} = \operatorname{argmin}_{\mathbf{q}} E(\mathbf{q}) = \operatorname{argmax}_{\mathbf{q}} P(\mathcal{D}|\mathbf{q}, \mathcal{M}) \quad (2.37)$$

$$\mathbf{H} = \left[\nabla_{\mathbf{q}} \nabla_{\mathbf{q}}^{\dagger} E(\mathbf{q}) \right]_{\mathbf{q}=\hat{\mathbf{q}}} = -\nabla_{\mathbf{q}} \nabla_{\mathbf{q}}^{\dagger} \left[\log P(\mathcal{D}|\mathbf{q}, \mathcal{M}) \right]_{\mathbf{q}=\hat{\mathbf{q}}} \quad (2.38)$$

where $\hat{\mathbf{q}}$ is the vector of maximum likelihood parameters, and \mathbf{H} is the Hessian or empirical *Fisher information matrix*. Inserting (2.36) and (2.35) into (2.20) gives

$$P(\mathcal{D}|\mathcal{M}) = \int P(\mathcal{D}|\mathbf{q}, \mathcal{M}) P(\mathbf{q}|\mathcal{M}) d\mathbf{q} = c^{\nu} \int \exp \left[-E(\mathbf{q}) \right] d\mathbf{q} \quad (2.39)$$

Approximating the negative log-likelihood by a second-order Taylor series expansion, the so-called *Laplace approximation*,

$$E(\mathbf{q}) \approx E(\hat{\mathbf{q}}) + \frac{1}{2}(\mathbf{q} - \hat{\mathbf{q}})^\dagger \mathbf{H}(\mathbf{q} - \hat{\mathbf{q}}) \quad (2.40)$$

where the superscript \dagger denotes matrix transposition, and inserting (2.40) into (2.39), we get

$$P(\mathcal{D}|\mathcal{M}) \approx c^\nu \exp \left[-E(\hat{\mathbf{q}}) \right] \int \exp \left[-\frac{1}{2}(\mathbf{q} - \hat{\mathbf{q}})^\dagger \mathbf{H}(\mathbf{q} - \hat{\mathbf{q}}) \right] d\mathbf{q} \quad (2.41)$$

$$= P(\mathcal{D}|\hat{\mathbf{q}}, \mathcal{M}) c^\nu \sqrt{\frac{(2\pi)^\nu}{\det \mathbf{H}}} \quad (2.42)$$

Taking logs, this gives:

$$\log P(\mathcal{D}|\mathcal{M}) = \log P(\mathcal{D}|\hat{\mathbf{q}}, \mathcal{M}) - \frac{1}{2} \log \det \mathbf{H} + \frac{\nu}{2} \log(2\pi c^2) \quad (2.43)$$

Equation (2.43), which in the neural network literature is referred to as the *evidence approximation* [26], [27], decomposes $\log P(\mathcal{D}|\mathcal{M})$ into two terms: the maximum log likelihood score, $\log P(\mathcal{D}|\hat{\mathbf{q}}, \mathcal{M})$, and a *penalty* or *regularization* term that depends on the Hessian \mathbf{H} . The integration (2.20) is thus reduced to an optimization, to obtain $\log P(\mathcal{D}|\hat{\mathbf{q}}, \mathcal{M})$, and the computation of the Hessian. For complex models, this computation of the Hessian can be quite involved [18], [19]. Therefore, a further approximation is often applied. Note that the Hessian \mathbf{H} is symmetric and positive semi-definite, as seen from (2.38), and it thus has ν real nonnegative eigenvalues, $\{\varepsilon_i\}$, $i = 1, \dots, \nu$. The determinant of a matrix is given by the product of its eigenvalues, which allows (2.43) to be rewritten as follows:

$$\log P(\mathcal{D}|\mathcal{M}) = \log P(\mathcal{D}|\hat{\mathbf{q}}, \mathcal{M}) - \frac{1}{2} \sum_{i=1}^{\nu} \log \left(\frac{\varepsilon_i}{2\pi c^2} \right) \quad (2.44)$$

The eigenvalues ε_i determine the curvature of the log-likelihood surface along the eigendirections at the maximum likelihood parameters $\hat{\mathbf{q}}$. This curvature increases with the sample size N , so the eigenvalues can be assumed to be proportional to N : $\varepsilon_i \propto N$. Now, introducing the further approximation of isotropy, that is, assuming the same curvature along all eigendirections:

$$\varepsilon_i \approx 2\pi c^2 N \quad \forall i \quad (2.45)$$

we get (recall that $\nu = \dim(\mathbf{q})$):

$$\log P(\mathcal{D}|\mathcal{M}) \approx \log P(\mathcal{D}|\hat{\mathbf{q}}, \mathcal{M}) - \frac{\nu}{2} \log N \quad (2.46)$$

This simple formula, which is a variant of the *minimum description length* in information theory [37], is known as the BIC (Bayesian information criterion)

approximation, introduced by Schwarz [38]. The first term is the maximum likelihood estimate for model \mathcal{M} . The second term is a regularization term, which penalizes model complexity and results from the integration; compare with the discussion in Section 2.2.1. However, for sparse data \mathcal{D} , neither the Laplace approximation nor the assumption of isotropy, (2.45), are reasonable. In particular, both approximations assume that the likelihood function is unimodal, which does not hold for many models. In fact, this deviation from unimodality becomes particularly noticeable for small data sets, which may render both the evidence and BIC approximations unreliable. A different, simulation-based approach that overcomes this shortcoming will be discussed in Section 2.3.7.

2.3.3 The EM Algorithm

The previous section has demonstrated that under certain approximations the integration (2.20) reduces to an optimization problem, namely, to find the maximum likelihood parameters $\hat{\mathbf{q}}$ for a given model \mathcal{M} . This optimization, however, may not be trivial due to the presence of hidden variables. Denote by \mathcal{D} the data corresponding to observed nodes in the graph. Denote by $\mathcal{S} = \{S_1, \dots, S_M\}$ the set of hidden nodes and their associated random variables. The log likelihood is given by

$$L(\mathbf{q}, \mathcal{M}) = \log P(\mathcal{D}|\mathbf{q}, \mathcal{M}) = \log \sum_{\mathcal{S}} P(\mathcal{D}, \mathcal{S}|\mathbf{q}, \mathcal{M}) \quad (2.47)$$

which involves a marginalization over all possible configurations of hidden states. If each hidden state S_i , $i = 1, \dots, M$, has K discrete values, we have to sum over K^M different terms, which for large values of M becomes intractable. To proceed, let $Q(\mathcal{S})$ denote some arbitrary distribution over the set of hidden states, and define

$$F(\mathbf{q}, \mathcal{M}) = \sum_{\mathcal{S}} Q(\mathcal{S}) \log \frac{P(\mathcal{D}, \mathcal{S}|\mathbf{q}, \mathcal{M})}{Q(\mathcal{S})} \quad (2.48)$$

$$KL[Q, P] = \sum_{\mathcal{S}} Q(\mathcal{S}) \log \frac{Q(\mathcal{S})}{P(\mathcal{S}|\mathcal{D}, \mathbf{q}, \mathcal{M})} \quad (2.49)$$

KL in (2.49) is the Kullback–Leibler divergence between the distributions Q and P , which is always non-negative and zero if and only if $Q = P$. The proof is based on the concavity of the log function and the normalization condition for probabilities: $\sum_{\mathcal{S}} Q(\mathcal{S}) = 1$:

$$\begin{aligned} \log x \leq x - 1 &\implies \log \frac{P(\mathcal{S})}{Q(\mathcal{S})} \leq \frac{P(\mathcal{S})}{Q(\mathcal{S})} - 1 \\ \implies Q(\mathcal{S}) \log \frac{P(\mathcal{S})}{Q(\mathcal{S})} &\leq P(\mathcal{S}) - Q(\mathcal{S}) \implies \sum_{\mathcal{S}} Q(\mathcal{S}) \log \frac{P(\mathcal{S})}{Q(\mathcal{S})} \leq 0 \\ \implies KL[Q, P] &\geq 0 \end{aligned}$$

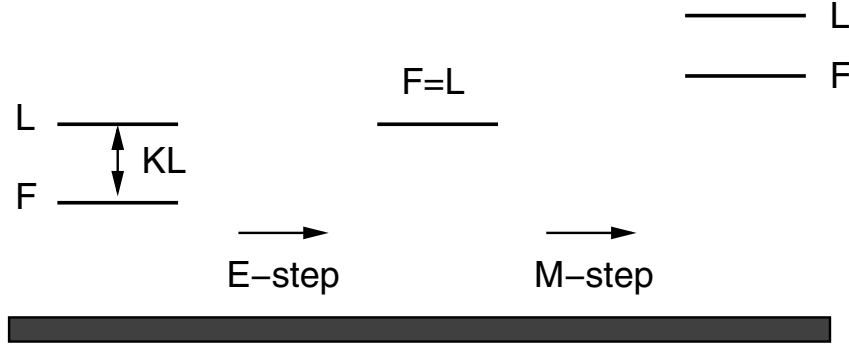


Fig. 2.22. Illustration of the EM algorithm. L , F , and KL are defined in equations (2.47)–(2.49). The algorithm is explained in the text.

From (2.47)–(2.49) we get:

$$L(\mathbf{q}, \mathcal{M}) = F(\mathbf{q}, \mathcal{M}) + KL[Q, P] \quad (2.50)$$

This is the fundamental equation of the Expectation Maximization (EM) algorithm [8], [33]. An illustration is given in Figure 2.22. F is a lower bound on the log-likelihood L , with a difference given by KL . The E-step holds the parameters \mathbf{q} fixed and sets $Q(\mathcal{S}) = P(\mathcal{S}|\mathcal{D}, \mathbf{q}, \mathcal{M})$; hence $KL(Q, P) = 0$ and $F = L$. The M-step holds the distribution $Q(\mathcal{S})$ fixed and computes the parameters \mathbf{q} that maximize F . Since $F = L$ at the beginning of the M-step, and since the E-step does not affect the model parameters, each EM cycle is guaranteed to increase the likelihood unless the system has already converged to a (local) maximum (or, less likely, a saddle point). The power of the EM algorithm results from the fact that F is usually considerably easier to maximize with respect to the model parameters \mathbf{q} than L . An example is given in Section 2.3.5.

2.3.4 Hidden Markov Models

A hidden Markov model (HMM) is a particular example of a Bayesian network with hidden nodes. In fact, the structure of an HMM is comparatively simple, which makes it an appropriate example for illustrating the concepts of the preceding subsection. Also, HMMs have been extensively applied in bioinformatics, and they will play an important role in later chapters of this book; see Section 5.10, Section 10.11.4, and Chapter 14. An illustrative example is given in Figure 2.23. Assume you are in a casino and take part in a gambling game that involves a die. You are playing against two croupiers: a fair croupier, who uses a fair die, and a corrupt croupier, who uses a loaded die. Unfortunately, the croupiers are hidden behind a wall, and all you observe is a sequence of die faces. The task is to predict which croupier is rolling the

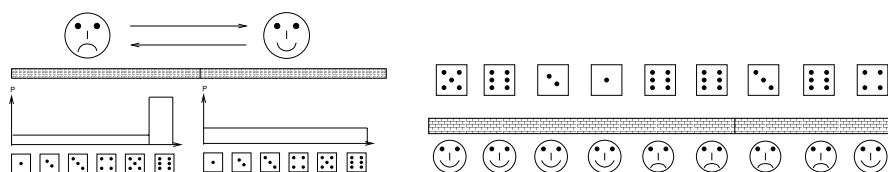


Fig. 2.23. The occasional corrupt casino. *Left:* Two croupiers are in a casino: a fair croupier, who uses a fair die, and a corrupt croupier, who uses a biased die. *Right:* The player only sees a sequence of die faces, but not the croupier, from whom he is separated by a wall. The task is to infer, from the sequence of observed die faces, which croupier has been rolling the die and to predict the breakpoint when the corrupt croupier is taking over. The idea for this illustration is taken from [9].

$$\begin{array}{cccccccc} \text{A} & \text{C} & \text{G} & \text{T} & \text{T} & \text{A} & \text{T} & \text{A} \\ \text{A} & \text{G} & \text{T} & \text{C} & \text{A} & \text{T} & \text{A} & \end{array} \quad \longrightarrow \quad \begin{array}{cccccccc} \text{A} & \text{C} & \text{G} & \text{T} & \text{T} & \text{A} & \text{T} & \text{A} \\ \text{A} & - & \text{G} & \text{T} & \text{C} & \text{A} & \text{T} & \text{A} \end{array}$$

Fig. 2.24. Pairwise DNA sequence alignment. The figure shows two hypothetical DNA sequences, each composed of the four nucleotides *adenine* (A), *cytosine* (C), *guanine* (G), and *thymine* (T). The sequences on the left are unaligned and seem to differ in all but one position. The sequences on the right have been aligned, and they differ only in two positions. Note that this alignment makes use of an extra symbol, the horizontal bar “-”, which indicates an *indel* (an *insertion* or a *deletion*, depending on the reference sequence).

```

TGGAGACCAC CGTGAACGCC CATCA - - - GG TCC T GCCCAA
TGGAGACCAC CGTGAACGCC CACCA - - - AT TCT T GCCCAA
TGGAGACCAC CGTGAACGCC GCCCA TCT AT TCT T GCCCAA
TGGAGACCAC CGTGAACGCC CATCA - - A AG TCT - GCCCAA
TGGAGACCAC CGTGAACGCC CACCA - - - GG TCT T GCCCAA

```

Fig. 2.25. Multiple DNA sequence alignment. The figure shows a small section of a DNA sequence alignment of five strains of Hepatitis-B virus. Rows represent strains, and columns represent sequence positions. The letters represent the four nucleotides adenine (A), cytosine (C), guanine (G), and thymine (T), while the horizontal bars indicate gaps.

die at a given time and to predict the breakpoint where the corrupt croupier is taking over (in order to nab him).

As a second example, consider the problem of aligning DNA sequences. Recall that DNA is composed of an alphabet of four nucleotides: *adenine* (A), *cytosine* (C), *guanine* (G), and *thymine* (T). After obtaining the DNA sequences of the taxa of interest, we would like to compare homologous nucleotides, that is, nucleotides that have been acquired from the same common ancestor. The problem is complicated due to the possibility of *insertions* and *deletions* of nucleotides in the genome (referred to as *indels*). Take, for instance, Figure 2.24. A direct comparison of the two sequences on the left gives the erroneously small count of only a single site with identical nucleotides. This is due to the

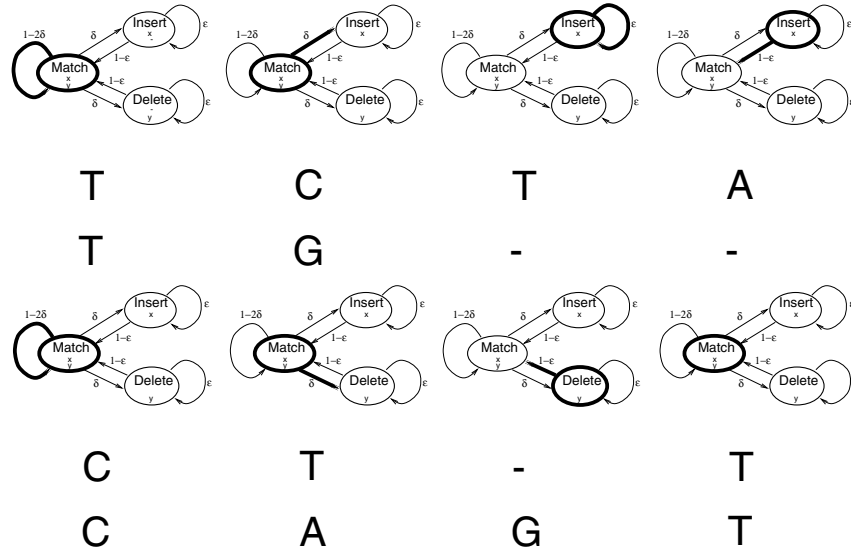


Fig. 2.26. Hidden states for pairwise DNA sequence alignment. The hidden states are represented by ellipses, and edges between these ellipses indicate possible transitions between the states. Active transitions are shown as thick lines. There are three different hidden states. (1) A *match* state emits a pair of nucleotides. (2) An *insert* state emits a nucleotide for the first sequence, and a gap for the second – so it “inserts” a nucleotide in the first sequence. (3) A *delete* state emits a nucleotide for the second sequence, and a gap for the first – hence it “deletes” a nucleotide in the first sequence.

insertion of a *C* in the second position of the first strand, or, equivalently, the deletion of a nucleotide at the second position of the second strand (the insertion of a so-called *gap*). A correct comparison leads to the alignment on the right of Figure 2.24, which suggests that the sequences differ in only two positions. The process of correcting for insertions and deletions is called DNA sequence alignment. Figure 2.25 shows a small subregion of a multiple DNA sequence alignment of five strains of Hepatitis-B virus.

The two examples given here have three important features in common. First, we can describe both processes in terms of a *hidden state* that has generated the observations. For the casino, this hidden state corresponds to the unknown croupier who is rolling the die. For the DNA sequence alignment, we can introduce conceptually a hidden state that indicates whether we have, at a given position, a nucleotide *match*, an *insertion*, or a *deletion*. An illustration is given in Figure 2.26. Second, the problem of finding the correct hidden states corresponding to a given sequence of observations is intrinsically stochastic. Observing, say, that the die face *six* occurs three times in a row gives some indication that the die may be biased. However, due to the inherent stochasticity of rolling a die this observation can also be obtained

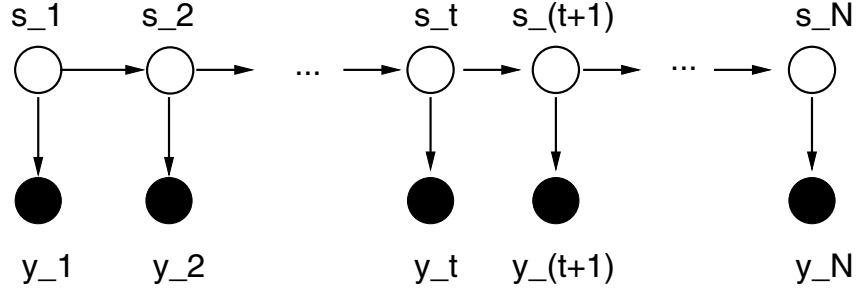


Fig. 2.27. Hidden Markov model. Black nodes represent observed random variables, white nodes represent hidden states, and arrows represent conditional dependencies. The joint probability factorizes into a product of *emission probabilities* (vertical arrows), *transition probabilities* (horizontal arrows), and the *initial probability*, that is, the probability of the initial state. The prediction task is to find the most likely sequence of hidden states given the observations.

from a fair die. Similarly, several mismatches between the nucleotides in corresponding DNA sequence positions may indicate that the sequences are misaligned. However, such mismatches can also occur in homologous sequences as a result of *mutations* during evolution. Consequently, we need probabilistic methods that allow robust inference in the presence of inherent stochasticity, and Bayesian networks are the ideal tools for this task. Third, and possibly most notably, both problems suffer from an explosion of the computational complexity. Given a sequence of observations – a sequence of die faces in the first example, or a sequence of nucleotide pairs in the second – we would like to find the *best* sequence of hidden states describing the observations. Given the intrinsic stochasticity mentioned above, “*best*” should be defined probabilistically as the mode of the posterior probability $P(\mathcal{S}|\mathcal{D})$, where \mathcal{S} represents a sequence of hidden states, and \mathcal{D} is the set of observations (the “*data*”). However, given K different hidden states ($K = 2$ for the casino, and $K = 3$ for the pairwise sequence alignment) and N sequence positions, there are K^N different hidden state sequences. Consequently, the number of hidden state sequences increases exponentially with the sequence length, which, in the most general scenario, prohibits an exhaustive search in sequence space.

To proceed, consider the Bayesian network in Figure 2.27, which contains two types of nodes. Filled circles represent observed random variables, \mathbf{y}_t (which, in general, can be vectors). Empty circles represent hidden states, S_t . The index t can refer to time, as in the casino example, or to location, as in the alignment problem. Applying the expansion rule for Bayesian networks (2.1) to the network in Figure 2.27 gives:

$$P(\mathbf{y}_1, \dots, \mathbf{y}_N, S_1, \dots, S_N) = \prod_{t=1}^N P(\mathbf{y}_t | S_t) \prod_{t=2}^N P(S_t | S_{t-1}) P(S_1) \quad (2.51)$$

We refer to the set of $P(\mathbf{y}_t|S_t)$ as the *emission probabilities* (associated with the vertical edges), to $P(S_t|S_{t-1})$ as the *transition probabilities* (which are associated with the horizontal edges), and to $P(S_1)$ as the *initial probability*.

Since the dependence structure between the hidden states is obviously Markovian, the Bayesian network in Figure 2.27 is called a *hidden Markov model*. A consequence of this simplification is that the complexity of an exhaustive search in the space of hidden state sequences is no longer exponential in the sequence length. From (2.51) we obtain the recursion:

$$\begin{aligned}
\gamma_n(S_n) &= \max_{S_1, \dots, S_{n-1}} \log P(\mathbf{y}_1, \dots, \mathbf{y}_n, S_1, \dots, S_n) \\
&= \max_{S_1, \dots, S_{n-1}} \left[\sum_{t=1}^n \log P(\mathbf{y}_t|S_t) + \sum_{t=2}^n \log P(S_t|S_{t-1}) + \log P(S_1) \right] \\
&= \log P(\mathbf{y}_n|S_n) + \max_{S_{n-1}} \left[\log P(S_n|S_{n-1}) + \max_{S_1, \dots, S_{n-2}} \left[\sum_{t=1}^{n-1} \log P(\mathbf{y}_t|S_t) \right. \right. \\
&\quad \left. \left. + \sum_{t=2}^{n-1} \log P(S_t|S_{t-1}) + \log P(S_1) \right] \right] \\
&= \log P(\mathbf{y}_n|S_n) + \max_{S_{n-1}} \left[\log P(S_n|S_{n-1}) + \gamma_{n-1}(S_{n-1}) \right] \tag{2.52}
\end{aligned}$$

Obviously:

$$\begin{aligned}
\max_{S_1, \dots, S_N} P(S_1, \dots, S_N | \mathbf{y}_1, \dots, \mathbf{y}_N) &= \max_{S_1, \dots, S_N} \log P(\mathbf{y}_1, \dots, \mathbf{y}_N, S_1, \dots, S_N) \\
&= \max_{S_N} \gamma_N(S_N) \tag{2.53}
\end{aligned}$$

and the mode, $P(\hat{\mathcal{S}}|\mathcal{D}) = P(\hat{S}_1, \dots, \hat{S}_N | \mathbf{y}_1, \dots, \mathbf{y}_N)$, is obtained by recursive backtracking, starting from the initialization $\hat{S}_N = \operatorname{argmax}_{S_N} \gamma_N(S_N)$, and continuing with the following iteration:

$$\hat{S}_{n-1} = \operatorname{argmax}_{S_{n-1}} \left[\log P(\hat{S}_n|S_{n-1}) + \gamma_{n-1}(S_{n-1}) \right] \tag{2.54}$$

This recursive iteration is called the *Viterbi algorithm* [36], which is a variant of *dynamic programming*. The computational complexity of a single step of the recursions (2.52) and (2.54) is $\mathcal{O}(K^2)$, that is, it only depends on the number of different states K , but is independent of the sequence length N . The total computational complexity of the algorithm is thus linear in N – rather than exponential in N – which enables us to carry out an exhaustive search even for long sequences. Note, again, that the hidden Markov assumption is at the heart of this reduction in computational complexity. This approximation corresponds to a casino where the decision about changing croupiers is made instantaneously, without considering earlier events in the past. In a pairwise DNA sequence alignment, the hidden Markov assumption restricts the

explicit modelling of the dependence structure between nucleotides to interactions between neighbouring sites. Finally, note that the log transformation in the previous equations is not required for a derivation of the algorithm. It is, however, important in practical implementations in order to prevent a numerical underflow for long sequences.

2.3.5 Application of the EM Algorithm to HMMs

Recall from Section 2.1 that a Bayesian network is defined by a triplet $(\mathcal{M}, \mathcal{F}, \mathbf{q})$. \mathcal{M} represents the network structure, given by Figure 2.27. \mathcal{F} represents the family of transition, emission, and initial probabilities, which are assumed to be known and fixed. The probabilities are thus completely specified by the parameter vector $\mathbf{q} = (\mathbf{w}, \boldsymbol{\nu}, \boldsymbol{\pi})$, where \mathbf{w} determines the emission probabilities, $P(\mathbf{y}_t|S_t, \mathbf{w})$, $\boldsymbol{\nu}$ determines the transition probabilities, $P(S_t|S_{t-1}, \boldsymbol{\nu})$, and $\boldsymbol{\pi}$ determines the initial probabilities, $P(S_1|\boldsymbol{\pi})$.⁵ For a sequence of observations $\mathcal{D} = (\mathbf{y}_1, \dots, \mathbf{y}_N)$ and a state sequence $\mathcal{S} = (S_1, \dots, S_N)$ we have the joint probability (from (2.51)):

$$\begin{aligned} P(\mathcal{D}, \mathcal{S}|\mathbf{q}) &= P(\mathbf{y}_1, \dots, \mathbf{y}_N, S_1, \dots, S_N|\mathbf{q}) \\ &= \prod_{t=1}^N P(\mathbf{y}_t|S_t, \mathbf{w}) \prod_{t=2}^N P(S_t|S_{t-1}, \boldsymbol{\nu}) P(S_1|\boldsymbol{\pi}) \end{aligned} \quad (2.55)$$

Assume we want to optimize the parameters in a maximum likelihood sense, that is, we want to maximize

$$L(\mathbf{q}) = \log P(\mathcal{D}|\mathbf{q}) = \log \sum_{\mathcal{S}} P(\mathcal{D}, \mathcal{S}|\mathbf{q}) \quad (2.56)$$

with respect to the parameter vector \mathbf{q} . The computation of L requires a summation over all hidden state sequences $\mathcal{S} = (S_1, \dots, S_N)$, that is, over K^N terms. For all but very short sequence lengths N , this direct approach is intractable. A viable alternative, however, is given by the expectation maximization (EM) algorithm, discussed in Section 2.3.3. Let $Q(\mathcal{S})$ denote an arbitrary probability distribution over the hidden state sequences, and F the function defined in (2.48). Inserting (2.55) into (2.48) gives:

$$F(\mathbf{q}) = A(\mathbf{w}) + B(\boldsymbol{\nu}) + C(\boldsymbol{\pi}) + H \quad (2.57)$$

where $H = -\sum_{\mathcal{S}} Q(\mathcal{S}) \log Q(\mathcal{S})$ is a constant independent of the parameters \mathbf{q} , and

⁵ Recall the convention that for different arguments, P denotes different functions. Also, note that the three probabilities stated here do not explicitly depend on t , that is, the Markov chain is assumed to be *homogeneous*. For example, if $S_t \in \{a_1, \dots, a_K\}$, then $P(S_t = a_i|S_{t-1} = a_k) = P(S_{t'} = a_i|S_{t'-1} = a_k) \forall t, t'$.

$$A(\mathbf{w}) = \sum_{\mathcal{S}} \sum_{t=1}^N Q(\mathcal{S}) \log P(\mathbf{y}_t | S_t, \mathbf{w}) = \sum_{t=1}^N \sum_{S_t} Q(S_t) \log P(\mathbf{y}_t | S_t, \mathbf{w}) \quad (2.58)$$

$$\begin{aligned} B(\boldsymbol{\nu}) &= \sum_{\mathcal{S}} \sum_{t=2}^N Q(\mathcal{S}) \log P(S_t | S_{t-1}, \boldsymbol{\nu}) \\ &= \sum_{t=2}^N \sum_{S_t} \sum_{S_{t-1}} Q(S_t, S_{t-1}) \log P(S_t | S_{t-1}, \boldsymbol{\nu}) \end{aligned} \quad (2.59)$$

$$C(\boldsymbol{\pi}) = \sum_{\mathcal{S}} Q(\mathcal{S}) \log P(S_1 | \boldsymbol{\pi}) = \sum_{S_1} Q(S_1) \log P(S_1 | \boldsymbol{\pi}) \quad (2.60)$$

Note that these expressions depend only on the marginal univariate and bivariate distributions $Q(S_t)$ and $Q(S_t, S_{t-1})$, but no longer on the multivariate joint distribution $Q(\mathcal{S})$. This marginalization outside the argument of the log function is at the heart of the reduction in computational complexity inherent in the EM algorithm. Having derived an expression for the function F in (2.48), we are set for the application of the EM algorithm, as described in Section 2.3.3.

The probabilities $Q(S_t)$ and $Q(S_t, S_{t-1})$ are updated in the *E-step*, where we set:

$$Q(S_t) \longrightarrow P(S_t | \mathcal{D}, \mathbf{w}, \boldsymbol{\nu}, \boldsymbol{\pi}) \quad (2.61)$$

$$Q(S_t, S_{t-1}) \longrightarrow P(S_t, S_{t-1} | \mathcal{D}, \mathbf{w}, \boldsymbol{\nu}, \boldsymbol{\pi}) \quad (2.62)$$

These computations are carried out with the *forward-backward* algorithm for HMMs [36], which is a dynamic programming method that reduces the computational complexity from $O(K^N)$ to $O(NK^2)$, that is, from exponential to linear complexity in N . The underlying principle is similar to that of the Viterbi algorithm, discussed after equation (2.54), and is based on the sparseness of the connectivity in the HMM structure. Since the forward-backward algorithm has been discussed several times in the literature before, it will not be described in this chapter again. Details can be found in the tutorial article by Rabiner [36], or textbooks like [1] and [9], which also discuss implementation issues.

Now, all that remains to be done is to derive update equations for the parameters $\mathbf{q} = (\mathbf{w}, \boldsymbol{\nu}, \boldsymbol{\pi})$ so as to maximize the function F in the *M-step* of the algorithm. From (2.57) we see that this optimization problem breaks up into three separate optimization problems for \mathbf{w} , $\boldsymbol{\nu}$, and $\boldsymbol{\pi}$. As an example, consider the optimization of $C(\boldsymbol{\pi})$ in (2.60) with respect to $\boldsymbol{\pi}$. Assume the hidden state S_t is taken from a discrete letter alphabet, $S_t \in \{a_1, \dots, a_K\}$. For the casino example, we have $K = 2$ “letters”, corresponding to the two croupiers. For the pairwise sequence alignment, we have $K = \binom{6}{2} - 1 = 14$ letters, corresponding to all 2-element combinations from $\{A, C, G, T, -\}$ except for the concurrence of two gaps. Define $\boldsymbol{\pi} = (\Pi_1, \dots, \Pi_K)$ and $P(S_1 = a_k | \boldsymbol{\pi}) = \Pi_k$ for positive, normalized scalars $\Pi_k \in [0, 1]$, that is, Π_k can be interpreted as a probability

distribution over the alphabet $\{a_1, \dots, a_K\}$. We can now rewrite (2.60) as follows:

$$C(\boldsymbol{\pi}) = \sum_{k=1}^K Q(k) \log \Pi_k = \sum_{k=1}^K Q(k) \log Q(k) - \sum_{k=1}^K Q(k) \log \frac{Q(k)}{\Pi_k}$$

The first term, $\sum_{k=1}^K Q(k) \log Q(k)$, does not depend on the parameters. The second term, $\sum_{k=1}^K Q(k) \log \frac{Q(k)}{\Pi_k}$, is the Kullback–Leibler divergence between the distributions $Q(k)$ and Π_k . In order to maximize $C(\boldsymbol{\pi})$, this term should be as small as possible. Now, recall from the discussion after equation (2.49) that the Kullback–Leibler divergence is always non-negative, and zero if and only if the two distributions are the same. Consequently, $C(\boldsymbol{\pi})$ is maximized for $\Pi_k = Q(S_1 = k)$.

The optimization of the other parameters, \mathbf{w} and $\boldsymbol{\nu}$, is, in principle, similar. For multinomial distributions, complete update equations have been derived in [1], [9], and [36], and these derivations will not be repeated here. Instead, let us consider the maximization of $B(\boldsymbol{\nu})$ in (2.59) for a special case that will be needed later, in Section 5.10. Assume that we have only two different transition probabilities between the hidden states. Let $\nu \in [0, 1]$ denote the probability that a state will not change as we move from position $t-1$ to position t . The probability for a state transition is given by the complement, $1 - \nu$. If a state transition occurs, we assume that all the transitions into the remaining $K-1$ states are equally likely. An illustrative application is shown in Figure 5.20. We can then write the transition probabilities in the following form:

$$P(S_t | S_{t-1}, \nu) = \nu^{\delta_{S_t, S_{t-1}}} \left(\frac{1 - \nu}{K - 1} \right)^{1 - \delta_{S_t, S_{t-1}}} \quad (2.63)$$

where $\delta_{S_t, S_{t-1}}$ denotes the Kronecker delta symbol, which is 1 when $S_t = S_{t-1}$, and 0 otherwise. It is easily checked that (2.63) satisfies the normalization constraint $\sum_{S_t} P(S_t | S_{t-1}) = 1$. Note that the vector of transition parameters, $\boldsymbol{\nu}$, has been replaced by a scalar, ν . Now, define

$$\Psi = \sum_{t=2}^N \sum_{S_t} \sum_{S_{t-1}} Q(S_t, S_{t-1}) \delta_{S_t, S_{t-1}} = \sum_{t=2}^N \sum_{S_t} Q(S_t, S_{t-1} = S_t) \quad (2.64)$$

and note that

$$\sum_{t=2}^N \sum_{S_t} \sum_{S_{t-1}} Q(S_t, S_{t-1}) [1 - \delta_{S_t, S_{t-1}}] = N - 1 - \Psi \quad (2.65)$$

Inserting (2.63) into (2.59) and making use of definitions (2.64) and (2.65) gives

$$B(\nu) = \Psi \log \nu + (N - 1 - \Psi) \log \left(\frac{1 - \nu}{K - 1} \right) \quad (2.66)$$

Setting the derivative of $B(\nu)$ with respect to ν to zero,

$$\frac{dB}{d\nu} = \frac{\Psi}{\nu} + \frac{N - 1 - \Psi}{\nu - 1} = 0 \quad (2.67)$$

we obtain for the optimal parameter ν :

$$\nu = \frac{\Psi}{N - 1} \quad (2.68)$$

This estimation is straightforward because, as seen from (2.64), Ψ depends only on $Q(S_{t-1}, S_t)$, which is obtained by application of the forward-backward algorithm in the E-step (see above). An example for optimizing \mathbf{w} in (2.58) will be given in Section 5.10. Note that the three parameter optimizations for \mathbf{w} , $\boldsymbol{\pi}$, and ν (or ν) constitute the M-step, which has to be applied repeatedly, in each loop of the EM algorithm.

2.3.6 Applying the EM Algorithm to More Complex Bayesian Networks with Hidden States

HMMs are a special class of Bayesian networks with hidden nodes. Their structure, \mathcal{M} , is particularly simple and known. The general scenario of learning arbitrary Bayesian networks with hidden states is more involved, but draws on the principles discussed in the previous sections. This section will provide a brief, not comprehensive, overview.

Let \mathcal{D} denote the set of observations associated with the observable nodes, and denote by \mathcal{S} the set of hidden states with their associated random variables. To optimize the network parameters \mathbf{q} in a maximum likelihood sense, we would like to apply the EM algorithm of Section 2.3.3. The E-step requires us to compute the posterior probability of the hidden states, $P(\mathcal{S}|\mathcal{D}, \mathbf{q}, \mathcal{M})$. For HMMs, this computation is effected with the forward-backward algorithm, as discussed in Section 2.3.5. The update equations in the M-step depend only on the univariate and bivariate marginal posterior distributions, $P(S_t|\mathcal{D}, \mathbf{q}, \mathcal{M})$ and $P(S_t, S_{t-1}|\mathcal{D}, \mathbf{q}, \mathcal{M})$, as seen from (2.57)–(2.62), which leads to the considerable reduction in the computational complexity, from $O(K^N)$ to $O(NK^2)$.

Section 4.4.3 will describe the application of a similar algorithm, *Pearl's message-passing algorithm* [34], to tree-structured Bayesian networks. The most general algorithm for computing the posterior probability of the hidden states is the *junction-tree algorithm* [25], [7]. This algorithm is based on a transformation of the DAG structure into a certain type of undirected graph, the so-called junction tree. The computational complexity of the EM algorithm is exponential in the size of the largest clique in this graph [21]. Here, a *clique* denotes a maximal complete subgraph, where a *complete* graph is

a graph with an edge between any pair of nodes, and a complete graph is *maximal* if it is not itself a proper subgraph of another complete graph. For HMMs, the size of the largest clique is two, hence we need to compute only univariate and bivariate posterior probabilities; see (2.61)–(2.62). The computational complexity is thus $O(K^2N)$, as stated before. If the size of the largest clique in the junction tree is m , the expressions in the M-step depend on m -variate posterior probabilities, and the computational complexity increases to $O(K^mN)$. For large values of m the computational costs thus become prohibitively large, which has motivated the exploration of faster, approximate techniques. Rather than set Q equal to the posterior probability $P(\mathcal{S}|\mathcal{D}, \mathbf{q}, \mathcal{M})$ in the E-step, which corresponds to an unrestricted free-form minimization of (2.49), one can define Q to be a member of a sufficiently simple function family, and then minimize (2.49) subject to this functional constraint. The simplest approach is to set Q equal to the product of its marginals, $Q(\mathcal{S}) = \prod_k Q(S_k)$, which corresponds to the *mean field approximation* in statistical physics (see, for instance, [2], Chapter 4). An application to inference in Bayesian network-like models can be found, for example, in [35]. An improved approach is to use a mixture of mean field approximators [20]. For a more comprehensive overview of these so-called *variational methods*, see [21]. Note, however, that by minimizing the Kullback–Leibler divergence in (2.49) – rather than setting it to zero – the likelihood is no longer guaranteed to increase after an EM step. In fact, this variational variant of the EM algorithm can only be shown to maximize a lower bound on the log-likelihood, rather than the log-likelihood itself [21].

Recall from the discussion in Sections 2.2.1 and 2.2.2 that the objective of inference is either to sample models from the posterior distribution $P(\mathcal{M}|\mathcal{D})$ or, if there is reason to assume that this distribution is peaked, to find the mode of $P(\mathcal{M}|\mathcal{D}) \propto P(\mathcal{D}|\mathcal{M})P(\mathcal{M})$, the *maximum a posteriori* (MAP) model (where due to the NP-hardness of the inference problem this mode, in practice, is usually a local maximum). Also, recall from Section 2.3.2 that under the BIC approximation, $\log P(\mathcal{D}|\mathcal{M}) = \log P(\mathcal{D}|\mathcal{M}, \mathbf{q}) - R(\mathbf{q})$, where $R(\mathbf{q}) = \frac{1}{2} \dim(\mathbf{q}) \log N$ is a regularization term; see (2.46). Now, it is straightforward to modify the EM algorithm such that it (locally) maximizes, for a given model \mathcal{M} , the penalized log-likelihood $\log P(\mathcal{D}|\mathcal{M}, \mathbf{q}) - R(\mathbf{q})$. Instead of maximizing F , defined in (2.48), in the M-step, we have to maximize the modified function $\tilde{F} = F - R(\mathbf{q})$. Then we repeat this procedure for different models \mathcal{M} , using heuristic hill-climbing techniques to find a high-scoring \mathcal{M} . However, each parameter optimization requires several EM cycles to be carried out. As discussed above, the E-step can be computationally expensive, and several E-steps are needed before a single change to the network structure \mathcal{M} can be made. To overcome this shortcoming, Friedman suggested a variant of the EM algorithm, the *structural EM algorithm* [10], where both the parameters \mathbf{q} and the model \mathcal{M} are optimized simultaneously in the M-step. This modification reduces the total number of E-steps that have to be carried out, although at the price of increased computational costs for those

E-steps that are carried out. Friedman also suggested combining the EM algorithm with the integration (2.20) in what he called the *Bayesian structural EM algorithm* (BSEM) [11]. Formally, BSEM is based on a modification of equations (2.48)–(2.49), where the dependence on \mathbf{q} is dropped so as to perform an optimization in model rather than parameter space. Recall that when certain regularity conditions are satisfied, the integral in (2.20) can be solved analytically. The idea in [11] is to carry out this integration within the E-step after imputing the missing values with their expectation values, where expectation values are taken with respect to the distribution obtained in the E-step. This approach is based on approximating the expectation value of a nonlinear function by the value of this function at the expectation value. Also, the expectation value has to be taken with respect to the posterior distribution $P(\mathcal{S}|\mathcal{D}, \mathcal{M})$ – as a result of the aforementioned modification of (2.48)–(2.49). Computing this expectation value is usually intractable. Hence, $P(\mathcal{S}|\mathcal{D}, \mathcal{M})$ is approximated by $P(\mathcal{S}|\mathcal{D}, \mathcal{M}, \hat{\mathbf{q}})$, where $\hat{\mathbf{q}}$ is the MAP parameter estimate for model \mathcal{M} ; see [11] for details.

2.3.7 Reversible Jump MCMC

The EM algorithm, discussed in the previous subsections, is an optimization algorithm. Its application is motivated by the BIC score (2.46), according to which an integration over the parameters \mathbf{q} can be replaced by an optimization. However, as discussed in Section 2.3.2, the BIC approximation becomes unreliable for sparse data. Also, for sparse data, the posterior distribution over structures, $P(\mathcal{M}|\mathcal{D})$, becomes diffuse and is not appropriately summarized by its mode, $\mathcal{M}^* = \operatorname{argmax} P(\mathcal{M}|\mathcal{D})$. Consequently, the optimization approach should be replaced by a sampling approach when the training data are sparse.

Now, recall from page 31 that the MCMC scheme is not restricted to a space of cardinal entities (model structures), but can readily be extended to continuous entities (model parameters). So rather than perform the sampling in the structure space $\{\mathcal{M}\}$, which is impossible due to the intractability of (2.20), we can sample in the product space of structures and parameters, $\{\mathcal{M}, \mathbf{q}\}$, and then marginalize over the parameters \mathbf{q} . Examples for this will be given in Chapters 4–6. However, care has to be taken when the dimension of the model \mathcal{M} changes. In modelling genetic networks, for instance, the model dimension may vary, changing every time we introduce or remove a hypothetical hidden agent (like a transcription factor⁶). In this case the probability distribution over the parameters \mathbf{q} becomes singular when the model dimension increases, and this has to be taken care of in the formulation of the algorithm. A generalization of the classical Metropolis–Hastings algorithm that allows for these dimension changes has been given by Green [14] and is usually referred to as the *reversible jump MCMC* or *Metropolis–Hastings–Green* algorithm. The details are beyond the scope of this chapter.

⁶ A transcription factor is a protein that initiates or modulates the transcription of a gene; see Figure 8.14.

2.4 Summary

This chapter has given a brief introduction to the problem of learning Bayesian networks from complete and incomplete data. The methods described here will reoccur several times in the remaining chapters of this book. For example, methods of phylogenetic inference from DNA or RNA sequence alignments, covered in Chapters 4 and 6, the detection of recombination between different strains of bacteria and viruses, discussed in Chapter 5, as well as the attempt to infer genetic regulatory interactions from microarray experiments, described in Chapters 8 and 9, are all based on the concepts and ideas outlined in the present chapter. The detailed procedures and methods for the particular applications will be discussed in the respective chapters.

Acknowledgments

Several ideas for this chapter have been taken from the tutorials by David Heckermann [16], Paul Krause [23], and Kevin Murphy [31], as well as a lecture given by Christopher Bishop at the 9th International Conference on Artificial Neural Networks in Edinburgh, 1999. I would like to thank Anja von Heydebreck, Marco Grzegorzczuk, David Allcroft and Thorsten Forster for critical feedback on a first draft of this chapter, as well as Philip Smith for proofreading the final version.

References

- [1] P. Baldi and P. Brunak. *Bioinformatics – The Machine Learning Approach*. MIT Press, Cambridge, MA, 1998.
- [2] R. Balian. *From Microphysics to Macrophysics. Methods and Applications of Statistical Physics.*, volume 1. Springer-Verlag, 1982.
- [3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995. ISBN 0-19-853864-2.
- [4] S. Chib and E. Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.
- [5] D. M. Chickering. A transformational characterization of equivalent Bayesian network structures. *International Conference on Uncertainty in Artificial Intelligence (UAI)*, 11:87–98, 1995.
- [6] D. M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H. J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics*, volume 5, pages 121–130, New York, 1996. Springer.
- [7] A. P. Dawid. Applications of general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.
- [8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B39(1):1–38, 1977.

- [9] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis. Probabilistic models of proteins and nucleic acids*. Cambridge University Press, Cambridge, UK, 1998.
- [10] N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In D. H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning (ICML)*, pages 125–133, Nashville, TN, 1997. Morgan Kaufmann.
- [11] N. Friedman. The Bayesian structural EM algorithm. In G. F. Cooper and S. Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 129–138, Madison, WI, 1998. Morgan Kaufmann.
- [12] N. Friedman, I. Nachman, and D. Pe’er. Learning Bayesian network structure from massive datasets: The “sparse candidate” algorithm. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 196–205, San Francisco, CA, 1999. Morgan Kaufmann Publishers.
- [13] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. Introducing Markov chain Monte Carlo. In W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors, *Markov Chain Monte Carlo in Practice*, pages 1–19, Suffolk, 1996. Chapman & Hall. ISBN 0-412-05551-1.
- [14] P. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732, 1995.
- [15] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.
- [16] D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*, Adaptive Computation and Machine Learning, pages 301–354, The Netherlands, 1998. Kluwer Academic Publishers.
- [17] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:245–274, 1995.
- [18] D. Husmeier. *Neural Networks for Conditional Probability Estimation: Forecasting Beyond Point Predictions*. Perspectives in Neural Computing. Springer, London, 1999. ISBN 1-85233-095-3.
- [19] D. Husmeier. The Bayesian evidence scheme for regularising probability-density estimating neural networks. *Neural Computation*, 12(11):2685–2717, 2000.
- [20] T. S. Jaakola and M. I. Jordan. Improving the mean field approximation via the use of mixture distributions. In M. I. Jordan, editor, *Learning in Graphical Models*, Adaptive Computation and Machine Learning, pages 163–173, The Netherlands, 1998. Kluwer Academic Publishers.
- [21] M. I. Jordan, Z. Ghahramani, T. S. Jaakola, and L. K. Saul. An introduction to variational methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 105–161, The Netherlands, 1998. Kluwer Academic Publishers.

- [22] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
- [23] P. J. Krause. Learning probabilistic networks. *Knowledge Engineering Review*, 13:321–351, 1998.
- [24] S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H. G. Leimer. Independence properties of directed Markov fields. *Networks*, 20:491–505, 1990.
- [25] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. *Journal of the Royal Statistical Society, Series B*, 50:157–224, 1988.
- [26] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.
- [27] D. J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472, 1992.
- [28] D. J. C. MacKay. Introduction to Monte Carlo methods. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 301–354, The Netherlands, 1998. Kluwer Academic Publishers.
- [29] D. Madigan and J. York. Bayesian graphical models for discrete data. *International Statistical Review*, 63:215–232, 1995.
- [30] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.
- [31] K. P. Murphy. An introduction to graphical models. Technical report, MIT Artificial Intelligence Laboratory, 2001. <http://www.ai.mit.edu/~murphyk/Papers/intro-gm.pdf>.
- [32] K. P. Murphy. Bayes net toolbox. Technical report, MIT Artificial Intelligence Laboratory, 2002. <http://www.ai.mit.edu/~murphyk/>.
- [33] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368, The Netherlands, 1998. Kluwer Academic Publishers.
- [34] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988.
- [35] C. Petersen and J. R. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.
- [36] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [37] J. J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
- [38] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.
- [39] H. Sies. A new parameter for sex-education. *Nature*, 332:495, 1988.
- [40] P. Spirtes, C. Meek, and T. Richardson. An algorithm for causal inference in the presence of latent variables and selection bias. In G. Cooper and C. Glymour, editors, *Computation, Causation, and Discovery*, pages 211–252. MIT Press, 1999.

Probabilistic Modeling in Bioinformatics and Medical
Informatics

Husmeier, D.; Dybowski, R.; Roberts, S. (Eds.)

2005, XX, 508 p., Hardcover

ISBN: 978-1-85233-778-0